GRAPH ALGORITHM TO FIND CORE PERIPHERY STRUCTURES USING MUTUAL K-NEAREST NEIGHBORS

Divya Sardana[1] and Raj Bhatnagar[2]

[1]Teradata Corp., Santa Clara, CA, USA
[2]Department of Electrical Engineering and Computer Science,
University of Cincinnati, OH, USA

ABSTRACT

Core periphery structures exist naturally in many complex networks in the real-world like social, economic, biological and metabolic networks. Most of the existing research efforts focus on the identification of a meso scale structure called community structure. Core periphery structures are another equally important meso scale property in a graph that can help to gain deeper insights about the relationships between different nodes. In this paper, we provide a definition of core periphery structures suitable for weighted graphs. We further score and categorize these relationships into different types based upon the density difference between the core and periphery nodes. Next, we propose an algorithm called CP-MKNN (Core Periphery-Mutual K Nearest Neighbors) to extract core periphery structures from weighted graphs using a heuristic node affinity measure called Mutual K-nearest neighbors (MKNN). Using synthetic and real-world social and biological networks, we illustrate the effectiveness of developed core periphery structures.

KEYWORDS

Graph Mining, Core Periphery Structures, MKNN, Complex Networks.

1. INTRODUCTION

Weighted complex networks have non-trivial topological properties as well as non-homogeneous edge weights [1]. Meso-scale structures help in studying the features of complex networks which are not easily evident at the local or global scale, e.g., community structure. Most algorithms for finding community structure in graphs result in disjoint subsets or clusters. However, in many complex networks, nodes play more than one role in the network, for example multi-functional proteins in protein-protein interaction (PPI) networks and people belonging to multiple social groups in a social network. Overlapping or fuzzy clustering algorithms result in multiple cluster assignments to nodes thereby helping to study relationships among clusters in the network. For e.g., in a co-authorship network, overlapping communities would represent authors who collaborate in multiple groups. Many overlapping clustering algorithms exist in literature that work for complex networks [2], [3], [4], [5], [6], [7], [8].

Another meso-scale structure called core periphery is also important in understanding complex networks. However, its research literature is not as wide spread as that exists for community structure. In [9], the authors illustrated that overlapping clusters naturally lead to the existence of dense cores surrounded by sparse peripheries in many complex networks. In [10], the authors use real world Twitter data to provide empirical evidence that community structure is always accompanied by a core periphery structure in a social network.

DOI: 10.5121/ijaia.2021.12101

The analysis of core periphery structures can be very useful in developing insights in many real-world networks. For example, in a social network, a dense core group of close friends is usually surrounded by a sparsely connected group of periphery acquaintances. A study of core periphery structures in such a network can help in identifying how trends spread across the network of friends. Similarly, in a protein-protein interaction (PPI) network, a dense core could be seen as a cohesively connected set of proteins surrounded by a loosely connected set of periphery proteins which dynamically connect to more than one core set depending upon their function.

Borgatti and Everett were the first to describe a core periphery structure as a dense and cohesively connected core which is surrounded by a less dense periphery with loose connections to it. [11]. Traditionally, the notion of core periphery structures has been used in diverse fields such as world systems [12], economics [13], organization studies [14] and social networks [15], [16]. Core periphery structures have been employed for the analysis of biological networks such as protein-protein interaction networks [17] and work groups [18]. With the accumulation of Big data in various real-world domains, the usefulness of core periphery structures has increased even more. In a recent paper [19], the authors perform an analysis of cores in a series of big crime data so as to characterize the modus operandi of criminals across the crime series.

In weighted graphs, both, the topological structure of nodes as well as non-trivial edge weight properties in the graph contribute towards density. Thus, a core periphery structure in a weighted graph can have a dense core as determined by high edge weight density or high structural density or both. We call this measure of density as SE-density. In figure 1, we demonstrate regions of varying density in a weighted graph. The subset of nodes (R,S,T,U,V) form a clique and is structurally very dense with low edge weight density (or low mean of edge similarities). On the other hand, the subset of nodes (A, B, C, D, E, F, G) form a subset with high edge weight density amongst each other with low structural density.

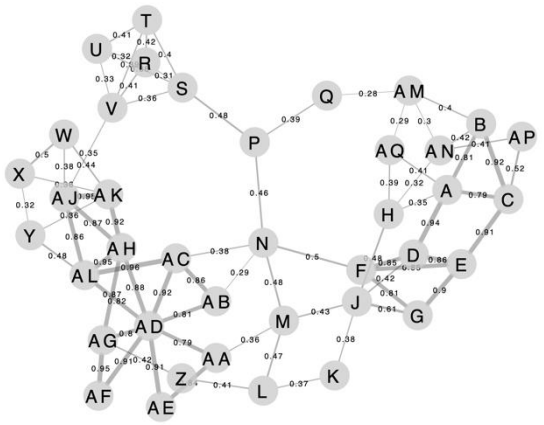

Figure 1. Synthetic Dataset 1

In this paper, we formally define core periphery structures that exist in weighted graphs. Next, we build a graph algorithm called Core Periphery-MKNN (CP-MKNN) which identifies core periphery structures in a weighted graph using a density based heuristic measure called MKNN (Mutual K-nearest neighbors). We further score and categorize the peripheries based upon the strength of their connection to the cores. The algorithm also identifies sibling relations between core-periphery structures which are connected together by loose periphery-periphery connections. We provide a comparison of the properties of core peripheries identified by CP-MKNN with another recently developed core periphery algorithm called ClusterONE-CP [20]. Through the use of synthetic and real world weighted social and PPI networks, we illustrate that CP-MKNN is

able to find denser cores than ClusterONE-CP as per structure or edge weight density. Lastly, we use a motivating real-world example in protein-protein interaction networks to express the usefulness of found core-periphery structures.

Next, we provide an outline of the paper. In section 2, we discuss the related work in the field of core periphery structures. We present our definition of core periphery structures in weighted graphs along with other relevant terms in section 3. We provide our methodology to find core periphery structures for a weighted graph in section 4. In section 5, we present an extensive experimental evaluation of our algorithm on two synthetic and real world social and PPI networks. In section 6, we conclude our findings.

2. RELATED WORK

While the notion of core-periphery structures has been in use for quite a while, it was Borgatti and Everett, who formalized the first model for finding core-periphery structures in a network in the late nineties [11]. They proposed a quality function comparing the network to an ideal model of core-periphery structures where nodes are connected to each other only if they have core membership. This algorithm is limited to the division of the whole graph into one core and one periphery. This model was extended by the authors in their following paper to identify more than one core and periphery structure. [21]. Another algorithm [22] based on the same idea was built using a more flexible model to assign a core score to each node.

