Chapter 3

CIM: Clique-based heuristic for finding influential nodes in multilayer networks

In this chapter[1], we propose a novel algorithm, clique-based influence maximization (CIM), to find the most influential nodes in a multilayer network. CIM comprises two stages. CIM finds all the maximal cliques in a multilayer network in the first stage. In the second stage, it finds seed nodes from the cliques. CIM ignores noted nodes in the network to increase efficiency and reduce redundancy. CIM generates better results for influence spread in multilayer networks than state-of-the-art algorithms.

The contributions of the proposed work are:

- We adapted the maximal clique selection algorithm [71] and extended it to multilayer networks.
- We propose a seed selection algorithm CIM by considering maximal clique structure for percolated and non-percolated cliques.
- We propose to ignore the *noted nodes* for influence maximization to reduce the redundancy and the spreading time without compromising the spread.
- We compare the proposed algorithm's performance against the state-of-the-art algorithms under the IC diffusion model on real-world social networks.

[1] K, Venkatakrishna Rao., Katukuri, M. and Jagarapu, M. CIM: clique-based heuristic for finding influential nodes in multilayer networks. Appl Intell 52, 5173–5184 (2022). https://doi.org/10.1007/s10489-021-02656-0

Table 3.1: Notations used in CIM.

Notation	Description
V, E	Nodes, edges in a graph
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
dd_v^l	Degree discount 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
u^l	Node u in layer l

3.1 Model of multilayer networks

A multilayer network model is intra and inter connected graph network $M = (V^M, E^M, V, L)$. Where $V^M \subseteq V \times L^1 \times L^2 \times .. \times L^m$ is the set of node layer combinations, $E^M \subseteq V^M \times V^M$ is the edge set in multilayer network containing the possible pair of combinations and elementary layers, V is set of nodes in multilayer network and $L = \{L^\alpha\}_{\alpha=1}^m, \alpha \geq 2$. Nodes need not be homogeneous and a node u can exist in many layers [70]. A node layer tuple u^l indicates a node u in layer l, $V^i \in V^M = [v_1^i, v_2^i, ...v_n^i]$ and $V = [V^1, V^2 ... V^m]$, where $\mid V \mid = m * n$. Different notations have been used in this chapter, Table 1 presents the notations used in this chapter.

3.2 CIM: Clique-based heuristic for finding influential nodes in multilayer networks

3.2.1 Identification of all maximal cliques

Moser et al. [53] proved that a graph with n vertices could have at most $3^{n/3}$ maximal cliques. Hence, a multilayer network with m layers and each layer with n nodes can have at most $3^{(m \times n)/3}$ maximal cliques. Any algorithm that finds all the maximal cliques of a multilayer network can explore $3^{(m \times n)/3}$ maximal cliques out of the network. Kerbosch et al. [5] propose an algorithm to find all the existing maximal cliques in a single layer network. The algorithm works

recursively for the generation of maximal cliques in the graph. It performs recursion for all the cliques, whether a maximal clique or not; due to which, the algorithm's time complexity is slightly higher. Tomita et al. [71] introduce a variant for the above algorithm, which finds all the existing maximal cliques. The algorithm backtracks the graph's branch search if it has no maximal clique so that time is reduced. Both algorithms generate all the existing maximal cliques. They use depth-first search (DFS) to generate all the maximal cliques in the given graph.

We adapt the algorithm proposed by [71] for a single layer network to multilayer networks to identify all the maximal cliques. This algorithm generates all the maximal cliques of a given multilayer network graph. The algorithm begins with an empty set and recursively expands the set in stages until all the maximal cliques are identified.

Algorithm 1 CIM: Clique-based Influence Maximization Algorithm

Initialize $Min, Upto$
/* $NOMC$- number of generated maximal cliques; K- number of Required Seed nodes, SOC- Size of Clique*/
$Min = K/NOMC$; $Upto$= K mod $NOMC$
Case 1: If $NOMC \geq K$ **then**:
    Sort all the cliques in descending order based on size
    Select highest degree node from each clique, Starting from largest size of the clique.
    This set of K nodes selected from first K cliques are seed set.
Case 2: If $NOMC \leq K$ **then**:
    Step 1: Select $Min(= K/NOMC)$ number of highest degree nodes from each clique
    Step 2: Select $Min + 1$ from the first K mod $NOMC$ cliques.
Case 3: If *any clique size* $\leq Min$ **then**:
    Select $Size\ of\ Clique\ (SOC)/2$ highest degree nodes from the clique.
    Remaining seed nodes will be selected based on satisfaction of case 1 or case 2.
Case 4: In percolated cliques,
    If the same node with highest degree is in different cliques then select that node from largest clique.
    Remaining seed nodes will be selected based on satisfaction of case 1 or case 2.

