Journal of Theoretical and Applied Information Technology
15th August 2019. Vol.97. No 15

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

GRAPH MINING TECHNIQUES FOR GRAPH CLUSTERING: STARTING POINT

[1]RASHED SALEM, [2] WAFAA ABDEL MONEIM, [3] MOHAMED HASSAN

[1]Faculty of Computers and Information, Menoufia University, Egypt

[2] Faculty of Computers and Informatics, Zagazig University, Egypt

[3] Faculty of Computers and Informatics, Zagazig University, Egypt

E-mail: [2]eng_is_wafaa@yahoo.com

ABSTRACT

Nowadays, large number of applications of graph clustering are available, with expanding the span of the graph the conventional methods of clustering are not appropriate to manipulate this issue which are costly for computation. So that, it is necessary to get a good algorithm to tackle this problem. Graph clustering algorithms are considered as the most effective techniques for solving various partitioning problems. Global graph clustering which based on the whole graph as input isn't convenient of large graphs. Local graph clustering algorithms solve this problem by working on a given vertex as input seed set without looking at the whole graph to find a good cluster. This research explores different graph clustering techniques based on the input parameters, *e.g*., local and global, as well as illustrating appropriate applications of graph clustering. This paper directed to help new researchers take a summary of graph clustering techniques that can be used for graph partitioning.

Keywords: *Graph Mining, Graph Clustering Methods, Big Graph Mining, Global Graph, Local Graph.*

1. INTRODUCTION

Datasets are identified by the big data term according to large size and complexity. The traditional techniques such as data mining methods cannot manipulate big data. Extracting useful knowledge or hidden pattern from these large datasets based on its velocity, variety, volume, veracity, and value is called big data analytics, that is a challenge of big data [1]. Graph databases are increased in everywhere. Structural relationships between objects build a graph model. There are many applications for graph model such as social networking, biology, chemistry, image processing, web link analysis, computer networks, and human genome assembly. Graph mining is the process of dealing with graph data by utilizing the methods of machine learning and data mining to detect useful and unexpected patterns [2]. Big graph mining is the process of extracting significative information from huge data of graph which reaches Tera and Petabytes [3].

Clustering is the unsupervised procedure. The process of splitting a set of input data into two categories is called clustering. Based on similarity measurements, similar objects are contained within the same clusters and dissimilar objects are dispersed across various clusters. Graph clustering is the process of dividing the graph vertices into groups based on if there is inside the group high edge density and outside the group low edge density. A cluster is formed as a group of vertices.

Review papers [4, 5] introduce graph clustering definitions, and cluster quality measures as well as the types of graph clustering algorithms, i.e., global and local. Using the whole graph as input for the clustering process is called global graph clustering, however using a certain seed vertex for the clustering process is called local graph clustering. There are many applications of graph clustering such as correlation clustering, graph partitioning, community detection, a protein-protein network, etc. Many algorithms of global clustering are described in [6,7,8, 9, 10, 11, 12].

There are many graph clustering algorithms for community discovery problem include Metis [13], Graclus [14] and Markov clustering [15]. Moreover, local graph clustering algorithms are used to deal with community detection such as social and web graphs, *e.g*., [16, 17, 18, 19, 20, 21, 22, 23, 24]. Many other local graph clustering algorithms with powerful guarantees are presented in [25, 26, 27, 28].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Conductance is utilized to mensuration the connectivity among vertices in a graph. Set conductance is calculated through the proportion of edges count departuring the set to the edges count touched by the set of vertices. The small conductance value stands for many internal edges within the set and few edges outside it. An efficient cluster is chosen by a partition of vertices whose outer links are smaller than its inner links.

2. GRAPH CLUSTERING TECHNIQUES

Figure 1 overview different techniques of graph clustering. Graph clustering algorithms are divided into global and local techniques based on the input parameter.

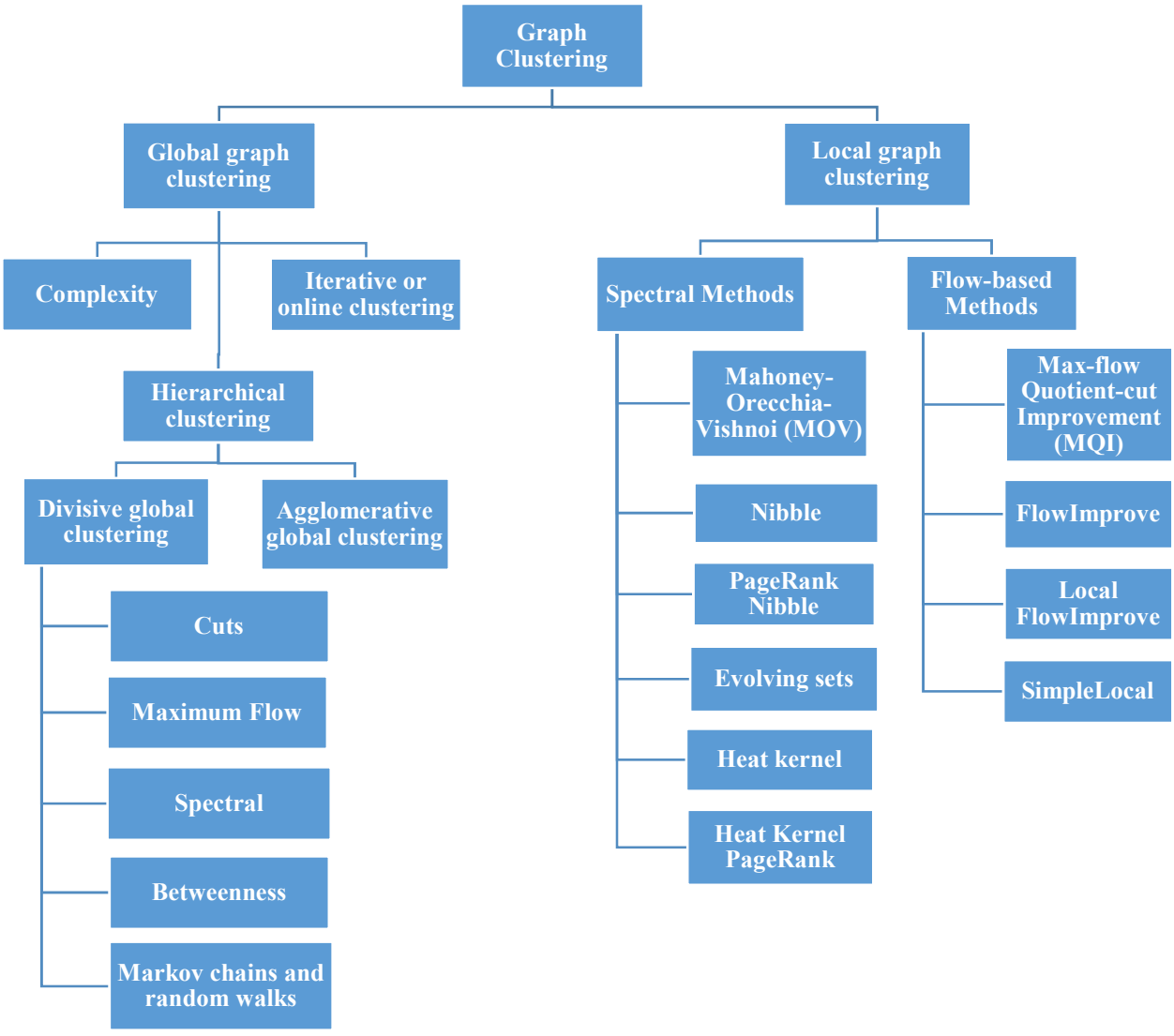

Figure 1: Different Techniques of Graph Clustering.

3. PRELIMINARIES

An undirected graph is represented by $G(N, E)$, which N is the vertex and E is the edge in the graph that connects the graph nodes. The graph nodes number is $n = |N|$, and the edges number is $m = |E|$. The vertex degree indicated by $d(x)$ is calculated by the edges count related to it. Set S is a parition of vertices, $vol(S)$ is the volume of S calculated by $\sum_{x \in s} d(x)$, $\partial(S)$ is the boundary of S

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

known by edges number departure the set and calculated by {(x, y)∈ E | x ∈ S, y ∉S}. A cluster conductance of S (ϕ(S)) is calculated by | ∂ (*S*) | / min(*vol(S)*, 2*m-vol(S)*). The conductance is used to mensuration the efficiency of the cluster. When the conductance value is small this means that high efficiency of the cluster, and when the conductance value is large this means that low efficiency of the cluster. The set *S* connect by edges with the graph remainder $\bar{S}$= *X* \ *S,* cut(*S*) = cut(*S*; $\bar{S}$). The vertices that share an edge with *S* but are not belonged to *S* stand for *Neigh*(*S*) that is calculated by *Neigh*(*S*) = {x ∈ $\bar{S}$: (x, *s*) ∈ *E* for some *s* ∈*S*}.

4. GLOBAL GRAPH CLUSTERING TECHNIQUES

This section discusses the methods of a global clustering for the input graph. Several researchers apply the global clustering of sparse graphs of million nodes [29, 30, 31]. In a global clustering, each vertex of a graph input is allocated to a cluster. Newman [32] previews global clustering methods.

