DECOUPLING-BASED APPROACH TO CENTRALITY DETECTION IN HETEROGENEOUS MULTILAYER NETWORKS

by

KIRAN MUKUNDA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2021

ACKNOWLEDGEMENTS

I am very grateful for all the guidance provided by Dr. Sharma Chakravarthy during the research work. This work would not have been complete without the constant motivation provided by him. His passion towards research and sincerity in work are an inspiration for life. I would like to thank my thesis committee members Ramez A Elmasri and David Levine for serving in my committee and providing valuable suggestions and feedback.

I would also like to extend my sincere gratitude towards Dr. Abhishek Santra for all the guidance and support throughout the year. I thank all my ITLab team members for providing the feedback and moral support.

I very thankful for my family and friends for their constant support and encouragement throughout my career. Without the guidance and support from all them it would not have been possible to complete this work. Finally, I am thankful to UTA for providing me this opportunity to present my work.

July 28, 2021

ABSTRACT

DECOUPLING-BASED APPROACH TO CENTRALITY DETECTION IN HETEROGENEOUS MULTILAYER NETWORKS

Kiran Mukunda, M.S.

The University of Texas at Arlington, 2021

Supervising Professor: Prof. Sharma Chakravarthy

Graph analysis is one of the techniques widely used for data analysis. It is used extensively on single graphs. Its ability to capture entities and relationships makes it an attractive data model. Search on graphs, such as finding triangles, cliques, shortest paths, etc., and aggregate analysis, such as communities, substructure, or centrality measures have well-defined algorithms for single graphs. The centrality measure, which is the focus of this thesis, identifies the most important nodes in a graph or network. While there are many centrality measures, the most commonly used ones are degree and betweenness centrality. Algorithms for analyzing these measures are numerous for single graphs.

In addition to graphs, multilayer networks (MLNs) are being used to model complex data sets. MLNs consist of several layers, each being a graph. If there are different types of entities in each layer and inter-layer edges are present, then the network is an example of a Heterogeneous Multilayer Network (HeMLN). Due to the lack of algorithms for HeMLNs, they are currently analyzed using aggregation of HeMLN layers including inter-layer edges into a single graph. An alternative projection-based approach is also used to analyze HeMLNs by transforming it into a single graph.

A decoupling-based framework has been proposed to avoid aggregation or projection and still obtain accurate results. This approach analyses the layers independently and composes the partial results to obtain results for a HeMLN. These algorithms have been shown to produce accurate results and are also efficient. Another advantage of this approach is that the layers can be analyzed in parallel. The composition algorithm produces the results of the entire HeMLN. To the best of our knowledge, there are no algorithms that compute centrality measures directly on HeMLNs. This thesis focuses on developing decoupling-based algorithms for degree and betweenness centrality measures for HeMLNs.

The challenge is to minimize the amount of information retained from each layer for use during composition to maximize accuracy. Also, keep the algorithm more efficient than its single graph counterpart, which is considered as the ground truth. This thesis proposes different heuristics and compares the results with the ground truth and naive algorithm. The **proposed heuristics *consistently* improve the accuracy as compared to the naive algorithm while taking less time than the single graph approach**.

Finally, the algorithms proposed in the thesis are tested against both real-world and synthetic data sets with different graph characteristics. This is important to demonstrate the efficacy of heuristics on an arbitrary graph. The results obtained are analyzed in detail to empirically establish the heuristic performance in terms of accuracy, time, and space complexity.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Graph model uses nodes and edges to represent entities and relationships. They are a type of NoSQL database. This representation helps us in leveraging different types of analysis based on relationships that exist in the connected data. For example, the relationship between the individuals as to how they communicate, cities where they live, etc. This representation overcomes some of the limitations of relational databases in capturing relationships without resorting to representations that require joins as in the relational model. This representation is also easier to visualize and understand. In the real-world, graph databases are becoming larger and more complex. This representation also facilitates querying and analysis in multiple ways. Centrality analysis helps understand the nodes critical in information flow under different assumptions. Hence, their computation in graphs, as well as multilayer networks is becoming increasingly important. Analyzing these using traditional methods have become cumbersome and highly inefficient. We will introduce multilayer networks after discussing centrality in a graph or a network.

There are several algorithms for analyzing the single graphs including community detection, substructure identification, centrality, finding cliques, etc. The distribution of edges within a graph is not always equal with a high concentration of edges within a special group of vertices and a low concentration of edges between them. This special group of vertices form a community within a graph [1] it can be used for example to detect a group of people with similar interest in social networks. Prominent and interesting substructures can be identified in a large graph [2] which can later be used for other applications such as data compression, etc. So analyzing graphs for various needs is important and can play an important role in solving problems of the real world.

Centrality nodes are the most influential nodes in any graph. Many people have proposed different interpretations of centrality measures such as the number of connections, bridge nodes, belongings, leader of a community, etc. Freeman (1979) analyzed various published papers and categorized them using communication networks as an example. He found three kinds of centrality measures, first one based on the degree of points and indexes of communication activity, the second one is based on betweenness which tells us about the control of communication within the network, last one is based on the closeness of the node to its community. Other centrality measures, such as Eigenvector, Katz centrality, Percolation centrality, page rank have also been defined. In this thesis we are focusing on degree and betweenness centrality measures.

Degree centrality (DC) tells us about the relative importance of a node within the given network based on the number of edges it has [3]. If the edges had weight, it would give us the relative strength of the node within the network. In the real-world data sets, usually, the degree of nodes follows the power law distribution i.e. a small group of nodes have higher degree compared to others. These usually form the *most influential nodes in a graph also called as hubs.*

Degree centrality is being used across networks to calculate the number of connections a node has within the network. For example, in a social network, the degree of a person indicates the number of connections that the person has in the network. After sorting the actors in descending order based on their degree values, top-k people with the most degrees can be identified. These *top-k people are the most influential within that social network.*

The time complexity of calculating the degree centrality is O(E) where E is total number of edges in the graph. As this is linear, it is widely used in analyzing very large graphs. Computing degree centrality is different for undirected and directed graphs. This thesis **focuses on undirected graphs** for simplicity.

Though degree centrality tends to give us some insights in the graph it is still a **local measure of importance**. For instance, it fails to capture the node that is present in the center of the graph but with very few edges lying on it. So, to identify these types of nodes there is a need to consider other characteristics of the nodes along with the degree.

Betweenness centrality is calculated based on the number of the shortest paths that pass through the node [4]. These nodes act as bridge nodes in the network. For example in a telecommunication network, more information passes through a node which has higher betweenness centrality value. Thus nodes with higher betweenness centrality value have higher control over the given network and it is important to identify these in a given network.

The underlying task for computing the betweenness centrality value is to compute all pair shortest path (APSP). It takes exponential time for computation ($O(V^3)$ where V is the number of vertices in the graph). As the computations are more for APSP it is quite challenging to calculate Betweenness centrality values for larger graphs. Hence, significant research is being done in approximating the betweenness centrality values through various methods using random walks [5], through sampling [6], etc. Unlike the degree centrality, this is a global measure as it is dependent on the entire network.

Using the above centrality measures we can identify the most influential nodes (hubs) in a single graph. In this thesis, we use two ways to identify the hubs:

1. **Top-k nodes** - by sorting the centrality values in *decreasing order of the centrality metric value* and taking the top-k nodes.
2. **Hubs** - any node with *above-average centrality value* is considered as a hub in that network.

All the above-mentioned analysis is for single graphs. However, for multilayer networks for multiple types of nodes, as per our knowledge there is no algorithm for centrality computation.

Multilayer networks provide an effective analysis model. It is efficient and flexible. Multilayer networks consist of multiple single graphs and a set of edges which connect these single graphs. Multilayer networks provide the flexibility of analyzing each layer individually and processing further based on the heuristics. Going forward we will see how the above-mentioned centrality measures can be calculated for a multilayer network.

Based on the type of entities that each layer consists of, multilayer networks can be of three types, they are Homogeneous, Heterogeneous, and Hybrid multilayer networks.

Homogeneous Multilayer Networks (HoMLNs) is where all the layers have the same entities/nodes. It is used to solve problems where there are multiple relationships among the same set of nodes, for example, if we take airline networks, different airline companies have different network for the same set of cities. For example, if one layer represents American airline network, another layer captures the Delta airline network for the same set of cities and so on. Analyzing these two layers would answer questions like which is the best city for an airline to create the next hub? [7–9].

Heterogeneous Multilayer Networks (HeMLNs) have different entities in each layer. For example in the IMDB data set, one layer represents the co-actors another layer represents the co-directors and third layer for movies. These layers are interconnected by inter-layer edges. So HeMLN consists of intra-layer and inter-layer edges with each layer having different entities [10].

Any synthetic graph can also be separated into multiple layers by simply grouping a set of nodes as a layer. This shows that any graph can be analyzed as an HeMLN once they are separated into layers without losing any data. Graph separation is used for efficient analysis of these data sets [11].

Hybrid multilayer networks (HyMLNs) is a combination of both HoMLNs and HeMLNs. For example, in the IMDB data set we have co-actors as a layer, co-directors as another layer now if we want to include another layer where two co-actors are connected if they are friends on Facebook we might want to use a combination of HoMLNs and HeMLNs. In this case, the IMDB co-actors and the Facebook friend layers act as HoMLNs and co-directors act as HeMLN. Any multilayer network which consists of both HeMLN and HoMLN form a Hybrid multilayer network (HyMLN) [12–16].

This **thesis mainly focuses only on HeMLNs**. Figure 1.1 shows an example of HeMLNs where layer one consists of co-actors and layer two consists of nodes representing co-directors and a third layer consisting of movies. Each node within a layer is connected by its intra-layer edges representing the relationship between them. In this example, co-actors are connected if they have acted in the same movie. Directors are connected if they have co-directed a movie together. In movie layer, if the two movies belong to the same genre they are connected by an edge.

There are also inter-layer edges for example if an actor has acted in a movie then an inter-layer edge connects the actor and the movie. If an actor has also directed a movie then there is an inter-layer edge between actor and director layers. As the

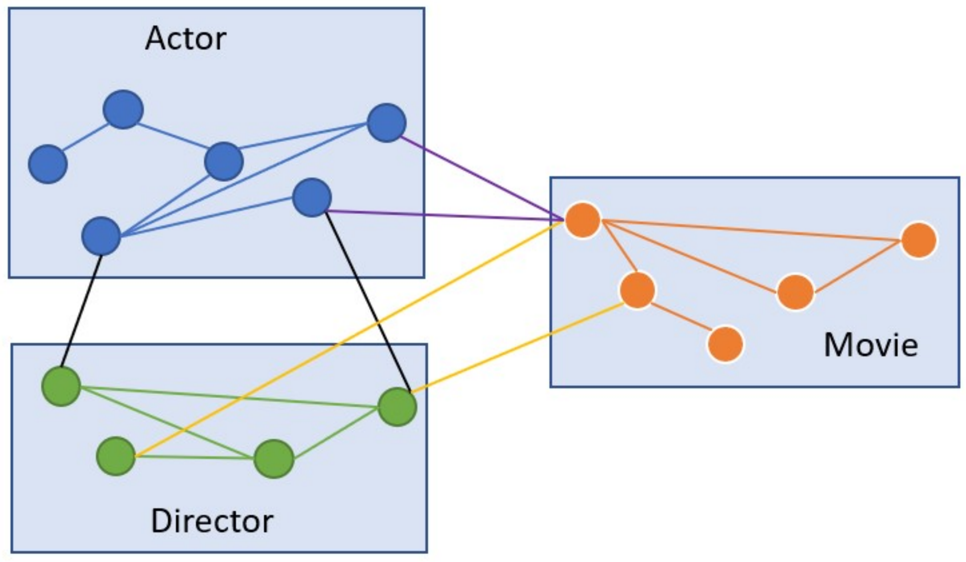


Figure 1.1: An example of Heterogeneous Multilayer Network - IMDB dataset.

entities are different in each layer and have inter-layer edges this IMDB data set belongs to Heterogeneous Multilayer networks.

There are many value-added applications of HeMLNs centrality such as to analyze transportation networks [17], Protein-protein interaction networks [18], disease behavior networks [18], etc. HeMLNs provide opportunities to solve computational challenges and efficiency issues [19].

There has been a lot of research on analyzing these multilayer networks. One way is to aggregate all the layers and compute the centrality measures, while this gets the work done but it is not the most efficient solution in some cases there is data loss as well. Apart from aggregation, there are solutions that use layer projections, network simplification approaches to solve multilayer networks. Attribute graphs are also used widely in these cases to represent additional information in the same single graph.

For HeMLNs one way to analyze centrality measures is to include all the layers into a single graph. All the nodes and the edges of each layer are included. Also, the inter-layer edges are retained to form a single large graph. Then single graph centrality measure algorithms are run on this large graph to get the most influential nodes. While this is a solution but it does not take advantage of the individual layers nor it is the most efficient solution.

While there are no well-defined framework to analyze multilayer networks, in this thesis we explore decoupling-based approach. It is based on a partition-based paradigm. Decoupling-based frameworks have been used in the past research activities to analyze the Multilayer networks and it is proven to be effective. Like in the cases of analyzing substructures in MLNs using decoupling approach [20] or detecting communities in MLNs using decoupling approaches [12] etc.

The advantages include the *parallel computations* on the individual layers. The entire Multilayer network data is not analyzed at once hence it *reduces the space complexity* as well. This framework provides an opportunity to analyze individual layers independently later combine the partial results. While these are some of the major advantages, it is quite challenging to identify the correct heuristic to get accurate results in this approach. Going forward we introduce and discuss in detail the various heuristics to identify centrality values in HeMLNs and showcase the corresponding results. We have also performed experiments to demonstrate the correctness of the heuristics that have been proposed.

1.1 Problem Statement

For a given Heterogeneous Multilayer Network we need to design and implement algorithms to detect various centrality-based hubs using the decoupling framework. Here we need to compute each layer individually and only once. A composition function needs to be defined to make use of the pre-computed individual layer results and compute the desired centrality measure. A list of hubs needs to be given out as the output of the computations. We have considered the two most used centrality measures for this thesis, they are:

1. Degree Centrality
2. Betweenness Centrality

The single graph approach constitutes the ground truth, where all the nodes and edges of all the layers are combined into a single graph and the hubs are calculated using existing single graph algorithms. The proposed heuristics have been compared this baseline. Extensive experimentation of the proposed heuristics has been performed on variety of data sets including synthetic and real-world data sets considering a diverse set of graph characteristics.

The added advantage of using the divide and conquer approach is to be explored in the performance analysis section. The accuracy is calculated using the Jaccard Similarity index to understand the similarity in results from our heuristics and the ground truth. Precision, Recall and F1 Scores are also used for analysis.

For simplicity, we are currently considering undirected and unweighted single graphs. This is the first thesis to explore the advantages and disadvantages of decoupling-based framework on HeMLN centrality measures.

1.2 Thesis Organization

Remainder of the thesis paper is organized as follows:

Chapter 2 is about the related work in degree and betweenness centrality measures.

Chapter 3 gives details of decoupling-based framework for multilayer networks.

Chapter 4 gives details about the data sets that are used in this thesis.

Chapter 5 is about the two heuristics for degree centrality and the related experiments.

Chapter 6 is about the two heuristics on betweenness (stress) centrality and the related experiments.

Chapter 7 includes conclusion and future work.

CHAPTER 2

RELATED WORK

The concept of centrality of a graph was first proposed by Bavelas in 1948 [[21]] and it is researched extensively thereafter. These concepts have been recently applied on large data sets and scientists are working on making its computation efficient and reduce the space complexity of algorithms. Traditionally, these centrality measures have been implemented as main memory algorithms. With the advent of social media and web 2.0 the amount of data being used for analysis has exploded in volume thus becoming challenging to compute these on large data sets. In this thesis, while we understand how these centrality measures are calculated for simple graphs, our goal is to develop algorithms for centrality on heterogeneous multilayer networks using the decoupling-based framework proposed in [12].

While there are multiple graph centrality measures that are defined, this thesis focuses on two important ones: degree and betweenness centrality.

2.1 Degree centrality

Degree of a node is defined as the total number of edges that are incident on it. It was used as a centrality measure by Shah in 1954. If the network is directed, then the total edges that are directed towards the node is called indegree and that are directed outwards from the node is called outdegree [22, 23].

For a given graph G(V, E) with $|V|$ vertices and $|E|$ edges the degree centrality of a node v is given by the equation 2.1

$$C_{\mathrm{D}}(v) = Number\, of\ 1 - hop\, neighbors \tag{2.1}$$

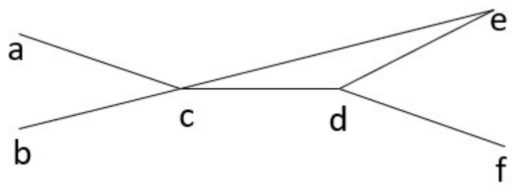


Figure 2.1: Simple graph for calculating degree centrality

Node IDs	**Degree value**	**Normalized degree centrality**
a	1	0.2
b	1	0.2
c	4	0.8
d	3	0.6
e	2	0.4
f	1	0.2

Table 2.1: Normalized degree centrality values for the graph in Fig 2.1

To normalize, the degree of a node is divided by the maximum number of nodes it can have direct edges with, that is $|V| - 1$ for an undirected graph.

$$C_{\mathrm{D}}(v) = \frac{degree(v)}{(|V| - 1)} \tag{2.2}$$

Figure 2.1 is a simple example of a graph and its normalized degree centrality values are shown in the table 2.1.

Centrality hubs as mentioned earlier are the most influential nodes in the network. **In this thesis we consider the nodes which are above or equal to average centrality value as a hub in the network**. Average degree of a graph can be calculated in a undirected graph if the number of nodes and total number of edges are known using the equation 2.3

$$Average\,degree = \frac{Total\,number\,of\,edges\, *\, 2}{Total\,number\,of\,nodes} \tag{2.3}$$

Group nodes	**Group degree centrality**
{c,d}, {c,f}	4
{a,d}, {a,c}, {b,d}, {c,e}	3
{a,e}, {a,f}, {b,c}, {b,e}, {b,f}, {d,f}, {e,f}	2
{a,b}, {d,e}	1

Table 2.2: Group degree centrality values for the graph in Fig 2.1

For the example in figure 2.1 the average degree centrality value is 0.4 so the hubs are **c,d** and **e**.

2.1.1 Other degree centrality formulations

1. Group degree centralization: Given a group of nodes in a graph, it is defined as the number of non-group nodes that are connected to the group nodes [24]. Multiple edges to the same non-group node is counted only once. For example, in figure 2.1 the group degree centrality of {c, d} is 4 that is nodes c and d as a group are connected to nodes a,b,e and f. Complete group degree centrality is shown in table 2.2

2. Time Scale Degree Centrality (TSDC): To include the presence and duration of the edges while calculating the degree centrality of a node, Time Scale Degree Centrality was proposed [25]. As an example, in figure 2.2, the nodes and edges are added as time progresses ($T > t_2 > t_1 > t_0$). In the fig 2.2 node d and edge 'cd' is added at time t1, node e and edge 'be' is added at time t2 etc. It's TSDC is given by the equation 2.4.

$$tsdc_{\mathrm{i}} = \sum_{j} x_{\mathrm{ij}}(T - t_{\mathrm{i}}) \tag{2.4}$$

Where T is the total time lapsed, t_i is the time where the node i first appeared and x_{ij} is the value in the adjacency matrix connecting nodes i and j. j belongs to the set of neighbours of node i. In Figure 2.2 the TSDC of node c is $T+(T-t_1)$

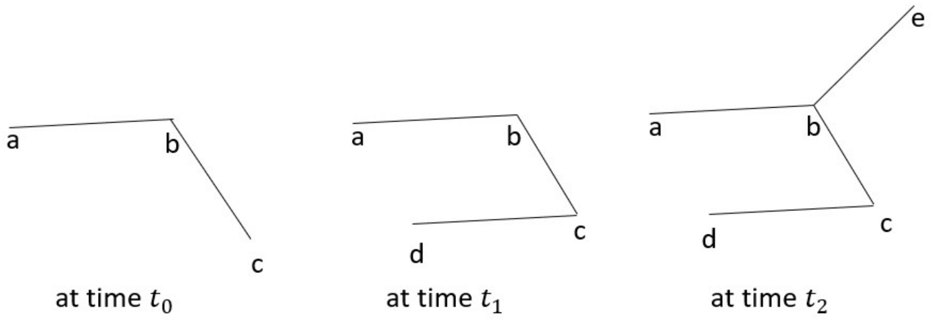


Figure 2.2: Sample graph where the characteristics of a graph vary based on time

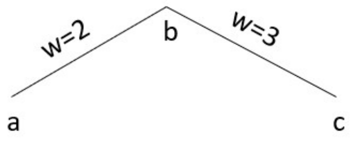


Figure 2.3: Sample graph for calculating Complex degree centrality.

if the weight of edges is assumed to be 1. Similarly for node b TSDC is T + T = 2T (for 2 edges 'ab' and 'bc' which are present throughout the time T)

3. Complex degree centrality (CDC): It is defined as "the geometric mean of the number of nodes to which this node is connected and the total weight of the edges" [26, 27]. Total weight is the sum of the weights of the edges. It is given by the equation 2.5

$$CDC_x = \sqrt[2]{DC_x * TR_x} \tag{2.5}$$

where DC is degree centrality of node x and TR is the total weight of the edges. For example in sample graph shown in figure 2.3

$$CDC(b) = \sqrt[2]{2 * 5} = 3.16$$

2.1.2 Applications of Degree Centrality

Some of the applications of degree centrality are as follows:

- Identifying the most influential nodes in a social network based on the number of connections.
- For predicting essential proteins based on weighted degree centrality [28]
- It is also used in the analysis of brain behavior like that of Parkinson's disease [29].

2.2 Betweenness centrality

Freeman (1977) mentions the betweenness centrality measure for social networks as early as 1977 [23]. He further explains the betweenness centrality as a global centrality index (unlike degree centrality which is local to a node) which gives us the most important nodes that are required for the information to flow across the network. Betweenness centrality measure as an individual number is of less importance but the relative measure gives us the relative importance of the node in the network. It helps in identifying the most centrally placed nodes in the graph [30].