A Kernighan-Lin algorithm-based methodology was proposed by Boyd et al. [23] to find core periphery structures in a social network. An enhanced version of this algorithm was built by Luo et al. [17] to identify k-plex cores in PPI networks. The idea of random walks has been used in [24] to assign a coreness score to nodes in a network as a centrality measure. An overlapping tiles-based model was developed by Yang and Leskovec [9] to identify overlapping communities and core periphery structures in a network. In [25], Bruckner et al. developed a core periphery identification technique for PPI networks using graph modification. Sardana et al. [20] proposed a core periphery algorithm called ClusterONE-CP based on a greedy growth based overlapping graph clustering algorithm called ClusterONE. de Jeude et al. [26] developed a p-value based method using multinomial hypergeometric distribution to assign a surprise like score to network partitions and categorize them into cores and peripheries. Three methodologies for find core periphery structures in directed graphs were described in [27]. Two of them are based upon the well know link analysis algorithm called Hyperlink-Induced Topic Search (HITS) [28] and one of them is based upon the concept of likelihood maximization.

The concept of MKNN heuristic was first coined in for use in data clustering in [29]. Sardana et al. utilized this heuristic for finding density-based clusters in a weighted graph [30]. In this paper, we use this heuristic measure to find core periphery structures in weighted graphs. Further, the work in this publication is a part of this paper's first author's PhD dissertation [31].

3. BACKGROUND

In a weighted graph, we formalize a core periphery structure as a dense, cohesive core set C, surrounded by a loosely connected periphery set P. The nodes in a core periphery structure are identified by the union ($C \cup P$). The core set C is denser than the periphery set P as per structural density or edge weight density. We further categorize the peripheries into two types: 1 and 2 depending upon the nature of density difference between the core and periphery. These two types of peripheries are defined in detail next.

1) Periphery Type 1: A periphery of type 1 is less dense than its core as per structure density measured using a term called Cohesion as defined below.

Definition 3.1. [Cohesion, η] For a set S, cohesion empirically captures the degree of intensity by which the nodes are structurally connected to each other. Mathematically, we define cohesion as follows.

$$\text{Cohesion}, \eta(S) = \frac{\text{Insim}(S)}{\text{Insim}(S) + \text{Cut}(S)} \quad (1)$$

Here *Insim(S)* corresponds to the sum of internal similarity weights for the set S, and *Cut(S)* represents the sum of edge weights connecting S with rest of the clusters. Formally, we define periphery P to be of type 1 for its core C if *Cohesion(C) > Cohesion (C ∪E ∪P)* where E is the edge set connecting C and P. Figure 2 demonstrates a core periphery structure of type 1 with a core of blue nodes having a higher structural density than its type 1 periphery of red nodes.

2) Periphery Type 2: A periphery type 2 is less dense than its core set as per edge weight density measured using the mean of edge weights (μ) of the edges belonging to a cluster. In other words, a periphery P connected to its core C using edge set E is labeled as type 2 if μ(Edge weights belonging to C) − μ(Edge weights belonging to (P U E)) > m (MEAN OFFSET). Here m or MEAN OFFSET is a user defined parameter with a default value of 0.2. Figure 2 demonstrates a core periphery structure of type 2 with a center core of blue nodes having a higher edge weight density than its type 2 periphery represented by green nodes. Note that the width of the edges has been drawn proportional to their edge weights.

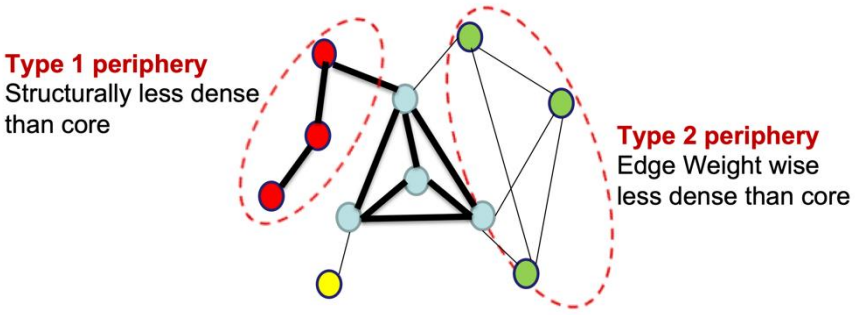

Figure 2. Example of two types of Core Periphery Structures in a weighted graph. The center blue core has two peripheries: Red (Type 1) and Green (Type 2).

We further assign a score or a distance to each core periphery structure based upon the density difference between its core and periphery constituents. On similar lines, a score based upon domain knowledge can also be constructed to make the core periphery relationships more meaningful. We construct an example core periphery score, called CPScore and an example core periphery distance called CPDistance as defined below. Let E be the edge set connecting the core and periphery sets.

Definition 3.2. [CPScore] CPScore for a core periphery structure is defined as the number of actual edges in the set C ∪ E ∪ P divided by the number of all possible edges in the set C ∪ E ∪ P. Intuitively, this score captures the structural density of the hypothetical cluster formed by merging the core C and its periphery P. A higher value of CP score is an indicator of high structural density between core and periphery. Formally, we calculate CPScore is as below.

$$\text{CPScore}(C_i, P_j) = \frac{\text{No. of edges in } C_i \cup E \cup P_j}{\text{No. of all possible edges in } C_i \cup E \cup P_j} \quad (2)$$

Definition 3.3. [CPDistance] CPDistance is defined as the number of standard deviations (σ) that the periphery edges are away from the mean (μ) of core edges. It gives a measure of edge weight density difference between the core and periphery. A lower value of CPDistance indicates a strong edge weight density similarity between the core and periphery. Formally, we calculate CPDistance as below.

$$\text{CPDistance}(C_i, P_j) = \frac{\left|\mu(w_{in}(C_i)) - \mu\left(w_{in}(E \cup P_j)\right)\right|}{\sigma(w_{in}(C_i))} \quad (3)$$

In the above formula, win corresponds to the edge weights of edges inside the cluster. The core periphery types and scores as defined above can be very useful in real-world networks to rank the strength of connection from cores to peripheries. For e.g., in a co-authorship network the CPScore and CPDistance can be used to prioritize potential co-authors for a core group based upon the quantity and quality of shared publications with its periphery group of authors.

4. METHODOLOGY

Consider an undirected, weighted graph, G = (V, E) with V as the vertex set and E as the edge set. The edge weights are represented by a similarity matrix (SM). Each element of SM contains a non-negative real number representing similarity between two vertices between 0 and 1. The CP-MKNN algorithm is performed in three phases as described next.