4.1 Complexity

This part displays some related problems of clustering with reference to a distance function:

- **Minimum *k*-clustering:** *k* clusters is the goal of data set partitioning that is dependent on minimizing the distance between clusters. Approximation of this problem within a factor of two.
- **Minimum *k*-centre:** a group of centres is constructed and minimized the distance between vertex and the nearest centre. Approximation factor of this problem is two.
- **Minimum *k*-median:** minimizing the sum of the distances between the vertex and the nearest centre whereas maintaining the centre set order to be fixed. Approximation of this problem within a factor near two.
- ***K*-means algorithm [33]:** *K*-means is one of the most popular algorithms for clustering. The process of clustering points set into *n* cluster by repeatedly providing the middle points of *n* clusters and everyone point is collected to group based on the nearest middle point. Choosing the middle point is based on minimizing the squares sum of the distances of intracluster.

4.2 Iterative or online clustering

The clustering has two cases are first whole data points are clustering at once, and the second is one object is cluster at a time to a convenient cluster. In iterative clustering, the group tasks that are performed for items when processed first can be considered either unchangeable or can be changed later to improve some computed clustering properties. Figure 2 displays the iterative clustering algorithm.

Iterative Improvement Algorithm:

```
Choose initial partition C1, … , Ck
repeat {
    unlock all vertices
    repeat {
        choose some Ci at random
        choose an unlocked vertex vj in Ci
        move vj to that cluster, if any, such that move
            gives maximum decrease in cost
        lock vertex vj
    } until all vertices locked
}until converge
```

Figure 2: the iterative clustering algorithm.

The process of online clustering includes one datum every time for running the clustering algorithm, knowing only the data that has been processed previously and has been used the Internet to be running. These online clustering algorithms supply a fractional process of data clustering which is coming from an anonymous stream of data to be clustered.

The online clustering allocates the clusters number to be utilized dynamically, usually based on a certain value of threshold to detect when a novel arriving datum have to be specified to a novel cluster rather than being combined into a present cluster. Several researches such as found in [34, 35, 36, 37] study the method of an online clustering for graphs.

4.3 Hierarchical clustering

Global clustering is defined by the term hierarchical structure, whereas each set of subsets is used to consist the upper level, and so on. This is even more useful for structuring the hierarchical graph to obtain more grouping or two groups are combined to get a more section into clusters. A dendrogram is the hierarchy clustering like a tree. Figure 3 shows the hierarchy clustering dendrogram.

There are two groups of hierarchical clustering algorithms:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- **Divisive (top-down)**

In divisive approach, it starts with all points in the same group, in each iteration the cluster is divided up into smaller groups depending on their properties, the process is terminated when each object belongs to one cluster [38, 39, 40, 41, 42, 43, 30, 44, 45].

- **Agglomerative (bottom-up)**

In agglomerative approach, forming an independent group is realized by starting with each point. It merges objects or groups that are adjacent to each other and continues until all the groups are combined into one [46, 47, 48, 49].

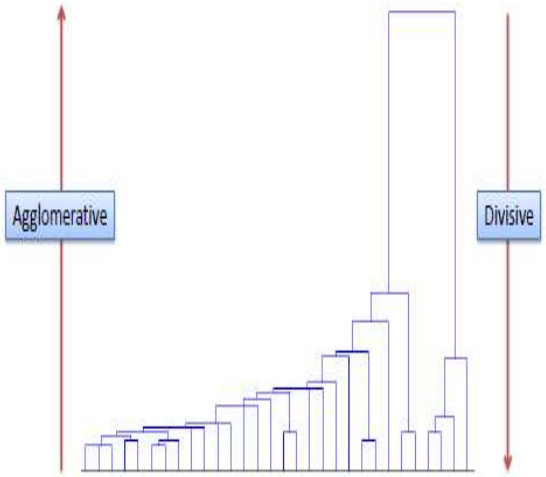

Figure 3: the hierarchy clustering dendrogram.

4.3.1 Divisive global clustering

Divise analysis called DIANA which is the inverse of agglomerative algorithm. Firstly, it starts with the root, whereas a lone cluster includes all objects. In each iteration, the different cluster is split into two groups. The process is repeated until all objects are included in their possess cluster. In this section, the diverse standard for defining the position to split the graph are described.

4.3.1.1 Cuts

The graph is partitioned into two groups by removing the cut. There are various types of graph which are used for the cut including undirected and directed graphs. The graphs can also be weighted or unweighted. The algorithm of the problem maximum-flow minimum-cut is described in [50, 51, 52]. The dense graph is cut into two or more groups rather than partition the graph into two groups. The division graph criterion is a low conductance [53, 54, 55]. The idea of the conductance depends on the boundary of the cluster to get excellent values rather than the clusters.

4.3.1.2 Maximum Flow

A flow network structure is known as a directed graph which includes S standing for a source, T standing for a sink and many vertices linked with edges. Capacity is defined for each edge, which is the maximum value of a flow that allowed for the edge.

The conditions of the flow network are described here:

- The input flow is equivalent to the output flow of any vertex.
- The flow of each is between 0 and capacity.
- The sum of flow outside of the source vertex is equivalent to the aggregate of flow inside of the sink vertex.
- Skew symmetry is used for edges flow network.

The maximum flow is characterized as the maximum flow value which network could permit to flow from source to sink. Multiple techniques are used to fix the problem of maximum flow such as Ford-Fulkerson algorithm [56], Dinic's Algorithm [57], Edmond-Karp [58], and Goldberg-Tarjan [59].

4.3.1.3 Spectral

The methods of spectral clustering are investigated in the survey [60, 61]. Spectral clustering [62] measures the matrix of similarity by depending on the eigenvector to split objects into clusters whereas the objects with high similarity are positioned in the same group and the objects with low similarity are positioned in the various clusters. The similarity matrix is used to build a graph, whereas objects represent vertices and the objects similarities represent the weights of the edge.

By analyzing the cluster, it contains more various objects functions based on the theory of spectral graph like MNCut [63], RatioCut [64], NCut [63], and MinCut [65].

$$\mathrm{MNCut}(\Delta)=\sum_{i=1}^{k}\frac{\mathrm{cut}(A_i,\bar{A}_i)}{\mathrm{vol}(A_i)}$$

where $\bar{A}_i$ stands for the complement of A.

The popular function of similarity is Gaussian kernel. The similarity is denoted by w_{ij} between two objects, which is computed by:

$$w_{ij}=\exp\left(\frac{-d^2(s_i,s_j)}{\sigma^2}\right)$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

While the Euclidean distance is $d(s_i, s_j)$ between two objects, s_i and s_j, the parameter σ dominances the neighborhood widths [66]. Figure 4 shows the spectral clustering algorithm.

Spectral clustering Algorithm:-

Input: A set of points, S = {s_1, s_2, …, s_n}, cluster number *K*.

Output: A partition, Δ = {A_1, A_2, …, A_K}.

1. Compute similarity matrix *W*.
2. Compute Laplacian matrix $L = D - W$.
3. Compute Normalized Laplacian matrix $L = I - D^{-1}W$
4. Compute the first K eigenvectors of *L*, denote as *U*.
5. Consider the rows of *U* as data points, and use *k*-means to cluster them into *K* clusters.
6. Assign s_i to cluster A_j if and only if row *i* of the matrix *U* was assigned to cluster A_j.

Figure 4: Spectral Clustering Algorithm.

There are many advantages of spectral clustering which are simple to execute, the results of clustering are good. They do not provide robust assumption on the cluster's statistics, and for many thousand points of sparse data is rapid. The spectral clustering disadvantages include expensive for computation of the large dataset and the parameters choice may be critical.

4.3.1.4 Betweenness

Newman and Girvan [31, 44] force weights on the edges which depend on features of the graph structure *G* to cluster an unweighted graph *G* = (*N*, *E*). The edge betweenness of (*x*, *y*) is the shortest routes number that linking any two nodes which use the edge to pass through.

The steps of the algorithm for community detection are listed as follows:

1. Firstly, the whole present edge betweenness in the network is computed.
2. The high betweenness of the edge is deleted.
3. Whole edges betweenness which influenced by the deleted is recomputed.
4. Iterate step 2 & 3 until there is no edges rest.

The betweenness of the edge e:

$$\sum_{s,t \neq v} \frac{\sigma_{st}(e)}{\sigma_{st}}$$

where the shortest routes number from *s* to *t* is represented by σ_{st} and $\sigma_{st}(e)$, *i.e.*, shortest paths number which use the edge *e* to pass through.

4.3.1.5 Markov chains and random walks

Firstly, a graph *G* and an edge *v* are given, the vertex neighbor is selected randomly, then shift to this neighbor, after that this vertex neighbor is selected and shift to it, and so on. The way of selecting vertices randomly is called a graph random walk.

A finite Markov chain with time-reversible is called a random walk [67, 68]. There is no big distinction between graph random walks theory and finite Markov chains theory. On a directed graph that have weights, each Markov chain is considered as a random walk. On undirected graphs, Markov chain with time-reversible is seen like random walks, and on regular symmetric graphs, symmetric Markov chains are seen like random walks.

The process of stochastic process is described by a Markov chain through states set based on the matrix of a transition probability.