3.2.2 Identification of seed users

This section presents the identification of seed nodes after finding all the existing maximal cliques in a multilayer network. CIM selects seed nodes from the cliques based on the number of generated maximal cliques (NOMC). Since a node in a clique is connected to all the other nodes, its influence is more. The following cases explain the selection of seed nodes from the

generated maximal cliques in a multilayer network. Algorithm 1 presents the procedure of seed nodes selection.

$$Min = K/NOMC$$
$$Upto = K \mod NOMC$$

Case 1: If generated $NOMC$ are more than or equal to the required number of seed nodes, sort the cliques in descending order based on their size. Select the highest degree node (yet to be selected) from each clique, starting from the largest clique. This set of K nodes selected from the first k cliques will be the seed set.

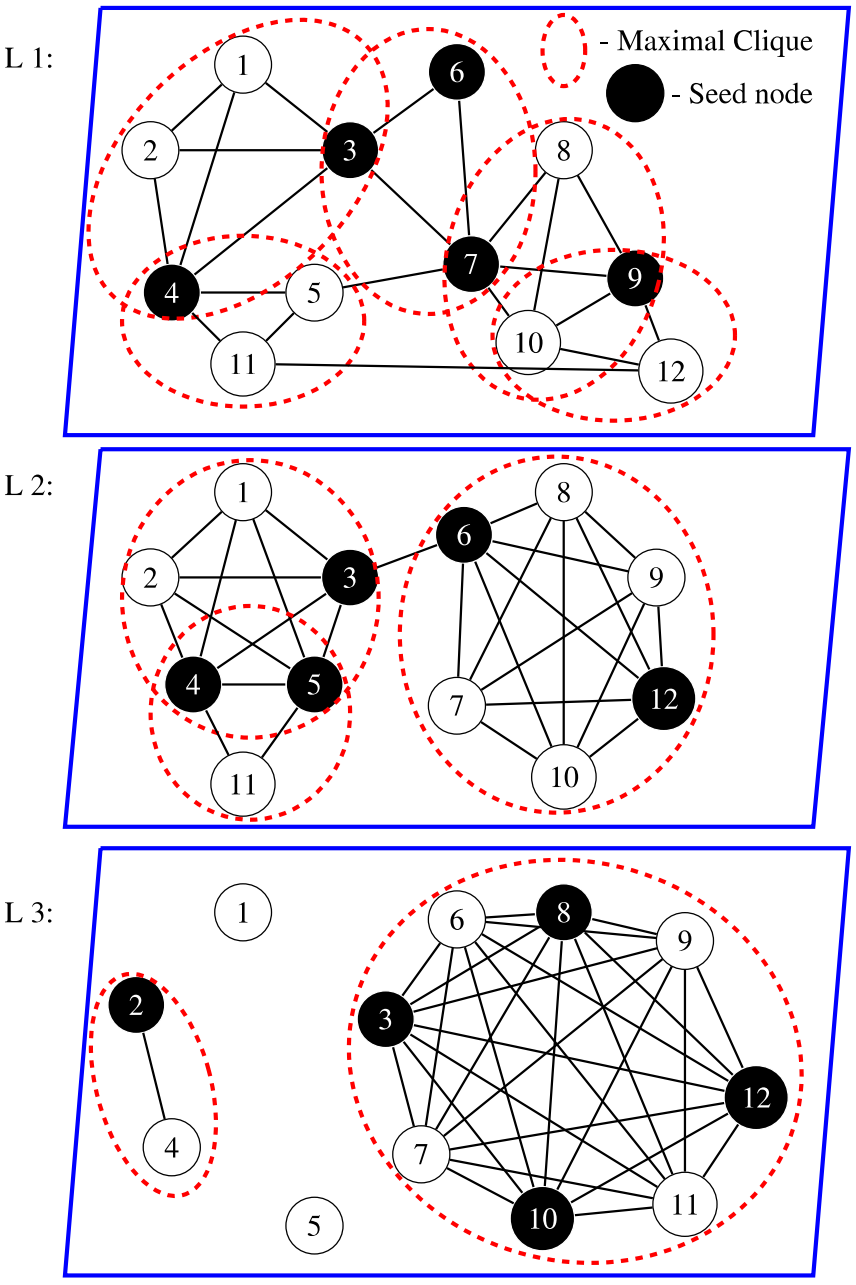
Case 2: If $NOMC$ are less than the required number of seed nodes K, then seed nodes will be selected in two steps. In the first step, select the Min (=$K/NOMC$) number of highest degree nodes (yet to be selected) from each clique. In the second step, select the nodes (i.e., Min+1 node, yet to be selected) from the first K *mod* $NOMC$ cliques.

