Chapter 5

CBIM: Community-based influence maximization in multilayer networks

One of the disadvantages of SIM is the lack of quality communities and seed nodes. This chapter[1], proposes CBIM, community based influence maximization in multilayer networks. CBIM overcomes the disadvantages of SIM. CBIM also finds the most influential nodes and it also uses community structure to find seed nodes. CBIM has two phases: In the first phase, CBIM uses the function $FIC(M)$ to find the small communities from a multilayer network based on dice neighborhood similarity. It uses the function $CSC(CS_{init}, \theta)$ to merge smaller communities and generate larger communities to improve communities' quality. In the second phase, CBIM computes Edge Weight Sum (EWS) for each node in a community and ranks the nodes based on EWS. CBIM uses the quota-based approach to select the seed node set from the communities based on the ranks. This chapter shows, a comparative study of various influence maximization (IM) algorithms that the CBIM algorithm performs better than the state-of-the-art. The simulation studies have shown that CBIM can detect a set of most influential nodes on real-time datasets under various settings and environments.

[1]Venkatakrishna Rao. K, C. Ravindranath Chowdary, CBIM: Community-based Influence Maximization in Multilayer Networks, Information Sciences, 2022, ISSN 0020-0255,https://doi.org/10.1016/j.ins.2022.07.103. Elsevier, 2022

Table 5.1: Notations used in this chapter.

Notation	Description
N	Number of nodes per layer in multilayer network
G^{α}	α layer graph in Multilayer network M
E^{α}	edges in α layer in Multilayer network M
K	Number of seed nodes
p	Propagation probability in IC model
$\sigma(S)$	Influence Spread achieved by seed set S
d_v^l	Degree of v in l^{th} layer
$nbrs(v^l)$	Neighbours of node v in l^{th} layer
$actnbrs(v^l)$	Active neighbours of node v in l^{th} layer
t_v^l	Number of neighbours of node v in layer l
	that are already selected as seed node
b_{w^l,v^l}	Propagation probability of a node v^l influence by w^l
u^l	Node u in layer l

The contribution of the proposed work is given as follows.

- We adapted a community detection algorithm [19] to a multilayer network.
- We propose EWS to assign edge weights based on degree and distance concerning that node in multilayer network.
- We propose a seed selection process based on a quota-based approach by considering community structure.
- Through experiments, we compare the performance of the proposed algorithm against the state-of-the-art algorithms under the IC and LT diffusion models on real-world social networks.

5.1 Basic model of multilayer networks

Let us consider a multilayer network $M = \{G^1, G^2, ..., G^{\alpha}, .., G^M\}$ [65], where each graph $G^{\alpha} = (V, E^{\alpha})$ is formed by the set of N nodes in each layer. $V = \{i; i = 1, 2, 3..N\}$ and E^{α} is set of edges in layer α. Each layer consists of same set of nodes N. Table 5.1 presents the notations used in this chapter.

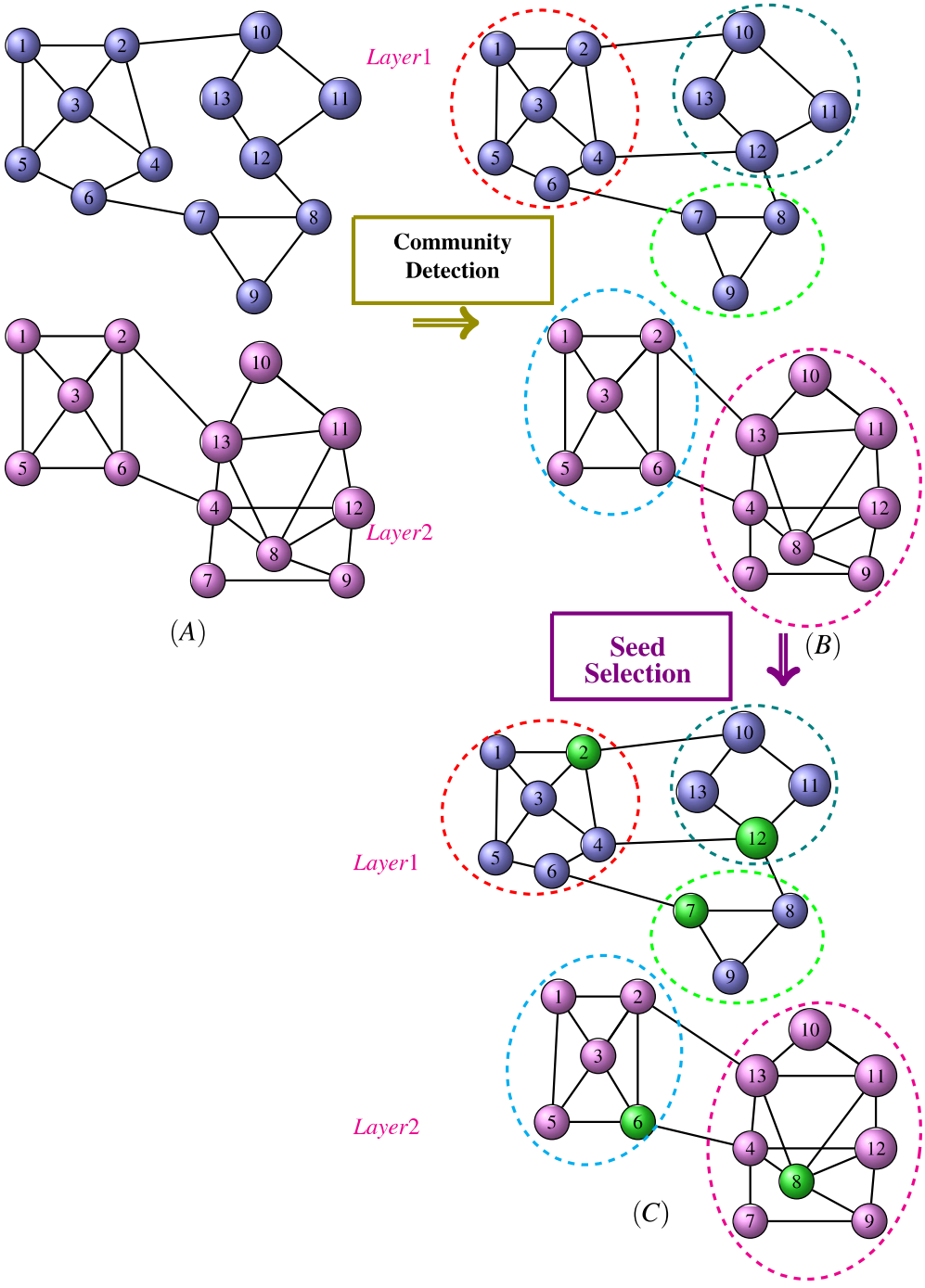


Figure 5.1: Overview of CBIM model

The information propagation dynamics of a multilayer network are different from a single layer network. To demonstrate the information spreading in a multilayer network, let us consider a simple multilayer network in Figure 5.1. Figure 5.1 (A) shows two different graphs in a multilayer network, where one graph in layer one and another graph in layer 2; 13 nodes exist in each layer. Figure 5.1 (B) shows the two graphs after identifying communities in a multilayer network, three communities are identified in layer 1 and two communities are identified in layer 2; layer 1 has $C1 = \{1,2,3,4,5,6\}, C2 = \{7,8,9\}$ and $C3 = \{10,11,12,13\}$; layer 2 has $C1 = \{1,2,3,5,6\}, C2 = \{4,7,8,9,10,11,12,13\}$. Figure 5.1 (C) shows seed nodes in different communities, i.e., nodes 2, 7, and 12 are selected as seed nodes in layer one, and node 6, 8 are selected as seed nodes in layer 2. These seed nodes will propagate the information to other nodes after running a diffusion model.