Graph *G* = (*N*, *E*) is donated as input, the matrix of adjacency denotes by *W*. The transition probabilities matrix of this Markov chain denotes by $M = (P_{ij})\ i, j \in V$

$$p_{ij} = \begin{cases} \frac{1}{d(i)}, & if\ ij \in E, \\ 0, & otherwise. \end{cases}$$

The diagonal matrix is represented by $D_{ii}=1/d(i)$ and the initial distribution by P_0. The stationary distribution is calculated by $\pi(v) = d(v)/2m$. Figure 5 presents Random Walk algorithm.

The advantages of random walk are simple to perform, no need to know knowledge of graph, and the footprint of the memory is small. The random walk disadvantages are unexpected cover, infinite case is worst and time consuming.

Random Walk algorithm.

Input:

- the adjacency matrix *W* of a graph G(*V*,*E*)
- A subset of nodes V_C having property *C*
- Initialization of nodes:

if $v \in V_C$ then $p_0(v) = 1 / |V_C|$ else $p_0(v)=0$

- Set transition matrix: $M = D^{-1}W$ where *D* is a diagonal matrix with $d_{ii} = \Sigma_j w_{ij}$
- Iteratively update until convergence or until $t=k$

$p_t = M^T p_{t-1}$

Output: p_t

Figure 5: The Random Walk algorithm.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.3.2 Agglomerative global clustering

Agglomerative clustering is also called AGNES. This algorithm is considered bottom-up approach, whereas each point is treated as one cluster (leaf). The two similar clusters are merged into novel cluster at each step. This process is repeated till all objects are combined into one cluster (root).

The steps of AGNES algorithm are listed as follows:

1. Each object is allocated into one cluster and the similarities is detected between clusters to the same as similarities the objects they include.
2. Clusters pair that are extreme similar is detected and merged then into one cluster.
3. The novel cluster needs to calculate the distances between it and each older clusters.
4. Step 2 & 3 are repeated until clustering whole objects into one cluster with *N* size.

5 LOCAL GRAPH CLUSTERING TECHNIQUES

This section discusses the methods of a local graph clustering usage. The local graph clustering algorithms take a seed node or group of seed nodes as input parameter. These techniques are divided into two groups which are spectral methods and Flow-based methods.

5.1 Spectral Methods

There are widespread methods in machine learning, applied mathematics, and data analysis which are spectral methods because of their best achievement in many application and strong theory. In undirected graphs, spectral methods perform a serious part, the graph has numerous properties based on quantities of spectral linked with matrices which demonstrate the graph. Networks are split large clusters by spectral methods which the edges number among cluster are minimized or the measurement of quality is maximized like modularity. Many techniques of spectral clustering are described here.

5.1.1 Mahoney-Orecchia-Vishnoi (MOV)

Mahoney, Orecchia, and Vishnoi [23] introduce a technique to build a locally-biased analogue of the second eigenvalue and its associated eigenvector. Theoretically and empirically, they are explained that this localized vector takes many of the perfect characteristics of the global second eigenvector. The basic advantage of this method is that the perfect solution for LocalSpectral collects a lot of the same structural characteristics as the global eigenvector, excluding in the locally biased setup. This method utilizes concepts from graph theory [69].

The LocalSpectral Optimization Program

The standard spectral optimization program is augmented with the constraint that the output cut well-correlated with the input seed set to describe the LocalSpectral. The inputs of LocalSpectral are (*A*, *s*, *k*) whereas *A* is donated as a graph, *s* as a seed vector and *k* as a correlation parameter, in addition to working in a vector space $\mathbb{R}^V$. The LocalSpectral (*A*, *s*, *k*) is the program of locally-biased spectral that is characterized as

$$\begin{aligned} \text{Min} \quad & y^T L_A y \\ \text{s.t.} \quad & y^T D_A y = 1 \\ & (y^T D_A I)^2 = 0 \\ & (y^T D_A s)^2 \geq \kappa \\ & y \in \mathbb{R}^V \end{aligned}$$

LocalSpectral optimal solutions are described and displayed that nearly-linear time is used to build a solution by fixing a linear equations system. The LocalSpectral (*A*, *s*, *k*) optimal solution is calculated effectively.

5.1.2 Nibble

Spielman and Teng [70, 71] present the premier nearly linear time for detecting an approximate sparsest cut *S* of a *G* graph. Nibble algorithm starts with a seed vertex υ depending on using the random walk algorithm. The inputs of nibble algorithms are a graph *G*, a seed edge υ, a conductance ϕ where ϕ ∈(0, 1), a positive integer *b*, and number of steps *t*. Nibble algorithm calculates a sweep cut on the weighted vector that calculated before on each step to retrieve the cluster with the small conductance value, if it is not obtained it continues running. The Nibble algorithm runs in time $O\ (2^b\ (\log^4 m)/\phi^5)$ and requires $\Phi(C) = O\ (\phi 3/\log^2 m)$. Figure 6 describes the pseudocode of nibble algorithm.

Nibble algorithm is based on approximating the random walk distributions for *t* steps. The random walk can be computed by $P = (GD^{-1}+I)/2$, where *G* refers to an unweighted graph and *D* stands for the diagonal matrix. The random walk distribution is represented by $P(\upsilon)$ beginning with node υ. To round the random walk, the truncation process is calculated by

$$p(\upsilon) = \begin{cases} p(\upsilon) & if\ p(\upsilon) \geq 2\epsilon d(i), \\ 0 & otherwise \end{cases}$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5.1.3 PageRank-Nibble Algorithm

Andersen, et al. [72] improved the Nibble algorithm running time O ($2b$ log^4 m/ϕ^5) and approximation ratio by producing PageRank-Nibble algorithm. The execution time of PageRank-Nibble is O (2^b $\log^3$ m/ϕ^2). To make the cuts, the PageRank-Nibble uses personalized PageRank vectors. The inputs of PageRank-Nibble are a seed vertex υ, a teleportation constant α where α ∈ [0, 1], and an integer b. The PageRank is calculated by

$$pr(\alpha, s) = \alpha s + (1 - \alpha)\, pr(\alpha, s)\, W,$$

where α is a teleportation a constant ∈[0, 1], s is a distribution (preference vector), and W stands for a random walk transition.

Input: graph G, vertex v, conductance $\phi \in (0, 1)$, positive integer b.
Output: C.
Initialize:
$\epsilon = 1 / (c_3 (\ell + 2)\, t_{last}\, 2^b)$.
$q_0 = \chi_v$ and $r_0 = [q_0]_\epsilon$.
For $t = 1$ to t_{last}
$q_t = Mr_{t-1}$.
$r_t = [q_t]$.
If there exists a j in which
$\Phi(S_j(q_t)) \le \varphi$,
$\lambda_j(q_t) \le (5/6)\mu(V)$,
$2^b \le \lambda_j(q_t)$, and
$I_x(q_t, 2^b) \ge 1/c_4(\ell + 2)2^b$,
Then return $C = S_j(q_t)$ and quit.
End For.
Return C = ∅.

Figure 6: Pseudocode of Nibble algorithm.

There are two distributions p and r which are used to approximate a PageRank vector pr (α, s). They have the property

$$p + pr(\alpha, r) = pr(\alpha, s).$$

Where p is an approximate PageRank vector and r is a residual vector. Figure 7 presents the PageRank-Nibble algorithm.

5.1.4 Evolving sets

Andersen and Peres [73] introduce EvoCut algorithm that relies on evolving sets. This method emulates the volume-biased evolving set process to detect a sparse cut. Furthermore, it is local clustering based on random which is a Markov chain on vertices sets. This method includes local approximation guarantee O ($\phi^{1/2}$ $\log^{1/2}$ n) and expected work/volume ratio O ($\phi^{-1/2}$ polylog (n)).

PageRank-Nibble (v, ϕ, b):

Inputs: a vertex v, a constant $\phi \in [0, 1]$, and an integer $b \in [1, B]$, where $B = [log_2 m]$.

Let $\alpha = \frac{\phi^2}{225 \ln(100\sqrt{m})}$.

Compute an approximate PageRank vector p = apr (α, χ_v, r) with residual vector r satisfying $\max_{u \in V} \le 2^{-b} \frac{1}{48B}$.

Check each set S_j^p with j ∈[1, |Supp (p)|], to see if it obeys the following conditions:

- Conductance: $\Phi(S_j^p) < \phi$,
- Volume: $2^{b-1} < \text{vol}(S_j^p) < \frac{2}{3}\text{vol}(G)$,
- Probability Change: $p[2^b] - p[2^{b-1}] > \frac{1}{48B}$,

If some set S_j^p satisfies all of these conditions, return S_j^p. Otherwise, return nothing.

Figure 7: PageRank-Nibble algorithm.

A Markov chain on a group of the vertex set of a graph is called evolving set process (ESP). The Markov chain transition rule is an easy technique which increases or decreases the present set. A Markov chain on V subsets is defined as the volume-biased evolving set process (volume-biased ESP) with a transition kernel that is calculated based on the following equation:

$$\hat{K}(S, \hat{S}) = \frac{\mu(\hat{S})}{\mu(S)} K(S, \hat{S})$$

where $K(S, \hat{S})$ refers to the ESP transition kernel.

The volume-biased ESP is simulated by GenerateSample while reaching to a stopping time τ. The inputs of the GenerateSample(x, T, B) method are a vertex x, an integer $T \ge 0$ and an integer $B \ge 0$. The method creates (S_0, . . . , S_τ) as a sample path and makes the set S_τ as output. Figure 8 displays the GenerateSample algorithm.