2.2.1 Variants of betweenness centrality

There are different variants of betweenness centrality that are currently used by the research community. But all these variants are measured based on the all-pair shortest paths (APSP) in the network.

All-pair shortest paths (APSP) can be computed using Floyd-Warshall algorithm in O(V^3) time. However, this just calculates the total distance between any two nodes. For betweenness centrality, there is a need to know the total number of shortest paths that pass through each node. Multiple shortest paths that have the same distance are also considered.

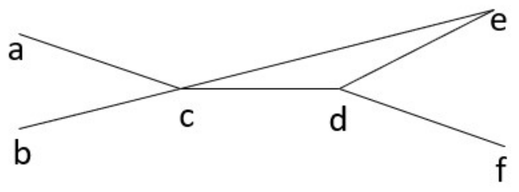


Figure 2.4: Sample graph for calculating Betweenness centrality

Node IDs	**Betweenness (Stress) centrality**	**Shortest paths in which the corresponding node is present**
a	0	
b	0	
c	7	{a,b}, {a,d}, {a,e}, {a,f}, {b,d}, {b,e} & {b,f}
d	4	{a,f}, {b,f}, {c,f} & {e,f}
e	0	
f	0	

Table 2.3: Betweenness centrality values of the nodes

Variant 1: Betweenness centrality can be calculated as the total number of shortest paths that pass through a node in a given network. This is also called as stress centrality and these two terms are used interchangeably [31,32]. Mathematically, the betweenness centrality measure is given by the equation 2.6.

$$C(v) = \sum_{s \neq v \in V} \sum_{t \neq v \in V} \sigma_{st}(v) \tag{2.6}$$

Where $\sigma_{st}(v)$ is the number of shortest paths from vertex s to t that passes through vertex v. For a sample graph (figure 2.4), to calculate the betweenness centrality values first step is to find out the APSP and sum up all the shortest paths that pass through the node. Table 2.3 shows the details. Node c and d are having higher betweenness centrality values as they are the bridge nodes. Leaf nodes are not present in any of the shortest paths so they are 0.

Variant 2 (Brandes): While the variant 1 considers the total number of shortest paths, it does not consider the sum of the fraction of the paths that pass through it out of all the shortest paths that exists between a pair of nodes. This is important as it reveals the nodes which are bridge nodes if the fraction is one then removing that node would remove the shortest path between those two nodes.

Following example differentiates between the two variants and the betweenness centrality values.

Betweenness centrality measure for variation 2 is given by equation 2.7.

$$C_B(v) = \sum_{s \in V} \sum_{t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{2.7}$$

Brandes proposed an algorithm which solves the problem in $\mathcal{O}(nm)$ and $\mathcal{O}(nm + n^2 \log n)$ time for unweighted and weighted networks respectively. It uses $\mathcal{O}(n+m)$ space [33]. This is the fastest known algorithm for calculating betweenness centrality

Brandes algorithm calculates betweenness values in two steps:

1. Runs the BFS starting from each node to calculate the All Pair Shortest Path.
2. Computing the dependencies $\delta(v)$ by traversing the shortest paths found in BFS by reverse order of distance.

Dependency of s on node u $\delta(s|u)$ is given by equation 2.8

$$\delta(s|u) = \sum_{v : u \in P_v} \frac{\sigma(s,u)}{\sigma(s,v)} (I_{v \in S} + \delta(s|v)) \tag{2.8}$$

Where $\sigma(s,u)$ is the number of shortest paths from node s to node u, $\sigma(s,v)$ is the number of shortest paths from node s to node v. $I_{v \in S}$ is 1 if $v \in S$. $\delta(s|v)$ is the dependency of node s on v, at the beginning it can be considered as 0 [34].

Endpoints: While calculating the APSP there are two options, to include source and target nodes (endpoints) in the shortest paths or to exclude them. In some applications such as that of information networks, the source and target nodes have the same control over its information as anyone passing through it [35].

If the endpoints are included, then the betweenness score of the nodes increases by the number of other nodes that are connected to it. In a network where every node can reach every other node betweenness score of all the nodes increases by a constant value 2n-2 where n is the total number of nodes in the network and by n-1 if it is undirected.

Considering the vital applications of the betweenness centrality measure, it is extensively researched by the community and multiple variations have been proposed thereafter.

<u>Applications of betweenness centrality:</u> Some of the applications of Betweenness centrality are as follows:

1. The urban traffic flow can be analysed by calculating the betweenness centrality values of the street network, as an example GPS-enabled taxi trajectory data (graph data) is used in the proposed paper [36] to analyse the traffic in Qingdao, China.
2. Out liner detection in the network data, betweenness centrality values are calculated repeatedly over a period of time to identify any new out liners, these out liners can be further analysed to detect anomalous data in the network [37].

Considering the vital applications of the betweenness centrality measure, it is extensively researched by the community and multiple variants have been proposed for that.

While these algorithms are focused on a single graph, there is a need to incorporate additional layers of information or various characteristics of the nodes and calculating these centrality measures to bring in the most important nodes of the whole network. To incorporate these new additions multilayer network was proposed, and it is being actively researched by the community. Next chapter talks extensively about multilayer networks and its analysis.

CHAPTER 3

MULTILAYER NETWORK ANALYSIS

Multilayer networks (also known as multiplexes) consist of multiple layers of simple graphs, each layer representing a feature of its entities and their relationships in the graph. These layers are connected by the relationship between the entities of different layers and are represented by the inter-layer edges of the network. Analysis of multilayer networks is being used in solving real world problems such as modeling and analysis of human brain networks [38], where brain regions are modelled as nodes and their structure or functional connection patterns as edges are evaluated, in dynamic social networks for clustering and anomaly detection [39], finding solutions to oil leakages [40], language analysis [41], etc.

<u>Advantages of modeling data into MLNs</u>: While considering MLNs the main advantage is that it can produce results based on multiple relationships (using a layer for each relationships) not just one as in single graphs. For example, in the health care domain instead of just looking at the ECG graphs/X-rays/MRI and coming to a conclusion, in MLNs one can include other factors such as health history of the person, weather and environment data, diet habits, etc. and derive an optimum conclusion [14].

<u>Disadvantages:</u> For computing the network properties such as centrality there are no algorithms currently available for MLNs that work without transforming an MLN into a simple or attribute graph. So the existing algorithms which are available for single graphs needs to be extended for MLNs and analyzed.

Graph representation: These HeMLNs can be represented using the adjacency matrix or a file with an unordered list of edges containing from and to node ids. The relationships among the nodes of the same layer form intra-layer edges and information about how each layer is connected to another layer forms the inter-layer edges.

Current approaches to solve MLNs use aggregation of all the layers in an MLN to a single graph or projection-based approaches for converting into a single graph. But this may result in loss of data. Efficiency of the solution can be an issue as the resulting graph size is likely to increase. Preserving structure and at the same time reducing the computation time is important. A decoupling-based framework has been proposed for solving computations directly on HeMLNs [12].

3.1 Decoupling Approach

A network decoupling-based approach has been used in this thesis for computing the centrality hubs of HeMLNs. There is been extensive research going on by using decoupling-based approach for different graph aggregate analysis as referenced in papers [12, 20]

Figure 3.1 shows the framework of the decoupling approach. It consists of analysis and composition functions. Each layer is analyzed independently using the analysis function. Then the partial results from any 2 layers are combined and along with the inter-layer edges, a composition function is used to output the results for the two layers. The binary composition can be applied to cover the entire MLN. Analyzing each layer can be done in parallel as they are independent [12].

With analysis function and composition function, graph processing of MLNs can be broken down and computed in parallel. This can benefit from parallelism and improve efficiency [12].

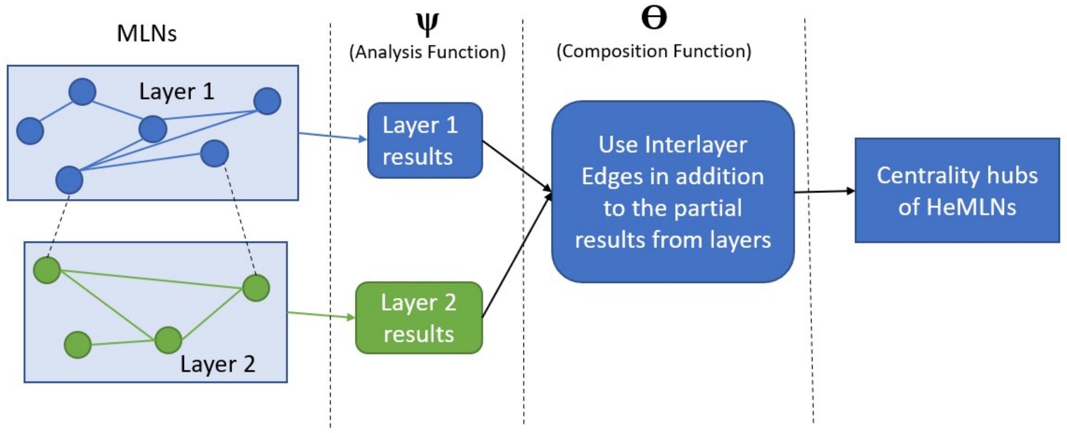

MLNs
Layer 1
Layer 2
Ψ
(Analysis Function)
Layer 1 results
Layer 2 results
Θ
(Composition Function)
Use Interlayer Edges in addition to the partial results from layers
Centrality hubs of HeMLNs

Figure 3.1: Decoupling approach

3.1.1 Advantages of Decoupling approach

- Structure preserving, there is no loss of data in this approach. So, the entire information can be utilized for analysis.
- Since each layer can be analyzed independently it can be run in parallel. Parallelism improves efficiency.
- Since small graphs are analyzed at any given time the memory usage can be reduced by writing the results of each layer into a file. Then utilizing these results in the composition function.
- Single graph algorithms can be utilized for analyzing graphs in each layer
- This approach is independent of network (or application) being analyzed.

3.1.2 Challenges using Decoupling approach

- Achieving high accuracy when compared to similar single network approaches is increasingly challenging.
- There is no information about other layers while analyzing one layer which makes it difficult to retain the most relevant information from that layer.

- As this is a new framework, new algorithms need to be developed using this framework.

3.2 Related work at ITLab

Decoupling based framework for various concepts such as community detection in HoMLNs and HeMLNs [7–9, 12, 16, 42], substructure discovery in HoMLNs [20] are being explored at IT Lab. As part of that work, decoupling based approach has been used and showed to be efficient in analyzing the MLNs.

3.2.1 Community detection

Community is a set of closely connected nodes in a network. It has been clearly defined and algorithms such as Louvain and Infomap are used. Community detection for MLNs are proposed using Decoupling approach and are proved to provide good results.

<u>Community detection in HeMLNs:</u> In the decoupling approach, analyzing function consists of identifying the communities in each layer, the composition function consists of bipartite graph matching using the interlayer edges to compute the overall HeMLNs community results [10, 12].

3.2.2 Substructure discovery

A decoupled based approach has been analyzed for substructure discovery in HoMLNs. The implementation is using Map/Reduce to utilize the efficiency achieved by parallelism. For each layer substructures of size k are identified independently. The identified substructures are expanded by adding one edge at a time. Then the composition function is applied to it, this process is repeated until the termination condition is set to true. The identified substructures are the substructures of the entire HeMLNs [20].

CHAPTER 4

DATASETS

Both synthetic and real world datasets have been used for validating accuracy and performance gain. Synthetic datasets are generated using subgen (from the AI Lab at Washington State University) [43] and Recursive-Matrix (R-MAT) [44]. For generating very large datasets a parallel version of R-MAT called Parallel R-MAT (PaRMAT) can be used [45]. This PaRMAT (Parallel R-MAT) PaRMAT divides the adjacency matrix into squares and executes using multiple threads. This multi threaded program can create very large graphs (with billions of edges). Synthetic data sets allow one to generate graphs with specific characteristics. This allows one to make sure the algorithms work for diverse graphs. Density, connected components, and distribution of edges in layers as well as the number of interlayer edges can be controlled. In addition to synthetic data sets, real world datasets have been used. They are taken from various sources including International Movie Database (IMDB) [46], The DBLP Computer Science Bibliography (DBLP) [47], Webgraph [48,49], etc. In general all these datasets are single graphs and these graphs need to be converted into HeMLNs for our purpose. In this thesis, we are using one of the two methods discussed below to generate HeMLNs from single graphs.

1. Method 1: Subgen graph generator provides an option to include vertex and edge labels. If the graph is generated using subgen, then vertices can be categorized using its vertex labels and edges can be categorized based on its edge labels. For example, vertex labels can be the layer to which the vertex belongs, edge labels can be intra layer edges or interlayer edges.

2. Method 2:If the graph is generated using RMAT or it is any real world dataset the graph might not contain any labels. In these scenarios the vertices can be categorized based on a random number generator. For each vertex, a random number is generated using which it is categorized into different layers. For example, if we are splitting a single graph into two layers, then a random number generator which generates two numbers based on a probability distribution is used. Each random number belongs to an individual layer and edges are classified based on the vertices on which it is incident upon. If both the vertices are in the same layer then it becomes an intralayer edge else, it is an interlayer edge.

Using the above methods, two or more layers are generated with intralayer and interlayer edges. As these methods ensure different nodes/entities in each layer, the multilayer network that is formed is a HeMLN. These generated HeMLNs are used in validating the heuristics that are discussed in later chapters.

4.1 Synthetic graphs

Synthetic graphs of different sizes are generated for analysis. Initially for developing heuristics various smaller datasets of 100, 1000, 5000 nodes and edges up to 10000 are used. Later, larger graphs up to 200,000 nodes and 10,000,000 edges of different sizes are created. Characteristics of one of the synthetic graphs with 100,000 nodes and 1,000,000 edges (represented as 100KV1ME)are discussed below.

4.1.1 Graph with 100,000 nodes and 1,000,000 edges (100KV1ME):

This graph is generated using PaRMAT. It is undirected graph with no duplicate edges or self loops. This is a single graph and has the characteristics shown in table

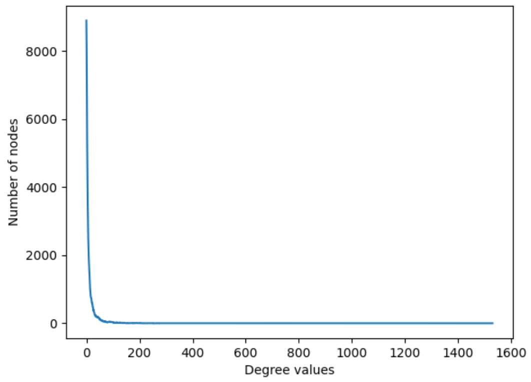
4.1. This dataset is further divided into two layers (layer 1 and layer 2) using method 2. In the table 4.1, each column represents the following:

- Dataset- the short hand representation of the dataset that is considered.
- Layers - represent the layers in the HeMLN
- No. of nodes - total number of nodes in the layer
- No. of edges - total number of edges in the layer
- Node distribution - percentage of nodes that are present in the layer when compared to the single graph
- Average degree - average degree of the layer
- Maximum degree - maximum degree of a node in the layer
- Minimum degree - minimum degree of a node in the layer
- Dangling nodes - total number of nodes with no edges.
- No. of connected components - total number of connected components in each layer
- Largest connected component - total number of nodes in the largest connected component that is present in the layer
- Sparsity - edge density in percentage (given by $\frac{Total\ number\ of\ edges\ in\ the\ layer * 100}{(N*(N-1)/2)}$ for an undirected graph where N is the total number of nodes in the graph)

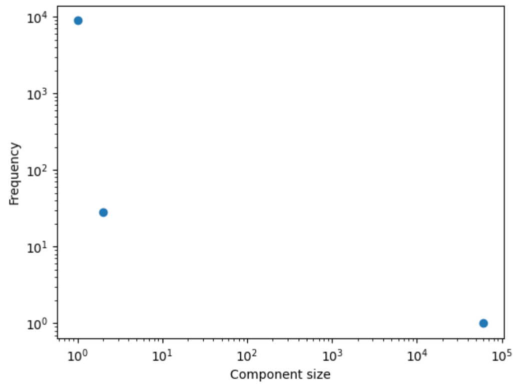
Dataset	Layers	No. of nodes	No. of edges	Node distribution (%)	Average degree	Maximum degree	Minimum degree	Dangling nodes	No. of connected components	Largest connected components	Sparsity(%)
100KV1ME		100000	1000000		20	2150	0	8976	9003	90972	0.01
	Layer 1	69984	496482	70	14.18	1529	0	8894	8923	61034	0.01
	Layer 2	30018	86976	30	5.79	324	0	7567	7658	22268	0.01
	Interlayer		416542								

Table 4.1: Characteristics of 100KV1ME dataset

Figure 4.1 represent the degree distribution and connected component size distribution in layer 1. For degree distribution the x axis represent the degree values

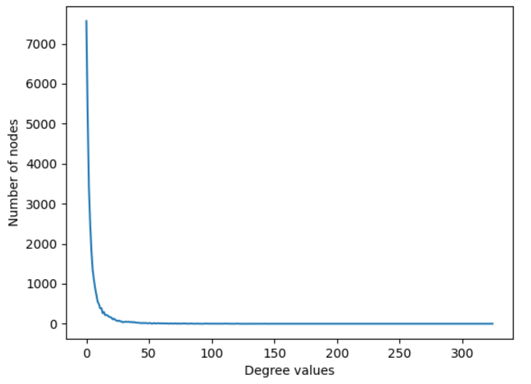


(a) Layer 1 degree distribution

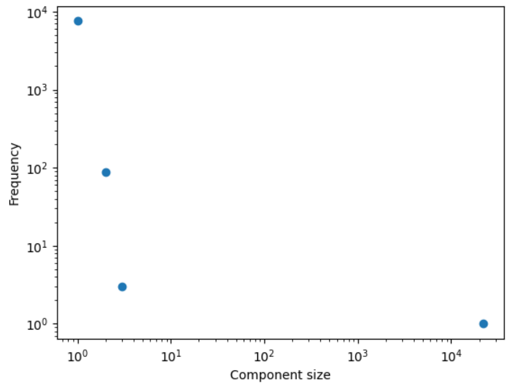


(b) Layer 1 connected component size distribution

Figure 4.1: Layer 1 network characteristics.



(a) Layer 2 degree distribution



(b) Layer 2 connected component size distribution

Figure 4.2: Layer 2 network characteristics.

of each node and y axis represent the number of nodes having that particular degree. For connected component size plot, x axis represent the component sizes and y axis represent the frequency of that particular component size. Similarly, figure 4.2 represents the same characteristics for layer 2.

4.1.2 Characteristics of all the synthetic datasets used in this thesis

Smaller datasets: These are smaller datasets starting from 100 nodes and 615 edges. These datasets are used initially to test the heuristics. Table 4.2 gives in detail information about the datasets.

In table 4.2,

- Dataset column represents the dataset name
- Layers column represent the different layers in the dataset. For these datasets, only two layers has been created and used for analysis.
- Number of nodes represent the total nodes in that layer
- Number of edges represent the total edges in that layer
- Node distribution as a percentage represent the node distribution among the layers.

Further the larger datasets are created by increasing both the number of nodes and edges. In this thesis we have used synthetic graphs upto 200,000 nodes and 10 million edges for analysis and experiments.

Dataset	Layers	Number of nodes	Number of edges	Node distribution (%)
100V615E				
	Layer 1	63	390	63
	Layer 2	37	143	37
	Interlayer		82	
1KV2KE				
	Layer 1	485	470	49
	Layer 2	515	491	52
	Interlayer		1036	
1KV6KE				
	Layer 1	697	3815	70
	Layer 2	304	788	30
	Interlayer		684	
5KV10KE-1				
	Layer 1	2467	2391	49
	Layer 2	2533	2560	51
	Interlayer		5069	
5KV10KE-2				
	Layer 1	3525	7436	71
	Layer 2	1475	1312	30
	Interlayer		1255	

Table 4.2: Smaller synthetic dataset characteristics

Datasets	**Layers**	**Nodes**	**Edges**	**Node distribution (%)**	**Average degree**	**Max degree**	**Min degree**	**Dangling nodes**	**No. connected components**	**Largest component**	**Sparcity (%)**
25kv100ke		25000	100000		8	465	0	4385	4422	20541	0.02
	Layer 1	17572	49588	70	5.643979058	241	0	4065	4124	13388	0.02
	Layer 2	7430	8757	30	2.357200538	134	0	3040	3128	4201	0.02
	Interlayer										
25kv200ke		25000	200000		16	926	0	2309	2320	22671	0.03
	Layer 1	17452	97894	70	11.21865689	636	0	2272	2287	15152	0.03
	Layer 2	7550	18174	30	4.814304636	201	0	1962	1987	5536	0.03
	Interlayer										
25kv300ke		25000	300000		24	1307	0	1399	1402	23597	0.05
	Layer 1	17604	150466	70	17.09452397	925	0	1428	1434	16166	0.05
	Layer 2	7398	25380	30	6.861313869	242	0	1459	1469	5921	0.05
	Interlayer										
25kv400ke		25000	400000		32	1716	0	996	999	24000	0.06
	Layer 1	17469	195675	70	22.40254165	935	0	1060	1063	16405	0.06
	Layer 2	7533	36026	30	9.564848002	535	0	1145	1154	6372	0.06
	Interlayer										
35kv400ke		35000	400000		22.85	1262	0	1966	1970	33028	0.03
	Layer 1	24580	194754	70	15.8465419	656	0	2100	2107	22468	0.03
	Layer 2	10422	36535	30	7.011130301	378	0	1985	2003	8401	0.03
	Interlayer										
45kv400ke		45000	400000		17.77	1257	0	4115	4130	40857	0.02
	Layer 1	31442	193573	70	12.3130208	664	0	4062	4082	27342	0.02
	Layer 2	13560	37333	30	5.506342183	392	0	3419	3459	10059	0.02
	Interlayer										
55kv400ke		55000	400000		14.5	1237	0	6130	6153	48824	0.01
	Layer 1	38697	192727	70	9.960823836	832	0	5972	6017	32637	0.01
	Layer 2	16305	37469	30	4.596013493	225	0	4815	4879	11361	0.01
	Interlayer										
100KV1ME		100000	1000000		20	2150	0	8976	9003	90972	0.01
	Layer 1	69984	496482	70	14.18844307	1529	0	8894	8923	61034	0.01
	Layer 2	30018	86976	30	5.794923046	324	0	7567	7658	22268	0.01
	Interlayer		416542								
100KV2ME		100000	2000000		40	4041	0	4260	4265	95732	0.02
	Layer 1	79890	1265733	80	31.69	3227	0	4455	4467	75413	0.02
	Layer 2	20112	83864	20	8.34	630	0	4004	4038	16041	0.02
	Interlayer		650403								
100KV3ME		100000	3000000		60	5686	0	2353	2354	97647	0.03
	Layer 1	90073	2451052	90	54.42	5160	0	2420	2421	87653	0.03
	Layer 2	9929	27996	10	5.64	321	0	2572	2596	7306	0.03
	Interlayer		520952								
100KV4ME		100000	4000000		80	7349	0	1629	1630	98371	0.04
	Layer 1	90108	3241201	90	71.94	6599	0	1722	1723	88386	0.04
	Layer 2	9894	39890	10	8.06	409	0	1997	2013	7866	0.04
	Interlayer		718909								
200KV1ME		200000	1000000		10	1423	0	36881	37097	162684	0.00
	Layer 1	160073	643022	80	8.03	1144	0	34761	35017	124794	0.00
	Layer 2	39929	39062	20	1.96	209	0	19337	19813	19597	0.00
	Interlayer		317916								
200KV5ME		200000	5000000		50	6702	0	7373	7375	192625	0.01
	Layer 1	119908	1792650	60	29.9	2993	0	7923	7938	111957	0.01
	Layer 2	80094	805373	40	20.11	2701	0	8038	8060	72014	0.01
	Interlayer		2401977								
200KV10ME		200000	10000000		100	12486	0	2842	2844	197156	0.03
	Layer 1	80085	1610469	40	40.22	2712	0	3716	3722	76359	0.03
	Layer 2	119917	3584044	60	59.78	7566	0	3438	3440	116477	0.02
	Interlayer		48055487								

Table 4.3: Synthetic graph characteristics of larger datasets

4.2 Real-world datasets:

Many real-world datasets have been used including International Movie Database (IMDB), The DBLP Computer Science Bibliography and datasets from The Laboratory for Web Algorithmics. The datasets from the The Laboratory for Web Algorithmics are very large real world datasets [48, 49]. Some of the real-world datasets and its characteristics have been discussed below.

