Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 4, April 2015, pg.96 – 105

RESEARCH ARTICLE

PROBABILISTIC PATH QUERIES IN NETWORKS PATH: EFFICIENT AND EFFECTIVE CLUSTERING METHODS

R.BRINDHA, M.E, CSE (PT), SCSVMV, KANCHIPURAM, bri_janu@yahoo.co.in

MRS.V.GEETHA, M.E, AP, DEPARTMENT OF CSE, SCSVMV KANCHIPURAM, vgeetha@kanchiuniv.ac.in

MRS.T.JAYANTHI, M.E AP, DEPARTMENT OF CSE, SCSVMV, KANCHIPURAM, tj@kanchiuviv.ac.in

ABSTRACT

Correlations may exist among adjacent edges in various probabilistic graphs. As one of the basic mining techniques, graph clustering is widely used in exploratory data analysis, such as data compression, information retrieval, image segmentation, etc. Graph clustering aims to divide data into clusters according to their similarities, and a number of algorithms have been proposed for clustering graphs, such as the pKwikCluster algorithm, spectral clustering, k-path clustering, etc. To develop efficient clustering algorithms for probabilistic graphs it becomes more challenging to efficiently cluster probabilistic graphs when correlations are considered. In this paper, we define the problem of clustering correlated probabilistic graphs. To solve the challenging problem, we propose two algorithms, namely the PEEDR and the CPGS clustering algorithm. For each of the proposed algorithms, we develop several pruning techniques to further improve their efficiency.

Index Terms—Clustering, correlated, probabilistic graphs, algorithm

1. INTRODUCTION

Network data analytics lie in the core of many scientific fields such as social, biological and mobile ad-hoc networks. Typically, such network data are associated with uncertainty. This uncertainty is either due to the data collection process or to machine-learning methods employed at preprocessing. Uncertainty may be also added to data for privacy-preserving reasons. We model such uncertain networks as probabilistic graphs. Every edge in a probabilistic graph is associated with a probability of existence. As an example, consider the probabilistic protein-protein interaction (PPI) networks.

In a probabilistic graph, any two edges *ei* and *ej* are called *conditionally independent* if *p(ei, ej)* = *p(ei)p(ej)*, and *conditionally dependent* if *p(ei, ej)* ≠*p(ei)p(ej)*. For the standard probabilistic graph model, any two edges are conditionally independent of each other. Typically, coexistence and mutual exclusion among adjacent edges are commonly observed in various graph oriented applications. For example, in Fig. 1(a), given two edges *e*2 and *e*3 with a mutual exclusion constraint, only the joint probabilities *p(e*2, *e*3*)* and *p(e*2, *e*3*)* may exist, where *ei* represents that *ei* does not exist. Obviously, *p(e*2, *e*3*)* ≠*p(e*2*)p(e*3*)*, and hence *e*2 and *e*3 are conditionally dependent on each other. Similarly, *e*1 and *e*2 are also conditionally dependent on each other due to a coexistence Constraint, correlations will lead to incorrect results.

Define the probabilistic graphs containing correlated adjacent edges as correlated probabilistic graphs as shown in Fig.2(a). As one of the basic data mining techniques, clustering is widely used in

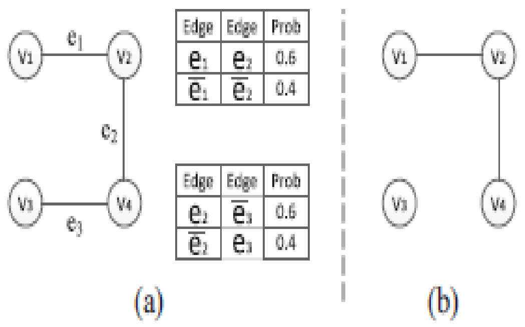


Fig. 1. Graph model: (a) Probabilistic graph with edge correlations.(b) Possible world graph.