Case 3: While selecting the seed nodes from each clique, If the size of any clique is less than or equal to Min, then select $Size\ of\ the\ clique(SOC)/2$ highest degree nodes (yet to be selected) from the clique, and remaining seed nodes will be selected based on case 1 or case 2.

Case 4: While finding all the maximal cliques, there may be percolated cliques. In percolated cliques, the same node with the highest degree may be present in different cliques. CIM selects the highest degree node from the larger clique, and the remaining seed nodes will be selected based on the satisfaction of case 1 or case 2.

Figure 3.1 Illustrates the selection of seed nodes in a multilayer network. We took the number of seed nodes as five. The network consists of three layers; layer one generates C1, C2, C3, C4, and C5 cliques, satisfying case 1. So node 3 from C1, node 7 from C2, node 6 from C3, node 4 from C4, and node 9 from C5 are selected as seed nodes. Layer two generates C6, C7, and C8 cliques, and it satisfies case 2, so the seed node selection is made in two steps. In the first step, select the highest degree node (i.e., Min=1) from each clique, i.e., 6 from C6, 4 from C7, and 5 from C8. In the second step, select the next highest degree node starting from the largest clique, i.e., 12 from C6 and 3 from C7. Layer three generates C9 and C10, the size of C10 is two and Min=2, so select $SOC/2$ highest degree nodes from C10, i.e., 2 or 4 from C10, and the remaining four seed nodes select from C9, i.e., 3, 8, 10 and 12.

Let the complexity of identifying all the cliques in a graph of n vertices be $O(Y(n))$. The complexity of finding the size of each clique and sorting the cliques is $O(rNOMC + NOMClogNOMC)$, where r is the size of the largest clique. The complexity of selecting K seed nodes will be $O(K)$ (assuming that the degrees of nodes are available and sorted within the clique).



$L1 = \{C1 = \{1,2,3,4\}, C2 = \{7,8,9,10\}, C3 = \{3,6,7\}, C4 = \{4,5,11\}, C5 = \{9,10,12\}$
$L2 = \{C6 = \{6,7,8,9,10,12\}, C7 = \{1,2,3,4,5\}, C8 = \{4,5,11\}$
$L3 = \{C9 = \{3,6,7,8,9,10,11,12\}, C10 = \{2,4\}$

Figure 3.1: Illustration of identifying five seed nodes from each layer after finding maximal cliques.

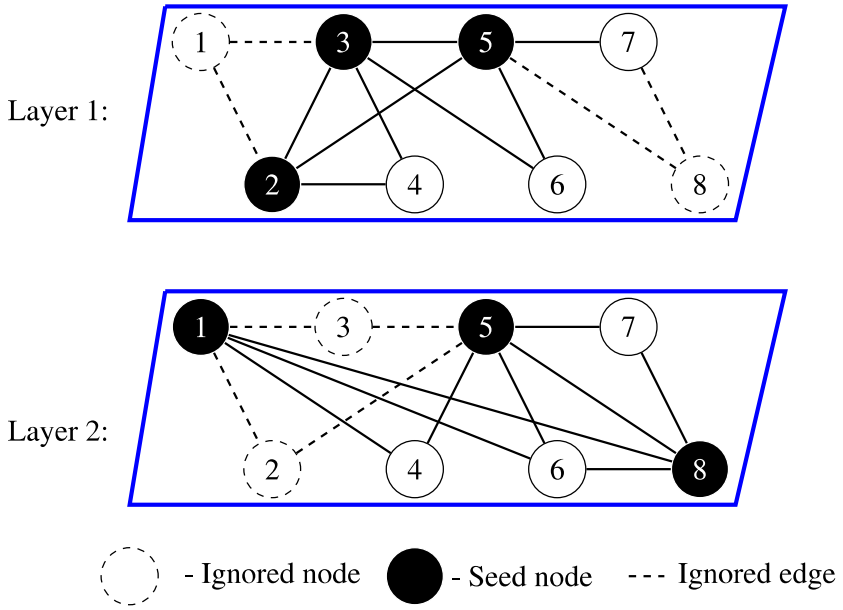


Figure 3.2: Illustration of ignoring noted nodes for influence maximization.