4.2.1 International Movie Database (IMDB)

This is a real world dataset containing information about movies produced around the world [46]. It has mainly 3 layers Actors, Directors and Movies. It can be modelled as a HeMLN as all three layers have different entities and they have inter layer edges connecting between them.

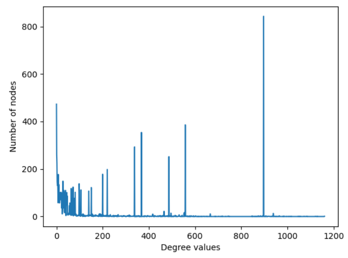
Tables 4.4 and 4.5 give the exact numbers.

Layers	**No. of Nodes**	**No. of Intralayer Edges**	**Average degree**	**Maximum degree**	**Minimum degree**	**Dangling nodes**	**Number of connected components**	**Largest Connected components**	Spasity
Actor	9486	996527	210.1	1161	0	473	651	7918	1.11
Director	4511	250845	111.21	517	0	296	425	3429	1.23
Movie	7952	8777618	2207.65	3167	0	1	10	3168	13.88

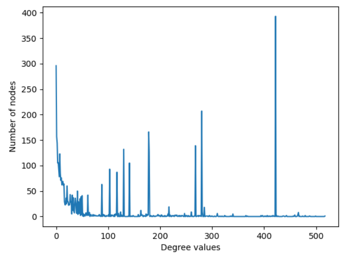
Table 4.4: Characteristics of IMDB dataset

Layers	**Number of Interlayer Edges**
Actor Director	32033
Actor Movie	31422
Director Movie	8581

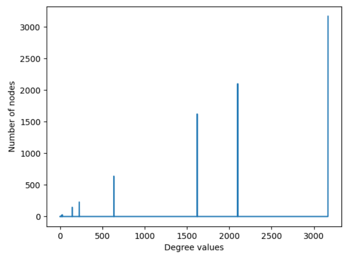
Table 4.5: Characteristics of IMDB dataset



(a) Actor layer



(b) Director layer



(c) Movies layer

Figure 4.3: Degree distribution of actor, director and movies layer.

4.2.2 The DBLP Computer Science Bibliography

DBLP dataset has information about papers published, authors of different papers, conferences in which the papers were published, published year etc [47]. This dataset can be modelled as a HeMLN by considering a layer of authors who have worked on the same paper, papers published in the same domain or papers published in a year. Considering these features the main 3 layers used in this thesis are CoAuthor, Papers and Year. Tables 4.6 and 4.7 give the exact numbers.

Layers	**Number of Nodes**	**Number of Intralayer Edges**
Coauthor	16918	2483
Papers	10326	12044080
Year	18	18

Table 4.6: Characteristics of DBLP dataset

Layers	**Number of Interlayer Edges**
Author Paper	37142
Author year	29984
Paper Year	10326

Table 4.7: Characteristics of DBLP dataset

4.2.3 Webgraph datasets:

These datasets are obtained from the The Laboratory for Web Algorithmics. These datasets are compressed using webgraph to reduce storage space. To use these datasets initially, the dataset has to be loaded using webgraph object and re-written into a file in the format used for other datasets. These also produce single large graphs, they will have to be further partitioned using one of the two methods discussed earlier. Some of the webgraph dataset characteristics are shown in table 4.8

Datasets	**Nodes**	**Edges**	**Average degree**	**Max degree**	**Min degree**	**Dangling nodes**	**No. connected components**	**Largest component**	**Sparcity**	**Details of the dataset**
word association-2011	10617	36116	6.8	269	0	2348	2351	8264	0.03	Its a graph describing the results of an experiment where the nodes correspond to words and edges represent a cue-target pair.
uk-2007-05@100000	100000	1521854	30	39941	0	3193	3465	95985	0.02	.uk domain graph, where nodes represent the URLs and edges connection between them
cnr-2000	325557	1752766	10.76	10749	0	17577	27779	214167	0.00	Italian CNR domain
amazon-2008	735323	2610591	7.1	393	0	20526	21028	707532	0.00	Similarity between books in Amazon store.
in-2004	1382908	8465313	12.24	11219	0	109682	138110	925340	0.00	.in domain

Table 4.8: Webgraph dataset characteristics

CHAPTER 5

DEGREE CENTRALITY

Degree of a node in a network (or graph) is the total number of edges that are incident on it. Degree hubs in a network are the nodes with degree greater than or equal to the average degree of the network [3, 50]. While degree hubs are defined for a single graph, to the best of our knowledge there are no algorithms to calculate the degree hubs for HeMLNs. We extrapolate the definition of a hub from a single graph to HeMLNs, if the HeMLN is transformed into a single graph by a union operation and retaining all the inter layer edges. In this thesis we propose different heuristics for composition function to achieve maximum accuracy and performance. We test our heuristics for identifying the degree hubs of a multilayer network against the ground truth.

Ground truth for degree centrality: For degree centrality in HeMLNs the ground truth is calculated by aggregating all the layers into a single network. It includes nodes from all the layers, intra-layer edges and inter-layer edges. Degree of each node is calculated and degree hubs are identified.

The degree hubs obtained from the ground truth are compared with the hubs obtained from the decoupling-based algorithms for accuracy. As discussed, Jaccard's coefficient, precision and recall are used for comparing the results with the ground truth.

Consider any two layers of a HeMLN with different entities/nodes. Each layer consists of a set of nodes and intra-layer edges. These two layers are connected by a set of inter-layer edges. As a whole, they form a two layer HeMLN. **All the heuristics proposed use two layer HeMLN for computations**[1].

As a first step, ground truth is calculated on these two layers separately and the hubs are identified. All the parameters such as average degree, time taken and the list of hubs along with the degree values are saved (written to a file).

As discussed in the decoupling approach, each layer is considered independently. The input to the analysis function are the corresponding layer graphs consisting of nodes and intra-layer edges. The output of analysis function is the list of node ids along with the corresponding degree values for each layer. The outputs are written to a file. For two layers, each analysis function is run independently (can be run in parallel). The analysis function of all the individual layers needs to be computed before proceeding to the composition function.

We propose two heuristics for calculating degree centrality in HeMLNs. *The goal of these heuristics is to keep minimal information from each layers, use the information from each layer and the inter-layer edges to compute the degree hubs of the entire HeMLN.* These heuristics are analysed based on accuracy and time when compared to that of ground truth. These heuristics and its results are discussed in the below sections.

Naive algorithm: A naive algorithm would be to take the hubs from each layer, union them and consider them as the hubs for the two layer HeMLN. This does not use any additional information from the layers. This would give a result that is not likely to match the ground truth (except in some unique cases.)

[1]Thus, heuristics for binary operator have been proposed, which can be easily extended to k layers through iterative application on the partial results.

This naive accuracy can be taken as the minimum accuracy which needs to be further improved using additional information from each layer along with inter layer edge information for composition. These become the heuristics.

The use of additional information from each layer can be seen as a spectrum with *no information (only hubs) on one side (naïve)* and *as much information (entire layers) as needed* to get the same accuracy as ground truth. The challenge is to maximize accuracy with minimum information use from each layer. This approach is used in refining the heuristic to improve accuracy. Of course, the cost of the decoupling approach (time taken) also changes and increases with more information used. **This is a trade off to make sure the cost does not exceed the cost of the ground truth and preferable maximize accuracy while keeping the cost significantly lower as compared to the ground truth.** Parallel computation of layers will also help in this.

5.1 Degree Centrality Heuristic 1

5.1.1 Design

As discussed, the decoupling approach includes two functions: analysis and composition functions. These two functions are discussed below.

Analysis function: Input to the analysis function is the nodes in the layer and the intra-layer edges. The output from this layer in heuristic 1 is the *degree hubs of that layer and its corresponding degree values.* So, from each layer we are keeping the degree hubs as the output along with the total number of nodes and edge information.

Composition function: In the composition function, the outputs from each layer along with the inter-layer edges are analyzed. The degree values obtained from each layer are loaded to the main memory. In this heuristic for each inter-layer edge, the degree of both the nodes on which the edge is incident upon are incremented by 1.

Degree average can be computed using the equation 2.3. Where total nodes is the sum of nodes from layer 1 and 2. Total edges is the sum of layer 1 intra-layer edges, layer 2 intra-layer edges and inter-layer edges. Figure 5.1 shows details of proposed heuristic 1. The complexity of composition is linear with respect to the number of edges and does not depend on the number of nodes and edges in layers. For each inter layer edge, a hash lookup is performed on both the nodes to increment the degree.

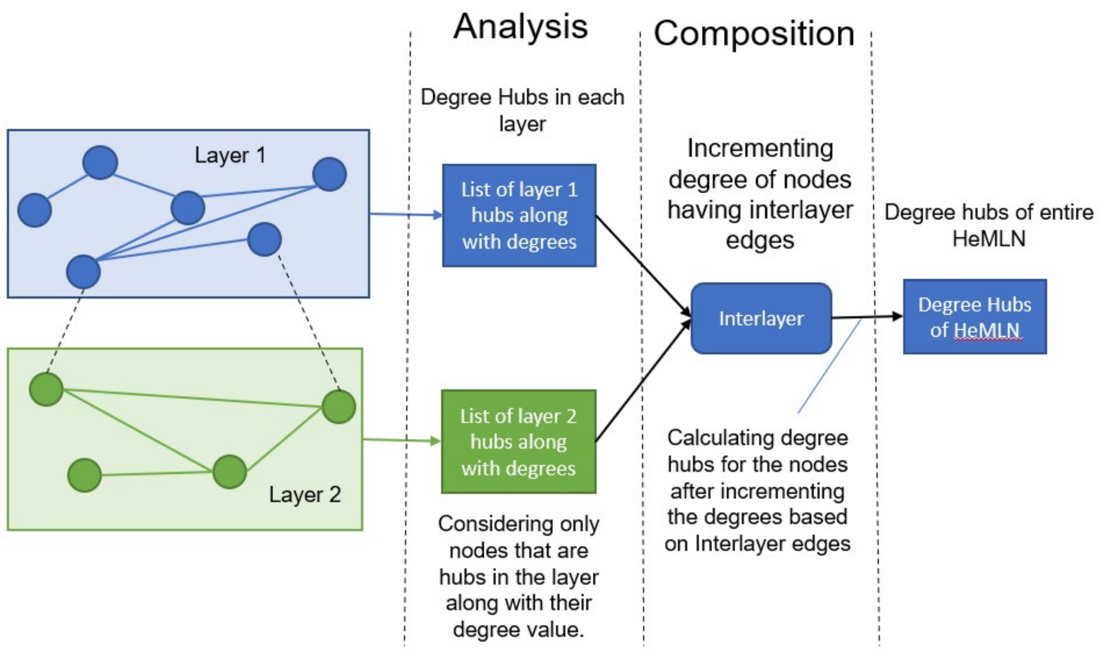

Analysis
Composition
Degree Hubs in each layer
Layer 1
List of layer 1 hubs along with degrees
Incrementing degree of nodes having interlayer edges
Degree hubs of entire HeMLN
Interlayer
Degree Hubs of HeMLN
List of layer 2 hubs along with degrees
Layer 2
Calculating degree hubs for the nodes after incrementing the degrees based on Interlayer edges
Considering only nodes that are hubs in the layer along with their degree value.

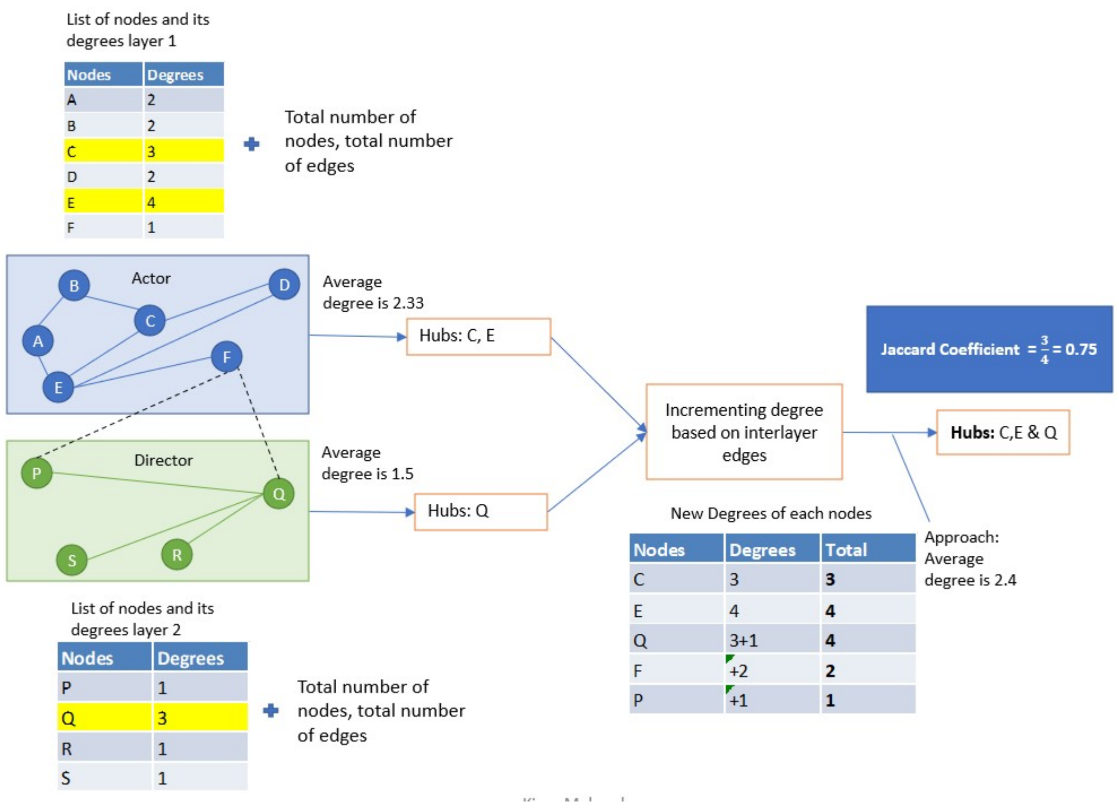
Figure 5.1: Network Decoupling approach for identifying Degree based hubs for HeMLNs using Heuristic 1

The composition function of the proposed heuristics is shown in algorithm 1. Where $Layer_i$ and $Layer_j$ are the two lists of layer hub nodes with the corresponding degrees obtained from the analysis function. $V_{i1}....V_{in}$ represent node IDs starting from 1 to n for layer i, *Hubs* represents the list of hubs along with the degree values as the output of the composition function. *IE* corresponds to the set of inter-layer edges between the two nodes (NodeID1 and NodeID2)

Algorithm 1 Composition algorithm for degree centrality heuristic 1

```
INPUT:
$Layer_i = \{V_i1 : DC_1, V_i2 : DC_2, ..., V_{in1} : DC_{n1}\}$
$Layer_j = \{V_j1 : DC_1, V_j2 : DC_2, ..., V_{jn2} : DC_{n2}\}$
$IE = \{(V_{i1}, V_{j1}), (V_{i2}, V_{j2}), ...\}$
ALGORITHM:
1: for e$\{NodeID1, NodeID2\} \in$ IE do
2:     $Layer_i[NodeID1] = Layer_i[NodeID1] + 1$
3:     $Layer_j[NodeID2] = Layer_j[NodeID2] + 1$
4: end for
5: Average degree=$\frac{Total\ number\ of\ edges\ in\ HeMLN * 2}{Total\ number\ of\ nodes\ in\ the\ entire\ HeMLN}$
6: for NodeID, DegreeValue $\in Layer_i \cup Layer_j$ do
7:     if DegreeValue $\geq$ Average degree then
8:         $Hubs.append(NodeID)$
9:     end if
10: end for
OUTPUT:
$Hubs = \{V_i1 : DC_{i1}, V_i2 : DC_{i2}, ..., V_{j4} : DC_{j4}....\}$
```

As an illustrative example, consider the following two layers as shown in figure 5.2. In this example, actor and director layers are analyzed individually and the list of degree hubs are identified. In this example, nodes C and E are the hubs from actors and Q is the hub from layer director. Node E has degree 4 and C,Q have degree 3. Further in the composition function, the degree of nodes are incremented by one for every interlayer edge. Edges FP and FQ are the interlayer edges. So, we increment the degrees of nodes P,F and Q by 1. In the composition function the degree of F is initially 0 as it is not a degree hub in the layer and we would not have saved the actual degree information of that node. The new degree hubs for HeMLN using heuristic 1 can be identified as C, E and Q as these nodes have greater than or equal to the average degree ($\geq$2.4).


List of nodes and its degrees layer 1
Total number of nodes, total number of edges
Actor
Average degree is 2.33
Hubs: C, E
Director
Average degree is 1.5
Hubs: Q
Incrementing degree based on interlayer edges
Jaccard Coefficient = $\frac{3}{4}$ = 0.75
Hubs: C,E & Q
Approach: Average degree is 2.4
New Degrees of each nodes
List of nodes and its degrees layer 2
Total number of nodes, total number of edges

Nodes	Degrees
A	2
B	2
C	3
D	2
E	4
F	1

Nodes	Degrees
P	1
Q	3
R	1
S	1

Nodes	Degrees	Total
C	3	**3**
E	4	**4**
Q	3+1	**4**
F	+2	**2**
P	+1	**1**

Figure 5.2: Example of heuristic 1

Node	Degree
A	1
B	2
C	3
D	2
E	3
F	3
P	2
Q	4
R	2
S	2

Table 5.1: Ground truth result

The average degree from ground truth is 2.4 and the hubs are **C, E, F** and **Q** as per the table 5.1.

We compare ground truth and heuristic 1 results using Jaccard co-efficient. So the union of the hubs include C,E,F and Q. Intersection of the results are C,E and Q. So the Jaccard co-efficient would be $\frac{3}{4}$=0.75.

5.1.2 Sample Data Set Results and Analysis

The results of this heuristics for sample data sets are shown in figure 5.3. The heuristics have been tested on both synthetic and real-world data sets. For sample data, **Heuristic 1 accuracy is always higher than that of naive approach**. We will have to test if it holds good for all the larger data sets. In heuristics 1 we were able to achieve a good accuracy of around 90% for actor-director layers, however, the accuracy is not the same for Director-Movie or Actor-Movie layers.