5.2 Community Based Influence Maximization (CBIM)

In this section, we discuss the proposed model, which consists of two phases. In the first phase, the function $FIC(M)$ finds the small communities from a multilayer network based on dice neighborhood similarity, and then the function $CSC(CS_{init}, \theta)$ merges some small communities of $FIC(M)$ and forms final communities to improve the quality of communities. In the second phase, we find the EWS for each node in a community and assign a ranking to the nodes based on the EWS, then we select the seed node set from the communities based on assigned ranking using the quota-based approach. The proposed method is outlined in Algorithm 5.

Algorithm 5 CBIM: Community-based Influence Maximization

1: ▷ $FIC(M)$ is Function for finding initial communities, CS_{init} is set of initial communities, FC is final community set, θ is community scale, $CSC()$ is community consolidation function
2: Multilayer network $M = \{G^1, G^2, ..., G^\alpha, .., G^M\}$
3: Form the initial community structure

$CS_{init} \leftarrow FIC(M)$ ▷ FIC() is discussed in Section 4.1.2

4: Combine small communities in *CS_init*

$FC \leftarrow CSC(CS\_init, \theta)$ ▷ $CSC()$ is discussed in Section 4.1.3

5: Find the EWS of all the nodes.
6: Identify seed nodes from communities based on quota-based approach using EWS.

5.2.1 Identification of communities

Here we discuss the first phase of CBIM, i.e., finding the communities in multilayer networks. Identification of communities is similar to a standard community detection algorithm. Finding the communities consists of two parts. In the first part, we identify initial communities based on neighborhood similarity. In the second part, we merge small communities based on the merging index factor to form final communities.

5.2.1.1 Neighborhood similarity selection

Several methods have been proposed in the literature for community detection, i.e., based on the distance between two nodes, paths between nodes, and the number of familiar neighbors, etc. However, those methods depend on global factors such as eigenvalues, euclidean distance, and prior knowledge; getting these factors is hard to acquire due to the large networks involved, and most of them are computationally complex. To overcome it, some neighborhood similarity measures have been tested to select the best communities for our influence maximization algorithm, such as the jaccard similarity coefficient [33], otsuka similarity coefficient [63], overlap similarity coefficient [72], and dice similarity coefficient [25]. Among these, dice similarity is giving some impressive results as it considers twice the number of common neighbors of nodes u^l and v^l.

Dice similarity coefficient (DSC)[25]: It is defined as the two times the number of common nodes in both neighbors sets divided by the total number of nodes in two neighbor sets. Dice similarity coefficient is $DSC(u^l, v^l)$,

$$DSC(u^l, v^l) = \frac{2 * |nbrs(u^l) \cap nbrs(v^l)|}{|nbrs(u^l)| + |nbrs(v^l)|} \tag{5.1}$$

$nbrs(u^l)$ is neighbors of node u in layer l and $nbrs(v^l)$ is neighbors of node v in layer l. After comparing various similarity indexes with our proposed algorithm, we have selected the dice similarity index for finding communities in the network since its performance is better than the remaining similarity measures. Figure 5.3 shows the IM performance of different similarity measures with our CBIM model.

5.2.1.2 Finding initial communities (FIC)

Here, we discuss part one of identifying communities, i.e., the initial community structure from a multilayer network. Cheng et al.[27] has proposed community detection in a single layer network using neighborhood similarity. We adapted [27] and applied it to multilayer networks for community detection. First, select the highest degree node from the network and make it a new community. Next, add the node with the maximum dice similarity coefficient into the new community. After this, select the next highest degree node from the remaining network and find the node with the maximum dice similarity coefficient. If the most similar node is already in a community, add the next highest degree node to that community. Otherwise, we create a new community for the next highest degree node and its most similar node. This process is repeated until every node is in some community. In Algorithm 6, part one explains the procedure of finding the initial community structure.

The initial community structure works based on node selection and node similarity from the above discussion. Generated communities in part one are too small, and a large number of small communities are generated. The idea of community detection will not be fulfilled with small communities, and it fails the fundamental characteristic of the community.

5.2.1.3 Community consolidation (CSC)

In the first part of community detection, there are many small communities. Edges with in community are less than outgoing edges of the community. Selecting seed nodes from these small communities will not give impressive results, as it violates the fundamental characteristic of a community. It is essential to merge small communities to make a final community structure. In the second part, we have proposed community consolidation, i.e., merging small communities to form sizable communities. In merging small communities, the first step is to find the communities to be merged, and the second step is to select the communities into which each small community is to be merged.

For the first step, we propose a merging index to find the communities that need to be merged. The merging index considers two factors: community conductance and community scale.

Definition 5 (Community conductance [79])**.** Conductance is to measure the quality of the community. Conductance is computed as given in Equation 5.2.

$$C_i^l(\gamma) = \frac{|E_i^{out}|}{2|E_i^{in}| + |E_i^{out}|} \tag{5.2}$$

where E_i^{out} is set of edges connecting community C_i^l with other communties and E_i^{in} is set

Algorithm 6 Community detection on neighborhood similarity measure

/* **Part 1 - Initial communities detection- FIC(M)** */

Initialize the variables NS and CS_{init} which are used to record node set and initial community structure.

$$NS \leftarrow V, CS_{init} \leftarrow \Phi$$

Select the highest degree node v^l from the node set (NS)

Get the most similar neighbour (sn^l) of v^l using Dice's neighbour similarity.

If sn^l is not in any community **then**

    Create a new community and assign v^l and sn^l to it.

$$K \leftarrow |CS_{init}|; C^l_{k+1} \leftarrow \{v^l, sn^l\}$$

    Insert the new community into commmunity structure.