3.3 Ignoring noted nodes for influence maximization

If a node is a seed node in one layer, then that node will be ignored for influence maximization in all the other layers where it is a non-seed node. We propose reducing the spreading time over the network and removing redundancy without compromising on the spread. Nodes ignored in other layers are termed as *noted nodes*. The idea is, if a node is activated in one layer, then it need not be reactivated in other layers since it is already influenced and information is available to that node. From this assumption, only non-seed nodes of a layer will be ignored for influence maximization, so ignoring these nodes will not affect much of the influence maximization. The number of newly influenced nodes after ignoring the nodes remains unchanged compared to the number of nodes influenced before ignoring them. The time complexity of influence maximization under the IC model after ignoring *noted nodes* is less comparatively. In Figure 3.2, nodes 2, 3, 5 are seed nodes in layer 1, and nodes 1, 5, and 8 are seed nodes in layer 2. Nodes 2, 3 and their links are ignored in the second layer since they are already seed nodes in layer one and non-seed nodes in layer 2. Similarly, nodes 1, 8 and their links are ignored for influence maximization in the first layer because they are seed nodes in the second and non-seed nodes in the first layer.

3.4 Experimental setup

To evaluate the performance, pymnet library [41] is used for generating synthetic multilayer networks. All the experiments were executed on a system with the following configuration, Intel Xeon(R) workstation with 2.20Ghz CPU, 32GB RAM, and Ubuntu 16.04.

3.4.1 Dataset description

To compare the proposed algorithm's performance with the State-of-the-art algorithms, we use several real-time datasets and synthetic network:

- Celegans multiplex GPI network [68], where multiplex consists of layers corresponding to different synaptic junctions.
- Drosophila multiplex GPI network [27], it consists of different types of genetic interaction for organisms in the general biological repository.
- Homo multiplex GPI network [24], it disseminates genetic and protein interaction data from humans and model organisms.
- ARXIV Net Science Multiplex [26] dataset consists of different Arxiv categories as layers in multiplex generated from the keyword "network" till 2014 journals.
- HOMO MULTIPLEX GPI NETWORK [26] is to disseminate genetic and protein interaction data from humans and model organisms.
- A synthetic four-layer network $G = \{L^1, L^2, L^3, L^4)$ is generated for the evaluation using the PYMNET library[41]. In the synthetic network, each layer is generated with a node count of 150, i.e., $\mid V^i \mid$=150. All the four layers together constitute 600 nodes, i.e.$\mid V \mid$=600. Edges are placed randomly between the nodes, irrespective of the layer. The edge generation probability parameter is fixed as 0.5 so that half of the possible edges will be present in the network.
- Sanremo 2016_final[23] is the final of Sanremo music festival 2016. The data has been generated in three layers, i.e., retweets, mentions, and replies between 2016-02-13 to 2016-02-14
- Moscow Athletics 2013 [60] consists of data collected from Twitter during the 2013 World Championship in Athletics. The multiplex network uses three layers corresponding to retweets, mentions, and replies between 2013- 08-05 and 2013-08-19.

- Newyork climate change 2014 [60] consists of data from Twitter during the 2014 people's climate march in New York. Data is based on retweets, mentions, and replies from 2014-09-19 to 2014-09-22.
- MLKing2013 [60] generated on Martin Luther King's 50th-anniversary speech "I have a dream.." in 2013. It consists of three layers corresponding to retweets, mentions, and replies observed between 2013-08-25 to 2013-09-02.
- Cannes2013 [60] focused on the Cannes film festival 2013. It consists of 3 layers corresponding to retweets, mentions, and replies observed between 2013-05-06 to 2013-06-03.
- London Multiplex Transport [26] consists of data from the London rail network, nodes are train stations, and routes between train stations are edges. Train stations are connected overground, docklands light railway (DLR), and underground.

Table 3.2: Properties of Datasets

Dataset	# of Nodes	# of Edges	# of Layers
Directed graph			
Celegans multiplex GPI network	3879	8181	6
Drosophila multiplex GPI network	8215	43366	7
Homo multiplex GPI network	18222	170899	7
Moscow Athletics 2013	88804	210250	3
Newyork climate change 2014	102439	353495	3
Undirected graph			
London Multiplex Transport Network	369	441	3
EU-Air Transportation multiplex	450	3588	37
Arxiv netscience multiplex	14489	59026	13
Sanremo 2016_final	56562	461838	3
MLKing 2013	327707	396671	3
Cannes 2013	438537	991854	3
Synthetic networks			
Pymnet synthetic network	600	179700	4

3.4.2 Baseline models

- **ShaPley value-based Influential Nodes (SPINs) [56]** : In SPIN, the shapely value concept in the cooperative game theory problem is used for finding the top k nodes. The

SPIN approach consists of two steps, computing a ranked list of nodes based on shapely value generation and choosing the top k nodes from the ranked list.