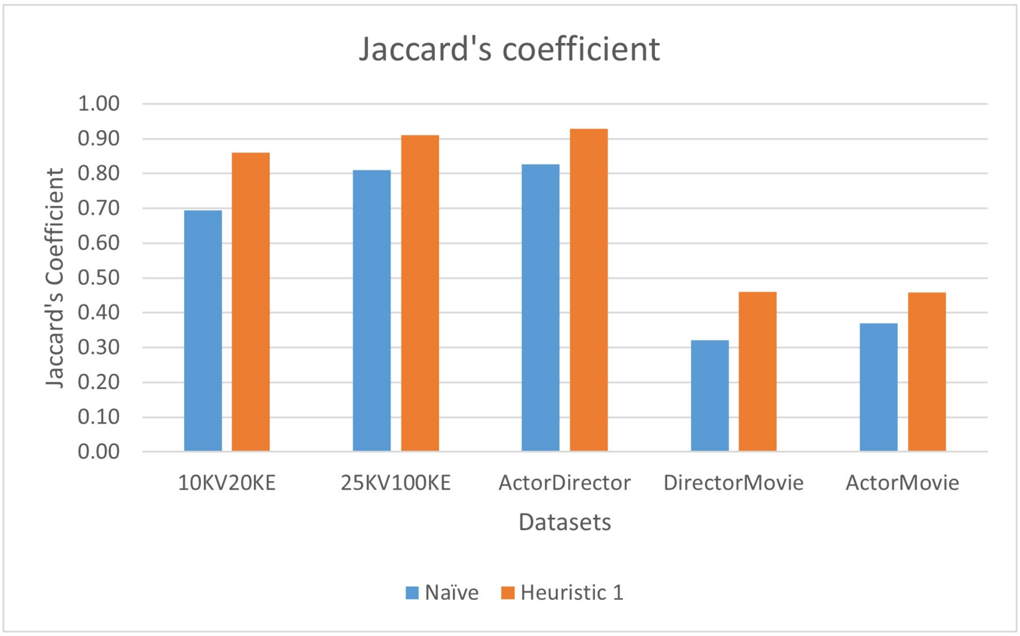


Figure 5.3: Heuristic 1 Jaccard's Coefficient (accuracy) results

Figure 5.4 shows precision and figure 5.5 recall plots for heuristics 1. It can be seen that the **precision is 100%** and **always better than that of naive approach**. This gives a lot of confidence in the heuristic that is proposed however, **further improvements that can be done to improve recall**.

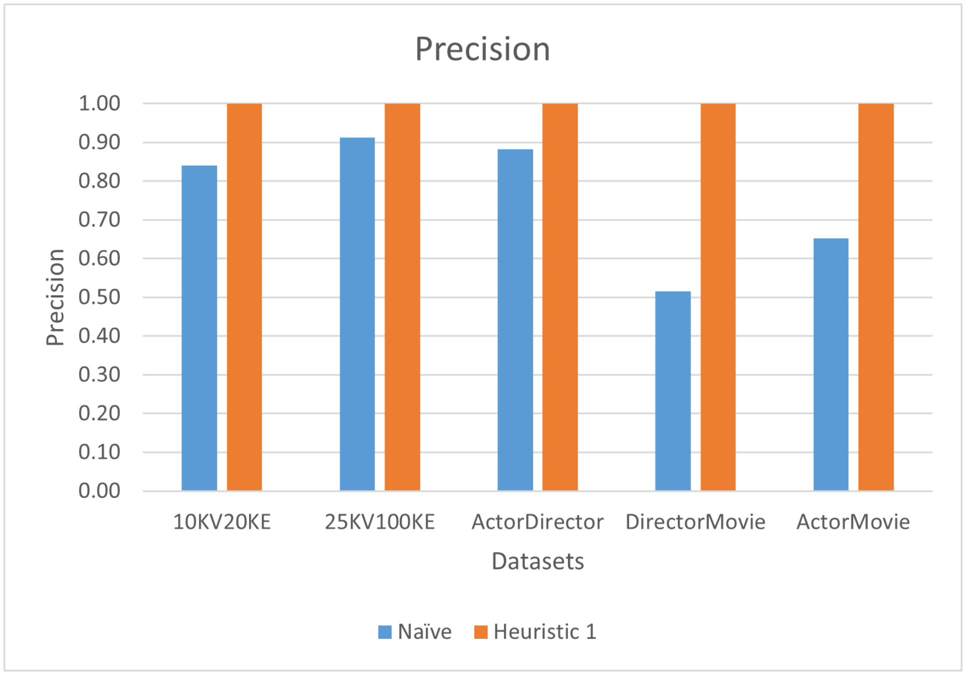


Figure 5.4: Precision plot for heuristic 1

By observing these plots we can say that the heuristic is giving 100% precision but because of false negatives recall is not 100%. We try to improve this in our heuristic 2, where we increase the amount of information retained from each layer to improve the accuracy.

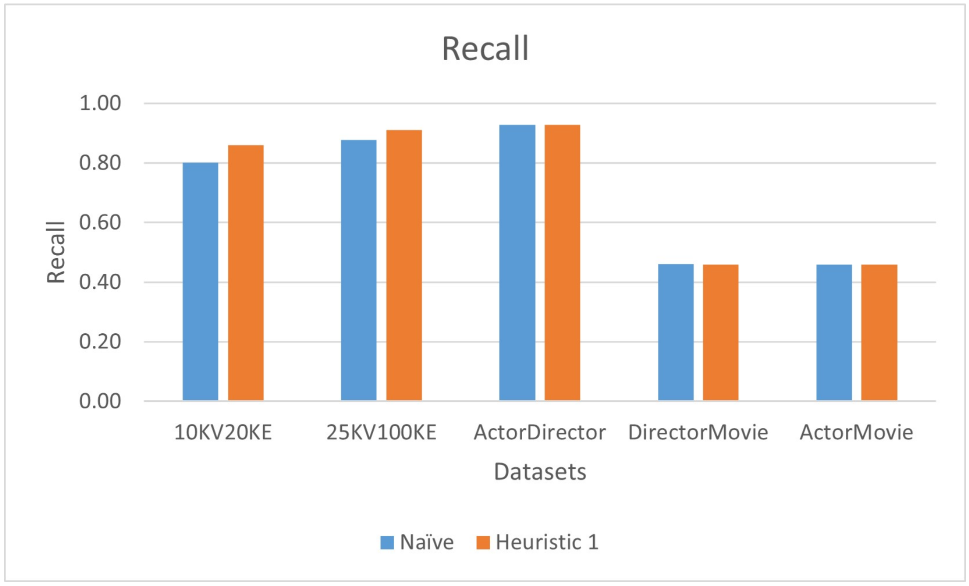


Figure 5.5: Recall plot for heuristic 1

5.2 Degree Centrality Heuristic 2

5.2.1 Design

Previous heuristics gave us 100% precision results however the **recall was inconsistent** and it tends to follow the naive accuracy for some data sets. Recall reduces if the false negatives are more. To improve recall in heuristic 2 we propose to keep the information of all the nodes from both the layers. So in heuristics 2, *along with the hub degree information the degree values of the remaining nodes is also retained* as an output of the layer-wise analysis step. The composition step remains the same as Heuristic 1, where degree of a node is updated if there is an incident inter-layer edge.

5.2.2 Sample Data Set Results and Analysis

Figure 5.6 compares the accuracy results of heuristics 1 and 2 for the same set of data sets. **Heuristic 2 is giving 100% accuracy for all the data sets.**

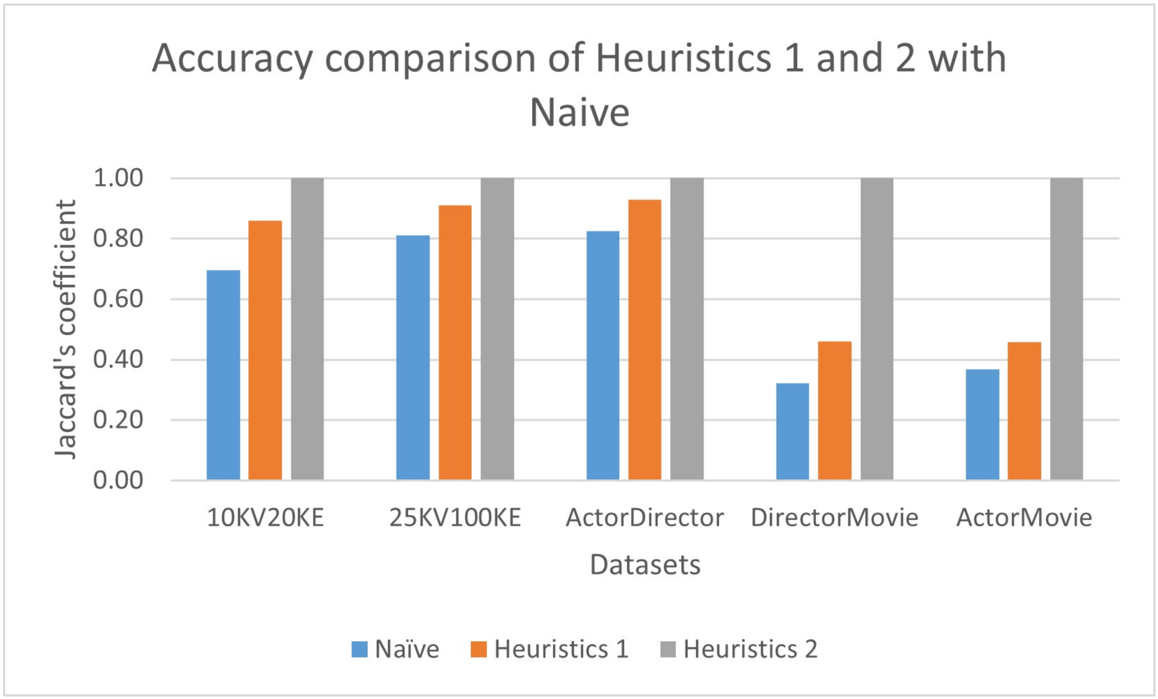


Figure 5.6: Accuracy comparison of heuristics 1 and 2

Figure 5.7 shows the performance evaluations of the various data sets using the heuristic 2 for identifying degree centrality. The time taken for heuristic 2 is calculated based on the sum of maximum time taken by the layers and the composition time. The results show us that with lesser time we are able to get 100% accuracy. The **maximum time savings that was achieved for these data sets is 33.76%**. It means that **heuristic 2 was able to produce the same results as that of ground truth with 33.76% less time**.

Figure 5.8 compares the amount of information retained from each layer by heuristics 1 and 2. It can be observed that the increase in accuracy in heuristic 2 comes at a cost of increase in amount of information retained from each layer. *The amount of additional information retained depends on the number of nodes in the layers.* Also, heuristic 2 always retains more information than that heuristic 1.

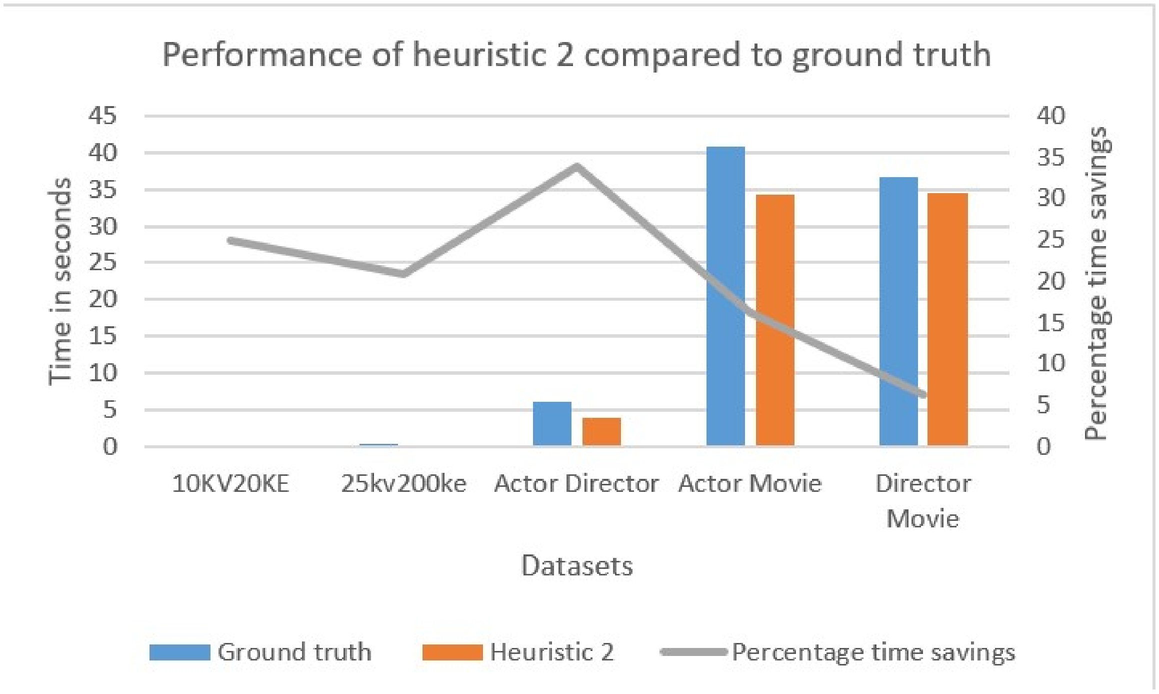


Figure 5.7: Performance evaluations for different real-world data sets.

This also proves that there is a trade off between amount of information used for computation and the accuracy of the results.

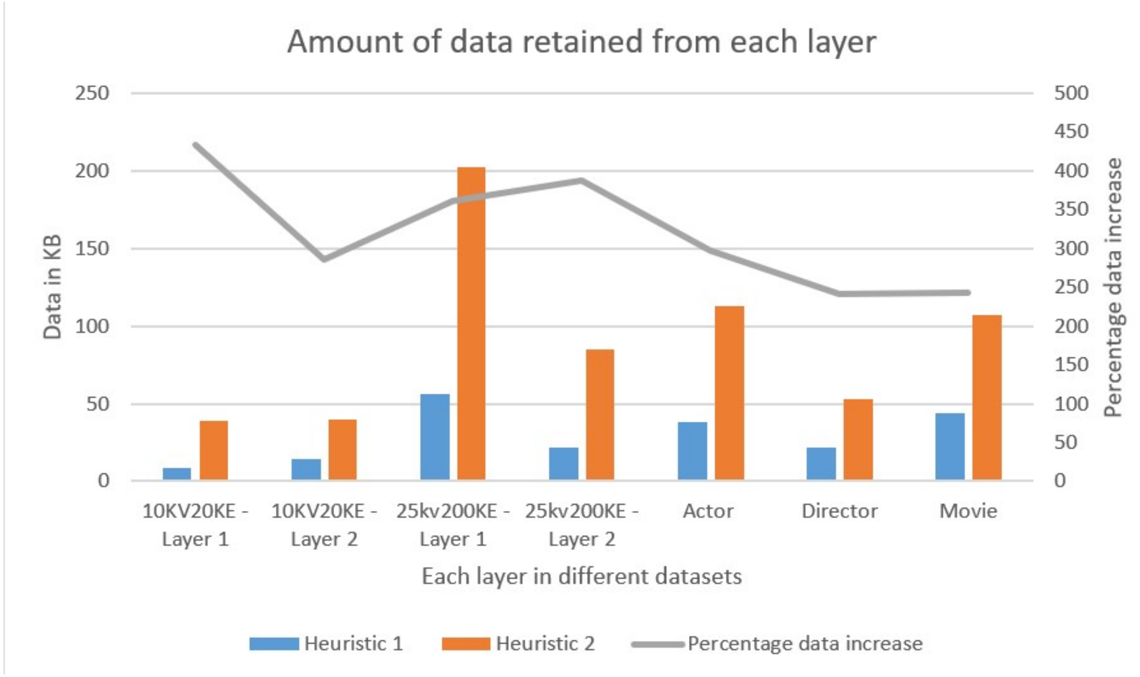


Figure 5.8: Comparison between heuristic 1 and 2 w.r.t amount information retained from each layer.

5.3 Large synthetic and real-world dataset experiments and performance evaluations

All the experiments are performed using heuristic 2 and the results are compared to that ground truth for accuracy. In this section we perform experiments using large synthetic data sets and real-world data sets, by varying the graph characteristics in order to validate the performance of the heuristic under different conditions.

Figure 5.9 shows the accuracy for synthetic data sets. This experiment varies sparsity for a data set, by fixing the number of nodes (100KV or 200KV), and varying the number of edges (1ME, 2ME, 3ME, 5ME, ..., 10ME). Upto 200,000 vertices and 10 million edge data sets are used for this experiment. In each case, accuracy for the heuristic is 100%.

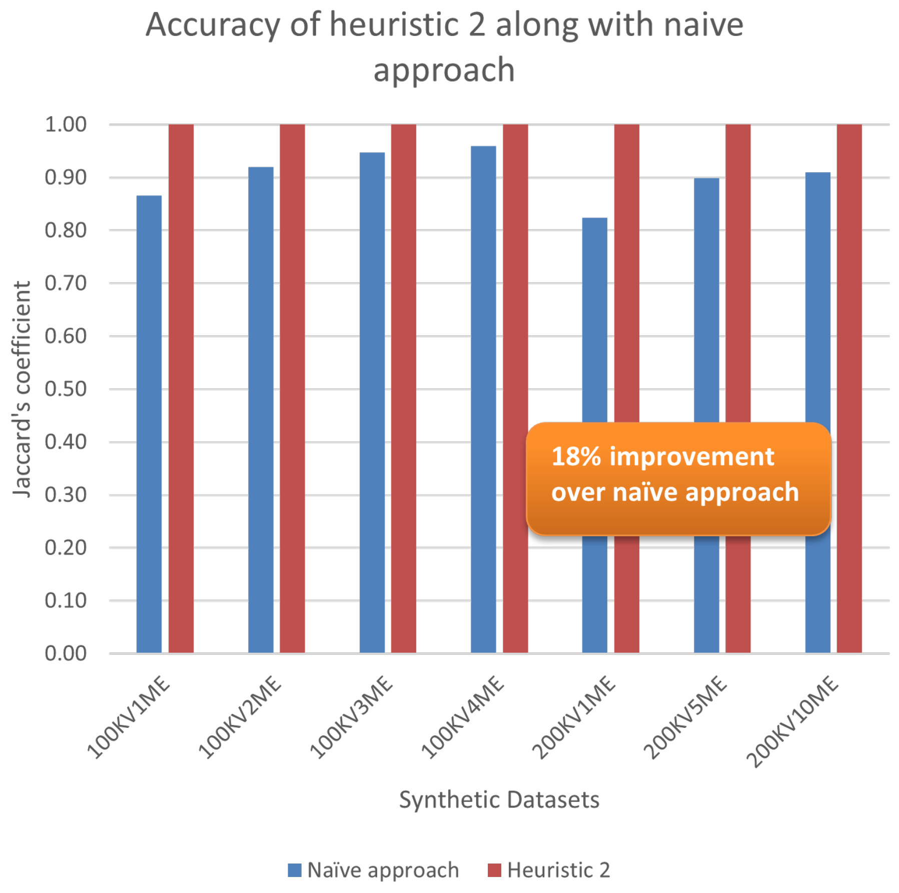


Figure 5.9: Accuracy results for large synthetic data sets (100K1ME to 200K10ME)

Figure 5.10 shows the performance comparison for synthetic data sets. We were able to **achieve minimum of 15% to maximum of 60% time savings** in these data sets compared to ground truth.

Similarly, the same heuristic was run on large real-world data sets. Figure 5.11 shows that accuracy for real-world data sets was also 100%.

Figure 5.12 shows the performance of heuristic 2 when compared to ground truth for large real-world data sets. The **minimum time savings is 2.6% and the maximum is 67%.**

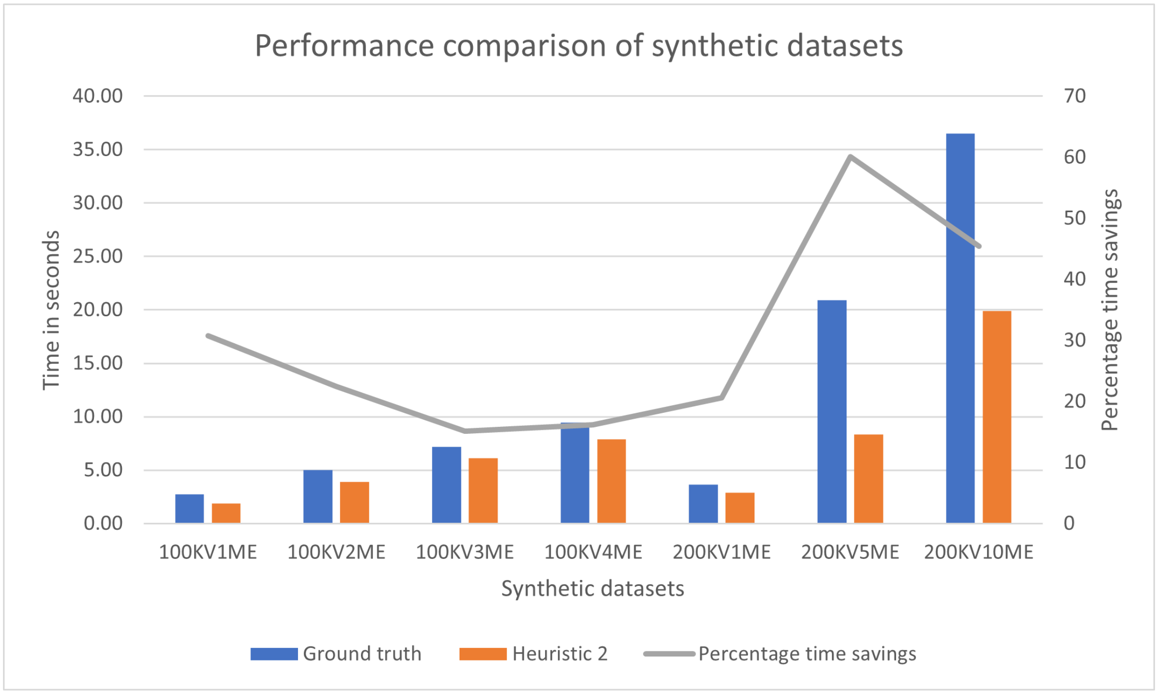


Figure 5.10: Performance comparison for large synthetic data sets (100K1ME to 200K10ME)