    Remove nodes v^l and sn^l from NS as they are classified

$$NS \leftarrow NS - \{v^l, sn^l\}$$

else

    Find the community to which sn^l belongs to and denote it as C^l_k

$$k \leftarrow locate(CS_{init}, sn^l)$$

    Insert node v^l into C^l_k and Remove the node v^l from NS

$$C^l_k \leftarrow C^l_k \cup \{v^l\} \; and \; NS \leftarrow NS - \{v^l\}$$

Repeat the steps from 2 to 10, until $NS = \Phi$

/* **Part 2 - Community consolidation- CSC**(CS_{init}, θ) */

Intialize Final Communities (FC)

$$FC \leftarrow CS_{init}$$

Step 1: Calculate community conductence (γ_i) and community scale (θ_i) for each community C^l_i in CS_{init}.

    Calculate the merging index (ψ_i) for each community in CS_{init}.

$$C^l_i(\psi) = C^l_i(\theta) * C^l_i(\gamma)$$

    Select the community with lowest merging index (C^l_x)

Step 2: Find the most similar community (C^l_y) to (C^l_x) and merge the two communities to form a new community (C^l_n).

$$s \leftarrow argmax_i\{Sim(C^l_x, C^l_y) | i = 1, 2, ...CS, x \neq y\} \; and \; C^l_n = C^l_x \cup C^l_y$$

    Calculate the merging index (ψ_n) for new community (C^l_n)

    Replace two communities C^l_x and C^l_y with new community C^l_n in final community set (FC)

Repeat the process from 15 to 18 until $\psi^l_i > \delta$

return FC

of edges in community C_i^l. The best communities are a densely connected set of nodes attached to the rest of the network via a few edges.

Definition 6 (Community scale [27])**.** Scale of community $C_i^l(\theta)$ is defined as

$$C_i^l(\theta) = \frac{|V_i^l|}{|V|} \tag{5.3}$$

where V_i^l is a set of nodes in community C_i^l and V is a set of nodes in a multilayer network based on the above definitions.

Definition 7 (Merging index)**.** Merging index of community $C_i^l(\psi)$ is the combination of community conductance and community scale, which is defined for community C_i^l as follows:

$$C_i^l(\psi) = C_i^l(\theta) * C_i^l(\gamma) \tag{5.4}$$

the first step can be solved by setting the merging index threshold, δ. If $\psi_i < \delta$ then community C_i^l is a candidate community for merging.

The second step is to select the communities with which each small community is merged based on the similarity between communities, i.e., small communities are merged into their adjacent similar communities. The similarity between the two communities C_i^l and C_j^l is given as follows:.

$$Sim(C_i^l, C_j^l) = \frac{\Sigma_{u^l \in C_i^l, v^l \in C_j^l} DSC(u^l, v^l)}{|C_j^l|} \tag{5.5}$$

where $DSC(u^l, v^l)$ is dice similarity between nodes $u^l \in C_i^l$, $v^l \in C_j^l$. C_i^l needed to be merged in C_j^l. $\Sigma_{u^l \in c_i^l, v^l \in c_j^l} sim(u^l, v^l)$ is sum of dice similarities between node u^l in community C_i and v^l in community C_j in layer l. In, Algorithm 6, Part 2 explains the community consolidation procedure. Part 2 is to improve the quality of community structure by merging small communities generated in Part 1. Identifying candidate communities is to be merged and computing similarity between communities is computationally expensive.

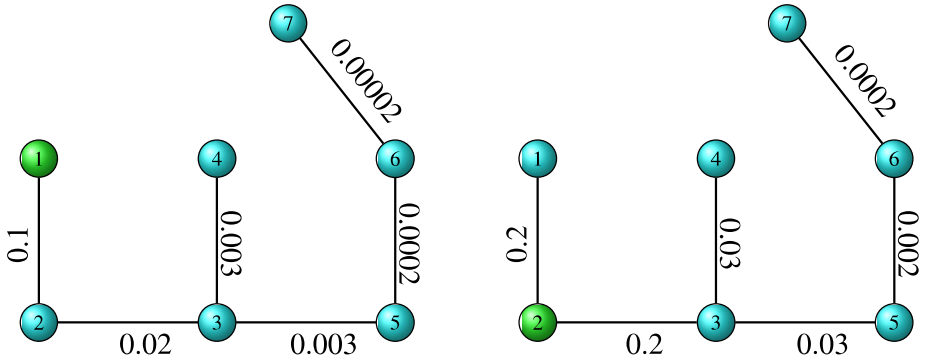


Figure 5.2: (A) Weight distribution to the edges for node 1 and EWS for node 1 is 0.12622 (B) Weight distribution to the edges for node 2 and EWS for node 2 is 0.4622

5.2.2 Selection of seed nodes

In this section we discuss the second phase of the CBIM and present seed node selection process after finding all the existing communities in a multilayer network. After finding the communities, CBIM calculates the EWS for each node in communities. Based on the EWS, CBIM select the seed nodes from each community using a quota-based approach. Algorithm 7 explains the procedure to select the seed nodes.

5.2.2.1 Edge Weight Sum (EWS)

The seed nodes are determined to maximize the influence over a network. The EWS is similar to Katz centrality [37]. Unlike Katz centrality, EWS considers degree and distance to compute the weight of the edge. If a node j is far away from node i, EWS penalizes for distance, but it rewards based on the degree of j. The main idea of computing EWS is to weight each node in the community based on the performance, before selecting seed nodes from the communities. Objective of calculating the EWS is to find the node's connectivity over the network. Node's connectivity depends not only on its degree but also on its neighbor's connections and neighbors of neighbors connections etc. EWS of a node i in layer l is calculated as

$$EWS_i^l = \Sigma_{hl=1}^{\infty}\Sigma_{j=1}^{n}[\alpha^{hl} * d_j^l] * (A^{hl})_{ij} \tag{5.6}$$

EWS_i^l is the edge weight sum of node i in layer l, n is the number of nodes in community (C_i^l), the adjacency matrix A is a n × n matrix. $(A^{hl})_{ji}$ is the number of walks of length hl starting from node i to node j. d_j^l is degree of node j in layer l. α is the attenuation factor, and

the value is fixed as 0.1 because it gave the best results for all kinds of graphs and experiments. Figure 5.2 gives an example of how to distribute the edge weight of a simple graph concerning nodes 1 and 2; it also explains the computation of the EWS of a node. EWS for node 1 in Figure 5.2 is 0.12622, node 2 is 0.4622, node 3 is 0.942, node 4 is 0.1642, node 5 is 0.482, node 6 is 0.4262 and node 7 is 0.12262.