The input of EvoCut (v, ϕ) algorithm is a node $v \in V$ as well as a conductance $\phi \in (0, 1)$ as target, and generates vertices set as output. Figure 9 shows the EvoCut algorithm.

5.1.5 HeatKernel

The heat kernel [74] is a deterministic approach, it begins with seed vertices for recognizing a community, this is a type of graph diffusion. Adjacency matrix is represented by A and D stands for the diagonal matrix of degrees that is calculated by $D_{ii} = di$. The random walk transition matrix is computed by $W = (D^{-1}A)^T = AD^{-1}$. A graph diffusion is calculated by the equation

$$\text{df} = \sum_{i=0}^{\infty} \alpha_i W^i s \quad (1)$$

where $\sum_i \alpha_i = 1$ and s is a stochastic vector. A small conductance community is calculated by a sweep procedure using a diffusion f estimate from a seed.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The heat kernel equation substitutes α_k with $t^i / i!$

$$hk = e^{-t}(\sum_{i=0}^{\infty} \frac{t^i}{i!}(W)^i)\ s = \exp\{-t(I-W)\}s \quad (2)$$

This algorithm is called HK-relax because of using coordinate-relaxation method for approximating h to execute this first approximate exp $\{tW\}$ with its degree N Taylor polynomial, $TN(tW)$ then computes $TN(tW)s$. An equation that uses Taylor polynomial to compute an approximation for a matrix G is:

$$\exp\{G\} = \sum_{i=0}^{\infty} \frac{1}{i!} G^i \approx \sum_{i=0}^{N} \frac{1}{i!} G^i \quad (3)$$

GenerateSample(x, T,B):

Input: is a starting vertex x, and two integers $T \geq 0$ and $B \geq 0$.

Output: is a set S sampled from the volume-biased ESP with stopping rule $\tau = \tau(T, B)$.

Internal state:

S = an instance of the set-with-boundary data structure.

X = the current location of the random walk particle.

The algorithm also maintains the current values of $\hat{\sigma}(S)$, $\mu(S)$, and cost $(S_0, \ldots, S)$.

Algorithm:

Initially, let $S = S_0 = \{x\}$ and $X = x_0 = x$.

At the beginning of step t, we have $S = S_{t-1}$ and $X = X_{t-1}$.

For $t = 1 \ldots \tau$, do step t as follows:

Stage 1. Select the vertices to add or remove from S:

(a) Given $X_{t-1} = x_{t-1}$, select $X_t = x_t$ with probability $p(x_{t-1}, x_t)$ and update $X \leftarrow x_t$.

(b) Lookup $p(x_t, S_{t-1})$ and pick Z uniformly at random from the interval $[0, p(x_t, S_{t-1})]$.

(c) Define $S_t = \{y \mid p(y, S_{t-1}) \geq Z\}$.

(d) Compute a list D of the vertices in $S_t \Delta S_{t-1}$ as follows.

For each $y \in \delta(S_{t-1})$:

i. Lookup $p(y, S_{t-1})$.

ii. If $y \in S_t \Delta S_{t-1}$, then add y to D.

(e) Update $\mu(S_t)$ and cost $(S_0, \ldots, S_t)$.

(f) If $t = T$ or cost $(S_0, \ldots, S_t) > B$, then $t = \_$, so halt and output $S_t = S_{t-1} \Delta D$.

Otherwise, proceed to the next stage.

Stage 2. Update S:

(a) Update S to $S_t = S_{t-1} \Delta D$ by adding or removing the vertices in D from S.

(b) Update $\hat{\sigma}(S_t)$ and compute $\phi(S_t) = \hat{\sigma}(S_t) / \mu(S_t)$.

(c) If $\phi(S_t) < \theta_T$, then $t = \tau$, so halt and output S_t.

Otherwise, proceed to the next step.

Figure 8: The GenerateSample Algorithm.

EvoCut(v, ϕ):

1. Let $T = (\phi^{-1}/100)$. If $T = 0$, then output $\{v\}$.

2. Let S = GenerateSample(v, T, ∞), and output S.

Figure 9: The EvoCut Algorithm.

The HK-relax algorithm

The inputs of this algorithm are transition matrix W of a random walk, s as a seed vector, and scalar $t > 0$. The following steps are followed to solve the linear system:

1- The vector of the solution is indicated by y and the start $nN \times 1$ residual by $r^{(0)} = e1 \otimes s$. The vertex i in residual block j is used to indicate the entry of $r(i, j)$.

2- All inputs from r that accept the following equation are removed frequently

$$r(i,j) \geq \frac{e^t \varepsilon d_i}{2N\psi_j(t)} \quad (4)$$

3- Start by filling the queue $Q(r)$ with the nonzero values of $r^{(0)}$ and replace $Q(r)$ values with the updates values of r if they satisfy the previous equation.

4- In each stage, pop the top entrance of $Q(r)$, recall it $r(i, j)$, and take out that entrance in r, doing $r(i, j) = 0$.

5- Append $r(i, j)$ to y_i.

6- Append $r(i,j)\frac{t}{j+1}W_{ei}$ to residual block r_{j+1}.

7- At each entrance of r_{j+1} that was upgraded, add that entrance to the backwards of $Q(r)$ if it accepts (4).

The Pseudo-code for the HK-relax technique is presented in figure 10.

5.1.6 HeatKernel PageRank

Heat Kernel PageRank [75] is an altered version of PageRank. The parameters of heat kernel pagerank are a seed node and heat or temperature constant. This algorithm is a random walk exponential sum from a seed vertex, measured by the temperature.

A $G = (V, E)$ stands for a graph, the random walk transition probability matrix W is a matrix catalogued by V and is calculated by

$$W(u,v) = \begin{cases} \frac{1}{d_u} & if\ \{u,v\} \in E, \\ 0 & otherwise \end{cases}$$

Where d_v stands for the vertex degree, the adjacency matrix is calculated by $W = D^{-1}A$, and the diagonal degree matrix is denoted by D. A G graph includes a random walk with a stationary distribution π if G is linked and non-bipartite. The stationary distribution π is computed by $\pi(u) = d_u / \Sigma_v d_v$.

Two parameters are α and the vector of preference f are used for PageRank $pr_{\alpha, f}$ that is computed by

$$pr_{\alpha,f} = \alpha f + (1-\alpha)\, pr_{\alpha,f} W.$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The heat kernel pagerank is computed by using two factors which are $t \geq 0$ as the temperature and a vector of preference f. The following equation presents the heat kernel pagerank:

$$p_{t,f} = e^{-t} \sum_{k=0}^{\infty} \frac{t^k}{k!} f W^k.$$

```
Input: graph G
parameters: seed, t, eps, N, psis
Initialize: residual r, queue Q
for s ∈ seed
    r[(s ,0)] ← 1./ len (seed )
    Q. append ((s ,0))
End For
while len(Q) > 0 do
  (v, j) ← Q. popleft
  rvj ← r[(v, j)]
  If v ∉ x Then
     x[v] += rvj
     r[(v, j)] = 0
     Mass = (t*rvj / (float (j) +1.))/ len (G[v])
  End If
  For u ∈ G[v]
     next ← (u,j+1)
     If j+1 == N Then
        x[u] += rvj /len(G(v))
        continue
     If next ∉ r Then
        Thresh ← math .exp(t)* eps*len(G[u])
        Thresh ← thresh / (N*psis [j +1])/2.
     End If
     If r[ next ] < thresh and r[ next ] + mass >= thresh Then
        Q. append( next )
     End If
     r[ next ] ← r[ next ] + mass
  End For
End While
```

Figure 10: The Pseudo-code for HK-relax algorithm.

Chung and Simpson [76] present the heat kernel pagerank approximation of a graph. Figure 11 describes ApproxHKPRseed algorithm. This algorithm approximates $p_{t,u}$ by an ϵ-approximate vector which we denote by $\widehat{p}_{t,u}$. The algorithm run time is $O(\frac{\log(\epsilon^{-1})\log n}{\epsilon^3 \log\log(\epsilon^{-1})})$.

ApproxHKPRseed (G, t, u, ϵ)

Input: a graph G, $t \in \mathbb{R}^+$, seed vertex $u \in V$, error parameter $0 < \epsilon < 1$.

Output: ρ, an ϵ-approximation of $\rho_{t,u}$.

Initialize a 0-vector ρ of dimension n, where $n = |V|$

$r \leftarrow \frac{16}{\epsilon^3}\log n$

$K \leftarrow c.\frac{\log(\epsilon^{-1})}{\log\log(\epsilon^{-1})}$ for some choice of constant c

For r iterations **do**

Start

Simulate a P random walk from vertex u where k steps are taken with probability $e^{-t}\frac{t^k}{k!}$ and $k \leq K$

Let v be the last vertex visited in the walk

$\rho[v] \leftarrow \rho[v] + 1$

End

End for

Return $1/r \cdot \rho$

Figure 11: The ApproxHKPRseed Algorithm.

5.2 Flow-based Methods

A flow-based method is used for improving graph cuts. These algorithms use network flow thoughts to uncover graph bottlenecks and route as much flow as possible. Many flow-based methods are described here.