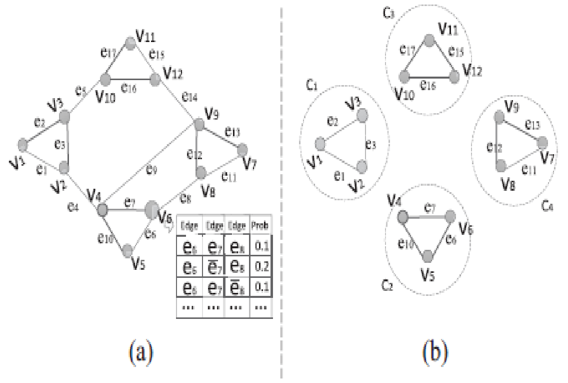


Fig.2. Correlated probabilistic graph and a cluster graph: (a) Correlated probabilistic graph.(b) Cluster graph.

To further improve the clustering quality, we propose an alternative solution, named the CPGS (Correlated Probabilistic Graphs Spectral) clustering algorithm. Spectral clustering is effective for clustering deterministic graph data. Then, these transformed points in the multi-dimensional space are iteratively clustered by the K- means algorithm. In addition, we develop several optimization strategies to speed up the clustering process. The major contributions of our work are summarized as follows.

•We formally define the problem of clustering correlated probabilistic graphs and investigate related properties.

• We propose a new algorithm, PEEDR, which is rather efficient for clustering correlated probabilistic graphs, and several pruning methods for this algorithm.

• We develop an algorithm, CPGS, for clustering correlated probabilistic graphs based on the spectral clustering algorithm, which can produce better cluster.

2. RELATED WORK

2.1 Algorithms for Clustering Deterministic Graphs

Deterministic graph clustering has been extensively studied in data mining research and a number of clustering algorithms have been developed. Ahmed *et al*. [1]. The different categories of clustering algorithms and recent efforts to design clustering methods for various kinds of graph data Jain *et al*. [7]As one of the most widely used graph clustering algorithms, spectral clustering has received increased interest of researchers. Spectral clustering relies on the eigen structure of a graph Laplacian matrix to partition vertices into disjoint clusters, with points in the same cluster having high similarity and points in different clusters having low similarity [9].

2.2 Querying and Mining of Probabilistic Graphs

Many classical data mining problems have been redefined in probabilistic graphs, such as the reachability query, shortest path query, *K*-NN query, etc. Jin *et al*. [3] studied the Distance-constraint Reachability query and presented a sampling algorithm to answer the NP-hard problem.

2.3 Querying and Mining the Probabilistic Data with Correlations

An efficient strategy was developed for query evaluation over such probabilistic databases by casting the query processing problem as inference problem in an appropriately constructed probabilistic graphical model. Lian *et al*. [16] investigated the nearest neighbor query on uncertain data with local correlations. Tight lower and upper bounds of the subgraph similarity probability were developed to prune the search space. Compared to these queries, clustering over correlated probabilistic graphs is more complicated.

3. PROBLEM DEFINITIONS

Define the model of a correlated probabilistic graph as $G = \{V, E, P, F\}$, where V is the set of vertices, E is the set of edges, P is the existence probability, and F is the joint probability distribution of edges. The output graph is modeled as a **cluster graph** which is composed of several disconnected clusters and each vertex in the graph only belongs to one cluster.

In this model, the joint probability distribution F is of the form $f(e1, e2, \cdots, ek) = p$, where ei denotes existence, ei denotes non existence for edge $ei \in E$, and p is the value of the joint probability.

A possible world graph serves as efficient probabilistic graphs. For a correlated probabilistic graph G = {V, E, P, F}, a possible world graph Gi = {V' , E'} is an instantiation sampled from G, where V'= V and E'cE.