- **KSN [42]**: Knapsack seeding of networks (KSN) is an approximation algorithm which parallelizes the problem in terms of the components of the layer. The difficulty lies in combining the solutions to the influence maximization problem on the separate layers to find the multiplex influence maximization (MIM). KSN achieves this by approximating the solution to the multiple-choice knapsack problem.

- **DISCO [45]**: Deep Learning-Based Influence Maximization(DISCO) is a learning-based framework. It extracts the nodes' features and trains the data using Deep Q Network (DQN) to identify seed nodes. It predicts the marginal influence maximization and selects the seed nodes.

- **DD**: The greedy heuristic is computationally expensive, so it may not be suitable for large and complex networks. Degree-based heuristics are performing well. The basic idea behind the degree discount heuristic is that the nodes with a maximum degree as seed nodes have more influence than the other nodes. However, if the two highest-degree nodes are neighbors, they will not result in influence maximization. In such a case, the degree discount heuristic discounts/reduces the degree of a node whenever it has a connection with a node, which falls under another seed node's influence. After discounting all the nodes' degrees, the seed set will be computed based on the higher-degree nodes.

3.5 Results

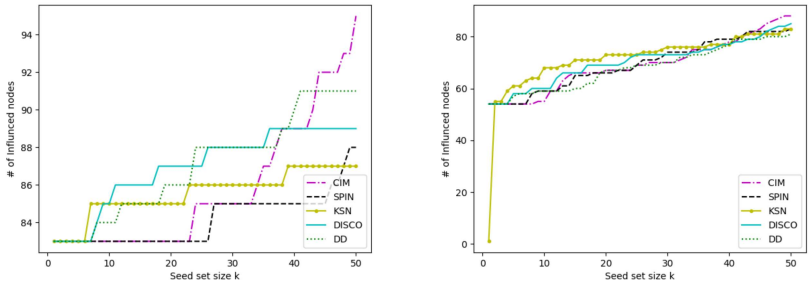
This section compares and analyzes the results with the datasets listed in Section 3.2. As the proposed algorithm is clique based influence maximization, it is more effective than other algorithms. Any node in a clique is connected to all other nodes of the clique, and due to this property, the influence maximization of our algorithm is better than other algorithms. We compare the efficiency of the algorithms on both synthetic and real time networks. We have discussed two parameters for analysis and comparison of the algorithms. The first parameter is to find the number of influenced nodes before and after ignoring *noted nodes*. The second parameter is the execution time to find the seed nodes, to find influenced nodes using the IC model, and to find influenced nodes after ignoring the *noted nodes*.

3.5.1 Synthetic networks

We consider synthesized multiplexes generated from PYMNET Model [41] with 600 nodes, and 1,79,700 edges, i.e., half of the maximum possible number of edges in 4 layers. We assigned a synthetic multiplex network to the IC diffusion model with edge weights and thresholds chosen randomly in the range [0,1].

3.5.1.1 Algorithm performance on synthesized networks

We have selected 50 seed nodes from the network and assigned them to the IC model for influence maximization. In Figure 3.3a, CIM performed better than other algorithms. After CIM, DD reported better results. DISCO has performed better than KSN and SPIN. SPIN influence count is less than all other algorithms. In Figure 3.3b, the CIM influence count is more than other algorithms in the synthetic dataset after ignoring *noted nodes*, but the influence counts DISCO, SPIN, and DD is almost the same.