4.1. Initialization Phase

The initialization phase involves two steps. First, it expands G to form an augmented graph Ga where secondary similarities are defined between all nodes up to four hops away. This is done by applying Dijkstra's algorithm in the four-hop neighborhood of each node. Second, a Mutual K-nearest neighbor (MKNN) matrix is generated using the similarities in Ga. MKNN is a node affinity measure as defined in [28] for use in graph clustering. As opposed to the classic KNN algorithm, MKNN is based on a two-sided relationship between vertices. We explain the concept of MKNN for K=4 using figure 3. Node G has 4 KNNs, namely, nodes A, B, D and E. However, nodes A, B, D and E already have found K=4 KNNs amongst the red nodes. MKNN defines a mutual relationship between two nodes, therefore, G forms an MKNN relationship with the next available node F with less than K=4 neighbors. Intuitively, for relatively low values of K, MKNN is able to capture set of nodes at approximately the same level of edge-weight based density. We use this heuristic to guide our core periphery formation algorithm.

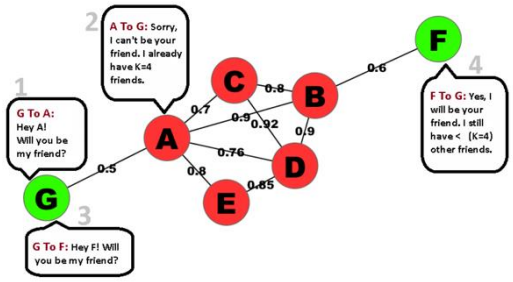

Figure 3. Example of Mutual K-nearest neighbors in a weighted graph

4.2. Phase 1 (Preliminary Phase)

The preliminary phase involves growing small sized dense subgraphs by merging some seed nodes with their MKNN neighbors. In this phase we call each node as an i-node. The i-nodes are ranked in a non-increasing order of density around them using a term called *radius* defined next. Given an i-node ini with similarities to its p MKNNs being (sm_1, sm_2...sm_p). Let the degrees of MKNN nodes be (d_1, d_2, ...d_p). We define radius, σ for in_i as follows.

$$\text{Radius}, \sigma(\text{in}_i) = \left(\frac{\sum_{i=1}^{p}(d_i * sm_i)}{p * d_{avg}}\right) \quad (4)$$

The average degree of a graph's i-nodes corresponds to d_{avg}. Based upon this formulation, i-nodes having a higher radius have denser surroundings and get a chance to merge with their MKNNs earlier than other i-nodes. We call this merging order defined by *radius* as the *merge initiator order*.

In phase 1, i-nodes merge with their MKNN neighbors in the *merge initiator order*. If in this process, some neighbors are found to already belong to some other subgroup, then this neighbor is merged with the initiator where it leads to the least increase in standard deviation of edges inside the subgroup. This process continues until all initiators are exhausted. Figure 4 illustrates the small subsets formed after phase 1 for synthetic dataset 1. The pseudocode for this phase is given in algorithm 1.

4.3. Phase 2 (Merge and Boundary Formation Phase)

In the preliminary phase, i-nodes merge with their MKNNs to generate small subgroups, which we denote as a c-node in phase 2. In the merge phase, (1) pairs of c-nodes with high level of similarity between them are merged with each other, and (2) boundary c-node sets are generated for each c-node which will later help in the formation of core-periphery structures. A connectivity matrix (CM) is constructed to define a notion of similarity among pair of c-nodes. We formulate it as below.

$$\text{CM}(cn_i, cn_j) = \left(\frac{\text{linkage}(cn_i, cn_j)}{\text{cut}(cn_i)} + \frac{\text{linkage}(cn_i, cn_j)}{\text{cut}(cn_j)}\right) \quad (5)$$

Here linkage (cn_i, cn_j) is calculated as the sum of primary similarities of edges that link cn_i's i-nodes and cn_j's i-nodes. Further, cut(cn_i) is used to capture the primary similarity summation

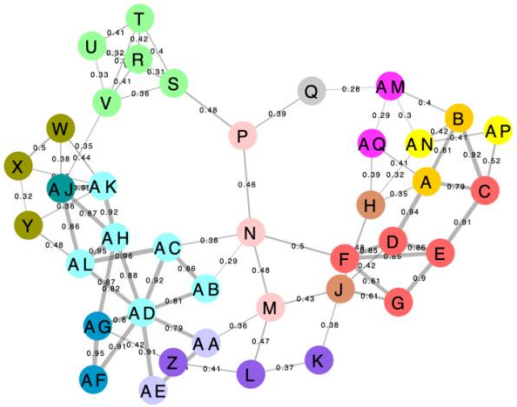

Figure 4. Dense subsets formed after Phase 1 for Synthetic Dataset 1

Algorithm 1: Form Preliminary clusters

Input:

SM: Matrix of similarity weights

L: Input i-node list ∈ V

$MKNN$: MKNN relation matrix

Output:

An array containing cluster identifiers for i-nodes

```
Calculate Radius of nodes ∈ L.
Sort nodelist L in decreasing order of each node's Radius.
Set initial cluster identifier as γ
for i-node l_i ∈ L do
    if cluster label C_i not set for l_i then
        Set cluster initiator as l_i
        C_i ← γ
        for i-node l_j ∈ set of MKNNs for i-node l_i do
            if C_j is empty then
                C_j ← γ
            else
                Change C_j to γ if this leads to
                a decrease in standard deviation of
                similarity edge weights in the cluster
                that results.
            end
        end
        γ ← γ + 1
    end
end
```

between cni's i-nodes and i-nodes belonging to other graph c-nodes. Intuitively, CM captures the relative strength of connection between two c-nodes as compared to their connections with all other graph c-nodes. Next, using CM values as a measure of similarity, MKNN relationships are defined among c-nodes. This helps to avoid considering every pair of c-nodes in CM for possible merging. For example, in figure 4, the core (C, D, E, F, G) forms MKNN neighbors as (A, B), (AN, AP) and (J, H). However, all c-nodes which are MKNN are not best to merge with each other. Before two c-nodes are merged, we ensure that they both are synchronized in terms of structural density and edge-weight density. This is done by using the two checks below for validating every merge.

1) **Check for structural density:** Two c-nodes pass the structural similarity check if the Cohesion of the prospective merged c-node is greater than the Cohesion of the initiator c-node.

2) **Check for edge-weight density:** Two c-nodes pass the edge weight density check if the mean difference of their edge weights is within a user defined parameter called MEAN OFFSET, m. This ensures that the two c-nodes are homogeneous in terms of edge weights and the variance of the merged cluster remains low.