4. PEEDR CLUSTERING ALGORITHM

4.1 Basic *PEEDR* Clustering Algorithm

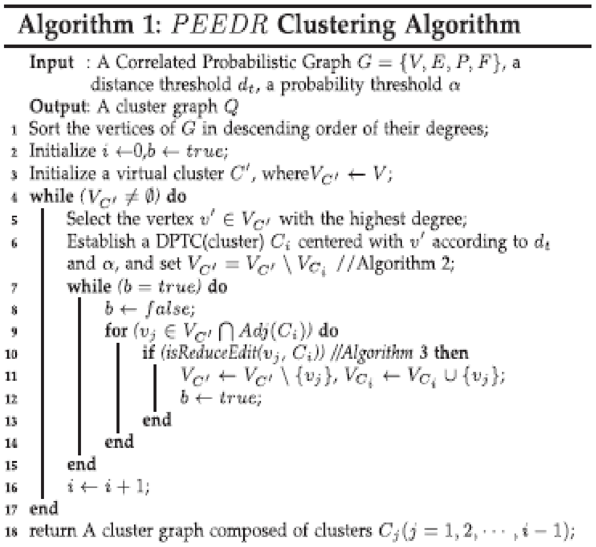
In this subsection, present a novel algorithm called Partially Expected Edit Distance Reduction (PEEDR), for clustering a correlated probabilistic graph G. To illustrate the PEEDR algorithm, first present a definition.

Definition 1 (Adjacent Vertex to Cluster).

Consider a graph $G = \{V, E, P, F\}$, and a cluster C in the cluster graph Q of G. Let V_C denote the set of nodes in cluster C. For any vertex $v \in V \setminus V_C$, that vertex v is adjacent tothe cluster C if it is adjacent to at least one vertex in V_C, denoted as $v \in Adj(C)$.

The PEEDR algorithm initializes a cluster with one vertex. Then for each vertex that is adjacent to the cluster, it is removed into the cluster if it reduces the expected edit distance from G to the current cluster graph. The above step is iteratively applied until we cannot expand the cluster. Next choose a vertex from the unclustered vertices and repeat the above procedure to generate another cluster. The procedure is repeated until all vertices of G are grouped into clusters. Consequently, get the final cluster graph. One open problem in the above clustering procedure is which vertex to choose in each iteration.

PEEDR algorithm is outlined in Algorithm 1. The algorithm first sorts all the vertices in descending order of their degrees (Line 1). It next initializes a **virtual cluster** *C*', where $V_{C'} = \{v|v \in V\}$, which keeps all the clustered vertices (Line 3). The algorithm finds the vertex with the maximum degree in V_C', denoted as *v*' (Line 5). As an optimization of our algorithm build a Distance-Probability-Threshold Clique DPTC centered at *v*', denoted as C_i (Line 6). The algorithm will check each vertex $vj \in V_{C'}$ that is adjacent to C_i and put v_j into C_i if it can reduce the objective function
D(G,Q) (Lines 9 ~ *15).*



```
Algorithm 1: PEEDR Clustering Algorithm
   Input  : A Correlated Probabilistic Graph $G = \{V, E, P, F\}$, a
            distance threshold $d_t$, a probability threshold $\alpha$
   Output: A cluster graph $Q$
1  Sort the vertices of $G$ in descending order of their degrees;
2  Initialize $i \leftarrow 0, b \leftarrow true$;
3  Initialize a virtual cluster $C'$, where $V_{C'} \leftarrow V$;
4  while ($V_{C'} \neq \emptyset$) do
5      Select the vertex $v' \in V_{C'}$ with the highest degree;
6      Establish a DPTC(cluster) $C_i$ centered with $v'$ according to $d_t$
       and $\alpha$, and set $V_{C'} = V_{C'} \setminus V_{C_i}$ //Algorithm 2;
7      while ($b = true$) do
8          $b \leftarrow false$;
9          for ($v_j \in V_{C'} \bigcap Adj(C_i)$) do
10             if (isReduceEdit($v_j$, $C_i$)) //Algorithm 3 then
11                 $V_{C'} \leftarrow V_{C'} \setminus \{v_j\}, V_{C_i} \leftarrow V_{C_i} \cup \{v_j\}$;
12                 $b \leftarrow true$;
13             end
14         end
15     end
16     $i \leftarrow i + 1$;
17 end
18 return A cluster graph composed of clusters $C_j (j = 1, 2, \cdots, i - 1)$;
```