Algorithm 7 Finding EWS and seed node selection

Input: Final Communities (FC)

Ouput: Seed Node set (SN)

1: Intialize the EWS, Seed node set (SN) and Community set (CS).

$$SN \leftarrow \Phi,\ CS \leftarrow FC,\ EWS \leftarrow \Phi$$

2: **for** C_i^l to CS **do**

3: **for** j to C_i^l **do**

4: Calculate the EWS for node j in community (C_i^l).

$$EWS_i^l = \Sigma_{hl=1}^{\infty}\Sigma_{j=1}^{n}[\alpha^{hl} * d_j^l] * (A^{hl})_{ij}$$

5: Insert the EWS of node j in community C_i^l $(EWS_j(C_i^l))$ factor into EWS of community C_i^l $(EWS(C_i^l))$.

$$EWS(C_i^l) \leftarrow EWS(C_i^l) \cup EWS_j(C_i^l)$$

6: **end for**

7: **end for**

8: Sort all the nodes in each community based on EWS in descending order.

9: Calculate required seed nodes from each community in quota based approach.

$$Quota(C_i^l) = k * \frac{n_i}{|V|}$$

10: Select the quota number of highest EWS nodes as seed nodes from each community (C_i^l).

$$SN = SN \cup Quota(C_i^l)$$

11: return SN

5.2.2.2 Quota based selection

Here we discuss the selection of seed nodes from the generated communities; after finding the EWS of all the nodes in each community, we sort all the nodes in descending order based on the EWS. Then select the $Quota(C_i^l)$ number of highest EWS nodes from the community C_i^l.

Calculation of $Quota(C_i^l)$ for community C_i in layer l is

$$Quota(C_i^l) = k * \frac{n_i}{|V|} \tag{5.7}$$

where k is the number of seed nodes, n_i is the number of nodes in community C_i^l in a multilayer network. $|V|$ is the number of nodes in multilayer networks. The Sum of all selected seed nodes from communities will form the seed node-set (SN).

5.2.2.3 Time Complexity

Here we discuss the time complexity of the CBIM model. We calculated time complexity for the community detection phase, i.e., for Algorithm 6, and seed node selection phase, i.e., Algorithm 7. Let the time complexity for finding the degree of each node in a multilayer network be $O(V^2)$. The time complexity for finding the highest degree node is $O(VlogV)$, and the time complexity for finding community conductance is $O(2VM)$, where M is the size of the community. The time complexity for the community scale is $O(1)$, and the total time complexity for community detection of CBIM is $O(V^2 + VlogV + 2VM + 1)$. In the seed node selection phase, the time complexity for finding the EWS is $O(CS^3)$, where CS is the size of community. The time complexity to sort the nodes in descending order is $O(MlogM)$. The time complexity to select the seed nodes is $O(K)$. Time complexity for seed node selection phase is $O(CS^3V + MlogM + V)$.

5.3 Baseline models

We compare CBIM with three models, that are KSN [42] introduced in Chapter 3 and IMCS [10] and CIM [36] in Chapter 4. We also compare CBIM with one more model as given below.

- **IM-ELPR [43]**: Influence maximization using Extended H-index, label propagation with relationship matrix (IM-ELPR) has three steps. In the first step, use the h-index centrality to use the seed nodes and the use label propagation technique to detect communities. In the second step, with the help of the relation matrix, merge small communities to form large communities. In the third step, k-influential nodes will be selected from communities.

5.4 Results

In this section, we compare the proposed algorithm, CBIM, with the algorithms described in Section 5.3, using the datasets listed in Section 3.2. We analyze the influence maximization of the algorithms and the execution time for finding the seed nodes. We compare CBIM with two community-based algorithms and two algorithm-based algorithms. We have reported the results of various algorithms on different datasets by fixing the seed set size k as 50.

We discuss two scenarios. In the first scenario, we present the performance of the proposed algorithms for influence maximization using IC and LT models. In the second scenario, we report the execution time for finding seed nodes and the execution time for influence spread using IC and LT models.

5.4.1 Comparison of neighborhood similarities for CBIM algorithm

This section compares the influence maximization of different similarity indexes with the CBIM algorithm on different datasets. Figure 5.3a presents the IM of various similarities with CBIM algorithm on Celegans multiplex GPI network dataset and dice similarity coefficient has performed better than other neighbor similarities. In addition to this, we also carried out four similarity experiments with some real-time datasets.

5.4.2 Performance of CBIM on Real networks using IC model

Here, we present influence maximization on six real-time datasets for influence maximization. Under each dataset in Table 3.2, graphs have been used to apply on the independent cascade model. We also consider directed graphs as undirected graphs by ignoring the edge directions and applying them to the IC model for influence maximization. We find the influenced nodes for discussed algorithms under the IC model. Figure 5.4a shows the influence maximization on London multiplex Transport network, CBIM has performed better than all other algorithms. CIM has performed better than IM-ELPR, KIN, and IMCS. IMCS has performed less than other algorithms. Figure 5.4b shows the influence maximization on Arxiv net Science multiplex, IM-ELPR, and CIM have influenced almost the same number of nodes, KSN and IMCS influenced the same number of nodes. CIM has performed better than other algorithms. Due to the graph connectivity, seed nodes and some nodes may influence the exponential number of nodes, leading to a sudden curve rise. Figure 5.5a shows the influence maximization on Celegans multiplex GPI network dataset; CBIM performed better than all other algorithms. IMCS has influenced

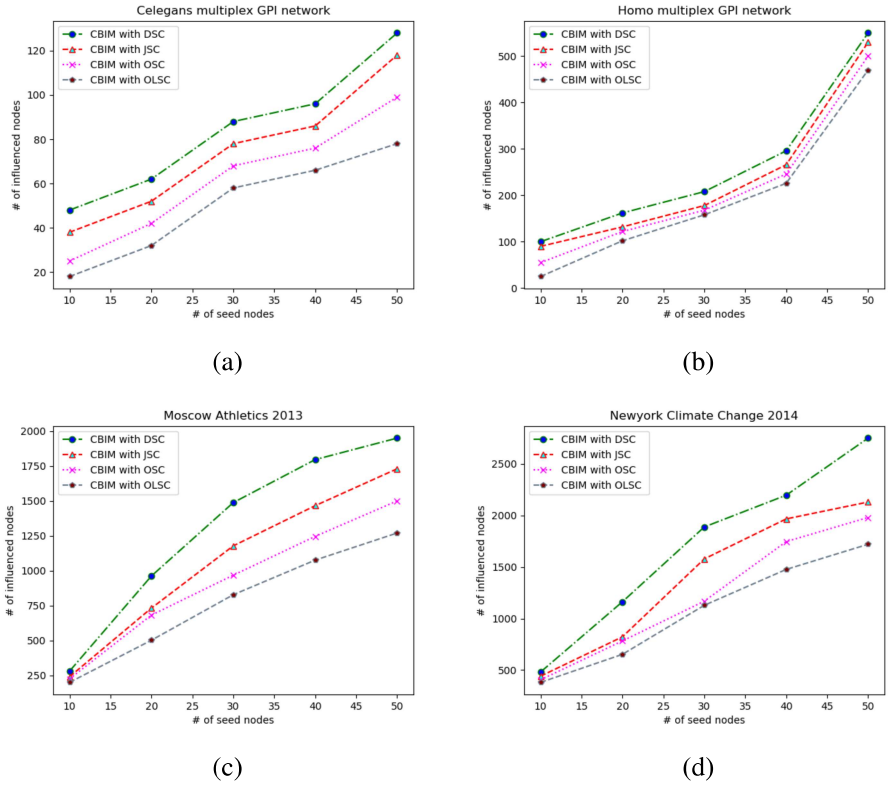


Figure 5.3: Influence spread of 50 seed nodes for different similarities with our proposed CBIM algorithm under various datasets.