The following rules are established to decide about merging and boundary set formation.

1) If c-node$_i$ and its MKNN c-node$_j$ satisfy both constraint A and B, then the two c-nodes

- merge with each other to form a bigger sized core if they are cores, or are peripheries belonging to the same set of cores. Otherwise, the two c-nodes

- form a weak periphery-periphery relationship. In this case, c-node$_j$ is put in boundary node set pp (periphery-periphery) of c-node$_i$

2) If c-node$_i$ and its MKNN c-node$_j$ satisfy only the structural density constraint A, c-node$_j$ is put in boundary nodeset low of c-node$_i$.

3) If c-node$_i$ and its MKNN c-node$_j$ satisfy only the edge weight density constraint B, c-node$_j$ is put in boundary nodeset inrange of c-node$_i$.

4) If c-node$_i$ and its MKNN c-node$_j$ do not satisfy any of the constraint A or B, c-node$_j$ is put in boundary nodeset low of c-node$_i$ if its mean is lower than the mean of c-node$_i$, or else in boundary nodeset high of c-node$_i$.

As an example, in figure 4, the red core (C, D, E, F, G) merges with its MKNN neighbor (A, B), whereas, it puts its MKNN subsets (AP, AN) and (J, H) in its boundary sets. The subsets (AP, AN) and (J, H) further merge with their own MKNN neighbors to grow in size. The pseudocode for phase 2 is given in algorithm 2.

Algorithm 2: Merge Phase

Input:

SM: Matrix containing graph similarities between all graph vertices

P: Input c-node list from phase 1(preliminary clusters)

C: Cluster label array from preliminary phase

Output:
An array C with cluster identifiers of each i-node
Sets called $low, high, inrange$ defining boundary nodes for each cluster.

```
Calculate CM_{|n|x|n|} where n is the length of P.
Find MKNN neighbors of all c-nodes ∈ P using CM for similarity weights.
Set D ← An array containing cluster identifier of each c-node
Sort list of c-nodes P in non-increasing order of their cohesion value.
Initialize c-node cluster label λ
repeat
    for c − node_i ∈ P do
        if D_i not set for cnode_i then
            Set cnode_i as cluster initiator
            D_i ← λ
            for c − node_j ∈ Set of MKNNs for c − node_i do
                if (cnode_j ∉ found cores for cnode_i and
                 structural sim. check is satisfied and
                 edge-weight sim. check is satisfied) then
                    if core_set(c − node_i) = core_set(c − node_j) then
                        Set D_j ← λ, λ ← λ + 1
                    else
                        Put cnode-j in boundary cnodeset pp
                        (periphery-periphery) for cnode-i
                    end
                else if (structural sim. check is satisfied &
                 edge-weight sim. check is not satisfied) then
                    Put cnode-j in boundary c-nodeset low of cnode-i.
                else if (structural sim. check is not satisfied &
                 edge-weight sim. check is satisfied) then
                    Put cnode-j in boundary c-nodeset in-range of cnode-i
                else if (structural sim. check is not satisfied &
                 edge-weight sim. check is not satisfied) then
                    Put c − node_j in boundary c-nodeset low or high of c − node_i
                    depending upon the their edgeweight mean difference.
                end
            end
        end
    end
    Refresh P with the new set of merged c-nodes Recalculate CM and MKNN
    matrices
until no merging takes place in an iteration;
```

4.4. Phase 3 (Core Periphery Structure Formation Phase)

After the merge phase completes, it leads to the formation of dense c-nodes and their corresponding boundary c-node sets, *low, inrange, high* and pp. The boundary sets vary in terms of density difference from the core. In this phase, all c-nodes are traversed in the order of their density measure Cohesion to explore their boundary c-node sets for the possibility of formation of core periphery and periphery periphery relationships. Further, each relationship is assigned a type as described below.

1) Core with Type 1 periphery: This type of periphery has a lower structural density than its core. c-nodes lying in boundary c-nodesets in-range of the core c-node are made as type 1 peripheries of the core.

2) Core with Type 2 periphery: This type of periphery has a lower edge weight density than its core. c-nodes lying in boundary c-nodesets low of the core c-node are put in type 2 periphery of the core. If the core c-node lies in boundary c-nodeset high of the periphery c-node, then this periphery is also made a type 2 periphery of the core.

3) Periphery-Periphery: This type of relationship is formed between two peripheries which do not share the same set of cores, but still have a weak connection between the two c-nodes lying in boundary c-nodeset pp are candidates for this relationship. These relationships seem to weakly connect two core periphery structures in a sibling relationship.

All the extracted relationships are next assigned with CPDistance and CPScore as defined before in the background section. A resultant core periphery structure of type 2 formed for Synthetic dataset 1 is shown in figure 5. Figure 6 shows an example of a core periphery structure with periphery type 1 for synthetic dataset 1. Further, the red core in figure 5 and the green core in figure 6 share a periphery (H, J, K, L, M, N, P, Q, Z), thereby forming a sibling relationship between the two cores. All the steps for identifying core periphery relationships are described in algorithm 3.

5. EXPERIMENTAL EVALUATION

We demonstrate the effectiveness of core periphery structures identified by CP-MKNN using two synthetic and two real protein-protein interaction network-based graph datasets. We further compare and contrast them with core-periphery structures generated by another algorithm called ClusterONE-CP. Both CP-MKNN and ClusterONE-CP allow a cluster to be both a core or a periphery or both and find core-periphery as well as well as periphery-periphery relationships. For running ClusterONE-CP, we set node penalty equal to 2 and overlap threshold equal to 0.5.

5.1. Results for Synthetic Datasets

We generated two weighted graphs to use as synthetic datasets: The first synthetic dataset has 39 vertices and 74 weighted edges (figure 1). The synthetic dataset 2 has 49 vertices and 97 weighted edges (figure 7) We use these synthetic datasets to illustrate the comparison of core periphery results obtained by CP-MKNN with those by ClusterONE-CP. In synthetic dataset 1, we embedded two edge weight dense cores, namely, core 1 with nodes A, B, C, D, E, F, G, H and core 2 with nodes AA, AB, AC, AD, AE, AF, AG. Further, there is one structurally core, core 3 with nodes R, S, T, U, V. The cores embedded in synthetic dataset 2 are Core 1 with nodes A, B, C, D, E, F, G, H; Core 2 with nodes J, K, L, M, N, P, Q and Core 3 with nodes R, S, T, U, V, W. All the cores are connected through peripheries. We use a measure called structural density

(graph density) [32] to measure the structural cohesiveness of clusters (both cores and peripheries). It is formally defined as

$$\text{Structural Density(C)}, \rho(\text{C}) = \left(\frac{|\text{edges(C)}|}{|\text{All possible edges(C)}|} \right) \quad (6)$$

Further, we use average variance of the weighted edges inside a core or periphery to determine how homogeneous are the edge weights inside the subset.