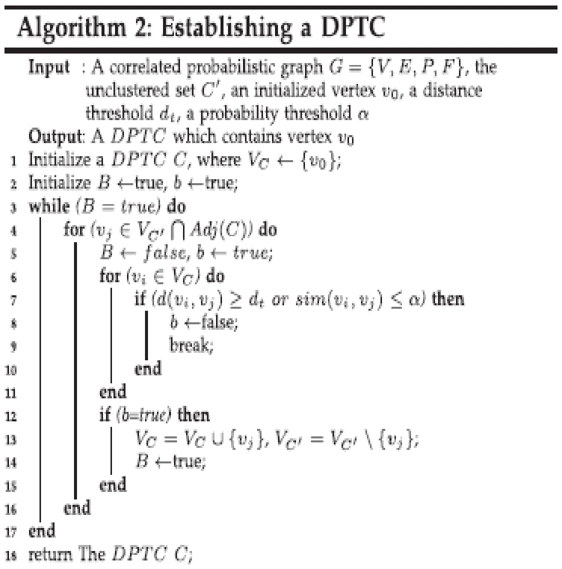
4.2 Optimizations for Clustering Process

4.2.1 Grouping Vertices into DPTCs(GVD)

Inspired by the concept of maximum clique, to build a Distance-Probability-Threshold Clique **(DPTC)** centered at the singleton cluster. The intuition behind DPTC is that vertices with small distances and high similarities are likely to be grouped into the same clusters. The clustering process of the *PEEDR* algorithm starts from a local graph and establishes the cluster graph gradually. As vertices will never be separated once grouped into a cluster, it is essentially a greedy algorithm.

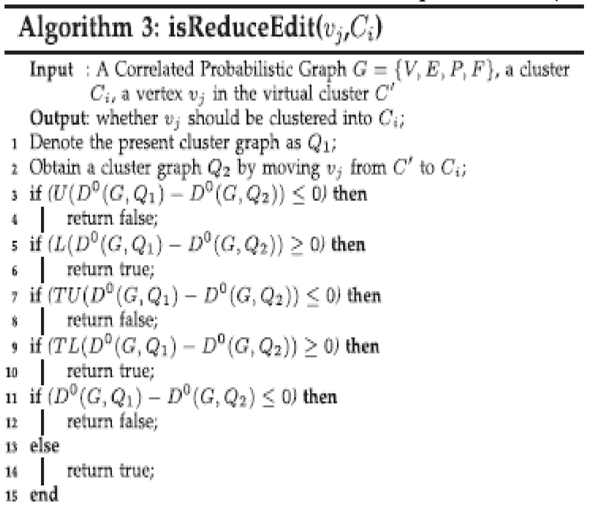
Definition 2 (DPTC).

Given a distance threshold d_t and a probability threshold α, we define a subgraph C of G as a Distance-Probability-Threshold Clique (DPTC), if for any pair of vertices $v_i, v_j \in V_C$, there exists a route r_k fromv_i to v_j satisfying the requirements $d_{rk}(v_i, v_j) \leq dt$ and $sim_{rk}(v_i, v_j) \geq \alpha$.



```
Algorithm 2: Establishing a DPTC
Input  : A correlated probabilistic graph $G = \{V, E, P, F\}$, the
         unclustered set $C'$, an initialized vertex $v_0$, a distance
         threshold $d_t$, a probability threshold $\alpha$
Output: A $DPTC$ which contains vertex $v_0$
1  Initialize a $DPTC$ $C$, where $V_C \leftarrow \{v_0\}$;
2  Initialize $B \leftarrow$true, $b \leftarrow$true;
3  while ($B = true$) do
4      for ($v_j \in V_{C'} \bigcap Adj(C)$) do
5          $B \leftarrow false$, $b \leftarrow true$;
6          for ($v_i \in V_C$) do
7              if ($d(v_i, v_j) \geq d_t$ or $sim(v_i, v_j) \leq \alpha$) then
8                  $b \leftarrow$false;
9                  break;
10             end
11         end
12         if ($b$=$true$) then
13             $V_C = V_C \cup \{v_j\}$, $V_{C'} = V_{C'} \setminus \{v_j\}$;
14             $B \leftarrow$true;
15         end
16     end
17 end
18 return The $DPTC$ $C$;
```