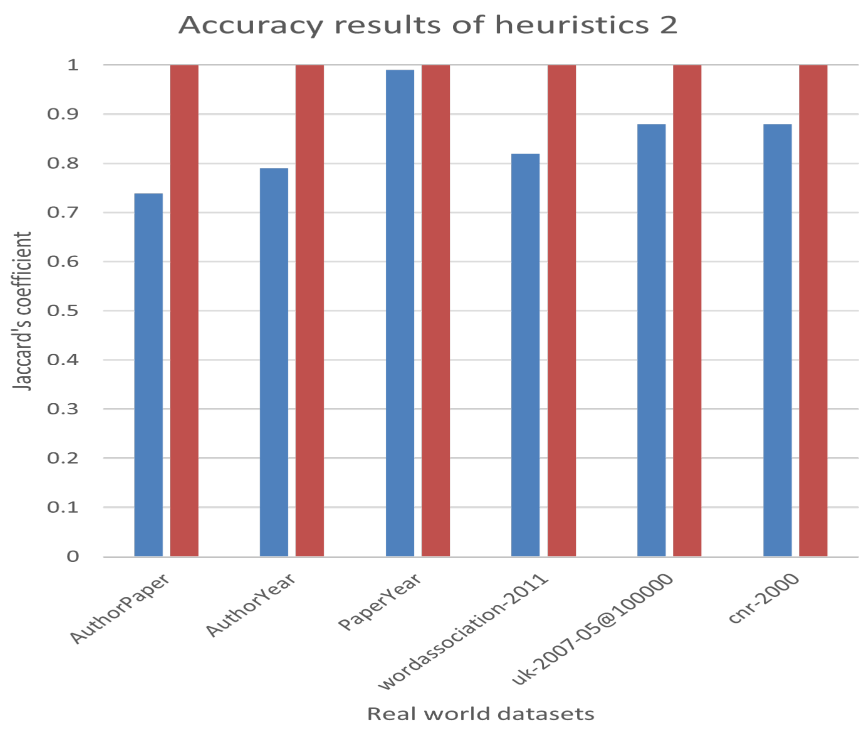


Figure 5.11: Heuristic 2 accuracy for large real-world datasets

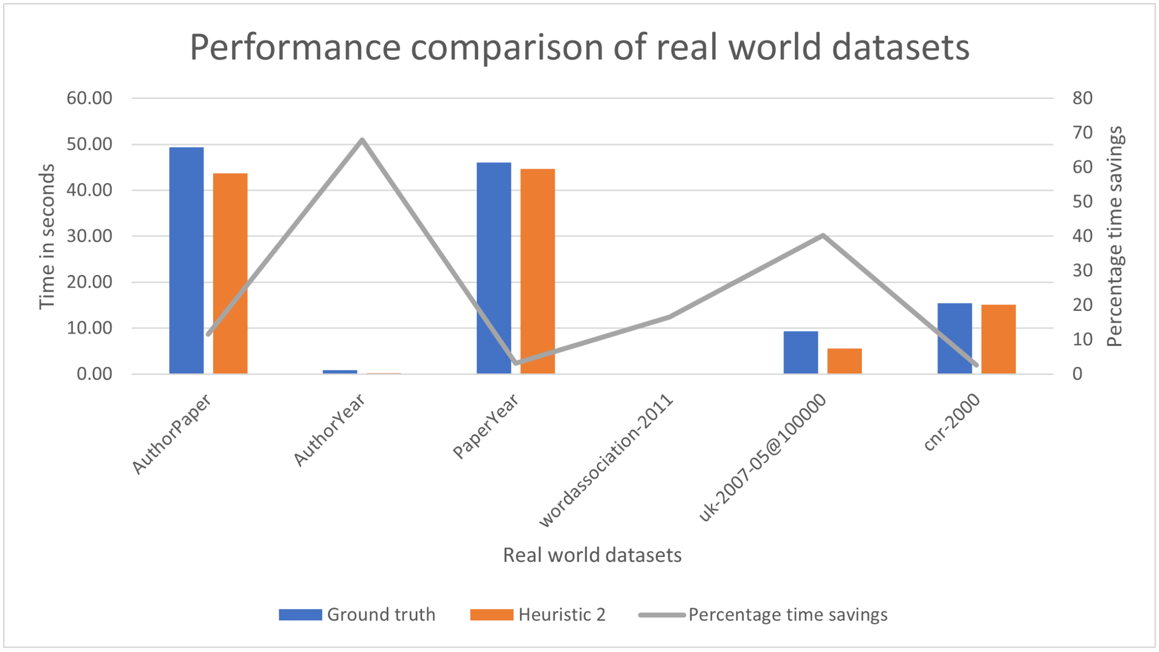


Figure 5.12: Performance comparison for real world datasets using heuristic 2

5.4 Extending the heuristics to k layers

As mentioned earlier, the decoupling-based approach can be extended beyond two layers, figure 5.13 shows the block diagram of the approach using IMDB dataset (Actor, Director and Movies are the 3 layers). To extend beyond two layers, the output format from the composition function should be the same as the input (partial results from layers). In this heuristic the output from each layer consists of nodes and its corresponding degrees, and the total number of nodes and edges. While considering k layers we need to consider the k*(k-1)/2 interlayer edge sets. Each interlayer edge set is considered one at a time. In figure 5.13 for 3 layers there are 3 interlayer edge sets: Actor-Director, Actor-Movie and Director-Movie. Actor-Director interlayer edges are considered while computing Actor-Director layers, and the remaining two interlayer edge sets involving movie layer is considered in the subsequent composition function. Like Actor or Director layers, Movie layer is analyzed individually, and partial results are saved. The results from the Actor-Director layer and partial results from Movie layer are used in the second composition function. In the second composition function two interlayer edge sets - Actor-Movie and Director-Movie are used. The composition algorithm remains the same that it to increment the degree value of the nodes having interlayer edge by 1. This heuristic proposed to save all the nodes and its degree information, so the accuracy is 100% even for three layers. Further, as the results are 100% both **commutative and associative properties hold good** for degree centrality heuristic 2 when used on k-layers.

Figure 5.14 shows the accuracy comparison of IMDB and DBLP datasets for three layers using heuristic 2. The accuracy of IMDB and DBLP is increased to 100% by using heuristic 2. The performance improvement for IMDB and DBLP datasets considering three layers are 14% and 3% respectively. Figure 5.15 shows the performance improvement for these two datasets.

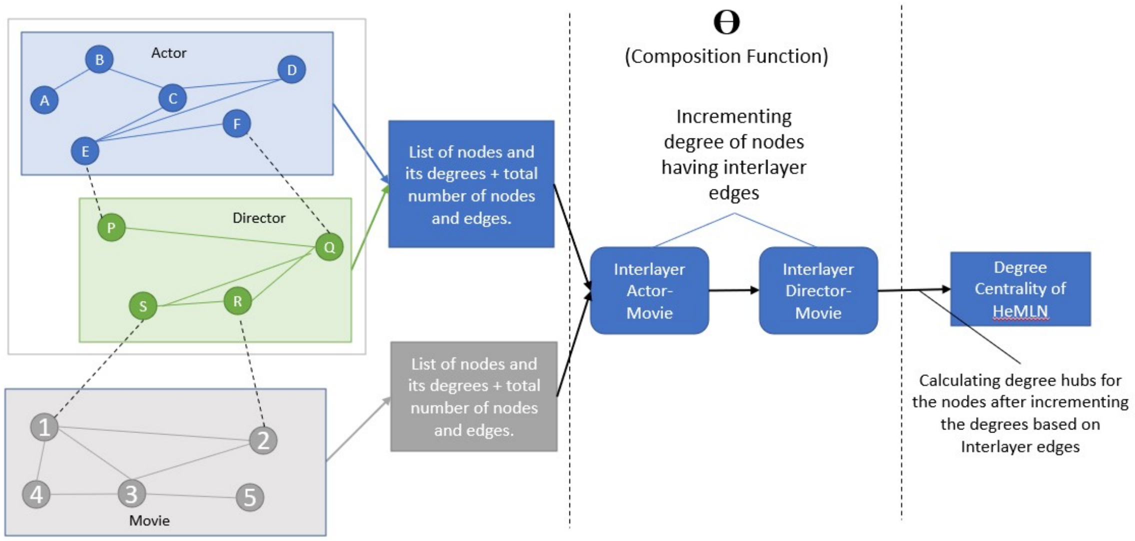

Actor
A
B
C
D
E
F
Director
P
Q
R
S
Movie
1
2
3
4
5
List of nodes and its degrees + total number of nodes and edges.
List of nodes and its degrees + total number of nodes and edges.
ϴ
(Composition Function)
Incrementing degree of nodes having interlayer edges
Interlayer Actor-Movie
Interlayer Director-Movie
Degree Centrality of HeMLN
Calculating degree hubs for the nodes after incrementing the degrees based on Interlayer edges

Figure 5.13: Extending the heuristic to 3 layers

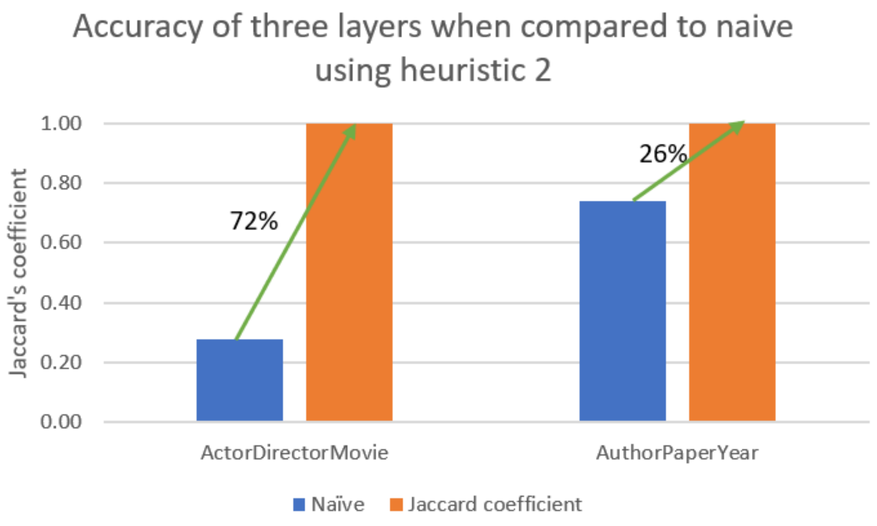


Figure 5.14: Accuracy of IMDB and DBLP datasets considering 3 layers

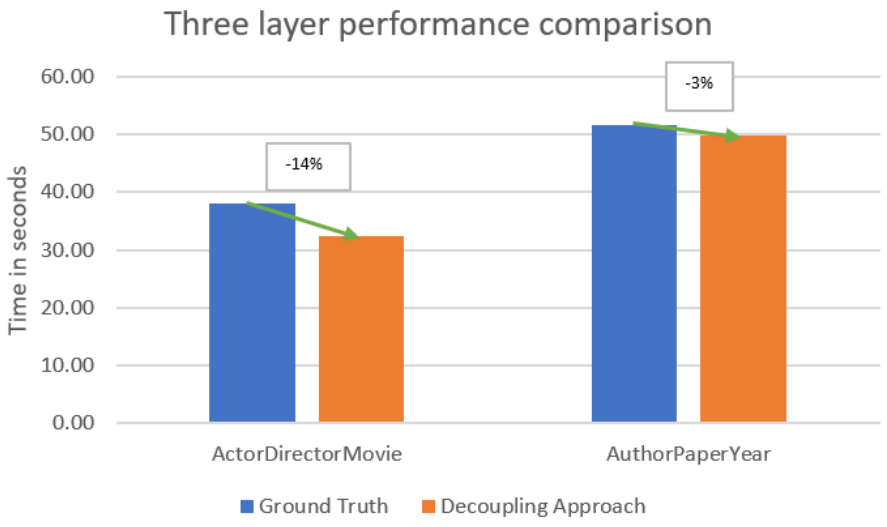


Figure 5.15: Three layer performance improvement for IMDB and DBLP datasets

5.5 Comparing the information retained from each layer and the accuracy

While analyzing the two heuristics we could see that the heuristic 2 retains the maximum information (all the nodes and its degree values), while heuristics 1 keeps only the hub information and this impacts the accuracy. In figure 5.16 we compare the accuracies of the heuristics based on the percentage of information retained. We compare the four different percentages starting from naive (minimum information) then we retain the top 25% of the nodes (sorted by the decreasing order of degree values) along with the hubs from each layer. Then increase the amount of information retained to 50%, 75% and 100% to compare the accuracy. We observed that the accuracy increases as we increase the amount of information retained from each layer.

Figure 5.17 shows the performance comparison of different datasets with respect to the amount of information retained. We can see that as we increase the percentage of information retained the composition time is slightly increased. This increase is because of the additional data that needs to be loaded on to the hash table in the composition function. There is no additional computation involved with increase in the amount of information retained.

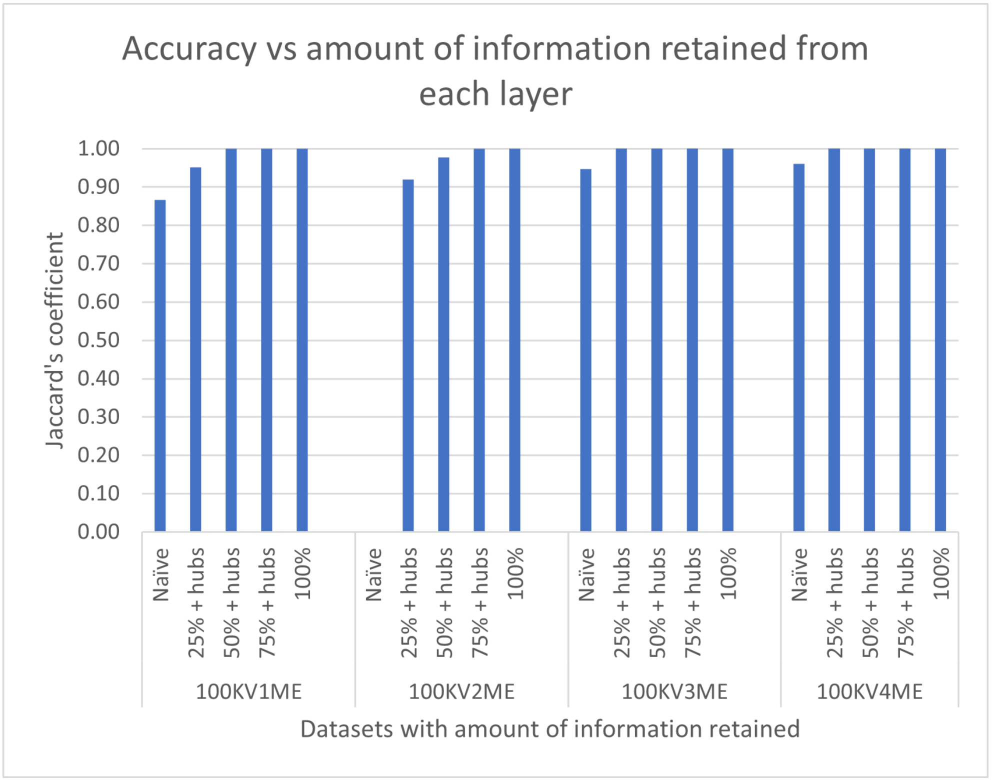


Figure 5.16: Accuracy comparison with respect to the amount of information retained from each layer for different datasets.

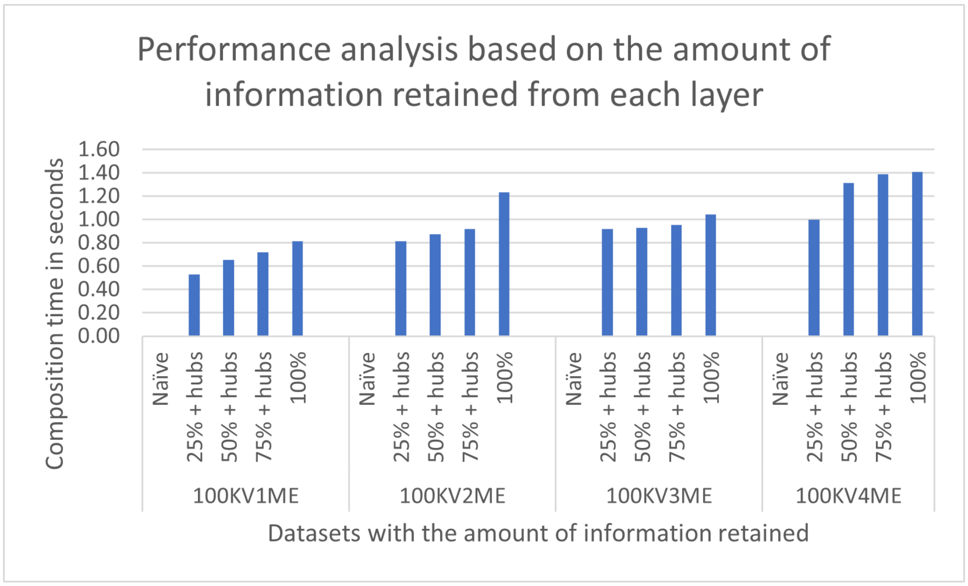


Figure 5.17: Performance comparison with respect to amount of information retained from each layer for different datasets.

CHAPTER 6

BETWEENNESS CENTRALITY

Betweenness centrality, a global measure of graph, is defined as the total number of shortest paths that pass through a node. In this thesis we propose two heuristics to approximate the betweenness centrality measure for HeMLNs. While there are many approximation algorithms for betweenness centrality computation on single graphs, such as using random walk [51] or using sampling [6], we believe this is the first attempt to explore decoupling-based approximation algorithm for a HeMLNs.

6.1 Naive approach

For approximating the betweenness measure using decoupling-based framework, the naive approach uses the union of the betweenness hubs obtained for each layer without any additional computation. The naive approach uses the minimum amount of information from each layer and the composition time is negligible. Hence, it gives maximum improvement in efficiency when compared to the ground truth.

6.2 Ground truth for betweenness centrality computation of HeMLN

Ground truth calculation is similar to that of degree centrality, and is done, by aggregating all the nodes and edges of a HeMLN into a single graph and applying the single graph algorithms to calculate the betweenness centrality hubs for the HeMLNs. The hubs obtained from ground truth are compared with the algorithms that use heuristics during decomposition for accuracy. The time taken by heuristics is expected to be less than that of ground truth time as composition uses partial results and ground truth uses an aggregated large graph.

6.3 Challenges

There are three main challenges for developing heuristic-based decoupling algorithms:

1. To identify the information to be computed and retained from each layer during its processing. Naive approach retains the minimal information. For others, this needs to be decided based on the heuristic applied.
2. To develop a composition algorithm to approximate the number of shortest paths based on the partial results of each layer and using the interlayer edges
3. To verify the accuracy and performance of developed betweenness algorithms on very large datasets with diverse characteristics. As layer and ground truth algorithms use single graphs and BFS for calculating all shortest paths between nodes, the complexity is high($O(V^3)$)

6.4 Intuition

For these algorithms, since we need to estimate or approximate the number of shortest paths through each node of the HeMLN, it seems appropriate to retain some additional information (e.g., the number of shortest paths for each node) for each layer while computing betweenness. Using this information from both layers, our heuristics approximate the number of shortest path through a node of the HeMLN using the inter layer edge information to calculate the betweenness hubs for the HeMLN. This requires analysis of nodes with and without inter layer edges to compute the HeMLN centrality hubs. While developing the heuristics, we realized that in addition to the between hubs for each layer, the degree of a node is likely to play a role as the number of paths going through a node is dependent on its degree.

Consider figure 6.1, layer 1 consists of nodes a through i and layer 2 consists of nodes p,q and r. Table 6.1 gives the calculated values of number of shortest paths that pass through each node, whether it is a betweenness hub or not and the degree values of the node as well. Similarly table 6.2 gives the values for layer 2. With these inputs the intuition is that if a node is a degree hub then the node has more neighbors than the rest of the nodes. More neighbours means more shortest paths passing through the node. This can also be seen in the sample graph figure 6.1b nodes d,f and p are degree hubs as well as betweenness hubs in the HeMLN.

The second intuition is that if a node is not a degree hubs, what is the effect of an interlayer edge on that node? Usually, the nodes that have interlayer edges tend to become the bridge nodes between layers. These bridge nodes have higher number of shortest paths. Also, it is sometimes not enough to just look at the nodes with interlayer edges as in the case of node i and q in figure 6.1b where there is an interlayer edge between them but still they do not come out as hubs in the HeMLNs. So in addition to this, our next promising node characteristic that we can consider is the layer betweenness hubs.

Layer betweenness hubs have high betweenness values in their respective layers. So it is be safe to assume that the layer hubs have higher chance of becoming hubs in entire HeMLN. This is also true in case of sample graph where all the hubs from layer 1 - d,e,f and layer 2 - p are the hubs in the HeMLN. So considering all the above mentioned intuitions this thesis proposes two heuristics to approximate the betweenness measure of the entire HeMLN.

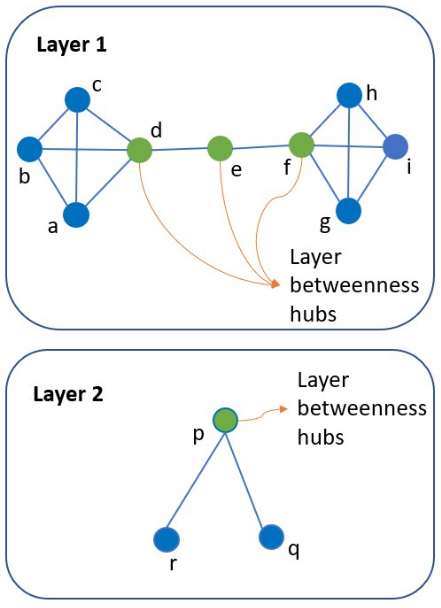
As per the example in figure 6.1, the probability of nodes with high degree value tend to become betweenness hubs as well because of the high connectivity.