Algorithm 3: Extract Core-Periphery Structures

Input:

S_n : Set containing c-nodes generated after merge phase

Output:

C_p: Set of relationships (C, P) between cores and peripheries with a label for the strength and type of each relationship

P_p: Set of periphery-periphery relationships (P, P) with a label for the strength of each relationship

```
Initialize C_P and P_p as empty data structures.
Store in O_n the c-nodes in S_n by the order of Cohesion.
for O_i ∈ O_n do
    Set P to merged c-nodes ∈ O_i 's boundary c-nodeset inrange, low
    for P_j ∈ P do
        if (P_j ∈ boundaryset low and μ(O_i) > μ(P_j)) or (P_j ∈ boundaryset
        inrange and Cohesion(O_i) > Cohesion(P_j)) then
            Form (O_i, P_j) as a core-periphery association
            Set Type(O_i, P_j) as the type of majority of boundary nodes ∈ P_j
            (Type 1: inrange, Type 2: low)
            Add (O_i, P_j) along with its type, CPDistance and CPScore to C_P
        end
        else if (P_j ∈ boundaryset pp then
            Form (O_i, P_j) as a periphery-periphery relationship
    end
end
```

5.1.1. Comparison of CP-MKNN vs. ClusterONE-CP for Synthetic datasets

In table 1, a comparison of properties of cores and peripheries is provided between CP-MKNN and ClusterONE- CP. It can be noted that both the datasets and both the algorithms, cores are denser than the peripheries as per structural density or edge weight density denoted by mean of edges, For CP-MKNN, in both the datasets, we see that all cores, core-peripheries and peripheries obtain a low variance of edge weights. This signifies homogeneous edge weights. For ClusterONE-CP, the variance results of synthetic dataset 2 indicate that the clusters are less homogeneous as per edge weights.

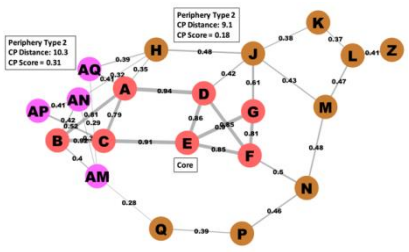

Figure 5. Core Periphery Structure with peripheries of Type 2 (Core: Red nodes, Peripheries: Brown and Pink nodes) for Synthetic Dataset 1

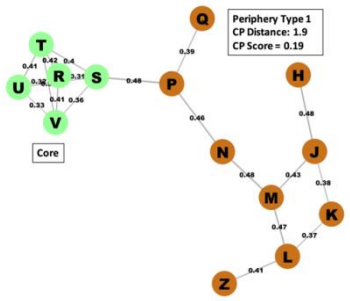

Figure 6. Core Periphery Structure with periphery Type 1(Core: Green nodes, Periphery: Brown nodes) for Synthetic Dataset 1

Table 1. Core Periphery Structures by CP-MKNN and ClusterONE-CP for Synthetic datasets

	No.	**Avg. Mean**	**Avg. Standard Deviation**	**Avg. Structural Density**
CP-MKNN for Synthetic Dataset 1				
Cores	3	0.71	0.043	0.56
Peripheries	3	0.39	0.061	0.45
Core Peripheries	0	0	0	0
ClusterONE-CP for Synthetic Dataset 1				
Cores	2	0.87	0.046	0.4
Peripheries	2	0.37	0.069	0.58
Core Peripheries	3	0.52	0.078	0.65
CP-MKNN for Synthetic Dataset 2				
Cores	3	0.88	0.025	0.48
Peripheries	2	0.43	0.068	0.64
Core Peripheries	2	0.65	0.089	0.38
ClusterONE-CP for Synthetic Dataset 2				
Cores	2	0.85	0.088	0.46
Peripheries	2	0.45	0.083	0.5
Core Peripheries	4	0.59	0.11	0.57

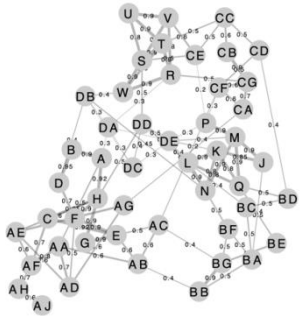

Figure 7. Synthetic Dataset 2

5.2. Results for Movie Co-Appearance Dataset

Using synthetic datasets, we demonstrated that CP-MKNN obtains cores which are denser than the core-peripheries and peripheries. In this section we demonstrate the difference between CP-MKNN and ClusterONE-CP using a movie co-appearance dataset called Les *Miserables* [33]. This dataset is network of interaction amongst 77 characters in a movie based on Victor Hugo's novel called *Les Miserables*. The clustering coefficient of this dataset is 0.57. Figure 8 illustrates a core found by CP-MKNN (blue nodes), K=2 and a core found by ClusterONE-CP (red and blue nodes combined). The blue nodes correspond to a very dense core comprising of the novel's actor, Maurius (MA); actress, Cosette (CO) and the actress's father, Jean Valjean (JV). The red nodes comprise other important characters in the novel. CP-MKNN separates the most important characters of the novel because it uses both structure and edge weight density constraints in constructing core periphery structures.

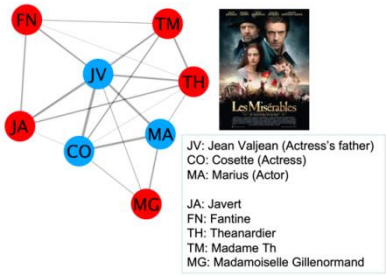

Figure 8. A dense core by CP-MKNN (K=2) (blue nodes) and ClusterONE-CP (blue and red nodes) on Les Miserables, a movie co-appearance dataset

5.3. Results for Real Social Network Dataset

We use a well-known social network dataset called Zachary's Karate Club Network, frequently used in the research community for evaluating community structures in graphs. Zachary's dataset is based on associations in a karate club among 34 people as its members, collected at a US university of a total of three years [34]. We represent each member of the club by its number from 1 to 34. The club's instructor represented by node 1 had a disagreement with his president, node 34 and the club got disintegrated into two factions. The first group mainly consisted of supporters of the president (node 34), and the second group represented people who supported the instructor (node 1). We use a weighted version of the network here. The edge weights are based upon the number of common activities that the club members took part in.