less number of nodes. IM-ELPR has performed better than heuristic algorithms. KSN and CIM performed better than the IMCS. Figure 5.5b shows the influence maximization on Homo multiplex GPI network dataset, CBIM influenced marginally more number of nodes than CIM and IM-ELPR. KSN and IMCS influenced the same number of nodes.

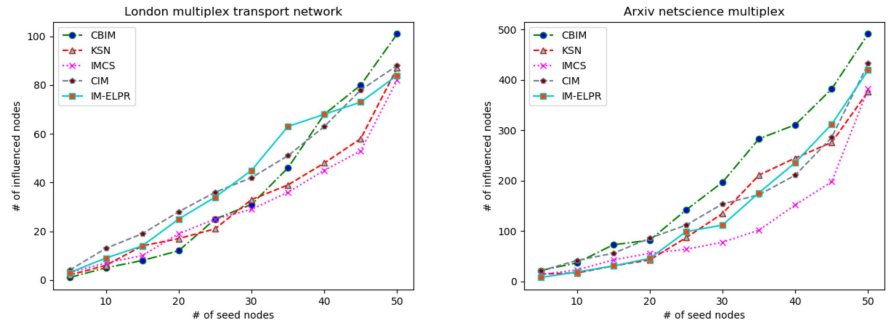
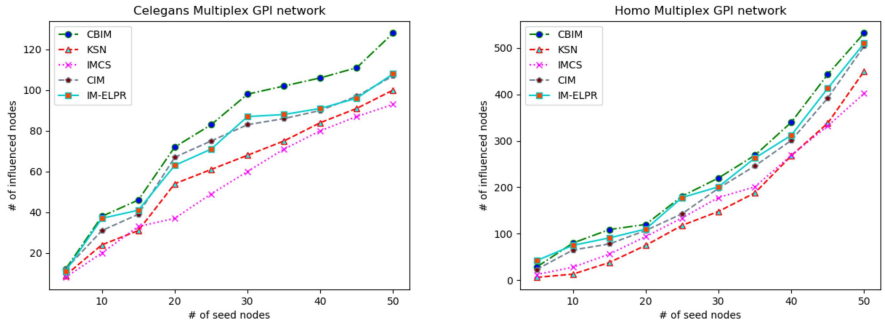


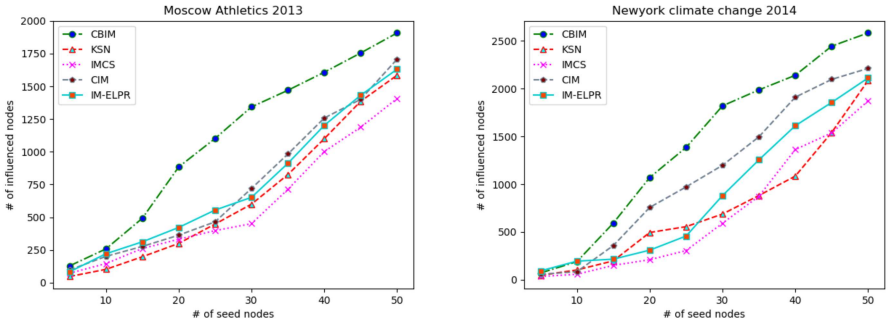
Figure 5.4: (a) Influence spread of 50 seed nodes for different algorithms using IC model on London Multiplex Transport Network. (b) Influence spread of 50 seed nodes for different algorithms using IC model on Arxiv netscience multiplex for influence maximization.

Figure 5.6a shows the influence maximization on Moscow Athletics 2013 datasets. CBIM has influenced more nodes, IMCS has influenced more nodes than KSN. CIM is performed better than KSN and IMCS. Figure 5.6b shows the influence maximization on Newyork climate change 2014. CIM has performed better than IM-ELPR, KSN, and IMCS. CBIM performed better than all other algorithms.



(a) (b)

Figure 5.5: (a) Influence spread of 50 seed nodes for different algorithms using IC model on Celegans multiplex GPI network. (b) Influence spread of 50 seed nodes for different algorithms using IC model on Homo multiplex GPI network for influence maximization.



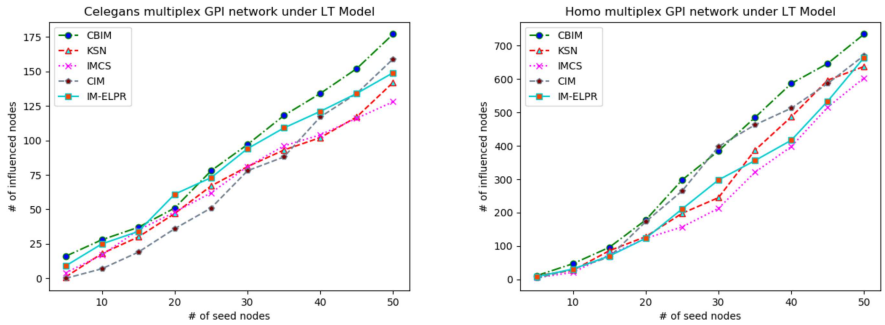
(a) (b)

Figure 5.6: (a) Influence spread of 50 seed nodes for different algorithms using IC model on Moscow Athletics 2013. (b) Influence spread of 50 seed nodes for different algorithms using IC model on Newyork climate change 2014 for influence maximization.

5.4.3 Performance of CBIM on Real networks using LT model

This section discusses the influence maximization of different algorithms under the LT model. In Table 3.2, directed graph datasets have been used to apply on linear threshold model. Figure 5.7a shows the influence maximization of discussed algorithms on Celegans multiplex GPI network, for the same seed nodes and same datasets, LT model has influenced marginally more number of nodes than IC model. CBIM, IM-ELPR, and CIM have influenced almost the same number of nodes. IMCS and KSN have influenced the same number of nodes. Figure 5.7b shows the influence maximization of discussed algorithms on the Homo multiplex GPI network. CBIM has influenced more nodes than other algorithms; CIM and IM-ELPR have performed marginally more than KSN.