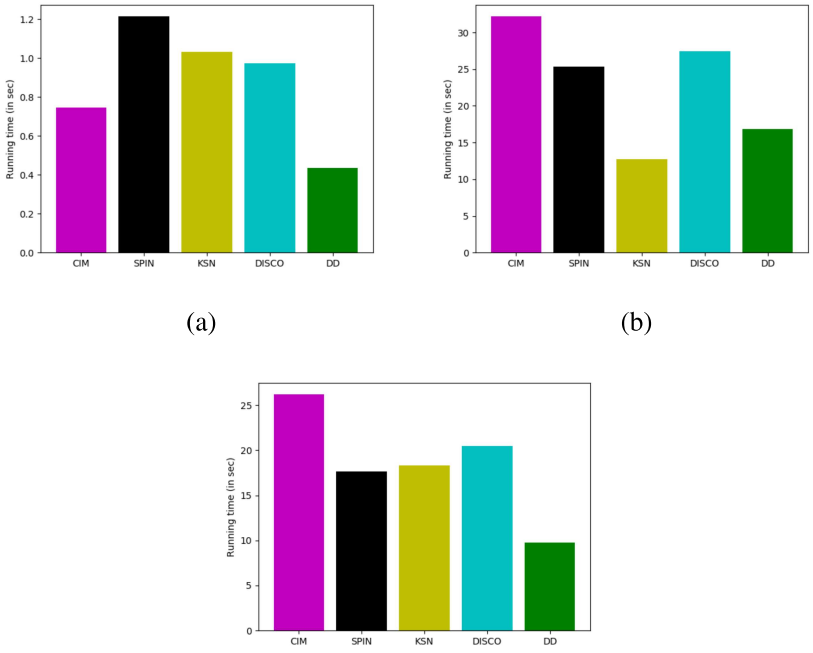


(a) (b)

Figure 3.3: (a) Influence spread of 50 seed nodes for different algorithms under IC model on Synthetic Network before ignoring noted nodes (b) Influence spread of 50 seed nodes for different Algorithms under IC model on Synthetic Network after ignoring the noted nodes.

3.5.1.2 Execution time on synthesized networks

Figure 3.4a gives the execution time to identify seed nodes of various algorithms. SPIN takes more time, and DD takes less time among all the algorithms. CIM takes less time than DISCO, KSN, and SPIN and takes more time than DD. Figure 3.4b gives the execution time of influence maximization under the IC model. CIM takes more time for influence maximization than any other algorithm since it gives more influence count, and KSN takes less time than all other algorithms. Figure 3.4c gives the running time of influence maximization under the IC model after ignoring *noted nodes*. In this, DD takes less time than any other algorithm. DISCO



(a) (b)

(c)

Figure 3.4: (a) Running time of different algorithms for finding 50 seed nodes in Synthetic Network. (b) Running time of IC Model for influence maximization on Synthetic Network. (c) Running time of IC Model on Synthetic Network after ignoring noted nodes.

takes more time than DD, SPIN, and KSN. The time complexity of all algorithms for influence maximization after ignoring *noted nodes* is less than the complexity before ignoring *noted nodes*.

3.5.2 Real networks

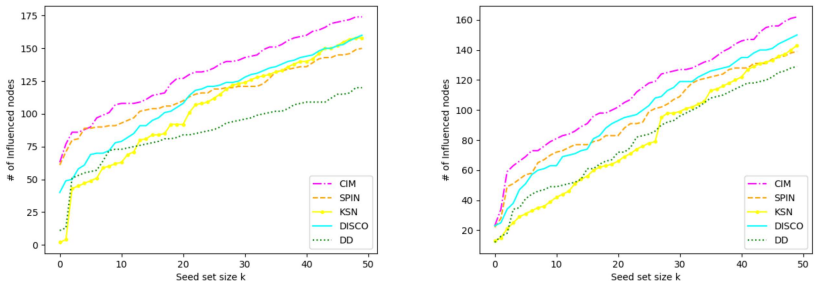
3.5.2.1 Performance of Algorithm on real networks

Here, we present influence maximization on three real-time datasets. Figure 3.5a shows influence maximization on Celegans multiplex GPI network dataset, and Figure 3.5b shows influence maximization on Celegans multiplex GPI network dataset after ignoring *noted nodes* in Celegans multiplex GPI network dataset. CIM produced the best results among all the algorithms under different test conditions. DISCO has influenced more nodes than KSN, SPIN, and DD. Some seed nodes can influence an exponential number of nodes based on the network. Due to this, some curves are suddenly rising in the results.