5.2.1 Max-flow Quotient-cut Improvement (MQI)

Max-flow Quotient-cut Improvement (MQI) algorithm [77] is the fast exact a flow-based algorithm. MQI improves the expansion or the cuts conductance of unweighted or weighted graphs. A heuristic partitioner of graph is get by merging MQI with Metis. This method detects the best advanced between whole cuts. The input parameters of MQI are undirected graph G and initial cut. The initial graph cut contains two sets which are X and $Y = \bar{X}$ where is the X complement. A novel directed graph is constructed dependent on the following steps which starts with a version of graph.

1. Whole Y-side nodes are discarded.
2. Each edge that links a pair of Y vertices is discarded.
3. Each edge that links a pair of X vertices is changed with a two of directed edges, everyone with weight a.
4. Source S and Sink T are added.
5. Every edge that links a Y node with an X node x is changed with an individual directed edge begin with S to x, with weight a.
6. An individual directed edge starts from everyone X vertex to T is inserted, with weight c.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The overall value of a flow is merged with the value of flow cross everyone edge to construct the solution of this max flow problem.

5.2.2 Flow Improve

Andersen and Lang [78] introduce Improve algorithm. The input of Improve algorithm is a vertices subset R and the output is a novel vertices subset S that has a minimum quotient cut result. This algorithm detects this set S by building and fixing a series of s-t minimum cut problems.

The augmented graph G_R (α) is built based on the input graph G, the set R and a parameter α ∈ (0, 1). The two nodes of the minimum cut problem which are source s and sink t are added into a group of vertices of a G graph to construct a subset of G_R (α). The edges set of G_R (α) consists of the edges set of R plus their weights, as well as adding edges start from s to any node v in R with the weight α vol (v) and edges begin with any node v in $V\backslash R$ to t with the weight α vol (v) $f(R)$, where $f(R)$ = vol (R)/ vol ($\bar{R}$) ≤ 1. The sum of edges weight linked to s is equivalent to the sum of edges weight linked to t. The problem of cut is to partition s and t in the augmented graph G_R (α) where s ∪ S and t ∪ $V\backslash S$. Figure 12 shows Flow Improve algorithm.

FlowImprove(R)
Input: a set $R \subseteq V$ satisfying vol (R) ≤ vol ($\bar{R}$).
Output: a new set $S \subseteq V$.
Initialization:
Let $S_0 = R$, let $i = 0$
Let $\alpha_0 = \tilde{Q}_R(R) = Q(R) < \infty$.
Main loop:
Compute the minimum s-t cut in the graph $G_R(\alpha_i)$, and let S_{i+1} be the subset of V corresponding to this cut.
Let $\alpha_{i+1} = \tilde{Q}_R(S_{i+1})$.
If $\alpha_{i+1} < \alpha_i$, let $i \leftarrow i + 1$ and repeat the loop. Otherwise halt and output S_i.

Figure 12: Flow Improve algorithm.

5.2.3 Local Flow Improve

Orecchia and Zhu [79] present the initial algorithm that merges the methods of spectral and flow to get the best approximation value for local graph clustering. This paper introduces LocalImprove that is the initial local algorithm of cut improvement, whereas the run-time is based on the input set size A instead of the whole graph size. Orecchia and Zhu [79] introduce two local algorithms, LocalFlow and LocalFlow^{exact}. The parameters of the algorithm are a graph G, a vertex subset $A \subset V$, a constant α ∈(0, 1), and a constant ε_σ.

The premier technique, LocalFlow combines the local and approximate value of maximum flows. LocalFlow is based on utilizing the updated release of Dinic's algorithm [80] that runs in local. A blocking flow algorithm is designed by LocalFlow algorithm that has run-time Õ (vol (A)/σ), and this algorithm utilizes Dinic's algorithm which the blocking flow algorithm is called repeatedly. The aim of the problem of blocking flow is detecting a graph s-t flow f. Dinic's algorithm is a simple iterative algorithm that begins with a zero-flow f. A graph blocking flow is calculated and represented by BlockFlow$_{G,f}$ (s; t) on each iteration, then gather it to the present flow f. It finishes until the blocking flow defeats to augment. Figure 13 describes the pseudocode of the LocalFlow algorithm.

Algorithm: LocalFlow$G$(A, α, ε_σ)
Input: $G = (V, E)$, $A \subset V$, $\alpha \in (0, 1)$, and $\varepsilon_\sigma \in \left(\frac{vol(A)}{vol(V-A)}, \infty\right)$
Output: an s-t flow f and a set S (representing an s-t cut($\{s\} \cup S$, $\{t\} \cup (V-S)$) in $G_A(\alpha, \varepsilon_\sigma)$
$G' \leftarrow G_A(\alpha, \varepsilon_\sigma)$.
$Bs \leftarrow \emptyset$.
$f \leftarrow 0$.
For $i \leftarrow 1$ **to** $I = {}^{def}$ $\left(\frac{5}{\alpha}\log(3vol(A)/\sigma)\right)$ **Do**
  $G'' \leftarrow G'(B_i)$.
  $f \leftarrow f$ + BlockFlow$_{G''}$, f (s, t) and breaks if BlockFlow fails to augment.
  $C \leftarrow$ the vertices in $N(A \cup B_i)$ whose edges to the sink get saturated in the new flow
  $B_i \leftarrow B_i \cup C$.
End for
If BlockFlow ever fails to find an augmenting path **Then**
  f is now an exact s-t maximum flow in G' and let its min-cut be S.
Else
  $S \leftarrow$ the cut among all layer cuts that minimizes conductance.
End if
Return (f, S).

Figure 13: Pseudocode of the LocalFlow algorithm.

The second algorithm, LocalFlow^{exact} links Andersen and Lang's guarantee [78] that merge local and exact maximum flows. The LocalFlow^{exact} algorithm is a release of Goldberg-Rao's algorithm [81] that runs locally. Figure 14 describes the pseudocode of the LocalFlow^{exact} algorithm.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Algorithm: LocalFlow^{exact} $G(A, \alpha, \varepsilon_G)$

Input: $G = (V, E)$, $A \subset V$, $\alpha \in (0; 1)$, and $\varepsilon_G \in (\frac{vol(A)}{vol(V-A)}, \infty)$

Output: the s-t maximum flow f and its dual s-t minimum cut S in $G_A(\alpha, \varepsilon_G)$.

$G' \leftarrow G_A(\alpha, \varepsilon_G)$.
$A = (3 \text{ vol } (A)/\sigma)^{1/2}$.
Scale G' up so that all capacities are integral, and $F \leftarrow$ poly (vol (A)/σ) is an upper bound on the s-t flow value.
$B_s \leftarrow \phi$.
$f \leftarrow 0$.
while $F \geq 1$ **do**
  $\Delta \leftarrow F/4\Lambda$.
  for $i \leftarrow 1$ to 6Λ **do**
    $l \leftarrow$ the length function.
    $G'' \leftarrow G'(B_s)$.
    $f \leftarrow f +$ BinaryBlockFlow$_{G'', l, \Delta}(s, t)$.
    $C \leftarrow$ the vertices in $N(A \cup B_s)$ whose edges to the sink get saturated in the new flow.
    $B_s \leftarrow B_s \cup C$.
  End for
  $F \leftarrow F/2$.
End while

Figure 14: Pseudocode of the LocalFlow^{exact} algorithm.

5.2.4 SimpleLocal

The SimpleLocal [82] uses the existing max-flow algorithms to introduce a new strongly-local flow algorithm. This algorithm is flexible and simple to execute and solves the same optimization problem as LocalImprove. Firstly, the three-Stage Local Max Flow procedure is computed then use this for computing the Simplelocal algorithm.

The inputs to build the augmented graph G_R (α) are $G = (N, E)$ stands for a graph, $R \subset N$ stands for an initial seed set that is based on the condition vol(R) ≤ vol ($\bar{R}$), and α ∈ (0, 1) is a parameter. Firstly, append s that stands for the source node and t that stands for the sink node to the graph. Secondly, append an edge (s, r) for every each $r \in R$ with weight αd_r. Finally, append an edge (x, t) for every $x \in \bar{R}$ with weight $\alpha f(R) d_x$ where $f(R)$ =vol (R) \ vol ($\bar{R}$). Figure 15 shows the augmented graph G_R (α).

Updating the capacity of the edges starting from $\bar{R}$ to t to $\alpha\varepsilon d_w$ for each vertices $w \in \bar{R}$ where ε = f(R) + δ for δ ≥ 0 to construct the modified augmented graph G'_R (α, δ). The process of solving a series of approximate max flow calculations on G'_R (α, δ) for many α values to detect finds a set S with low conductance called LocalImprove that is based on upgrading Dinic's max-flow technique [79] and detecting blocking flows transaction on the local graph that refers to a subgraph of G'_R (α, δ) called the local graph. A local subgraph of G'_R (α, δ) is built and updated to compute the goal of LocalImprove. A three-stage method is improved for exact maximum flow calculations on G'_R (α, δ) instead of using Dinic's method to calculate approximate maximum flows.

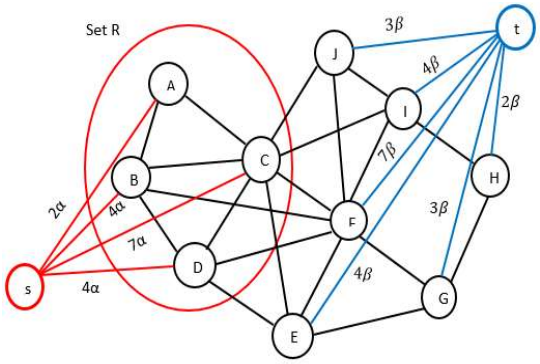

Figure 15. The augmented graph G_R (α) used by Improve.