Figure 9 represents core periphery structures obtained by CP-MKNN on the Karate Club dataset and figure 10 displays core periphery structures obtained by ClusterONE-CP on the same dataset.

The size of nodes has been drawn proportional to their degree and the width of edges has been drawn proportional to the edge weights. CP-MKNN is able to identify the two factions correctly around the two nodes 1 and 34. ClusterONE-CP further splits one of the factions into two parts. Further, CP-MKNN identifies nodes 10, 20 and 29 as peripheries for both the factions. It indicates that perhaps, these three nodes have relationships with both the cores and suggest a relationship between the cores. Further, CP-MKNN provides a core periphery association of type 1 between node 1's core and node 34's core with a low value of *CPDistance* (0.11) and *CPScore* (0.17). This indicates that the two factions are similar to each other in terms of edge weight density even if they are separated in terms of structural density. Thus, CP-MKNN not only divides the network into cores and peripheries, but we are also able to derive insights about how the members of different cores might be associated with each other. This information can be very useful for understanding the flow of information in social network analysis.

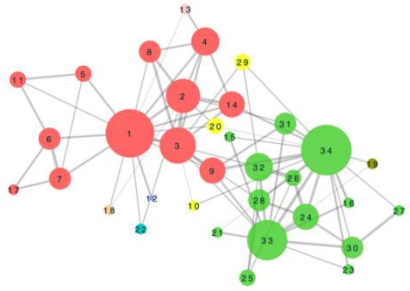

Figure 9. Core Periphery structures obtained by CP-MKNN on Karate Club dataset (Faction1 (Core1): Red nodes, Faction (Core 2): Green nodes, Shared periphery: Yellow nodes)

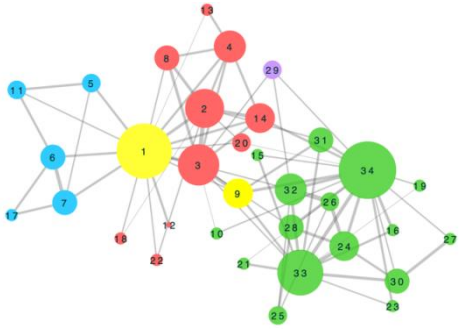

Figure 10. Core Periphery structures obtained by ClusterONE-CP on Karate Club dataset (Faction 1 (Split into two cores): Red and Blue nodes, Faction 2 (Core3): Green nodes)

5.4. Results for Real PPI Network Dataset

We use two widely used PPI datasets in the research community for the evaluation of core periphery structures generated by CP-MKNN. These datasets are named Gavin (1855 proteins with 7669 edges) [35] and Krogan (2708 proteins with 7123 edges) [36]. Further, we use the MIPS catalog of gold standard complexes [37] as a validation dataset. In this dataset, there are a total of 203 protein complexes and 1189 proteins. We use accuracy as defined by Brohee and Helden [38] for assessing the quality of core periphery structures. This measure is formulated as the geometric mean of sensitivity (Sn) and positive predicted value (PPV). Sn and PPV are mathematically formulated as below.

$$S_n = \left(\frac{\sum_{i=1}^{m} \max_{1 \le j \le n} t_{ij}}{\sum_{i=1}^{m} m_i} \right) \quad PPV = \left(\frac{\sum_{j=1}^{n} \max_{1 \le j \le m} t_{ij}}{\sum_{j=1}^{n} \sum_{i=1}^{m} t_{ij}} \right) \quad (7)$$

Here n is the count of clusters predicted by the algorithm and n is the number of protein complexes in the gold standard database. mi corresponds to the protein count in complex i. T is the contingency table of protein complexes with two dimensions, actual and predicted.

5.4.1. Comparison of CP-MKNN vs. ClusterONE-CP for Real PPI datasets

We present an accuracy comparison between CP-MKNN and ClusterONE-CP in table 2. It can be noted that accuracy value obtained for both the algorithms is very similar in the case of both the PPI datasets. This suggests that clusters obtained by CP-MKNN are meaningful and match well to the gold standard complexes similar to ClusterONE-CP. Further, the clusters obtained for CP-MKNN tend to have a higher average mean of edge weights indicating that CP- MKNN is able to separate very dense cores from their sparser peripheries.

Next, we present a comparison of the properties of cores and peripheries as obtained by CP-MKNN and ClusterONE-CP on the real PPI datasets in table 3. Both the algorithms identify the clusters as being core, periphery or core-periphery (both core and periphery). We notice that for both CP-MKNN and ClusterONE-CP, cores obtain a higher average mean and average structural density over the peripheries. This difference is statistically significant as per Student's t-test with $\alpha = 0.05$. Further, both the algorithms find core-peripheries to have a statistically significantly higher average mean than peripheries for both the datasets. The difference in structural density of core-peripheries and peripheries is not found to be statistically significant. The table also displays the difference in the essentiality values of cores vs peripheries. Essential proteins are those that are thought be critical for the survival of the organism. It has been studied that the essentiality of proteins in a PPI network could be related to their structural characteristics in the network [39] [17]. We use an essential proteins database [40] to determine the essentiality of proteins. For both the datasets, we find that cores obtained by CP-MKNN have a higher average essentiality over the peripheries as per t-test with $\alpha = 0.05$. This is in alignment with the existing studies that the essentiality of proteins could be related to their structural properties in the PPI network For ClusterONE-CP, this trend is supported only for the Gavin dataset.

Table 2. Accuracy comparison for CP-KNN vs ClusterONE-CP for PPI datasets

Algorithm	**No. of Clusters (size>2)**	**Sn**	**PPV**	**Accuracy**	**Avg. Mean**
Gavin Dataset					
CP-MKNN	233	0.44	0.41	0.43	0.41
ClusterONE-CP	245	0.56	0.39	0.47	0.38
Krogan Dataset					
CP-MKNN	293	0.42	0.46	0.44	0.75
ClusterONE-CP	332	0.49	0.44	0.46	0.71

5.4.2. Comparison of relationship types identified by CP-MKNN for Real PPI datasets

We provide a comparative study of different types of relationships (core-periphery (type 1 and type 2) and periphery- periphery) for CP-MKNN in table 4 for Gavin dataset. The table displays the number of relations for each type, along with the average CPDistance and the average CPScore of the peripheries in each type. As described before, CPDistance is a measure of edge weight distance between core and periphery, whereas CPScore is a measure of structural similarity between core and periphery. A smaller value of CPDistance signifies that core and periphery are closer as per edge weight density. A higher value of CPScore suggests that the core and periphery are closer to each other as per structural density. As evident from table 4, type 1 peripheries are closer their cores than type 2 peripheries as per edge weight density (lower value of average CPDistance). On the other hand, type 2 peripheries are closer to their cores than type 1 peripheries as per structural density (higher value of average CPScore). These results are in line with our construction of peripheries of different types.