Figure 3.6a presents influence maximization on Drosophila Multiplex GPI Network dataset

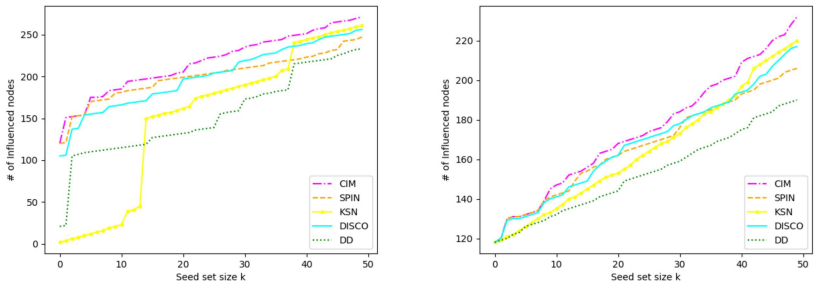
and Figure 3.6b presents influence maximization on Drosophila Multiplex GPI Network dataset after ignoring *noted nodes* in Drosophila Multiplex GPI Network dataset. CIM performed better than other algorithms in both cases. KSN and DISCO have influenced the same number of nodes.

Figure 3.7a shows influence maximization on Homo multiplex GPI network and Figure 3.7b shows influence maximization on Homo multiplex GPI network after ignoring the *noted nodes* in Homo multiplex GPI network. DISCO performance is marginally more than KSN and DD, DISCO and SPIN influenced same number of nodes. In some cases, influenced nodes before ignoring *noted nodes* is marginally more than the influenced nodes after ignoring the *noted nodes* because *noted nodes* do not come under influence count. In three real-time datasets, results are similar, and in synthetic dataset, results are different.



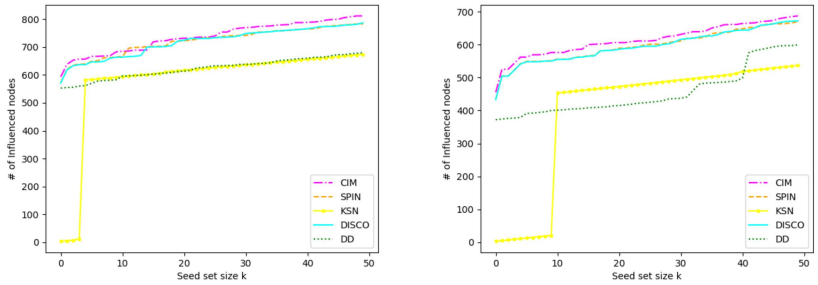
(a) (b)

Figure 3.5: (a) Influence spread of 50 seed nodes for different algorithms under IC model on Celegans multiplex GPI network. (b) Influence spread of 50 seed nodes for different algorithms under IC model on Celegans multiplex GPI network after ignoring noted nodes.



(a) (b)

Figure 3.6: (a) Influence spread of 50 seed nodes for different algorithms under IC model on Drosophila multiplex GPI network for influence maximization. (b) Influence spread of 50 seed nodes for different algorithms under IC model on Drosophila multiplex GPI network after ignoring *noted nodes* for influence maximization.



(a) (b)

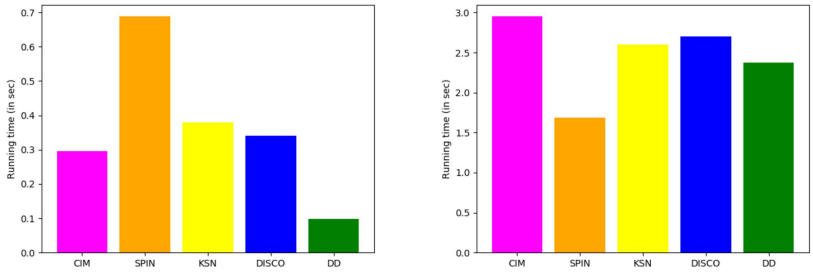
Figure 3.7: (a) Influence spread of 50 seed nodes for different algorithms under IC model on Homo multiplex GPI network for influence maximization. (b) Influence spread of 50 seed nodes for different algorithms under IC model on Homo multiplex GPI network after ignoring noted nodes for influence maximization.

3.5.2.2 Execution time on real networks

Here, we compare the three different scenarios in mentioned datasets; first scenario is the execution time to find the seed nodes of various algorithms. Second scenario is the execution time for the influence maximization using IC model. The third scenario is the execution time for influence maximization after ignoring *noted nodes*. Figure 3.8 shows the three scenarios on Celegans multiplex GPI network. Figure 3.9 shows the three scenarios on the Drosophila multiplex GPI network dataset, and Figure 3.10 shows the three scenarios on the Homo multiplex GPI network dataset. Comparatively, the results on all the three datasets are similar in each scenario. In the first scenario, SPIN takes more time, and DD takes less time among all the algorithms. CIM takes less time than DISCO, KSN, and SPIN and takes more time than DD. In the second time scenario, among all the algorithms, in real-time datasets, SPIN takes less time, and CIM takes more time for influence maximization. KSN takes more time than SPIN, DD and less time than DISCO. Though CIM takes more time for synthetic datasets, KSN takes less time. DISCO takes more time than SPIN, DD, and KSN. Similarly, in the third scenario, among all the algorithms, in real-time datasets, SPIN takes less time, and CIM takes more time. DISCO takes more time than SPIN, DD, and KSN.