Nodes	Number of SP	Layer hubs	Degree Values
a	0	No	3
b	0	No	3
c	0	No	3
d	15	Yes	4
e	16	Yes	2
f	15	Yes	4
g	0	No	3
h	0	No	3
i	0	No	3

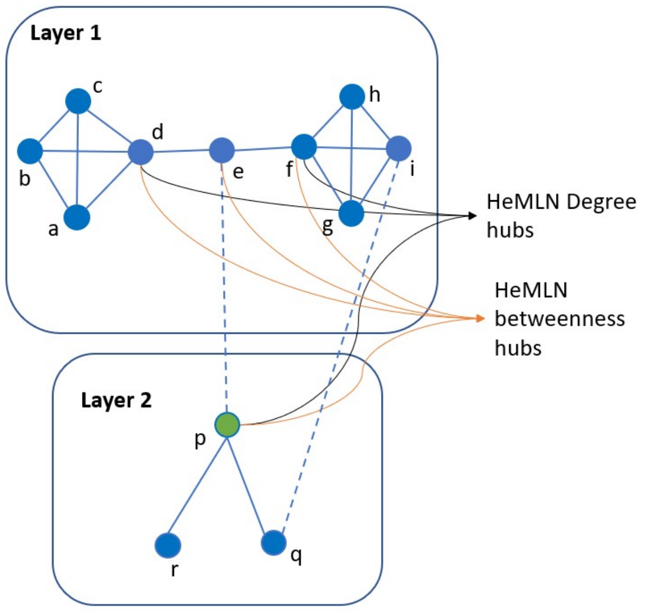
Table 6.1: Characteristics of nodes in layer 1

Nodes	Number of SP	Layer hubs	Degree Values
p	1	Yes	2
q	0	No	1
r	0	No	1

Table 6.2: Characteristics of nodes in layer 2



(a) Sample graph layers



(b) Complete HeMLN

Figure 6.1: A sample graph to illustrate the intuition behind approximating betweenness centrality measure.

Similarly, the nodes with interlayer edges act as bridge nodes between layers (bridge nodes means high betweenness measure) so there is high probability that even these nodes might be the betweenness centrality hubs in the HeMLNs. Along with these if the node is already a betweenness hub in the layer then the node is centrally placed in the layer (these nodes already have high betweenness values) so there is a high probability that these might become the hubs of the entire HeMLNs. Based on this intuition, we propose two heuristics in this thesis and compare the results with the ground truth.

6.5 Degree-dominant heuristic

As discussed in the earlier section, the parameters that can be considered while approximating betweenness centrality measure are the degree values, betweenness measure in the layer, and the nodes with interlayer edges. When each layer is analyzed individually, the degree and its betweenness measure are calculated. Based on the average betweenness measure in each layer the layer hubs are identified. The entire output is saved for further computation in the composition function.

Using the partial results from the layers, along with the interlayer edges based on the intuition the following composition algorithm is proposed and its decision tree is shown in figure 6.2. This decision tree is applied for each node in the HeMLN.

As discussed in the intuition the nodes with $\geq$ average degree value have more edges than the rest of the nodes so their betweenness measure tends to be high. With this assumption, for each node, we first check if the node has a degree value higher than that of the average if so, then the maximum number of shortest paths (of all the nodes having interlayer edges in the other layer) is added to this node for approximation.

Secondly, the nodes are checked for interlayer edges, these nodes have a high probability of becoming bridge nodes between layers. Further if the node is already a hub in the layer the probability of that node becoming a hub in HeMLN is also high so this heuristic proposes to add an average number of shortest paths (average shortest path of the nodes in another layer that have interlayer edges) to these nodes. For the rest of the nodes we add the minimum number of shortest paths. As per the decision tree there are three values that are used in this heuristic that is maximum, average and the minimum number of shortest paths.

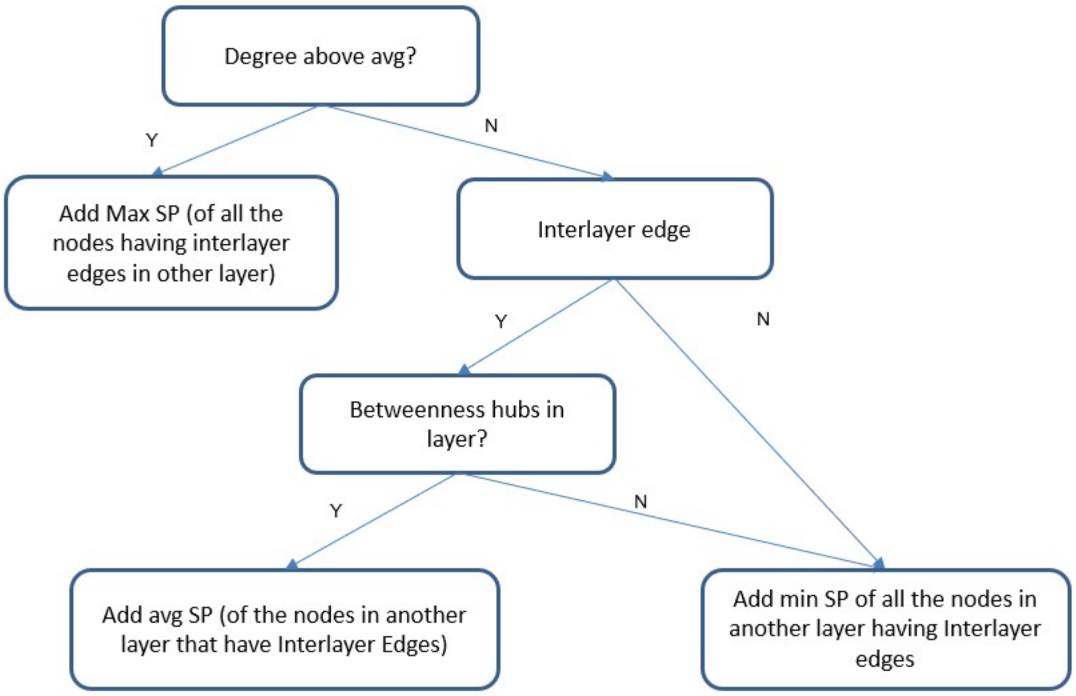

Degree above avg?
Y
N
Add Max SP (of all the nodes having interlayer edges in other layer)
Interlayer edge
Y
N
Betweenness hubs in layer?
Y
N
Add avg SP (of the nodes in another layer that have Interlayer Edges)
Add min SP of all the nodes in another layer having Interlayer edges

Figure 6.2: Degree-dominant heuristic decision tree

Composition algorithm for the Degree-dominant heuristic is shown in Algorithm 2 where the inputs, *ni_nodes* and *nj_nodes* consists of nodes of their respective layers.

The *interlayer_edges* represents the interlayer edge set. The *shortest_path* represent the number of shortest paths that pass through each node. $NodeWithIE$ consists of list of nodes that have interlayer edges (this can be obtained through the interlayer edge set). The algorithm follows the decision tree that is discussed earlier.

The above heuristic is initially tested on smaller datasets starting from 100 nodes to up to 10,000 nodes. Also, this heuristic is tested on word association dataset. Figure 6.3 shows the accuracy of degree-dominant heuristic on various datasets. Naive accuracy is also plotted in blue color for comparison.

It can be noted from the figure 6.3 that the degree-dominant heuristic has performed better in majority of the datasets that were tested. There was a **maximum of 35% increase in accuracy**. However in the last three datasets the accuracy has dropped below the naive approach.

Figure 6.4 shows the precision plot for the same datasets. Even here the precision has increased for first five datasets and later drops significantly.

Further the same heuristic was tested on large datasets. Figure 6.5 shows the accuracy comparison on large datasets. It can be seen that the accuracy results are further below the naive approach for all these datasets as well. Figure 6.6 shows the precision comparison on large datasets, even the precision is lower compared to naive approach. As precision is low we can infer that there are more false positives in the list of hubs identified by the heuristic. False positives need to be reduced to improve the accuracy.

6.5.1 Analysis of Degree-dominant heuristic:

Degree-dominant heuristic provided good precision in some datasets where there was an increase of 20% and reached up to 90% precision. This is promising as there was a good increase in precision when compared to naive approach. However the same

Algorithm 2 Composition algorithm for betweenness centrality degree-dominant heuristic

INPUT:

$ni\_nodes = \{V_{i1}, V_{i2}, ..., V_{in1}\}$

$nj\_nodes = \{V_{j1}, V_{j2}..., V_{jn2}\}$

$interlayer\_edges = \{(V_{i1}, V_{j1}), (V_{i2}, V_{j2}), ...\}$

$shortest\_path = \{V_{i1} : BC_{i1}, V_{i2} : BC_{i2}...V_{in} : BC_{in}, V_{j1} : BC_{j1}, V_{j2} : BC_{j2}...V_{jn} : BC_{jn},\}$

ALGORITHM:

```
1:  for $node_i, node_j \in interlayer\_edges$ do
2:      if $MinShortestPathLayer_i > shortest\_path[node_i]$ then
3:          $MinShortestPathLayer_i = shortest\_path[node_i]$
4:      end if
5:      if $MinShortestPathLayer_j > shortest\_path[node_j]$ then
6:          $MinShortestPathLayer_j = shortest\_path[node_j]$
7:      end if
8:      if $MaxShortestPathLayer_i < shortest\_path[node_i]$ then
9:          $MaxShortestPathLayer_i = shortest\_path[node_i]$
10:     end if
11:     if $MaxShortestPathLayer_j < shortest\_path[node_j]$ then
12:         $MaxShortestPathLayer_i = shortest\_path[node_j]$
13:     end if
14: end for
15: $NodesWithIE = \{V_{i1}, V_{i2}, ...V_{j1}, V_{j2}, ...\}$
16: find the average number of SP of the nodes $\in NodesWithIE$ in each layer
    ($AvgShortestPathLayer_i$, $AvgShortestPathLayer_j$)
```

```
let $Betweenness_{HeMLN}$ consists of all the betweenness values obtained from layers
for NodeID $\in Layer_i \cup Layer_j$ do
    if NodeID in $ni\_nodes$ then
        if DegreeValue of NodeID$>$ Average degree then
            add $MaxShortestPathLayer_j$ to $Betweenness_{HeMLN}[NodeID]$
        else
            if NodeID $\in$ NodesWithIE and NodeID $\in$ layer betweenness hub then
                add $AvgShortestPathLayer_j$ to $Betweenness_{HeMLN}[NodeID]$
            else
                add $MinShortestPathLayer_j$ to $Betweenness_{HeMLN}[NodeID]$
            end if
        end if
    else
        if DegreeValue $>$ Average degree then
            add $MaxShortestPathLayer_i$ to $Betweenness_{HeMLN}[NodeID]$
        else
            if NodeID $\in$ NodesWithIE and NodeID $\in$ layer betweenness hub then
                add $AvgShortestPathLayer_i$ to $Betweenness_{HeMLN}[NodeID]$
            else
                add $MinShortestPathLayer_i$ to $Betweenness_{HeMLN}[NodeID]$
            end if
        end if
    end if
end for
```

```
41: calculate the new average number of shortest paths and assign it to $AverageNoSP$
42: for NodeID $\in ni\_nodes \cup nj\_nodes$ do
43:     if $Betweenness_{HeMLN}[NodeID] \geq AverageNoSP$ then
44:         add NodeID to $BetweennessHubs$ list
45:     end if
46: end for
```

OUTPUT:

$BetweennessHubs = \{V_1, V_5....\}$

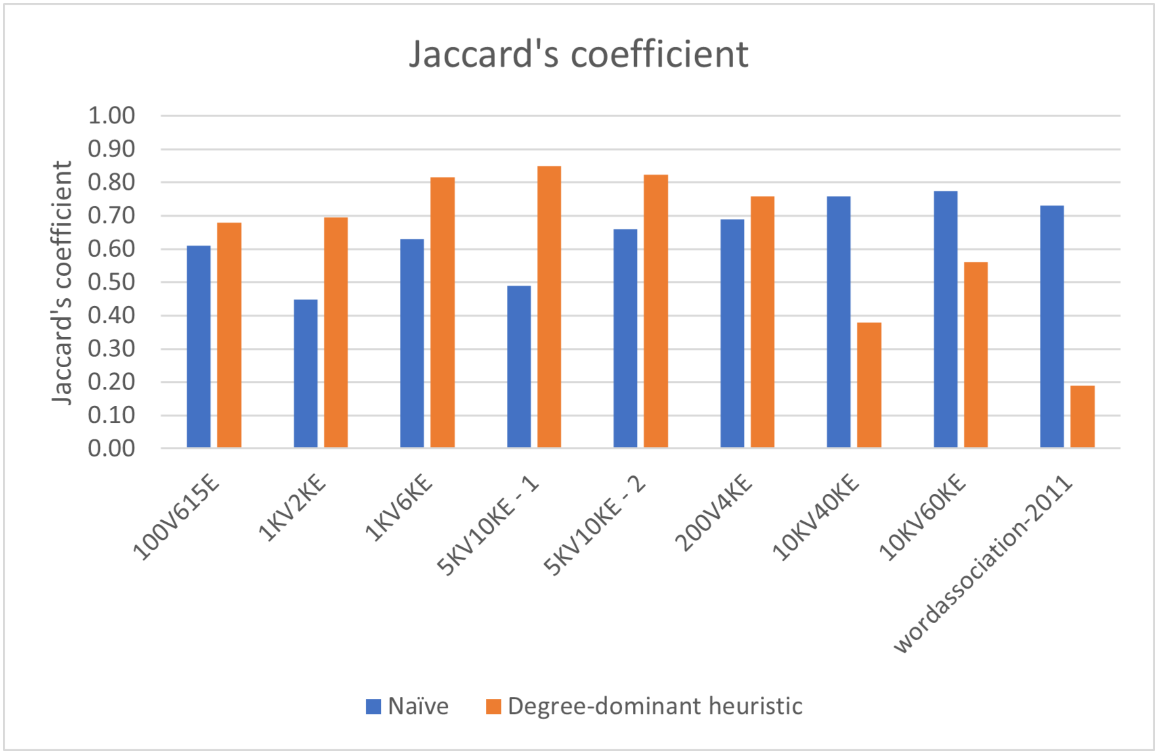


Figure 6.3: Jaccard's coeffecient plot for various small datasets using degree-dominant heuristic.

heuristic gave precision lesser than that of naive approach for the last four datasets and for the larger datasets. It can be noted that this heuristic is not consistent for all datasets with respect to accuracy. Also, for word association dataset the accuracy

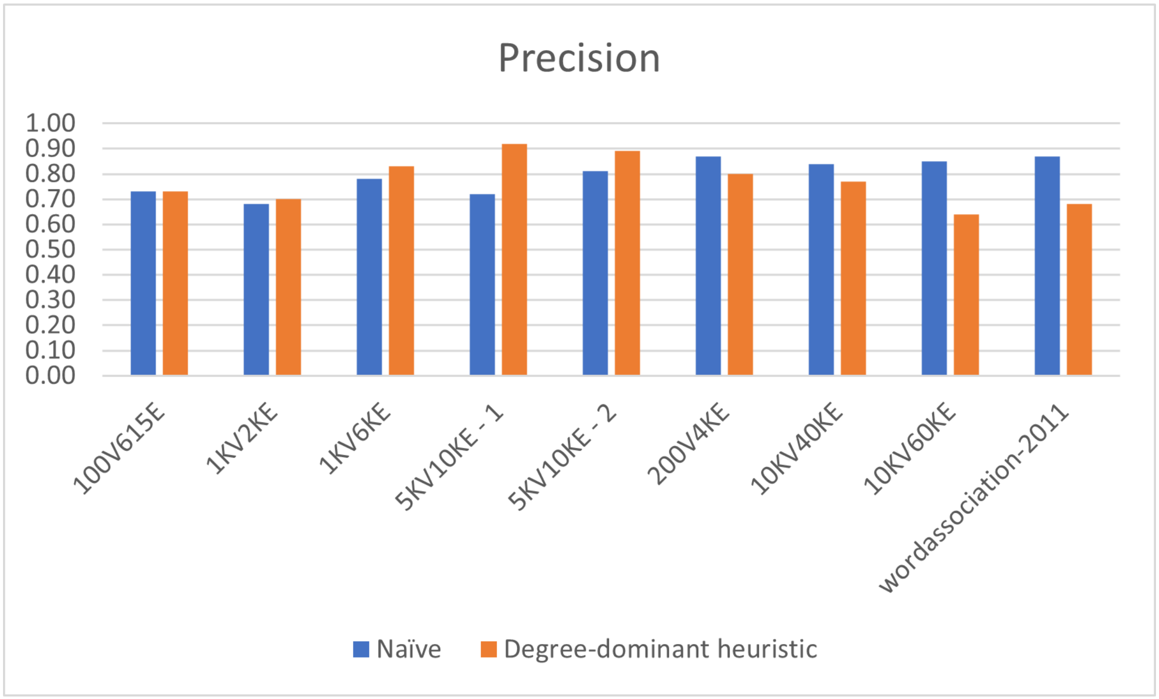


Figure 6.4: Precision plot for various small datasets using degree-dominant heuristic.

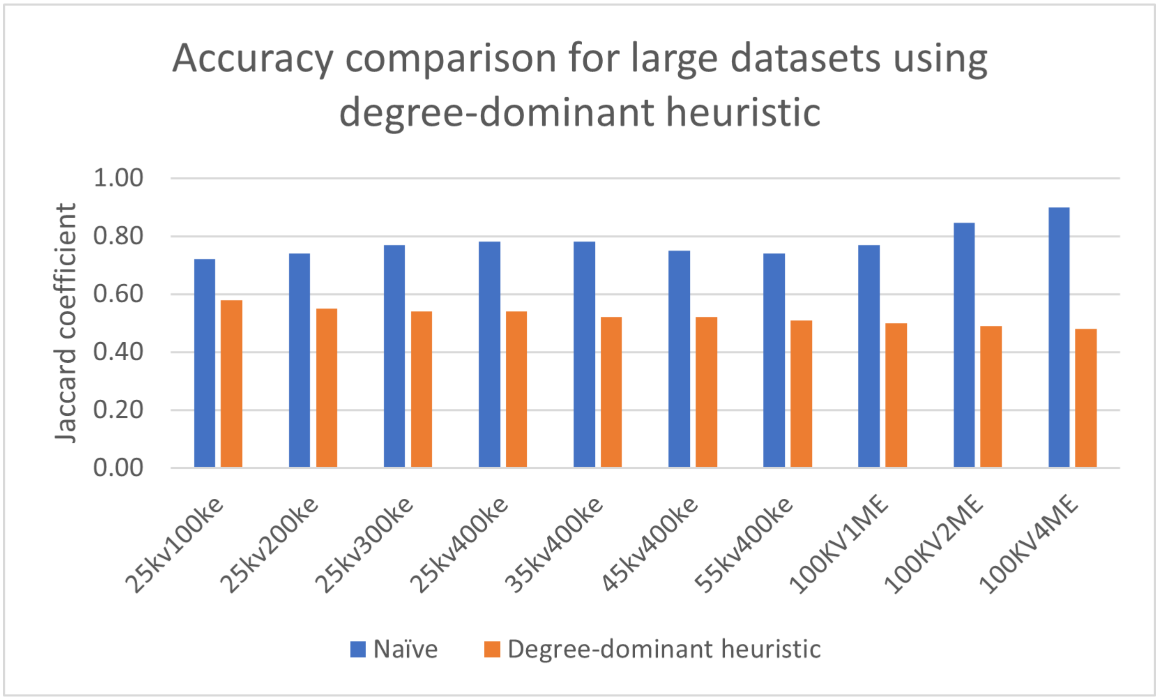


Figure 6.5: Accuracy comparison of degree-dominant heuristic on large datasets

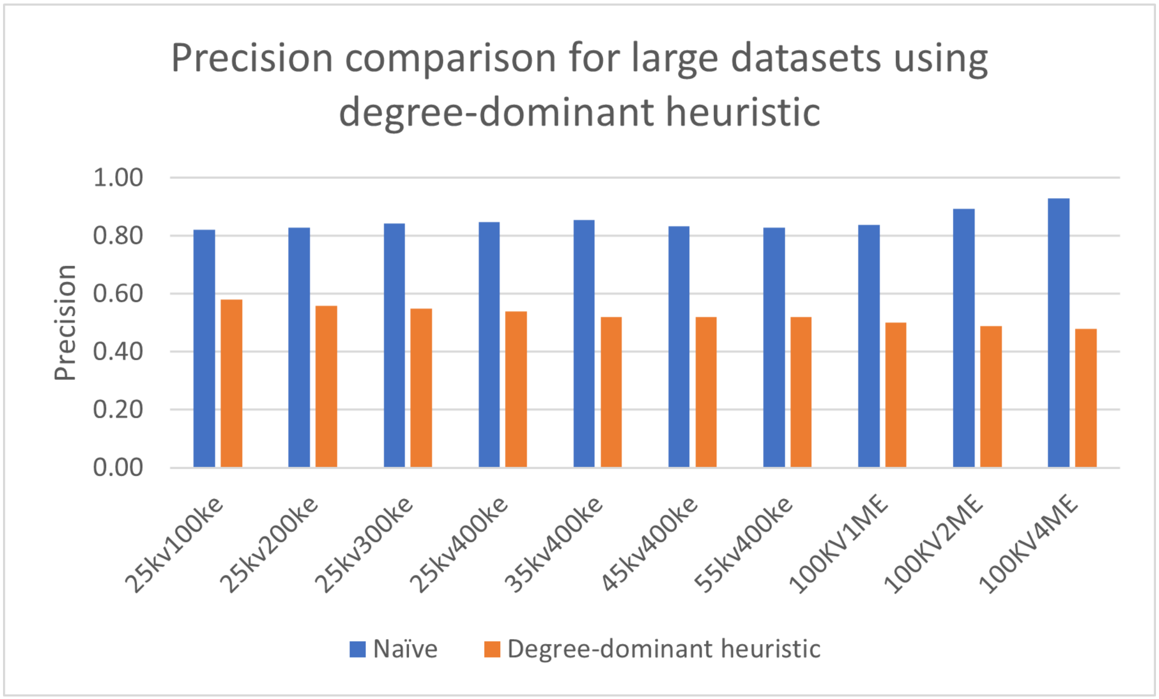


Figure 6.6: Precision comparison of degree-dominant heuristic on large datasets

dropped to almost 19% when the naive was 86% accurate. In the next heuristic we try to improve the accuracy by improving the precision.

6.6 Degree-betweenness heuristic

In degree-betweenness heuristic the same parameters are analysed in each layer, however with the change in composition algorithm this heuristic proposes to reduce the inconsistency with the accuracy. In degree-betweenness heuristic we propose to include an if condition where the node should be a degree hub also a hub in the layer and should have an interlayer edge. If all these three conditions are satisfied the node can have a very high chance of becoming a hub in the entire HeMLN. satisfying all three conditions means the node has more edges than rest of the nodes, it has an interlayer edge so it might be a bridge node also if it is hub in the layer then its betweenness measure is also high. Figure 6.7 shows the decision tree of degree-betweenness heuristic.