Algorithm 2 describes how to establish a DPTC. Given a distance threshold d_t and a probability threshold α, we start from the original DPTC C which contains only one vertex (Line 1). For each vertex $v_j \in V_{C'} \cap Adj(C)$, if for any vertex $v_j \in V_C$, there exists a route r_k from v_i to v_j that makes d_{rk} $(v_i, v_j) \leq d_t$ and sim_{rk} $(v_i, v_j) \geq \alpha$, then v_i will be added to the DPTC. The above steps are iteratively executed until no vertices meet the requirement (Lines 3~17).



```
Algorithm 3: isReduceEdit($v_j$,$C_i$)
Input  : A Correlated Probabilistic Graph $G = \{V, E, P, F\}$, a cluster
         $C_i$, a vertex $v_j$ in the virtual cluster $C'$
Output: whether $v_j$ should be clustered into $C_i$;
1  Denote the present cluster graph as $Q_1$;
2  Obtain a cluster graph $Q_2$ by moving $v_j$ from $C'$ to $C_i$;
3  if ($U(D^0(G, Q_1) - D^0(G, Q_2)) \leq 0$) then
4      return false;
5  if ($L(D^0(G, Q_1) - D^0(G, Q_2)) \geq 0$) then
6      return true;
7  if ($TU(D^0(G, Q_1) - D^0(G, Q_2)) \leq 0$) then
8      return false;
9  if ($TL(D^0(G, Q_1) - D^0(G, Q_2)) \geq 0$) then
10     return true;
11 if ($D^0(G, Q_1) - D^0(G, Q_2) \leq 0$) then
12     return false;
13 else
14     return true;
15 end
```

isReduceEdit (Algorithm 3) and several optimization techniques (Sections 4.2.2 and 4.2.3). Then we update the set $V_{C'} = V_{C'} \setminus V_{Ci}$ and create the next cluster from the vertices in the virtual cluster C'. The procedure is repeated until all the vertices in C' are grouped into disconnected clusters.

4.2.2 Pruning By Loose Bounds of Objective Function(PLB)

In line 9 of Algorithm 1, for each vertex $v_j \in V_{C'} \cap Adj(C_i)$,we need to check whether v_j should be moved from the virtual cluster $C'$to C_i. This is determined by $D^0(G,Q_1)-D^0(G,Q_2)$, where Q_1 and Q_2 are cluster graphs, and Q_2 is obtained from Q_1 by moving v_j from C' to C_i. If $D^0(G,Q_1)-D^0(G,Q_2) \geq 0$, v_j will be moved from C' to C_i; otherwise, v_j remains in C'. However, computing $D^0(G,Q_1)$ and $D^0(G,Q_2)$ according to the definition of estimated expected edit distance is highly complex. Instead of calculating $D^0(G,Q_1)$ and $D^0(G,Q_2)$, To estimate
upper and lower bounds for them. Only the existence states of the edges adjacent to v_j are changed when generating Q_2 from Q_1. Assume that m edges are changed and the EO order O_0 of

those changed edges is $< e_1, e_2, \dots, e_m >$. The **partial estimated expected edit distance** from G to a cluster graph Q is defined
by

$$D_p^0(G,Q) = m - \sum_{i=1}^{m} p^0\left(X_Q(e_i)\right).$$

Then, we have

$$D^0(G,Q_1) - D^0(G,Q_2) = D_p^0(G,Q_1) - D_p^0(G,Q_2)$$
$$= -\sum_{i=1}^{m} p^0\left(X_{Q_1}(e_i)\right) + \sum_{i=1}^{m} p^0\left(X_{Q_2}(e_i)\right).$$

4.2.3 Pruning By Tight Upper Bounds of Objective Function (PTUB)

A tighter upper bound of the partial expected edit distance in this section.Suppose the O_0 order of those edges that have changed from Q_1 to Q_2 is $< e_1, e_2, \dots, e_m >$. For any cluster graph Q, the partial expected edit distance $D^0_p(G,Q)$ is defined in Section 4.2.2.