Figure 5.8a shows the influence maximization of discussed algorithms on Moscow Athletics 2013; CIM has performed better than all other algorithms. IM-ELPR has performed better than KSN and IMCS. Figure 5.7b shows the influence maximization of discussed algorithms on the Newyork climate change 2014 dataset; However, IM-EPLR has performed better than CIM at the beginning, IM-EPLR and CIM have influenced the same number of nodes at the end.

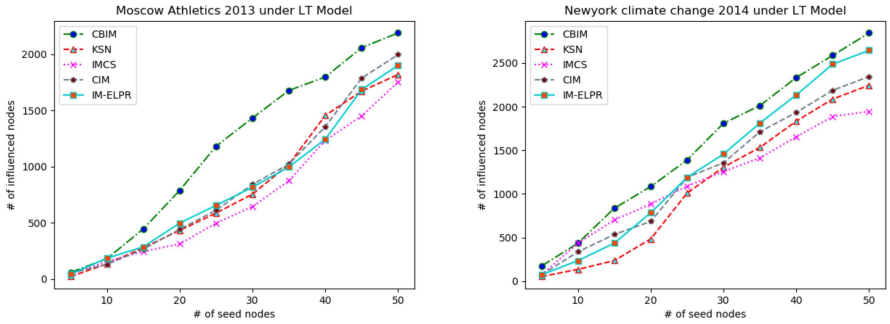


(a) (b)

Figure 5.7: (a) Influence spread of 50 seed nodes for different algorithms using LT model on Celegans multiplex GPI network. (b) Influence spread of 50 seed nodes for different algorithms using LT model on Homo multiplex GPI network for influence maximization.

5.4.4 Execution time on Real networks

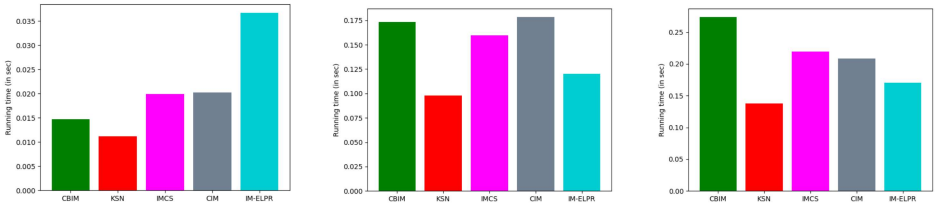
This section discusses two scenarios; the first is to find seed nodes using different algorithms. The second scenario is the performance of influence maximization using the IC model. In most cases, CBIM is more efficient than other algorithms for finding seed nodes. But, some times CBIM takes more time than other algorithms to spread information as it influences more nodes



(a) (b)

Figure 5.8: (a) Influence spread of 50 seed nodes for different algorithms using LT model on Moscow Athletics 2013. (b) Influence spread of 50 seed nodes for different algorithms using LT model on Newyork climate change 2014 for influence maximization.

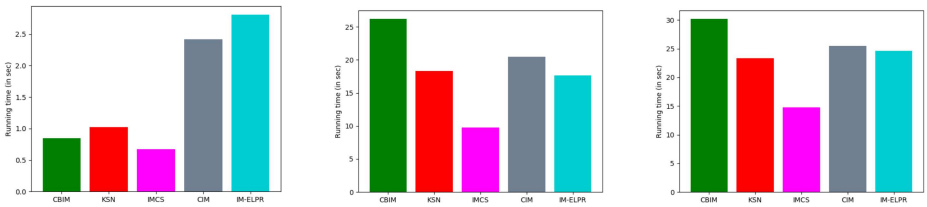
relatively. Figure 5.9ashows the execution time of identifying seed nodes using the Celegans multiplex GPI network. IM-EPLR takes more time than other algorithms, and KSN is efficient than other algorithms. IMCS and CIM take almost the same time. CBIM is efficient than IMCS, CIM, and IM-EPLR. Figure 5.9b shows the execution time of Influence maximization using the IC model on the Celegans multiplex GPI network. CBIM and CIM have taken almost the same time, but they take more time than KSN, IM-EPLR, and IMCS. Figure 5.9cshows the execution time of influence maximization using the LT model on the Celegans multiplex GPI network. CBIM takes more time, and KSN is efficient than other algorithms. CIM takes more time than IM-EPLR and KSN.



(a) (b) (c)

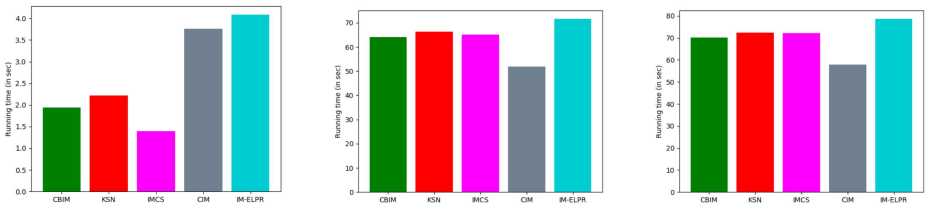
Figure 5.9: (a) Execution time of different algorithms for finding seed nodes under Celegans multiplex GPI network. (b) Execution time of different algorithms for spreading of 50 seed nodes using IC model in Celegans multiplex GPI network. (c) Execution time of different algorithms for spreading of 50 seed nodes using LT model in Celegans multiplex GPI network.

Figure 5.10a shows the execution time of finding seed nodes by algorithms on the Homo multiplex GPI network. IMCS is efficient than other algorithms. CBIM takes more time than IMCS, but it is efficient than IM-EPLR, CIM, and KSN. Figure 5.10bpresents the execution time of Influence maximization using the IC model on the Homo multiplex GPI network. IMCS is



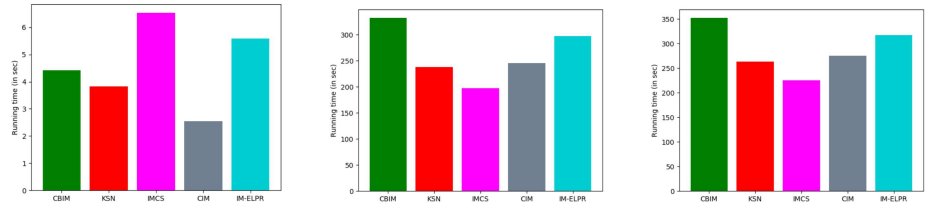
(a) (b) (c)

Figure 5.10: (a) Execution time of different algorithms for finding seed nodes under Homo multiplex GPI network. (b) Execution time of different algorithms for spreading of 50 seed nodes using IC model in Homo multiplex GPI network. (c) Execution time of different algorithms for spreading of 50 seed nodes using LT model in Homo multiplex GPI network.