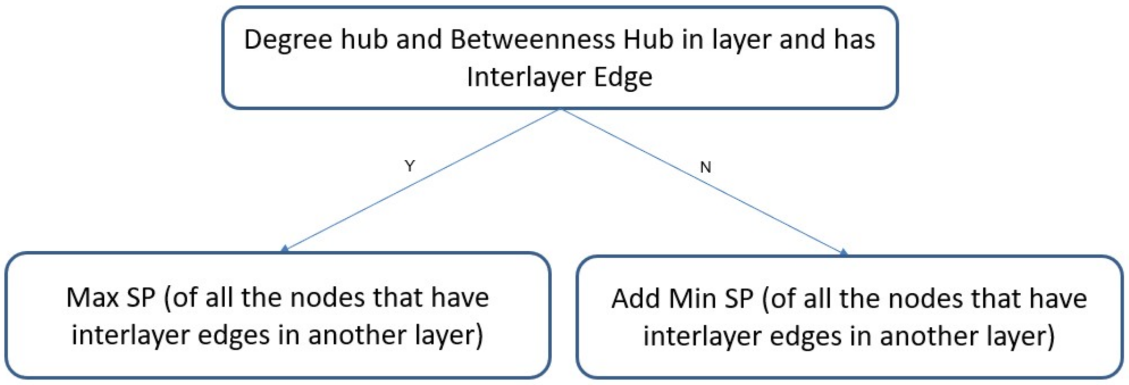

Degree hub and Betweenness Hub in layer and has Interlayer Edge
Y
N
Max SP (of all the nodes that have interlayer edges in another layer)
Add Min SP (of all the nodes that have interlayer edges in another layer)

Figure 6.7: Decision tree for degree-betweenness heuristic

Algorithm 3 explains in detail the composition algorithm for the degree-betweenness heuristic. As mentioned earlier the inputs, *ni_nodes* and *nj_nodes* consists of nodes of their respective layers. The *interlayer_edges* represents the interlayer edge set. The *shortest_path* represent the number of shortest paths that pass through each node. $NodeWithIE$ consists of list of nodes that have interlayer edges (this can be obtained through the interlayer edge set). The algorithm follows the decision tree that is discussed earlier.

Figure 6.8 compares the accuracy of naive approach, degree-dominant and degree-betweenness heuristics on the same set of datasets. The accuracy improvements need to be understood with respect to the naive approach. In the datasets that were tested, **degree-betweenness heuristic accuracy is consistently better than that of naive approach as compared to** . In the datasets that were tested the **accuracy increase was in the range of 1% to 22%** over the naive approach. This validates our intuition and our estimation is better than degree-dominant heuristic across datasets.

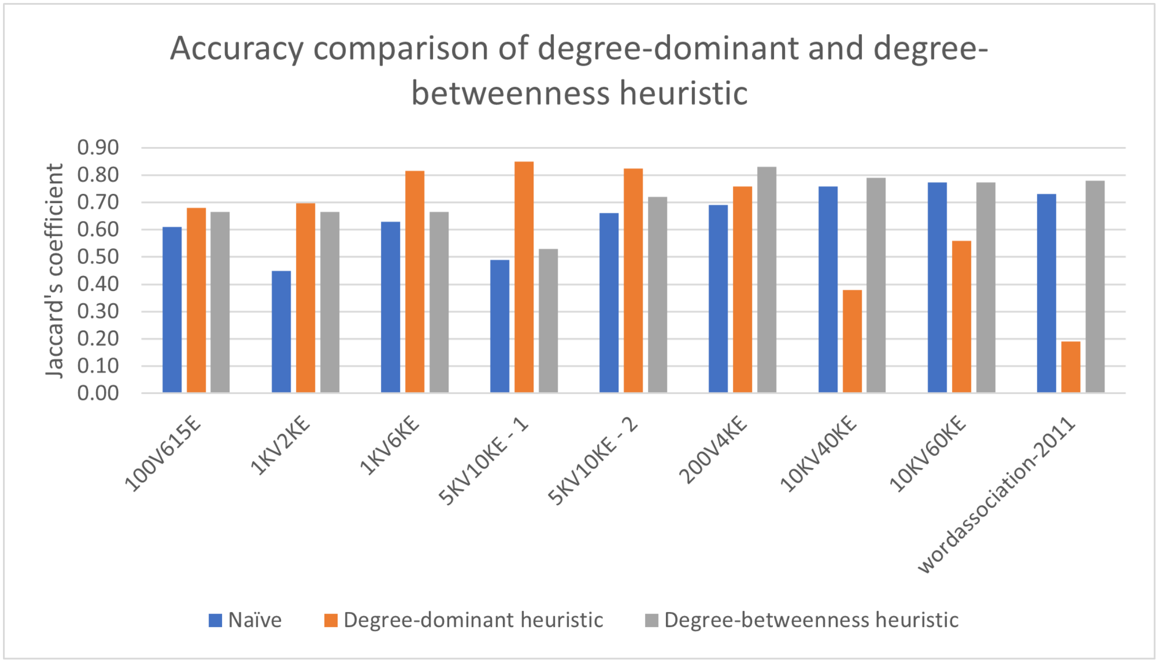


Figure 6.8: Comparing the accuracy of naive approach, degree-dominant heuristic and degree-betweenness heuristic

Algorithm 3 Composition algorithm for betweenness centrality degree-betweenness heuristic

INPUT:

$ni\_nodes = \{V_{i1}, V_{i2}, ..., V_{in1}\}$

$nj\_nodes = \{V_{j1}, V_{j2}..., V_{jn2}\}$

$interlayer\_edges = \{(V_{i1}, V_{j1}), (V_{i2}, V_{j2}), ...\}$

$shortest\_path = \{V_{i1} : BC_{i1}, V_{i2} : BC_{i2}...V_{in} : BC_{in}, V_{j1} : BC_{j1}, V_{j2} : BC_{j2}...V_{jn} : BC_{jn}, \}$

ALGORITHM:

```
 1: for $node_i, node_j \in interlayer\_edges$ do
 2:     if $MinShortestPathLayer_i > shortest\_path[node_i]$ then
 3:         $MinShortestPathLayer_i = shortest\_path[node_i]$
 4:     end if
 5:     if $MinShortestPathLayer_j > shortest\_path[node_j]$ then
 6:         $MinShortestPathLayer_j = shortest\_path[node_j]$
 7:     end if
 8:     if $MaxShortestPathLayer_i < shortest\_path[node_i]$ then
 9:         $MaxShortestPathLayer_i = shortest\_path[node_i]$
10:     end if
11:     if $MaxShortestPathLayer_j < shortest\_path[node_j]$ then
12:         $MaxShortestPathLayer_i = shortest\_path[node_j]$
13:     end if
14: end for
15: let $NodesWithIE = \{V_{i1}, V_{i2}, ...V_{j1}, V_{j2}, ...\}$
16: find the average number of SP of the nodes $\in$ $NodesWithIE$ in each layer
    ($AvgShortestPathLayer_i$, $AvgShortestPathLayer_j$)
```

```
17: let $Betweenness_{HeMLN}$ consists of all the betweenness values obtained from layers
18: **for** NodeID $\in$ $ni\_nodes$ **do**
19:     **if** (DegreeValue > Average degree) and (NodeID $\in$ NodesWithIE) and (NodeID
        $\in$ layer betweenness hub) **then**
20:         add $MaxShortestPathLayer_j$ to $Betweenness_{HeMLN}[NodeID]$
21:     **else**
22:         add $MinShortestPathLayer_j$ to $Betweenness_{HeMLN}[NodeID]$
23:     **end if**
24: **end for**
25: **for** NodeID $\in$ $nj\_nodes$ **do**
26:     **if** (DegreeValue > Average degree) and (NodeID $\in$ NodesWithIE) and (NodeID
        $\in$ layer betweenness hub) **then**
27:         add $MaxShortestPathLayer_i$ to $Betweenness_{HeMLN}[NodeID]$
28:     **else**
29:         add $MinShortestPathLayer_i$ to $Betweenness_{HeMLN}[NodeID]$
30:     **end if**
31: **end for**
32: calculate the new average number of shortest paths and assign it to $AverageNoSP$
33: **for** NodeID $\in$ $ni\_nodes \cup nj\_nodes$ **do**
34:     **if** $Betweenness_{HeMLN}[NodeID] \geq AverageNoSP$ **then**
35:         add NodeID to $BetweennessHubs$ list
36:     **end if**
37: **end for**
**OUTPUT:**
$BetweennessHubs = V_1, V_5....$
```

Figure 6.9 compares the precision of naive approach, degree-dominant and degree-betweenness heuristics. Degree-betweenness heuristic precision is consistently better than that of naive approach. For these datasets **precision increase was in the range of 6% to 18%**. More improtantly degree-betweenness heuristic is giving good results for the datasets where degree-dominant heuristic performed poorly. Degree-betweenness heuristic is promising and will be further tested on very large datasets as part of experiments.

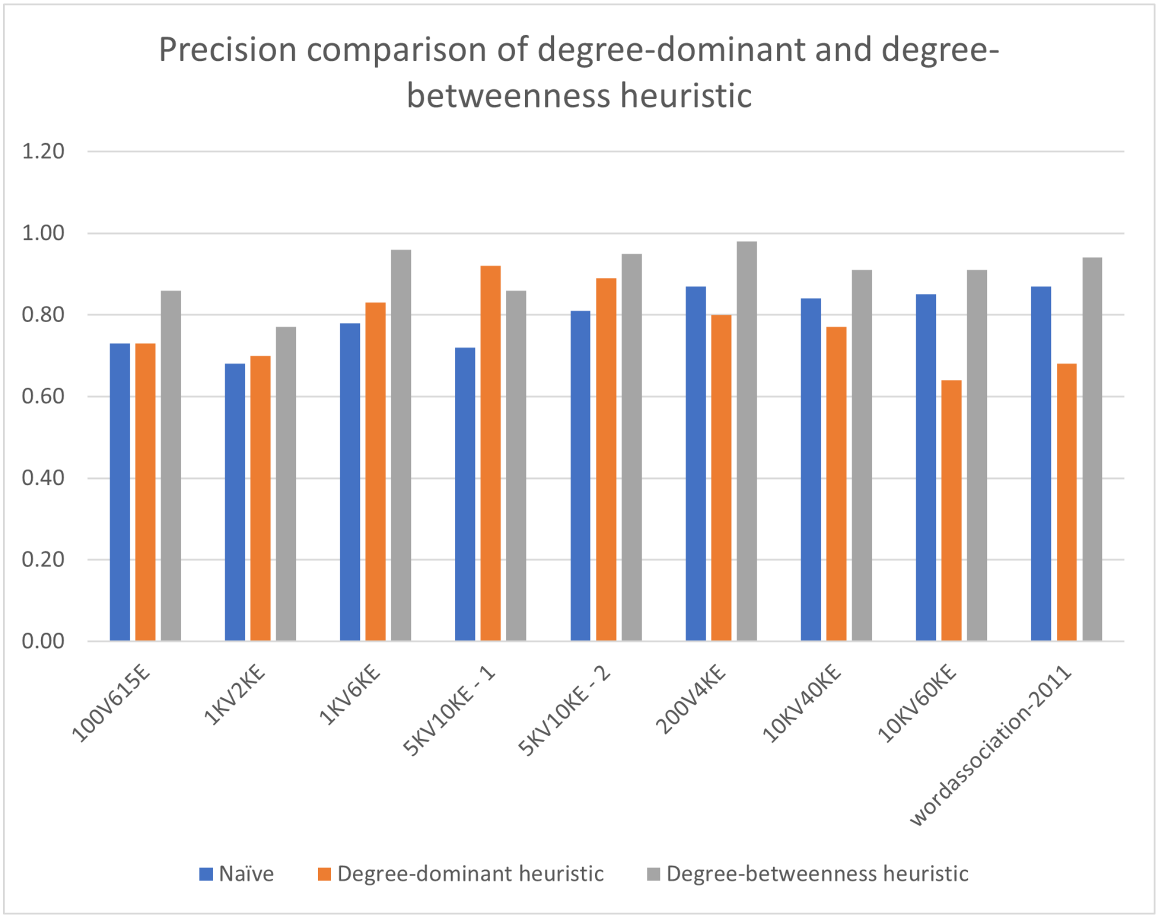


Figure 6.9: Comparing the accuracy of naive approach, degree-dominant heuristic and degree-betweenness heuristic

As discussed, betweenness centrality is computationally expensive. The maximum amount of time taken for calculating betweenness centrality is mainly consumed by the all pair shortest path(APSP) computation. Being a main memory algorithm, to analyze larger datasets, we will have to reduce the APSP time by parallelizing the APSP computation. This is possible as APSP can be parallelized. In this thesis we use **High performance computing (HPC)** for computing the betweenness measure on larger datasets. Since BFS is used to compute APSP, with multiprocessing we can compute it in parallel. Figure 6.10 shows the overall algorithm to compute APSP in parallel using 100 processes at a time. In this algorithm each process computes the BFS keeping one node as the root, similarly we can initiate 100 processes at a time. This can be increased to the total number of nodes based on the system capabilities. For each process the input is a graph (adjacency matrix) and the output is the number of shortest paths that pass through each node. The output from each process is stored in a stack. While each process produces an output, at the end of all the processes the output is combined and the process is repeated until all the shortest paths are computed. This algorithm is used in analysis function and ground truth for calculating the betweenness centrality values for large datasets.

Degree-betweenness heuristic was run on larger datasets up to 4 million edges. Additional information about the dataset characteristics is discussed in chapter 4. Figure 6.11 shows the results of degree-betweenness heuristic on larger datasets starting from 25,000 nodes and 100,000 edges to 100,000 nodes and 4 million edges. Degree-betweenness heuristic is consistent with the results and it is always **better than the naive approach**.

Figure 6.12 shows the performance of degree-betweenness heuristic on larger datasets. The precision tends to follow the naive accuracy. For all the datasets

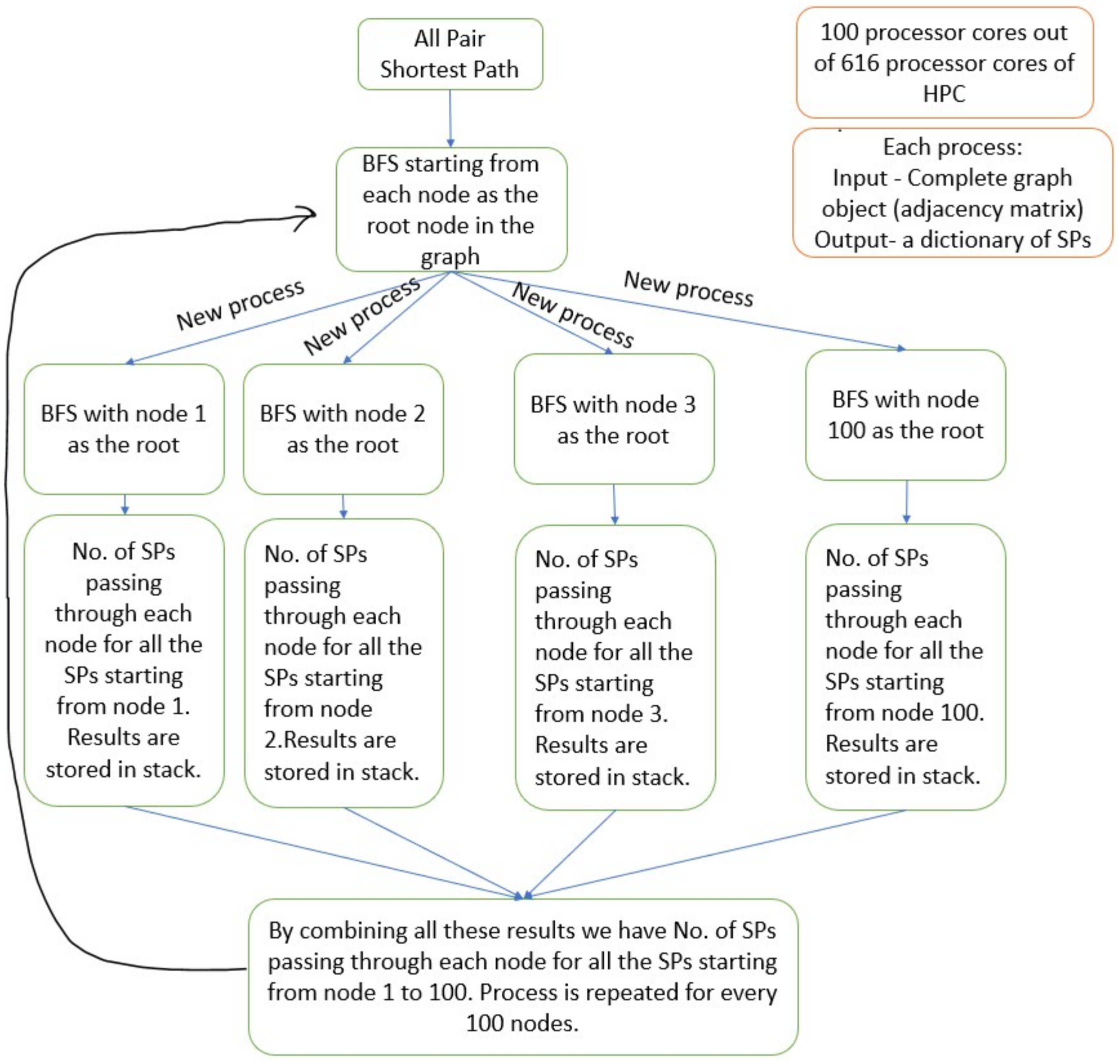

All Pair Shortest Path
100 processor cores out of 616 processor cores of HPC
Each process: Input - Complete graph object (adjacency matrix) Output- a dictionary of SPs
BFS starting from each node as the root node in the graph
New process
New process
New process
New process
BFS with node 1 as the root
BFS with node 2 as the root
BFS with node 3 as the root
BFS with node 100 as the root
No. of SPs passing through each node for all the SPs starting from node 1. Results are stored in stack.
No. of SPs passing through each node for all the SPs starting from node 2.Results are stored in stack.
No. of SPs passing through each node for all the SPs starting from node 3. Results are stored in stack.
No. of SPs passing through each node for all the SPs starting from node 100. Results are stored in stack.
By combining all these results we have No. of SPs passing through each node for all the SPs starting from node 1 to 100. Process is repeated for every 100 nodes.

Figure 6.10: Betweenness centrality calculations on HPC

the **degree-betweenness heuristic precision is more than that of naive approach**.

Figure 6.13 compares the time taken by ground truth, naive and degree-betweenness heuristic. degree-betweenness heuristic with decoupling-based composition performs significantly better than the ground truth but takes more time than naive approach. For the datasets that were tested the **% time savings is in the range of 29% to 69%**.

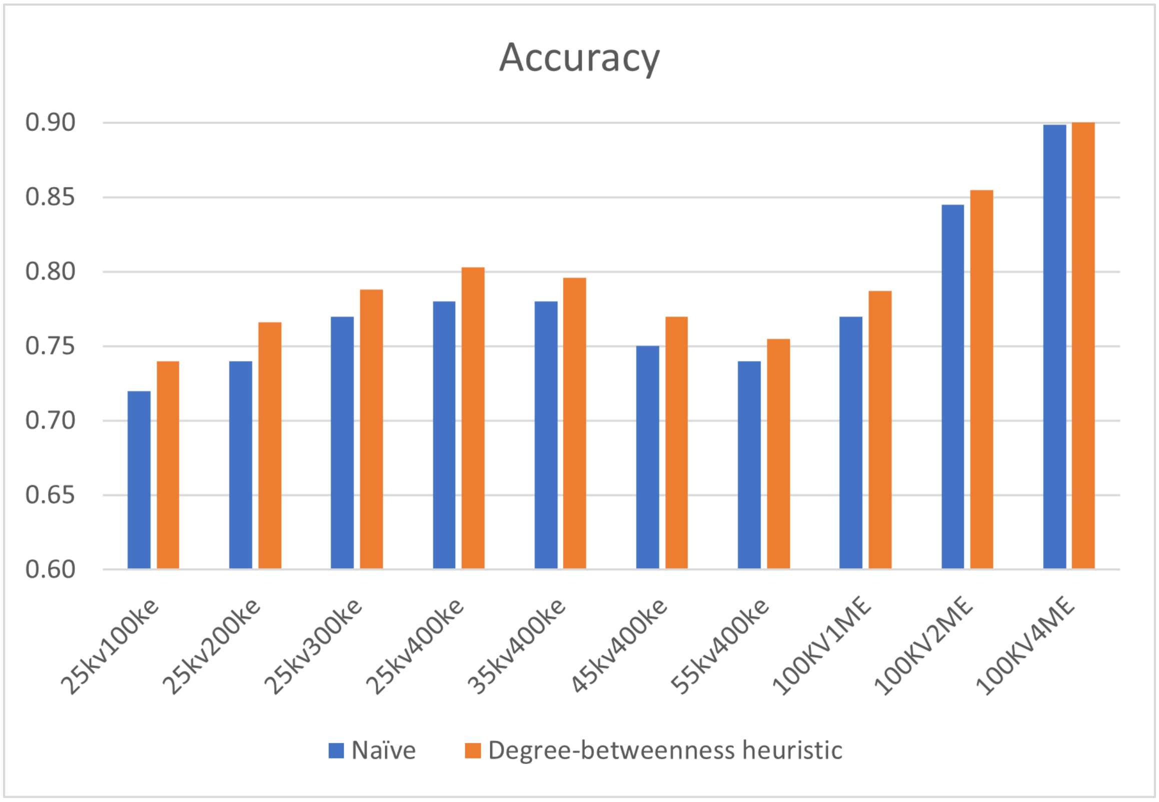


Figure 6.11: Accuracy comparison of degree-betweenness heuristic on larger datasets

The performance improvement mainly depends on the node distribution among the layers. Figure 6.14 shows the time savings among various node distributions. Layers with 50:50 node distribution (50% nodes in each layer) have **the highest % time savings in this heuristic is around 90%** while the HeMLN with 90:10 (90% nodes in one layer and 10% nodes in another layer) has the least % time savings. The **% time savings reduces as the difference in the node distribution increases**. This is because the heuristic time depends on the maximum time taken by the layers. If the node distribution is equally distributed across layers then both the layers take almost equal amount of time and we would achieve maximum parallelization. If all the nodes are present in one layer then the maximum time is equal to the time taken by the larger layer which is almost equal to the ground truth.