4.2.4 Optimizing By Redefining Objective Function (OROF)

In the proposed algorithms, joint probability tables are repeatedly read when calculating the edges' conditional probabilities.

5. CPGS CLUSTERING ALGORITHM

Spectral clustering refers to a class of techniques which rely on the eigen-structure of a graph Laplacian matrix to partition vertices into disjoint clusters with high intra-cluster and low inter-cluster similarity.

Based on this observation, one straightforward approach to extending the spectral clustering algorithm for correlated probabilistic graphs works as follows:

1) We map conditional probabilities into weights between each pair of adjacent vertices.

2) We extend the Dijkstra method to find the K nearest neighbuor (K-NN) of each vertex. It enumerates part of the possible world graphs and calculates the probability that avertex is the K-NN of the others when correlations exist among edges

3) We establish a Laplacian matrix according to the K-NN results, and compute the eigenvectors of it according to the power method.

4) We represent the vertices by points in a K-dimensional space, and cluster these points with a K-means algorithm. We call the straightforward method Spectral, and it will be used as a benchmark method in our experiments.

5.1 Establishing DPTCs

In the *CPGS* algorithm, cluster a graph by establishing DPTCs and representing these DPTCs as objects to be clustered. This can improve the time efficiency as the number of objects to be clustered is much smaller than the number of vertices. This paper represents vertices in the graph by points in a multi-dimensional space as the distance between vertices in graphs is complex to calculate compared with points in a multi-dimensional space.

5.2 Searching the *K*-NN of each DPTC

In *CPGS*, we need to find the K nearest neighbor (KNN) DPTCs of each DPTC to establish a graph Laplacian matrix. Before presenting the approaches to finding the KNN DPTCs, we define the similarity between two adjacent DPTCs based on the definition of estimated expected edit distance.

Given a correlated probabilistic graph $G = \{V, E, P, F\}$, and two adjacent DPTCs C_i and C_j in a DPTC graph G_D,the **similarity** between them is defined as $sim(C_i,C_j) = \Sigma e_p \in S(C_i,C_j)$

$p^k(X_D(e_p))$, where $S(C_i,C_j)$ is the set of edges between C_i and C_j in the correlated probabilistic graph, and $X_D(e_p)$ is the existence state of e_p in the DPTC graph G_D.

5.3 Establishing a Laplacian Matrix and Calculating Eigenvectors

In this subsection, how to establish a Laplacian matrix from the graph matrix *W* and an approach to calculating its eigenvectors. The *K* eigenvectors corresponding to the *K* largest eigenvalues of the Laplacian matrix are used to represent each object, namely DPTC in our algorithm, with a point in a multi-dimensional space .

First how to establish a symmetric Laplacian matrix *L(G)* from *W*. Set $W_{ij} = Max\{W_{ij},W_{ji}\}$, and thus *W* is a symmetric matrix. Then, a diagonal matrix *D* can be obtained, where $D_i = \Sigma j W_{ij}$. Subsequently, we have the Laplacian matrix $L(G) = D - W$.

6. EXPERIMENTS

6.1 Experimental Setup

The algorithms are implemented in Microsoft Visual Studio C++ on a PC with a 4 dual core CPU and 8GB memory. Use two real-life graph datasets in our experiments.

PPI network: Use a **PPI network** from the STRING database[4]. The network is modeled as a probabilistic graph by representing proteins as vertices, pairwise interactions as edges, and the reliability of each pairwise interaction as edge probability. This graph has 4, 179 vertices and 39, 309 edges.

YouTube social network: In the YouTube social network1, each vertex represents a user and an edge between two vertices represents there exists a connection between them. This dataset comprises 1, 134, 890 vertices and 5, 987, 624 edges. The edge existence probability is randomly generated to indicate the link reliability between users.

Correlation Simulation: These two datasets do not contain the correlation probabilities among adjacent edges. To generate these probabilities, we first present several definitions. For the *m* adjacent edges, define the **correlation rate** θ among these edges sharing the same vertex, so that the correlation only exists among $\lfloor m\theta \rfloor$ edges, and the other edges are independent of each other.