(a) (b) (c)

Figure 5.11: (a) Execution time of different algorithms for finding seed nodes under Moscow Athletics 2013. (b) Execution time of different algorithms for spreading of 50 seed nodes using IC model in Moscow Athletics 2013. (c) Execution time of different algorithms for spreading of 50 seed nodes using LT model in Moscow Athletics 2013.



(a) (b) (c)

Figure 5.12: (a) Execution time of different algorithms for finding seed nodes under Newyork climate change 2014. (b) Execution time of different algorithms for spreading of 50 seed nodes using IC model in Newyork climate change 2014. (c) Execution time of different algorithms for spreading of 50 seed nodes using LT model in Newyork climate change 2014.

efficient than other algorithms and CBIM takes more time than all other algorithms. KSN is efficient than CIM, CBIM, and more than IMCS, IM-EPLR. Figure 5.10c shows the execution time of Influence maximization using the LT model on the Homo multiplex GPI network. IMCS

is efficient than other algorithms, and CBIM takes more time than all other algorithms. CIM and IM-EPLR take almost the same time.

Figure 5.11a shows the execution time of finding seed nodes under-discussed algorithms on Moscow Athletics 2013. IM-EPLR takes more time than other algorithms, and IMCS is efficient than any other algorithms. Figure 5.11b shows the execution time of Influence maximization using the IC model on Moscow Athletics 2013, CIM takes less time, CBIM and IMCS takes similar time. Figure 5.11c shows the execution time of Influence maximization using the LT model on the Moscow Athletics 2013 dataset. IM-ELPR takes more time than any other algorithm; KSN, CBIM, and IMCS take almost the same time for influence maximization. CIM is efficient than any other algorithm.

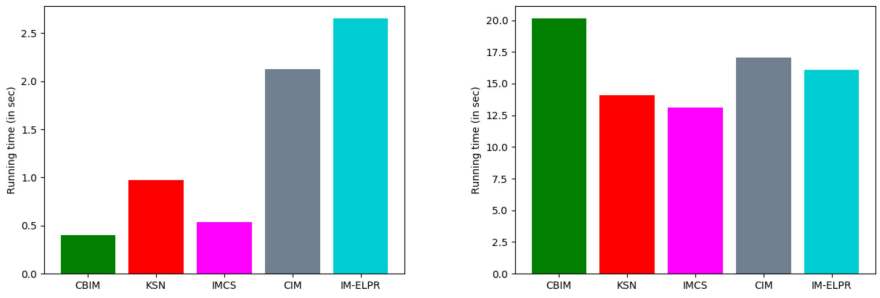
Figure 5.12a shows the execution time for finding seed nodes of discussed algorithms on the Newyork climate change 2014 dataset. IMCS is efficient than any other algorithm. Figure 5.12b shows the execution time of influence maximization using the IC model on the Newyork climate change 2014 dataset. KSN is efficient than any other algorithms. CBIM and IMCS take almost a similar time. Figure 5.12c shows the execution time of influence maximization using the LT model on the Newyork climate change 2014 dataset, IMCS is efficient, and CBIM takes more time, algorithms have taken similar time under the IC model and LT model.

Figure 5.13a shows the execution time for finding seed nodes of discussed algorithms on Arxiv net science multiplex, CBIM is efficient, and IM-EPLR takes more time than all other algorithms. KSN takes more time than CBIM and IMCS, is efficient than IM-EPLR and CIM. Figure 5.13b shows the execution time of influence maximization under the IC model on Arxiv net science multiplex, CBIM takes more time, and IMCS takes less time. Figure 5.14a shows the execution time for finding seed nodes of discussed algorithms on the London Multiplex Transport Network, IM-EPLR takes more time than all other algorithms, CBIM and KSN take almost the same time. Figure 5.14b shows the execution time of influence maximization using the IC model on London Multiplex Transport Network, CBIM takes more time, KSN and CIM take almost the same time and efficient than all other algorithms.

5.4.5 Comparision with CIM and SIM

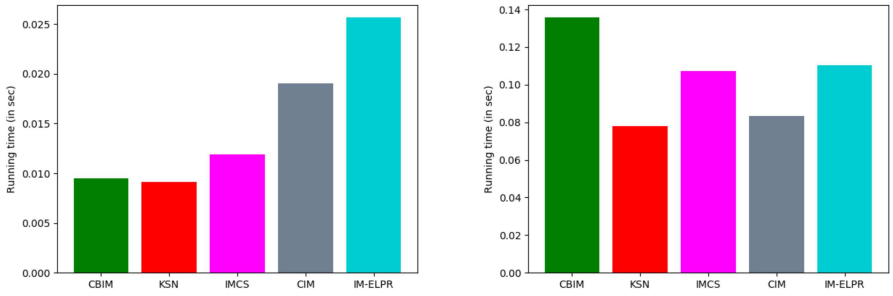
This section compares and analyzes the results of CBIM, SIM, and CIM. Results are in two scenarios, i.e., the influence spread and execution time for finding seed nodes on real-time datasets, which are listed in Section 3.2. Figure 5.15a shows the influence spread on the Sanremo music festival 2016 dataset. CBIM has performed better than SIM and CIM in influence spread. Figure 5.15b presents the running time to find seed nodes on the Sanremo music festival 2016 dataset. SIM takes more time than CBIM and CIM.

Figure 5.16a shows the influence spread on Moscow Athletics 2013 dataset. CBIM has



(a) (b)

Figure 5.13: (a) Execution time of different algorithms for finding seed nodes under Arxiv netscience multiplex. (b) Execution time of different algorithms for spreading of 50 seed nodes using IC model in Arxiv netscience multiplex.

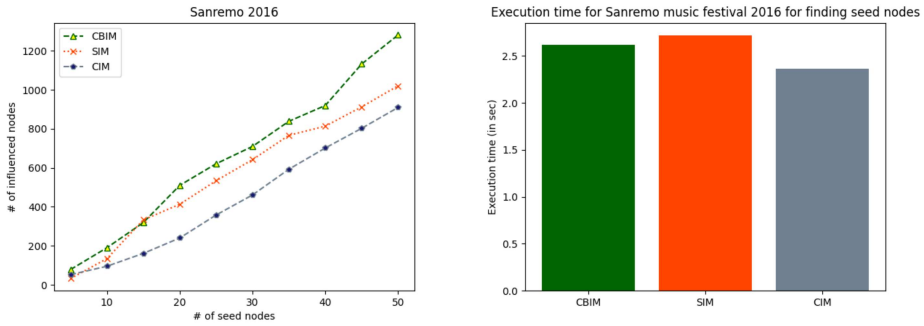


(a) (b)

Figure 5.14: (a) Execution time of different algorithms for finding seed nodes under London Multiplex Transport Network. (b) Execution time of different algorithms for spreading of 50 seed nodes using IC model in London Multiplex Transport Network.