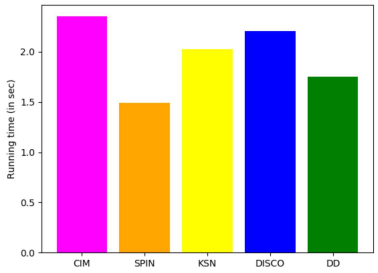
3.6 Summary

We propose CIM, an efficient algorithm for influence maximization in multilayer networks under the independent cascade diffusion model. CIM performs better than other algorithms. CIM considers the graph's connectivity, due to which the probability of influencing a large number of nodes is higher. We came to this conclusion after conducting extensive experiments



(a)

(b)

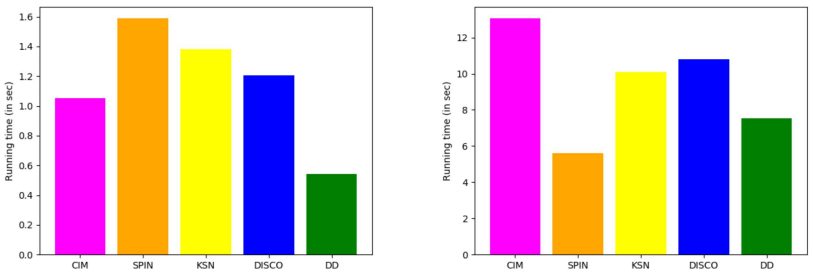


(c)

Figure 3.8: (a) Running time of different algorithms for finding 50 seed nodes in Celegans multiplex GPI network (b) Running time of IC Model on Celegans multiplex GPI network for influence maximization. (c) Running time of IC Model on Celegans multiplex GPI network after ignoring *noted nodes* for influence maximization.

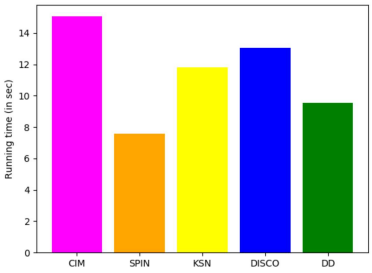
on different datasets and a synthetic network.

Complex networks are a continually focused concept in the realm of network science. Influence maximization emerged as an essential topic in complex networks. A performance evaluation test bed is developed and used to analyze and compare the results of the algorithms.



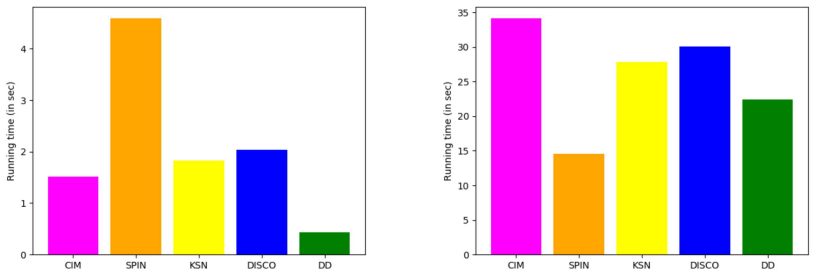
(a)

(b)



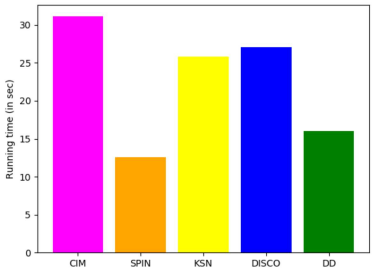
(c)

Figure 3.9: (a) Running time of different algorithms for finding 50 seed nodes in Drosophila multiplex GPI network (b) Running time of IC Model on Drosophila multiplex GPI network for influence maximization. (c) Running time of IC Model on Drosophila multiplex GPI network after ignoring *noted nodes* for influence maximization.



(a)

(b)



(c)

Figure 3.10: (a) Running time of different algorithms for finding 50 seed nodes in Homo multiplex GPI network (b) Running time of IC Model on Homo multiplex GPI network for influence maximization. (c) Running time of IC Model on Homo multiplex GPI network after ignoring *noted nodes* for influence maximization.