5.2.4.1 Three-Stage Local Max Flow Procedure

3StageFlow is used to calculate of a modified augmented graph G'_R (α, δ) maximum s-t flow. Local graph L = (N_L, E_L) is defined as a part of the modified augmented graph G'_R (α, δ) that includes

- Add s, t to the graph G'.
- Edges starting from s to the R set.
- Edges between nodes in R.
- Edges starting from R to Neighbor(R).
- Edges starting from t to the Neighbor(R).

F is a flow vector that starts with zero vector, and flow (F) is aggregated amount of all flow starting from s to t. Figure 16 describes the 3StageFlow flowchart.

Step 1. Expansion

For much flow in the local graph starting from s to t, there is needed to expand graph at the start of each repetition. The expanded set of vertices is indicated by X. All neighbors of $x \in G'$ for each vertex $x \in X$ that are not a portion of L are included as well as add the edges between x and whole its neighbors. An edge is included between every new vertex addition to L and the sink t. The local graph is not expanded in the first step, so the set $X = \phi$.

Step 2. Max-Flow Computation

After the first step is completed correctly, maximum flow f is calculated by using any available max-flow subroutine after extending the local graph L. F value is updated to $F + f$. To structure flow residual graph L_f, the capacity c_{ij} of an edge in EL is changed by c_{ij} - f_{ij}, where the edge (i, j) flow is indicated by f_{ij} and the value f_{ij} is used to replace the capacity c_{ji} of an edge in E_L.

Step 3. Updates

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

This step analyzes the flow effects and decide if the local graph need to expand. The residual graph of f is used to develop the local graph to discover an unsaturated edges chain of the vertices set that remain connected to s, these vertices set called the source set S.

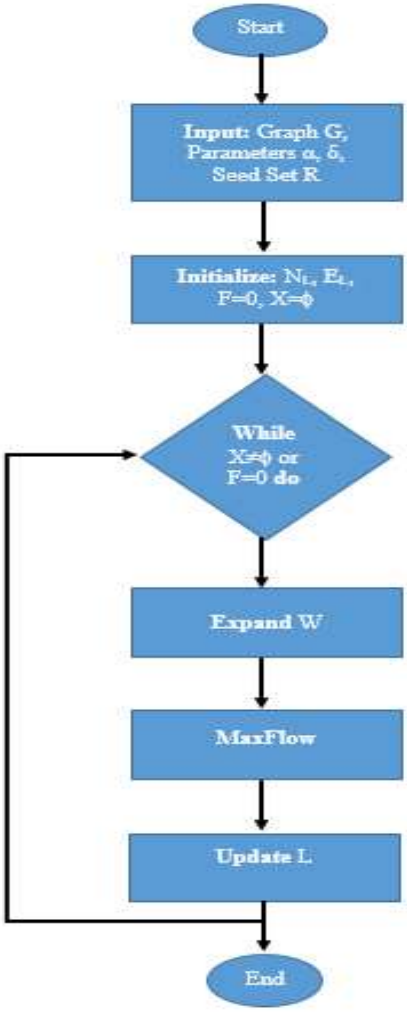

Figure 16: The 3StageFlow flowchart.

5.2.4.2 SimpleLocal Algorithm

A good conductance cut of SimpleLocal is computed by requesting 3StageFlow repeatedly to detect the very little α, like that the G'_R (α, δ) maximum s-t flow is less than αvol(R). Figure 17 illustrates the SimpleLocal flowchart.

6 APPLICATIONS OF GRAPH CLUSTERING

Graph mining techniques are described in many applications to achieve more of enhancements in graph clustering. This section presents these applications.

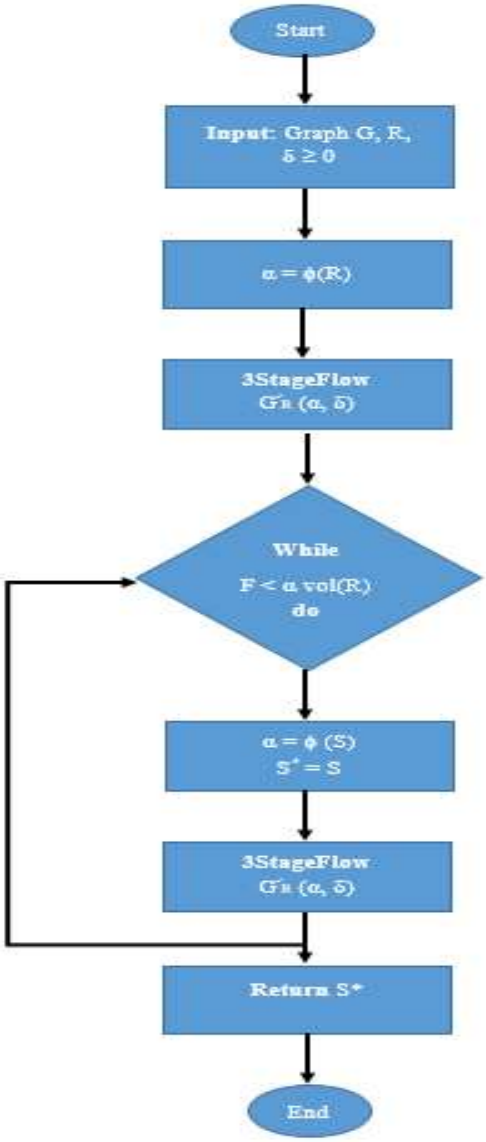

Figure 17: The SimpleLocal flowchart.

- **Community Detection in Web Applications and Social Networks**

Variety actual world graphs are available such as social networks, web graphs, and biological networks, the problem is detecting communities (clusters) of these graphs. Example of applications includes dividing the social networks like Facebook and LinkedIn into clusters of friends, scientific collaboration networks into research communities, and World Wide Web into clusters of linked webpages.

The individuals are interacted with all other inside a group more repeatedly than those outside the group this form the community. Community detection [83] uses a network for detecting clusters

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

where the memberships of individuals' group are not clearly given. Many algorithms for detecting community in graphs are studied by Newman and Girvan [84].

Community is divided into two types, *i.e.*, overlapping communities and non-overlapping communities which are described in Figure 18.

- **Non-Overlapping communities:** It is featured with dense connectivity within the community, sparse across communities
- **Overlapping communities:** It is possible for each individual to have many communities simultaneously.

The trouble of community detection is known as a NP-problem and it has been discussed in computer science for decades as the problem of graph partitioning. Numerous algorithms have been suggested, including spectral clustering, random walk-based methods, modularity-based methods, hierarchical clustering, and user profile based methods.

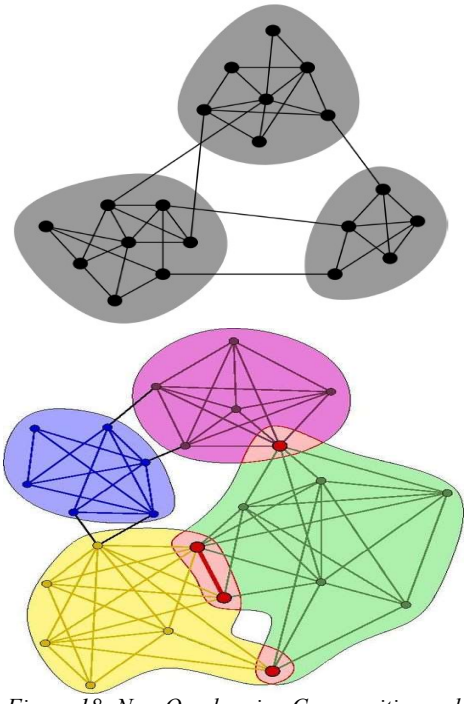

Figure 18: Non-Overlapping Communities and Overlapping communities.

- **Discovery of protein complexes**

The biological functions are carried out by proteins which reacting as complexes. The serious job is discovering the proteins complexes that could realize the link between biological network functions and structures as well as predicting the unknown proteins function.

In the period of proteomics, massive data of protein interactions have been created from many experimental techniques and computational methods. They are used to forecast protein function and discover protein complexes from the networks of protein–protein interaction (PPI) [85].

The rules of cellular organization and biological functions of proteins are understood to help the protein complexes prediction. A PPI network is constructed as an undirected graph, where proteins are illustrated by nodes and proteins interactions are symbolized by edges. Proteins react with others like a compound to implement their biological functions in cells, like DNA replication, transcription and protein degradation. Thus, in PPI networks, the dense subgraphs are known as protein complexes.

PPI network has clusters that are extremely interconnected or heavy areas which may illustrate complexes. The process of identifying protein complexes is like detecting the clusters of the graph. Protein complexes are identified by improving many algorithms of graph clustering by utilizing the information encoded in the network topology.

- **Image segmentation**

An image is a way of transferring information which includes much valuable information. There is a serious region in digital image technology application to understand and extract useful information from the image. The initial step is the image segmentation to understand the image.

Image pixels are classified dependent on some criteria by using image segmentation method [86] that splits an image into discrete regions numbers, where the pixels include large similarity in every region and large dissimilarity among regions. Image segmentation is an important tool in various regions such as image processing, health care, pattern recognition, and traffic image. Image segmentation have various techniques such as cluster-based, edge-based, neural network-based, and threshold-based.