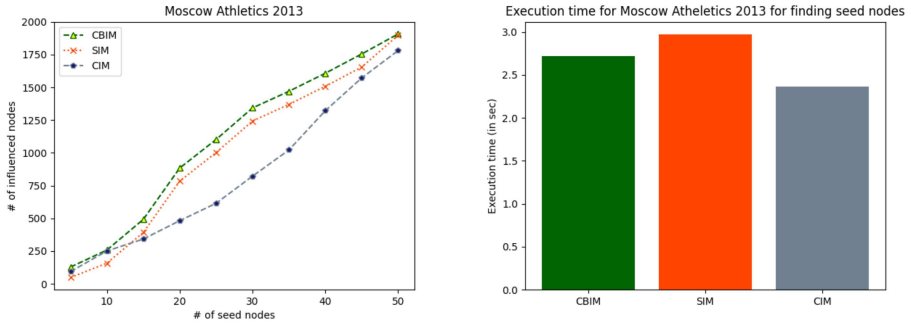
performed better than SIM and CIM in influence spread. Figure 5.16b presents the running time to find seed nodes on the Moscow athletics 2013 dataset. SIM takes more time than CBIM and CIM. Figure 5.17a shows the influence spread on the Newyork climate change 2014 dataset. CBIM has performed better than SIM and CIM in influence spread. Figure 5.17b presents the running time to find seed nodes on the Newyork climate change 2014 dataset. SIM takes more time than CBIM and CIM.

Figure 5.18a shows the influence spread on Martin Luther King's 50th anniversary 2013 dataset. CBIM has performed better than SIM and CIM in influence spread. Figure 5.18b presents the running time to find seed nodes on Martin Luther King's 50th anniversary 2013 dataset. SIM takes more time than CBIM and CIM. Figure 5.19a shows the influence spread on the Cannes film festival 2013 dataset. CBIM has performed better than SIM and CIM in influence spread. Figure 5.19b presents the running time to find seed nodes on the Cannes film



(a) (b)

Figure 5.15: (a) Number of influenced people for 50 seed users for various algorithms using IC model on Sanremo music festival 2016 dataset. (b) Execution time for various algorithms to find 50 seed users on Sanremo music festival 2016 dataset.



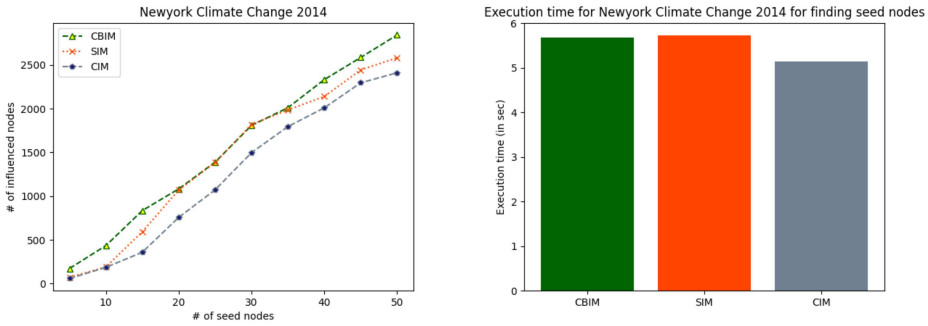
(a) (b)

Figure 5.16: (a) Number of influenced people for 50 seed users for various algorithms using IC model on Moscow Athletics 2013 dataset. (b) Execution time for various algorithms to find 50 seed users on Moscow athletics 2013 dataset.

festival 2013 dataset. SIM takes more time than CBIM and CIM.

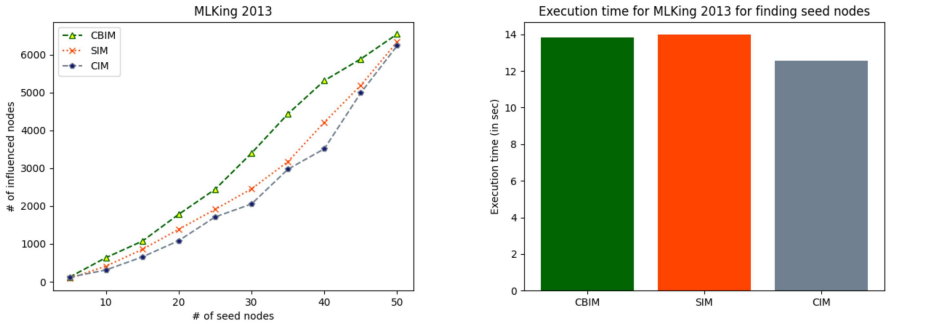
5.5 Summary

This chapter proposes CBIM, an efficient community-based influence maximization algorithm using IC and LT models in multilayer networks. CBIM selects seed nodes that influence more than the competing algorithms for the same number of nodes. The edge density is high in communities generated by CBIM. Due to this, individuals are likely to have frequent interactions and influence each other. Therefore, CBIM is more effective than traditional algorithms. To the best of our knowledge, this is the first attempt to consider both degree and distance for selecting



(a) (b)

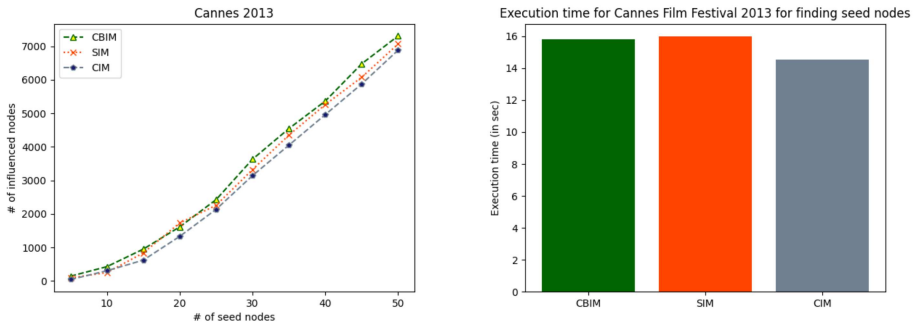
Figure 5.17: (a) Number of influenced people for 50 seed users for various algorithms using IC model on Newyork climate change 2014 dataset. (b) Execution time for various algorithms to find 50 seed nodes on newyork climate change 2014 dataset.



(a) (b)

Figure 5.18: (a) Number of influenced people for 50 seed users for various algorithms using IC model on Martin Luther King's 50th anniversary 2013 dataset. (b) Execution time for various algorithms to find 50 seed users on Martin Luther King's 50th anniversary 2013 dataset.

seed nodes from the communities in multilayer networks. The influence spread of CBIM is higher than traditional algorithms.



(a) (b)

Figure 5.19: (a) Number of influenced people for 50 seed users for various algorithms using IC model on Cannes film festival 2013 dataset. (b) Execution time for various algorithms to find 50 seed users on Cannes film festival 2013 dataset.