Table 3. Core Periphery Structures by CP-MKNN and ClusterONE-CP for PPI datasets

	No.	**Avg. Mean**	**Avg. Stand. Dev.**	**Avg. Struct. Density**	**Avg. Essentiality**
CP-MKNN for Gavin Dataset					
Cores	67	0.56	0.14	0.84	0.38
Peripheries	45	0.28	0.024	0.52	0.28
Core Peripheries	101	0.35	0.083	0.49	0.33
ClusterONE-CP for Gavin Dataset					
Cores	22	0.52	0.15	0.91	0.37
Peripheries	52	0.32	0.06	0.58	0.16
Core Peripheries	149	0.38	0.12	0.57	0.34
CP-MKNN for Krogan Dataset					
Cores	88	0.89	0.093	0.58	0.37
Peripheries	60	0.51	0.096	0.52	0.16
Core Peripheries	112	0.77	0.095	0.41	0.32
ClusterONE-CP for Krogan Dataset					
Cores	14	0.93	0.059	0.70	0.23
Peripheries	55	0.48	0.079	0.54	0.22
Core Peripheries	247	0.77	0.13	0.45	0.29

Table 4. Comparison of relationships types obtained for CP-KNN (K=4) for Gavin dataset

Relationship Type	**# Relationships**	**Avg. Mean Difference**	**Avg. Cohesion Difference**
CP-Type 1	214	0.85	0.20
CP-Type 2	194	3.04	0.32
Periphery-Periphery	109	4.9	0.22

Next, we provide a motivating example of a real core- periphery and periphery-periphery relationship obtained by CP-MKNN on the Krogan PPI dataset in figure 11. For each core or periphery subunit, we also note down the MIPS gold standard complex it mapped on to. CP-MKNN finds a protein subset which maps to Exosome complex as a core for a periphery subset which maps to RNA-polymerase-III com- plex. Further, CP-MKNN finds a periphery-periphery relationship between this RNA-polymerase-III complex periphery and another periphery that maps to RNA-polymerase-II complex. It has been researched that exosome complex is a central factor in processing stable RNA species produced by RNA polymerases I, II, and III [41]. This suggests that both the core-periphery and periphery-periphery relations identified by CP-MKNN are meaningful and can be used to gain insights into the nature of complex networks.

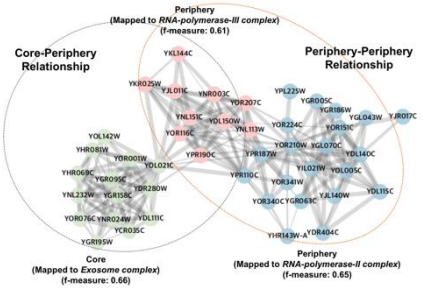

Figure 11. An example Core-Periphery and Periphery-Periphery relationship obtained by CP- MKNN on Krogan PPI dataset

6. CONCLUSIONS

To summarize, we developed a core periphery structure identification algorithm in weighted and undirected graphs using a heuristic measure called MKNN. We provide with a definition of core periphery structures suited for weighted graphs. We illustrated the usefulness of core periphery structures generated by CP-MKNN using synthetically generated datasets as well as real world social networks and biological PPI networks. We also provided a comparative analysis of core periphery results by CP-MKNN with those obtained by another core periphery algorithm called ClusterONE-CP. We demonstrate that CP-MKNN finds denser cores than ClusterONE-CP as per structure or edge weight density. We further scored and categorized the obtained core periphery structures and provided with comparative examples. We demonstrated that CP-MKNN is able to separate very dense cores as per structural density and edge weight density constraints from their peripheries in weighted graphs. These relationships can be very useful to gain insights about the nature of complex networks. This paper is a proof of concept of the algorithm for identifying core periphery structures in weighted graphs. As a future direction of work, there is scope for making the algorithm scalable for large graphs. Further, the core periphery structures defined in this paper can be applied to graphs which vary over time to see how the relationships and the core

periphery scores evolve over time. This can be helpful in studying how trends change over time in complex networks represented as weighted graphs.

REFERENCES

[1] A. Barrat, M. Barthelemy, R. Pastor-Satorras, and A. Vespignani, "The architecture of complex weighted networks," in Proc. Nat. Acad. of Sci. of the USA, 2004, vol. 101, no. 11, pp. 3747- 3752.

[2] Y.-Y. Ahn, J. P. Bagrow, and A. Lehmann, "Link communities reveal multi-scale complexity in networks," Nature, vol. 466, pp. 761–764, 2010.

[3] E.M. Airoldi, D.M. Blei, S.E. Fienberg, and E.P. Xing, "Mixed membership stochastic block models," J. Mach. Learn. Res., vol. 9, pp. 1981–2014, 2007.

[4] M. Sales-Pardo, R. Guimera, A. Moreira, and L.A.N Amaral, "Extracting the hierarchical organization of complex systems," in Proc. Nat. Acad. of Sci. of the USA, 2007, vol. 104, pp. 18 874–18 874.

[5] I. Psorakis, S. Roberts, M. Ebden, and B. Sheldon, "Overlapping community detection using Bayesian non-negative matrix factorization," Phys. Rev. E, vol. 83, 2011.

[6] G. Palla, I. Derenyi, I. Farkas and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," Nature, vol. 435, no. 7043, pp. 814–818, 2005.

[7] T.S. Evans, and R. Lambiotte, "Line graphs, link partitions, and overlapping communities," Phys. Rev. E, vol. 80, Art. ID. 016105, 2009.

[8] T. Nepusz, H. Yu, and A. Paccanaro, "Detecting overlapping protein complexes in protein-protein interaction networks," Nature Methods, vol. 9, pp. 471-472, 2012.

[9] J. Yang, and J. Leskovec, "Overlapping Communities Explain Core–Periphery Organization of Networks," in Proc. of the IEEE, 2014, vol. 12, pp. 1892-1902.

[10] J. Yang, M. Zhang, K.N. Shen, X. Ju, and G. Xitong, "Structural correlation between communities and core-periphery structures in social networks, Evidence from twitter data," Expert Systems with Applications, vol. 111, pp. 91-99, 2018.