This problem is studied by many researchers and different techniques are introduced for image segmentation. These techniques are partitioned into many groups which are clustering (fuzzy and hard), histogram thresholding, region splitting and merging, region growing, physical

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

model-based, edge-based, neural network and genetic algorithm based approaches, and fuzzy approaches.

Table 1 shows the discussed global graph clustering algorithms. Table 2 shows the discussed local graph clustering algorithms.

7 COMPERHENSIVE REVIEW

Table 1: Comprehensive Review of Global Graph Clustering Methods.

Global Graph Clustering		
Complexity Methods		
Algorithm	**Year**	**Reference**
K-means	1979	Hartigan and Wong [33]
Iterative or Online Methods		
Algorithm	**Year**	**Reference**
Online or Iterative clustering	2011, 2007, 2002, 2010	Li et al. [34], Zanghi et al. [35], Toussaint [36], Seeland et al. [37]
Hierarchical Methods		
Divisive Hierarchical Clustering		
Algorithm	**Year**	**Reference**
Cuts	2001, 1956	Cormen et al. [50], Elias et al.[51], Ford and Fulkerson[52]
Maximum Flow	1956,1970, 1972, 1988	Ford and Fulkerson [56], Dinic's [57], Edmond and Karp [58], and Goldberg-Tarjan [59].
Spectral	1992	Hagen and Kahng [62]
Betweenness	2003, 2004	Newman and Girvan [44, 31]
Markov Chains and Random Walks	1996, 2001	Lovász [67], Aldous and Fill [68]
Agglomerative Hierarchical Clustering		
Algorithm	**Year**	**Reference**
Agglomerative clustering (AGNES)	2003, 2004, 2007	Carrasco et al. [46], Donetti et al. [47], Du [48], Hopcroft et al. [49]

Table 2: Comprehensive Review of Local Graph Clustering Methods.

Local Graph Clustering		
Spectral Methods		
Algorithm	**Year**	**Reference**
Mahoney-Orecchia-Vishnoi (MOV)	2012	Mahoney et al. [23]
Nibble	2004, 2013	Spielman and Teng [70, 71]
PageRank Nibble	2006	Andersen et al. [72]
Evolving Sets	2009	Andersen and Peres [73]
Heat Kernel	2014	Kloster and Gleich [74]
Heat Kernel PageRank	2009	Chung [75]
Flow-based Methods		
Algorithm	**Year**	**Reference**
Max-flow Quotient-cut Improvement (MQI)	2004	Lang and Rao [77]
FlowImprove	2008	Andersen and Lang [78]
Local FlowImprove	2014	Orecchia and Zhu [79]
SimpleLocal	2016	Veldt et al. [82]

8 CONCLUSION

This paper provides an overall review of different graph mining methods. Firstly, it demonstrated the related work of graph clustering. Secondly, it dealt with many graph clustering techniques that based on the input data with more details. Finally, it described some of the applications of using these techniques to improve the clustering process.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

There are many applications of graph clustering in computing, because of large graphs, the traditional clustering methods can be expensive for computing the real-world graphs of interest. Global graph clustering methods require the whole graph as input, so it is not appropriate for partitioning, but the local graph clustering methods require a given seed node or set of seed nodes to get the cluster. Scalability problems led to the development of local graph clustering algorithms.

As discussed in this article, graph clustering methods are classified into groups, *i.e.*, global and local. From this review, it noticed that local graph clustering methods resolve the problem of partitioning which based on a given node as input seed set to get a good cluster without looking at the whole. This review assists the researchers in a graph clustering to select the appropriate method from these different methods to partition the graph efficiently.

Besides previous algorithms, the investigations on the algorithms are still being done and new algorithms are being developed continually. Local graph clustering can be easily understood, faster than traditional algorithms that touch the entire graph and find good clusters in a graph with work proportional to the size of the cluster rather than that of the entire graph. For future works, we suggest developing local graph clustering algorithm.

9 REFERENCES

[1] W. Fan, and A. Bifet, "Mining Big Data: Current Status, and Forecast to the Future," SIGKDD Explorations Newsletter, vol. 14(2), 2013, pp. 1–5.

[2] U. Kang and C. Faloutsos, "Big graph mining: Algorithms and discoveries," SIGKDD Explorations, vol. 14 (2), 2012.

[3] S. Aridhi and E. M. Nguifo, "Big graph mining: Frameworks and techniques," Big Data Research, Vol. 6, 2016, pp. 1-10.

[4] S. E. Schaeffer, "Graph clustering," Computer science review, vol. 1(1), 2007, pp. 27-64.

[5] C. C. Aggarwal, and H. Wang, "A survey of clustering algorithms for graph data," Managing and mining graph data, vol. 40, 2010, 275-301.

[6] J. Ni, H. Fei, W. Fan, and X. Zhang, "Cross-network clustering and cluster ranking for medical diagnosis," In: ICDE, 2017.

[7] J. Ni, M. Koyuturk, H. Tong, J. Haines, X. Rong, and X. Zhang, "Disease gene prioritization by integrating tissue-specific molecular networks using a robust multinetwork model," BMC Bioinform, vol. 17(1), 453, 2016.

[8] D. Zhou, and C.J.C. Burges, "Spectral clustering and transductive learning with multiple views," In: ICML,2007.

[9] A. Kumar, P. Rai, and H. Daume, "Co-regularized multi-view spectral clustering," In: Advances in neural information processing systems, 2011.

[10] A. Kumar, and H. Daume, "A co-training approach for multi-view spectral clustering," In: ICML. 2011.

[11] W. Cheng, X. Zhang, Z. Guo, W. Yubao, P.F. Sullivan, and W. Wang, "Flexible and robust co-regularized multi-domain graph clustering," In: KDD, 2013.

[12] J. Ni, H. Tong, W. Fan, and X. Zhang, "Flexible and robust multi-network clustering," In: KDD, 2015.

[13] K. George, and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," SIAM Journal on Scientific Computing, vol. 20(1), 1998, pp. 359–392.

[14] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts." In SIGKDD'04, 2004, pp. 551–556. ACM.

[15] S. v. Dongen, "Graph clustering by flow simulation," PhD Thesis, 2000.

[16] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," Internet Mathematics vol. 6(1), 2009, 29-123.

[17] L. G. S. Jeub, P. Balachandran, M. A. Porter, P. J. Mucha, and M. W. Mahoney, "Think locally, act locally: Detection of small, medium-sized, and large communities in large networks," Physical Review E, vol. 91(1): 012821, 2015.

[18] Y. Wu, R. Jin, J. Li, and X. Zhang, "Robust local community detection: on free rider effect and its elimination," Proceedings of the VLDB Endowment vol. 8(7), 2015, pp. 798-809.

[19] I. M. Kloumann and J. M. Kleinberg "Community membership identification from small seed sets," Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014.

[20] J. J. Whang, D. F. Gleich, and I. S. Dhillon, "Overlapping community detection using seed set expansion," Proceedings of the 22nd ACM

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

international conference on information & knowledge management, ACM, 2013.

[21] W. Cui, Y. Xiao, H. Wang, and W. Wang, "Local search of communities in large graphs," In: SIGMOD, 2014.

[22] Y. Wu, Y. Bian, X. Zhang, "Remember where you came from: on the second order random walk based proximity measures," Proc. VLDB Endow., Vol. 10(1), 2016, pp. 13–24.

[23] M. W. Mahoney, L. Orecchia, and N. K. Vishnoi, "A local spectral method for graphs: With applications to improving graph partitions and exploring data graphs locally," Journal of Machine Learning Research vol. 13(Aug), 2012, pp. 2339-2365.

[24] K. Voevodski, S.-H. Teng, Y. Xia .Voevodski, K., et al., "Finding local communities in protein networks," BMC bioinformatics vol. 10(1): 297, 2009.

[25] Z. A. Zhu, S. Lattanzi, and V. Mirrokni, "Local Graph Clustering Beyond Cheeger's Inequality," arXiv preprint arXiv:1304.8132, 2013.

[26] Z. A. Zhu, S. Lattanzi, and V. Mirrokni, "A Local Algorithm for Finding Well-Connected Clusters," ICML (3), 2013.

[27] S. O. Gharan, and L. Trevisan, "Approximating the expansion profile and almost optimal local graph clustering," Foundations of Computer Science (FOCS), IEEE 53rd Annual Symposium on, IEEE, 2012.

[28] T. C. Kwok, L. C. Lau, and Y. T. Lee "Improved Cheeger's inequality and analysis of local graph partitioning using vertex expansion and expansion profile," Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2016.

[29] J.E. Hopcroft, O. Khan, B. Kulis, and B. Selman, "Natural communities in large linked networks": in: Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining, KDD, ACM, New York, NY, USA, 2003.

[30] M.E.J. Newman, "Fast algorithm for detecting community structure in networks," Physical Review E 69 (6) 066133, 2004.

[31] M.E.J. Newman, and M. Girvan, "Finding and evaluating community structure in networks," Physical Review E 69 (2) 026113, 2004.

[32]M.E.J. Newman, "Detecting community structure in networks," The European Physical Journal B 38 (2), 2004, pp. 321–330.