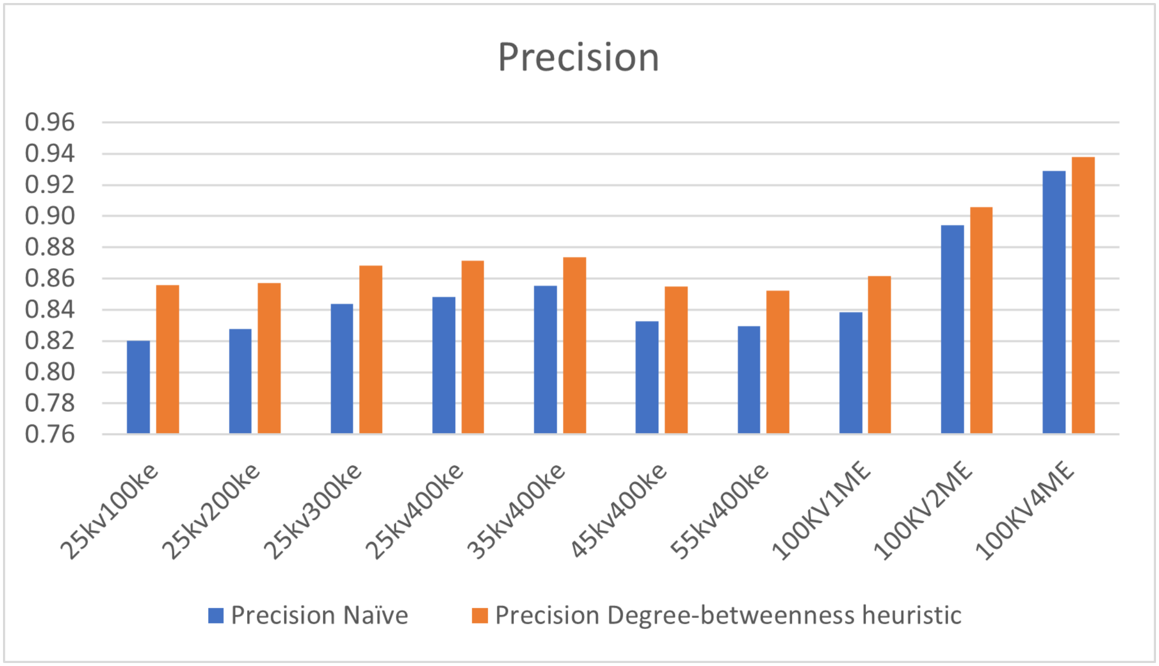


Figure 6.12: Precision comparison of degree-betweenness heuristic on larger datasets

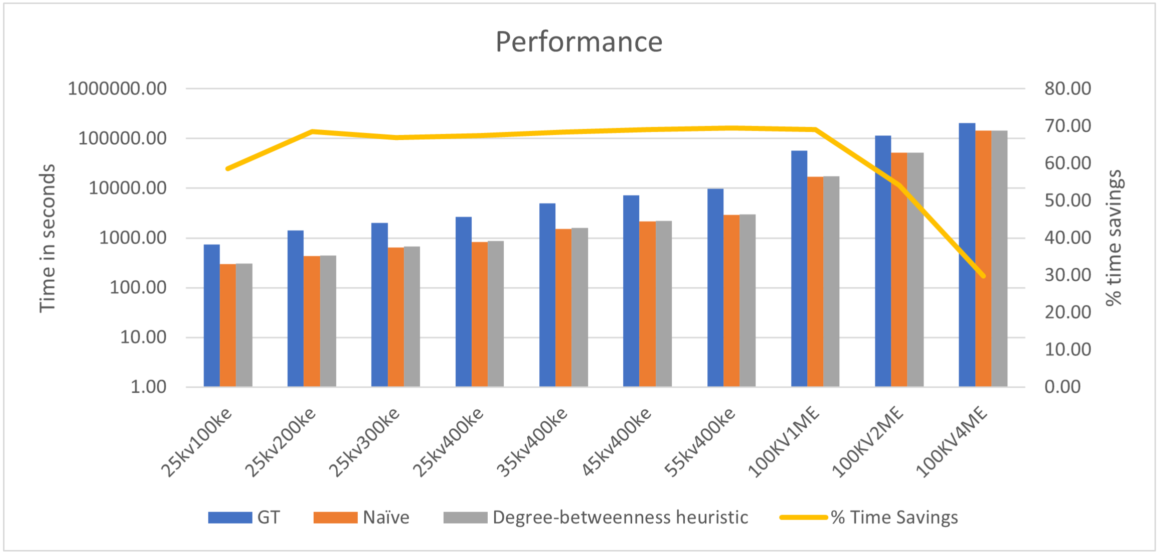


Figure 6.13: Performance of degree-betweenness heuristic on larger datasets

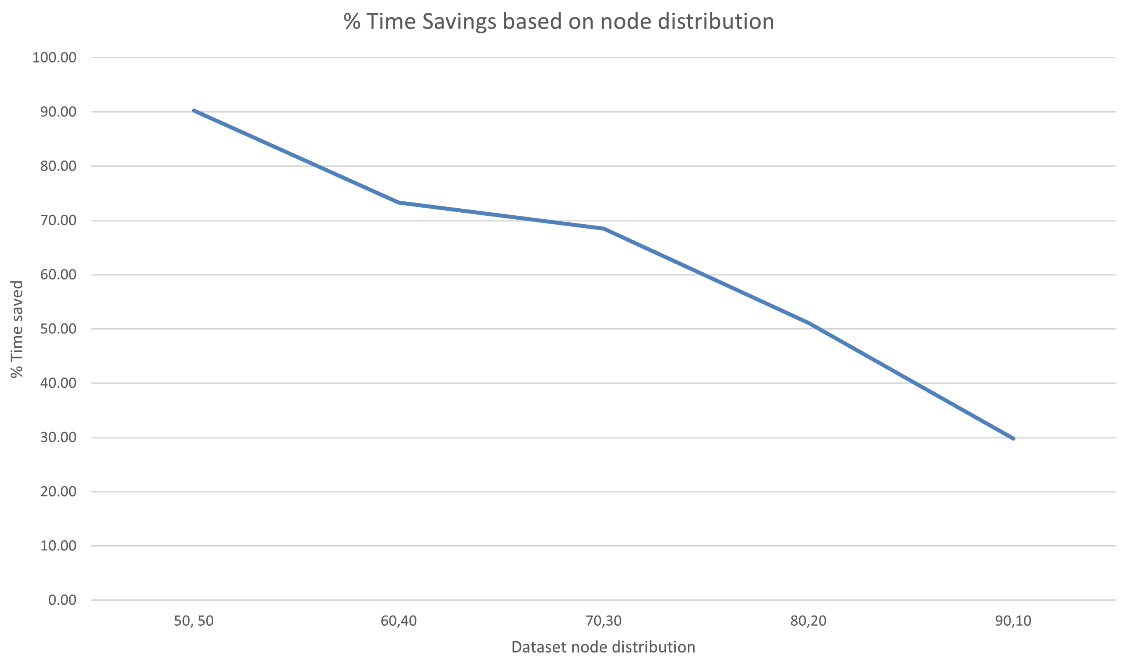


Figure 6.14: Betweenness centrality performance comparison based on node distribution

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This thesis proposes decoupling based algorithms for degree and betweenness centrality computation on HeMLNs. For each several heuristics were develop and the most promising two of them are presented. For degree centrality, with the increase in the amount of partial results retained from each layer we were able to increase accuracy reaching **100%** when compared to ground truth. The **performance improvement is in the range of 2- 67%** for the datasets that were tested. This depends on graph characteristics as well as edge distribution between layers and the number of interlayer edges. This is significant as we are able to achieve 100% accuracy and reduce the time taken as compared to ground truth.

For betweenness centrality, again two heuristics have been presented. Accuracy of degree-dominant heuristic is not consistent across data sets with different graph characteristics. Hence, it was improved to develop a heuristic that is consistent across graph characteristics. Our algorithm based on degree-betweenness heuristic provides consistent improvement in accuracy across datasets and graph characteristics. The accuracy obtained from degree-betweenness heuristic is better than naive algorithm for all the datasets that were tested. With degree-betweenness heuristic a maximum of 94% precision was achieved which is significant when compared to the percentage of time saved. For betweenness centrality we were also able to conclude that the percentage time saved depends on the node distribution among the layers in the HeMLN. **For 50% node distribution around 90% time was saved** with degree-betweenness heuristic.

All the heuristics proposed have provided good results and we believe that it can be further improved. Decoupling-based approach is a new framework for HeMLNs and based on the results obtained, we believe it can be extended to other centrality measures, such as Eigenvector, Pagerank, etc.

REFERENCES

[1] S. Fortunato, “Community detection in graphs,” *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[2] D. J. Cook and L. B. Holder, “Substructure discovery using minimum description length and background knowledge,” *Journal of Artificial Intelligence Research*, vol. 1, pp. 231–255, 1993.

[3] P. Bródka, K. Skibicki, P. Kazienko, and K. Musiał, “A degree centrality in multi-layered social network,” in *2011 International Conference on Computational Aspects of Social Networks (CASoN)*, 2011, pp. 237–242.

[4] A. McLaughlin and D. A. Bader, “Scalable and high performance betweenness centrality on the gpu,” in *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* IEEE, 2014, pp. 572–583.

[5] K. Nakajima and K. Shudo, “Estimating high betweenness centrality nodes via random walk in social networks,” *Journal of Information Processing*, vol. 28, pp. 436–444, 2020.

[6] M. Riondato and E. M. Kornaropoulos, “Fast approximation of betweenness centrality through sampling,” *Data Mining and Knowledge Discovery*, vol. 30, no. 2, pp. 438–475, 2016.

[7] A. Santra, S. Bhowmick, and S. Chakravarthy, “Efficient community re-creation in multilayer networks using boolean operations,” in *International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland*, ser. Procedia Computer Science, P. Koumoutsakos, M. Lees, V. V. Krzhizhanovskaya,

J. J. Dongarra, and P. M. A. Sloot, Eds., vol. 108. Elsevier, 2017, pp. 58–67. [Online]. Available: https://doi.org/10.1016/j.procs.2017.05.246

[8] ——, "Hubify: Efficient estimation of central entities across multiplex layer compositions," in *2017 IEEE International Conference on Data Mining Workshops, ICDM Workshops 2017, New Orleans, LA, USA, November 18-21, 2017*, R. Gottumukkala, X. Ning, G. Dong, V. Raghavan, S. Aluru, G. Karypis, L. Miele, and X. Wu, Eds. IEEE Computer Society, 2017, pp. 142–149. [Online]. Available: https://doi.org/10.1109/ICDMW.2017.24

[9] K. Samant, E. Memeti, A. Santra, E. Karim, and S. Chakravarthy, "Cowiz: Interactive covid-19 visualization based on multilayer network analysis," in *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*, 2021, pp. 2665–2668. [Online]. Available: https://doi.org/10.1109/ICDE51399.2021.00299

[10] A. Santra, K. S. Komar, S. Bhowmick, and S. Chakravarthy, "A new community definition for multilayer networks and A novel approach for its efficient computation," *CoRR*, vol. abs/2004.09625, 2020, https://arxiv.org/abs/2004.09625.

[11] H. N. Djidjev, "Linear algorithms for graph separation problems," in *Scandinavian Workshop on Algorithm Theory.* Springer, 1988, pp. 216–222.

[12] A. Santra, "Analysis of complex data sets using multilayer networks: A decoupling-based framework," Ph.D. dissertation, The University of Texas at Arlington, July 2020, https://itlab.uta.edu/students/alumni/PhD/Abhishek_Santra/ASantra_PhD2020.pdf.

[13] A. Santra and S. Bhowmick, "Holistic analysis of multi-source, multi-feature data: Modeling and computation challenges," in *Big Data Analytics - 5th International Conference, BDA 2017, Hyderabad, India, December 12-15, 2017, Proceedings*, ser. Lecture Notes in Computer Science, P. K. Reddy, A. Sureka,

S. Chakravarthy, and S. Bhalla, Eds., vol. 10721. Springer, 2017, pp. 59–68. [Online]. Available: https://doi.org/10.1007/978-3-319-72413-3_4

[14] S. Chakravarthy, A. Santra, and K. S. Komar, "Why multilayer networks instead of simple graphs? modeling effectiveness and analysis flexibility and efficiency!" in *Big Data Analytics - 7th International Conference, BDA 2019, Ahmedabad, India, December 17-20, 2019, Proceedings*, ser. Lecture Notes in Computer Science, S. Madria, P. Fournier-Viger, S. Chaudhary, and P. K. Reddy, Eds., vol. 11932. Springer, 2019, pp. 227–244. [Online]. Available: https://doi.org/10.1007/978-3-030-37188-3_14

[15] ——, "Humble data management to big data analytics/science: A retrospective stroll," in *Big Data Analytics - 6th International Conference, BDA 2018, Warangal, India, December 18-21, 2018, Proceedings*, ser. Lecture Notes in Computer Science, A. Mondal, H. Gupta, J. Srivastava, P. K. Reddy, and D. V. L. N. Somayajulu, Eds., vol. 11297. Springer, 2018, pp. 33–54. [Online]. Available: https://doi.org/10.1007/978-3-030-04780-1_3

[16] K. S. Komar, A. Santra, S. Bhowmick, and S. Chakravarthy, "EER $\rightarrow$ MLN: EER approach for modeling, mapping, and analyzing complex data using multilayer networks (mlns)," in *Conceptual Modeling - 39th International Conference, ER 2020, Vienna, Austria, November 3-6, 2020, Proceedings*, ser. Lecture Notes in Computer Science, G. Dobbie, U. Frank, G. Kappel, S. W. Liddle, and H. C. Mayr, Eds., vol. 12400. Springer, 2020, pp. 555–572. [Online]. Available: https://doi.org/10.1007/978-3-030-62522-1_41

[17] J. D. Wilson, J. Palowitch, S. Bhamidi, and A. B. Nobel, "Community extraction in multilayer networks with heterogeneous community structure," *J. Mach. Learn. Res.*, vol. 18, no. 1, p. 5458–5506, Jan. 2017.

[18] Z. Hammoud and F. Kramer, “Multilayer networks: aspects, implementations, and application in biomedicine,” *Big Data Analytics*, vol. 5, 07 2020.

[19] B. Lee, S. Zhang, A. Poleksic, and L. Xie, “Heterogeneous multi-layered network model for omics data integration and analysis,” *Frontiers in Genetics*, vol. 10, p. 1381, 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/fgene.2019.01381

[20] A. Rai, “MLN-SUBDUE: Decoupling Approach-based Substructure Discovery In Multilayer Networks (MLNs),” Master’s thesis, The University of Texas at Arlington, May 2020, https://itlab.uta.edu/students/alumni/MS/Anish_Rai/ARai_MS2020.pdf.

[21] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck, “Computing classic closeness centrality, at scale,” in *Proceedings of the Second ACM Conference on Online Social Networks*, ser. COSN ’14. New York, NY, USA: Association for Computing Machinery, 2014, p. 37–50. [Online]. Available: https://doi.org/10.1145/2660460.2660465

[22] D. Sharma and A. Surolia, *Degree Centrality.* New York, NY: Springer New York, 2013, pp. 558–558. [Online]. Available: https://doi.org/10.1007/978-1-4419-9863-7_935

[23] L. C. Freeman, “Centrality in social networks conceptual clarification,” *Social networks*, vol. 1, no. 3, pp. 215–239, 1978.

[24] M. G. Everett and S. P. Borgatti, “The centrality of groups and classes,” *The Journal of mathematical sociology*, vol. 23, no. 3, pp. 181–201, 1999.

[25] S. Uddin and L. Hossain, “Time scale degree centrality: A time-variant approach to degree centrality measures,” in *2011 International Conference on Advances in Social Networks Analysis and Mining.* IEEE, 2011, pp. 520–524.

[26] H. Kretschmer and T. Kretschmer, “A new centrality measure for social network analysis applicable to bibliometric and webometric data,” *Collnet Journal of Scientometrics and Information Management*, vol. 1, no. 1, pp. 1–7, 2007.

[27] W. Maharani, A. A. Gozali, *et al.*, “Degree centrality and eigenvector centrality in twitter,” in *2014 8th international conference on telecommunication systems services and applications (TSSA).* IEEE, 2014, pp. 1–5.

[28] X. Tang, J. Wang, J. Zhong, and Y. Pan, “Predicting essential proteins based on weighted degree centrality,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 2, pp. 407–418, 2013.

[29] M. Zhong, W. Yang, B. Huang, W. Jiang, X. Zhang, X. Liu, L. Wang, J. Wang, L. Zhao, Y. Zhang, *et al.*, “Effects of levodopa therapy on voxel-based degree centrality in parkinson’s disease,” *Brain imaging and behavior*, vol. 13, no. 5, pp. 1202–1219, 2019.

[30] N. Kourtellis, T. Alahakoon, R. Simha, A. Iamnitchi, and R. Tripathi, “Identifying high betweenness centrality nodes in large social networks,” *Social Network Analysis and Mining*, vol. 3, no. 4, pp. 899–914, 2013.

[31] A. McLaughlin and D. A. Bader, “Accelerating gpu betweenness centrality,” *Communications of the ACM*, vol. 61, no. 8, pp. 85–92, 2018.

[32] D. Koschützki, K. A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski, “Centrality indices,” in *Network analysis.* Springer, 2005, pp. 16–61.

[33] U. Brandes, “A faster algorithm for betweenness centrality,” *Journal of mathematical sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[34] D. Erdős, V. Ishakian, A. Bestavros, and E. Terzi, “A divide-and-conquer algorithm for betweenness centrality,” in *Proceedings of the 2015 SIAM International Conference on Data Mining.* SIAM, 2015, pp. 433–441.

[35] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation," *Social Networks*, vol. 30, no. 2, pp. 136–145, 2008.

[36] S. Gao, Y. Wang, Y. Gao, and Y. Liu, "Understanding urban traffic-flow characteristics: a rethinking of betweenness centrality," *Environment and Planning B: Planning and Design*, vol. 40, no. 1, pp. 135–153, 2013.

[37] H. M. Shashikala, R. George, and K. A. Shujaee, "Outlier detection in network data using the betweenness centrality," in *SoutheastCon 2015.* IEEE, 2015, pp. 1–5.

[38] M. De Domenico, "Multilayer modeling and analysis of human brain networks," *Giga Science*, vol. 6, no. 5, p. gix004, 2017.

[39] B. Oselio, A. Kulesza, and A. O. Hero, "Multi-layer graph analysis for dynamic social networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 4, pp. 514–523, 2014.

[40] R. Casarin, M. Iacopini, G. Molina, E. Ter Horst, R. Espinasa, C. Sucre, and R. Rigobon, "Multilayer network analysis of oil linkages," *The Econometrics Journal*, vol. 23, no. 2, pp. 269–296, 2020.

[41] S. Martinčić-Ipšić, D. Margan, and A. Meštrović, "Multilayer network of language: A unified framework for structural analysis of linguistic subsystems," *Physica A: Statistical Mechanics and its Applications*, vol. 457, pp. 117–128, 2016.

[42] K. Komar, "Data-Driven Modeling of Heterogeneous Multilayer Networks And Their Community-Based Analysis Using Bipartite Graphs," Master's thesis, The University of Texas at Arlington, August 2019, http://itlab.uta.edu/students/alumni/MS/Kanthi_Sannappa_Komar/KanthiK_MS2019.pdf.

[43] "AI@WSU," http://ailab.wsu.edu/subdue/download.htm, 2011, [Online].

[44] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-mat: A recursive model for graph mining," in *Proceedings of the 2004 SIAM International Conference on Data Mining.* SIAM, 2004, pp. 442–446.

[45] F. Khorasani, R. Gupta, and L. N. Bhuyan, "Scalable simd-efficient graph processing on gpus," in *Proceedings of the 24th International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '15, 2015, pp. 39–50.

[46] OpenSourceData, "The internet movie database," ftp://ftp.fu-berlin.de/pub/misc/movies/database/, 2018, [Online].

[47] "Dblp dataset," http://dblp.uni-trier.de/xml/, 2018, [Online].

[48] P. Boldi and S. Vigna, "The WebGraph framework I: Compression techniques," in *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004).* Manhattan, USA: ACM Press, 2004, pp. 595–601.

[49] P. Boldi, M. Rosa, M. Santini, and S. Vigna, "Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks," in *Proceedings of the 20th international conference on World Wide Web*, S. Srinivasan, K. Ramamritham, A. Kumar, M. P. Ravindra, E. Bertino, and R. Kumar, Eds. ACM Press, 2011, pp. 587–596.

[50] J. Golbeck, "Chapter 3 - network structure and measures," in *Analyzing the Social Web*, J. Golbeck, Ed. Boston: Morgan Kaufmann, 2013, pp. 25 – 44. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780124055315000031

[51] M. E. Newman, "A measure of betweenness centrality based on random walks," *Social networks*, vol. 27, no. 1, pp. 39–54, 2005.

BIOGRAPHICAL STATEMENT

Kiran Mukunda was born in Bengaluru, Karnataka, India. He received his Bachelors degree in Electronics and Communication Engineering from R.V. College of Engineering, Bengaluru, India in May 2015. After graduation he worked for Oracle India Private Limited, Bengaluru as a Senior Application Engineer from June 2015 to August 2019. He studied his masters degree in computer science at The University of Texas at Arlington from August 2019 to July 2021. He likes working in data mining, databases and software engineering domains.