[11] S.P. Borgatti, and M.G. Everett, "Models of Core/Periphery structures," Social Networks, vol. 21, pp. 375–395, 1999.

[12] D. Smith, and D. White, "Structure and dynamics of the global economy: network analysis of international trade 1965–1980," Social Forces, vol. 70, pp. 857–893, 1992.

[13] P. Krugman, The Self-Organizing Economy. Blackwell, United King- dom: Oxford, 1996.

[14] R.R. Faulkner, Music on Demand: Composers and Careers in the Hollywood Film Industry. New Brunskwick, NJ, USA: Transaction Publishers, 1987.

[15] E.O. Laumann, and F.U. Pappi, Networks of Collective Action: A Perspective on Community Influence Systems. New York, USA: Academic Press, 1976.

[16] P. Doreian, "Structural equivalence in a psychology journal net- work," American Society for Information Science, vol. 36, no. 6, pp. 411–417, 1985.

[17] F. Luo, B. Li, X. Wan, and R. Scheuermann, "Core and periphery structures in protein interaction networks," BMC Bioinformatics, vol. 10, pp. S8, 2009.

[18] J.N. Cummings, and R. Cross, "Structural properties of work groups and their consequences for performance," Social Networks, vol. 25, no. 3, pp. 197–210, 2003.

[19] T. Wang, C. Rudin, D. Wagner, and R. Sevieri, "Finding Patterns with a Rotten Core: Data Mining for Crime Series with Cores." Big Data, vol. 3, no. 1, pp. 3-21, 2015.

[20] D. Sardana and R. Bhatnagar, "Core Periphery structures in weighted graphs using greedy growth," in IEEE/WIC/ACM International Conference on Web Intelligence, 2016, pp. 1-8.

[21] M.G. Everett, and S.P. Borgatti, "Peripheries of cohesive subsets," Social Networks, vol. 21, pp. 397, 1999.

[22] M.P. Rombach, M.A. Porter, J.H. Fowler, and P.J. Mucha, "Core-periphery structure in networks," SIAM Journal on Applied Mathematics, vol. 74, no. 1, pp. 167-190, 2014.

[23] J.P. Boyd, W.J. Fitzgerald, and R.J. Beck, "Computing core/periphery structures and permutation tests for social relations data," Social Networks, vol. 28, pp. 166-178, 2006.

[24] F.D. Rossa, D. Fabio, and C. Piccardi, "Profiling core-periphery network structure by random walkers," Scientific Reports vol. 3, no. 1, pp. 1-8, 2013.

[25] S. Bruckner, F. Huffner, and C. Komusiewicz, "A graph modification approach for finding core–periphery structures in protein interaction networks," Algorithms for Molecular Biology, vol. 10, no. 1, pp. 16, 2015.

[26] J.V.L. de Jeude, G. Caldarelli, and T. Squartini, “Detecting core- periphery structures by surprise,” EPL (Europhysics Letters), vol. 125, no. 6, 2019.
[27] A. Elliott, A. Chiu, M. Bazzi, G. Reinert, and M. Cucuringu, "Core–periphery structure in directed networks," in Proc. Royal Society A, 2020, vol. 476, no. 2241.
[28] J.M. Kleinberg, “Authoritative sources in a hyperlinked environment,” Journal of ACM, vol. 46, pp. 604-632, 1999.
[29] Z. Hu, and R. Bhatnagar, “Clustering algorithm based on mutual K-nearest neighbor relationships,” Statistical Analysis and Data Mining: The ASA Data Science Journal, vol. 5, no. 2, pp. 100-113, 2012.
[30] D. Sardana, and R. Bhatnagar, “Graph Clustering Using Mutual K- Nearest Neighbors,” in Active Media Technology, 2014, pp. 35-48.
[31] D. Sardana, “Analysis of Meso-scale Structures in Weighted Graphs,” PhD diss., University of Cincinnati, 2017.
[32] J. Nešetřil, and P.O. De Mendez, “Sparsity: graphs, structures, and algorithms”, Springer Science & Business Media, vol. 28, 2012.
[33] D.E. Knuth, “The Stanford Graphbase: A platform for combinatorial computing,” New York: ACM Press, pp. 74-87, 1993.
[34] W. Zachary, “An information flow model for conflict and fission in small groups,” Journal of Anthropological Research, vol. 33, pp. 452-473, 1977.
[35] A.C. Gavin, M. Bosche, R. Krause, P. Grandi, M. Marzioch, A. Bauer, J. Schultz, J.M. Rick, A.M. Michon, C.M. Cruciat, and M. Remor, “Functional organization of the yeast proteome by systematic analysis of protein complexes,” Nature, vol. 415, no. 6868, pp. 141-147, 2002.
[36] N.J. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, A.P. Tikuisis, and T. Punna, “Global landscape of protein complexes in the yeast Saccharomyces cerevisiae,” Nature, vol. 440, no. 7084, pp. 637–643, 2006.
[37] H.W. Mewes, D. Frishman, U. Güldener, G. Mannhaupt, K. Mayer, M. Mokrejs, B. Morgenstern, M. Münsterkötter, S. Rudd, and B. Weil, “MIPS: A database for genomes and protein sequences,” Nucleic Acids Res, vol. 30, pp. 31-34, 2002.
[38] S. Brohee, and J.V. Helden, “Evaluation of clustering algorithms for protein-protein interaction networks,” BMC bioinformatics, vol. 7, no. 1, pp. 1, 2006.
[39] H. Jeong, S.P. Mason, A.L. Barabasi, “Lethality and centrality in protein networks,” Nature, vol. 411, no. 6833, pp. 41-42, 2001.
[40] W.H. Chen, P. Minguez, M.J. Lercher, and P. Bork, “OGEE: an online gene essentiality database,” Nucleic acids research, vol. 40, no. D1, pp. D901-D906, 2012.
[41] C. Kilchert, S. Wittmann, and L. Vasiljeva, “The regulation and functions of the nuclear RNA exosome complex,” Nature Reviews Molecular Cell Biology, 2016.

AUTHORS

Divya Sardana: Dr. Divya Sardana is a Senior Data Scientist at Teradata Corp., Santa Clara, CA, USA. She has a PhD in Computer Science from the University of Cincinnati, OH US. Her research targets development of scalable machine learning and graph algorithms for the analysis of complex datasets in interdisciplinary domains of data science.

Raj Bhatnagar: Dr. Raj Bhatnagar is a Professor at the department of EECS at University of Cincinnati, Cincinnati, OH, USA. His research interests encompass developing algorithms for data mining and pattern recognition problems in various domains including Bioinformatics, Geographic Information Systems, Manufacturing, and Business.