[33] J.A. Hartigan, M.A. Wong, Algorithm AS 136: A k-means clustering algorithm, Applied Statistics 29, 1979, pp. 100–108.

[34] K. Li, F. Yao, and R. Liu , "An online clustering algorithm", Proceedings of the Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2011.

[35] H. Zanghi, C. Ambroise, and V. Miele, "Fast online graph clustering via Erd˝osRényi mixture," Tech. Rep. 8, JouyenJosas/ Paris/Evry, France, 2007.

[36] G.T. Toussaint, Proximity graphs for nearest neighbor decision rules: Recent progress, in: Proceedings of the Thirty-Fourth Symposium on Computing and Statistics, Interface2002, The Interface Foundation of North America, Fairfax Station, VA, USA, 2002.

[37] M. Seeland, T. Girschick, F. Buchwald, and S. Kramer, "Online structural graph clustering using frequent subgraph mining," Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases, 3, 2010, pp. 213-228.

[38] U. Brandes, M. Gaertler, D. Wagner, Experiments on graph clustering algorithms, in: G. Di Battista, U. Zwick (Eds.), Proceedings of the Eleventh European Symposium on Algorithms, in: Lecture Notes in Computer Science, vol. 2832, SpringerVerlag GmbH, Heidelberg, Germany, 2003.

[39] T.N. Bui, F.T. Leighton, S. Chaudhuri, M. Sipser, Graph bisection algorithms with good average case behavior, Combinatorica 7 (2), 1987, pp. 171–191.

[40] G.W. Flake, R.E. Tarjan, K. Tsioutsiouliklis, Graph clustering and minimum cut trees, Internet Mathematics 1 (1), 2004, pp. 385–408.

[41] S. Fortunato, V. Latora, M. Marchiori, Method to find community structures based on information centrality, Physical Review E 70 (056104), 2004.

[42] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proceedings of the National Academy of Sciences, USA 99, 2002, pp. 8271–8276.

[43] E.J.L. Johnson, A. Mehrotra, G.L. Nemhauser, Mincut clustering, Mathematical Programming 62 (1), 1993, pp. 133–151.

[44] M.E.J. Newman, M. Girvan, Mixing patterns and community structure in networks, in: R. PastorSatorras, M. Rubi, A. Díaz Guilera (Eds.), Statistical Mechanics of Complex Networks:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Proceedings of the XVIII Sitges Conference on Statistical Mechanics, in: Lecture Notes in Physics, vol. 625, SpringerVerlag GmbH, Berlin, Germany, 2003.

[45] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (8), 2000, pp. 888–901.

[46] J.J.M. Carrasco, D.C. Fain, K.J. Lang, L. Zhukov, Clustering of bipartite advertiser keyword graph, in: Proceedings of the Third IEEE International Conference on Data Mining, Workshop on Clustering Large Data Sets, 2003.

[47] L. Donetti, M.A. Muñoz, Detecting network communities: A new systematic and efficient algorithm, Journal of Statistical Mechanics P10012, 2004.

[48] H. Du, An algorithm for detecting community structure of social networks based on prior knowledge and modularity, Complexity 12 (3), 2007, pp. 53–60.

[49] J.E. Hopcroft, O. Khan, B. Kulis, B. Selman, Natural communities in large linked networks, in: Proceedings of the Ninth International Conference on Knowledge

Discovery and Data Mining, KDD, ACM, New York, NY, USA, 2003.

[50] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 2nd ed., MIT Press and McGraw Hill, Cambridge, MA, USA, 2001, pp. 643–700.

[51] P. Elias, A. Feinstein, C.E. Shannon, Note on maximum flow through a network, IRE Transactions on Information Theory IT2, 1956, pp. 117–119.

[52] L.R. Ford Jr., D.R. Fulkerson, Maximum flow through a network, Canadian Journal of Mathematics 8, 1956, pp. 399–404.

[53] J.J.M. Carrasco, D.C. Fain, K.J. Lang, L. Zhukov, Clustering of bipartite advertiser keyword graph, in: Proceedings of the Third IEEE International Conference on Data Mining, Workshop on Clustering Large Data Sets, 2003.

[54] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (8), 2000, pp. 888–901.

[55] J. Šíma, S.E. Schaeffer, "On the NP completeness of some graph cluster measures," in: J. Wiedermann, G. Tel, J. Pokorný, M. Bieliková, J. Štuller (Eds.), Proceedings of the Thirty-second International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM, in: Lecture Notes in Computer Science, vol. 3831, Springer-Verlag GmbH, Berlin, Heidelberg, Germany, 2006.

[56] L. R. Ford and D. R. Fulkerson, "Maximal Flow Through a Network," Can. J. Math. 8, 1956, pp. 399-404.

[57] A. Dinice, "Algorithm for Solution of a Problem of Maximum Flow in Networks With Power Estimation," Soviet Math. Dokl. 11, 1970, pp. 1277- 1280.

[58] J. Edmonds and R.M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," J. Assoc. Comput. Mach. 19, 1972, pp. 248-264.

[59] A.V. Goldberg And R.E. Tarjan, "A New Approach to the Maximum-Flow Problem," Journal of the Association for Computing Machinery, Vol 35. No. 4, 1988, pp. 921-940.

[60] M.C.V. Nascimento and A.C.P.L.F. De Carvalho, "Spectral methods for graph clustering–a survey," European Journal of Operational Research, 211(2), 2011, pp. 221–231.

[61] S.V.Suryanarayana, "A Survey: Spectral Clustering Applications and its Enhancements," International Journal of Computer Science and Information Technologies Vol. 6, 2015, pp. 185-189.

[62] L. Hagen and A. Kahng, "New spectral methods for ratio cut partitioning and clustering," IEEE Trans. Computer Aided Design, 11(9), 1992, pp. 1074-1085.

[63] J. Shi, and J. Malik, "Normalized cuts and image segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), 2000, pp. 888–905.

[64] L. Hagen, A.B. Kahng, "New spectral methods for ratio cut partitioning and clustering," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 11(9), 1992, pp. 1074–1085.

[65] C.H.Q. Ding, X.F. He, H.Y. Zha, M. Gu, H.D. Simon, "A min-max cut algorithm for graph partitioning and data clustering," In: Proceedings of 1st IEEE International Conference on Data Mining. 2001, pp. 107–114.

[66] M. Meila, "The multicut lemma," UW Statistics Technical Report 417, 2001.

[67] L. Lovász, Random walks on graphs: A survey, in: Bolyai Society Mathematical Studies, 2, in: Combinatorics, Pál Erd˝ os is Eighty, vol. 2, Bolyai Mathematical Society, 1996, pp. 353–397.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[68] D.J. Aldous, and J.A. Fill, Reversibe Markov Chains and Random Walks on Graphs. http://www.stat.berkeley.edu/aldous/RWG/book.html, 2001.

[69] F.R.K. Chung. Spectral Graph Theory, volume 92 of CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[70] D. A. Spielman, and S.-H. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, ACM, 2004.

[71] D. A. Spielman, and S.-H. Teng. "A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning," Society for Industrial and Applied Mathematics, vol. 42(1), 2013, pp. 1–26.

[72] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," Proceedings of 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), 2006.

[73] R. Andersen, and Y. Peres. "Finding sparse cuts locally using evolving sets," Proceedings of the forty-first annual ACM symposium on Theory of computing, ACM, 2009.

[74] K. Kloster, and D. F. Gleich, "Heat kernel based community detection," Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014.

[75] F. Chung, "A local graph partitioning algorithm using heat kernel pagerank," Internet Mathematics vol. 6(3), 2009, pp. 315-330.

[76] F. Chung and O. Simpson, "Computing heat kernel pagerank and a local clustering algorithm," in 25th International Workshop, IWOCA 2014, 2014, pp. 110–121.

[77] K. Lang and S. Rao, "A flow-based method for improving the expansion or conductance of graph cuts," in Proc. 10th Int. IPCO Conf. Integer Program. Combinatorial Optim., 2004, pp. 325–337.

[78] R. Andersen, and K. J. Lang. "An algorithm for improving graph partitions," Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2008.

[79] L. Orecchia, and Z. A. Zhu, "Flow-based algorithms for local graph clustering," Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2014, pp. 1267–1286.

[80] E. A. Dinic. "Algorithm for solution of a problem of maximum flow in networks with power estimation," Soviet Math Doklady, 11, 1970, 1277-1280.

[81] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," SIAM Journal on Scientific Computing, 20(1), 1998, pp. 359-392.

[82] N. Veldt, D. F. Gleich, and M. W. Mahoney, "A simple and strongly-local flow-based method for cut improvement," International Conference on Machine Learning (ICML), 2016.

[83] S.R. Chintalapudi, and M.K. Prasad, "A survey on community detection algorithms in large scale real world networks", in Computing for Sustainable Global Development (INDIACom), 2nd International Conference on, IEEE., 2015, pp. 1323-1327.

[84] M. E. J. Newman, M. Girvan, "Finding and evaluating community structure in networks," Physical review. E, Statistical, nonlinear, and soft matter physics, 69(2), 2004.

[85] B. Snel, P. Bork, and M.A. Huynen. "The identification of functional modules from the genomic association of genes." Proceedings of the National Academy of Sciences 99, no. 9, 2002, pp. 5890-5895.

[86] J. Shi, and J. Malik, "Normalized cuts and image segmentation," Departmental Papers (CIS), 2000.