6.2 Performance of *PEEDR* Clustering Algorithm

Evaluate the performance of the *PEEDR* algorithm and its optimizations.

Efficiency of Optimizations: This set of experiments studies the effect of the optimizations for *PEEDR* in terms of the running time. Specifically, we compare the following algorithms:

BASIC: The *PEEDR* algorithm that is based on the DPTCs and does not employ any other optimizations proposed in Section 4.2.

OPSV: The difference of OPSV from the BASIC algorithm is that OPSV sorts vertices in descending order of their degrees.

PLB: Based on the OPSV algorithm, PLB adopts the pruning technique proposed in Section 4.2.2.

PTUB: Based on the PLB algorithm, the PTUB algorithm adopts the pruning technique in Section 4.2.3.

OROF: The *PEEDR* algorithm implemented with all the pruning techniques proposed in Section 4.

Fig. 4(a) and 4(b) report the efficiency of the different optimizations by varying the data size from 800 to 4, 000 on PPI and YouTube datasets, respectively. Fig. 5(a) and 5(b) illustrate the efficiency of the different optimizations by varying the correlation rate θ from 0.2 to 0.8.

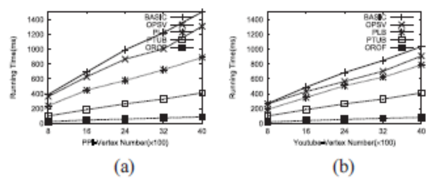


Fig. 4. *PEEDR*: efficiency vs vertex number: (a) PPI-vertex number(×100). (b) Youtube-vertex number(×100).

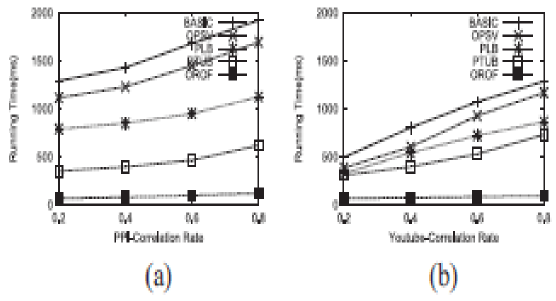


Fig. 5. *PEEDR*: efficiency vs θ: (a) PPI-correlation rate. (b) Youtube-correlation rate.

The effectiveness of ***PEEDR*****:** Evaluate the effectiveness of the *PEEDR* algorithm. The BASIC algorithm may generate different cluster graphs from the other algorithms. In other words, PLB, PTUB and OROF do not affect the effectiveness of the *PEEDR* algorithm. Thus, we only discuss the effectiveness of the BASIC and the OPSV algorithms.
Fig. 6(a) and 6(b) show the accuracy rate by varying the vertex number from 800 to 4000 on PPI and YouTube networks. The accuracy decreases as the vertex number increases, since the larger the vertex number, the bigger the deviation of the generated cluster graph from the optimal cluster graph is.

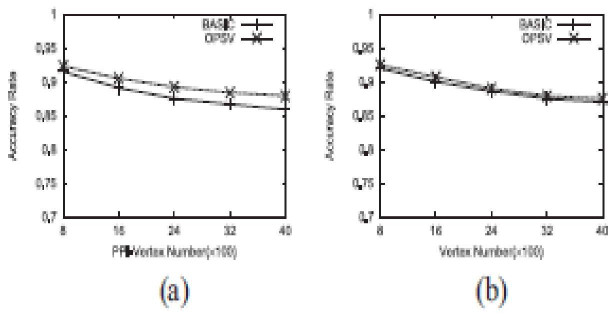


Fig. 6. *PEEDR*: effectiveness vs vertex number: (a) PPI-vertex number(×100). (b) Vertex number(×100).

OPSV performs better than BASIC as sorting vertices makes the vertices with higher degrees become the cluster centers, and thus OPSV generates better cluster graphs. Fig. 7 illustrates the effectiveness of the OPSV algorithm by varying θ onboth PPI and YouTube.

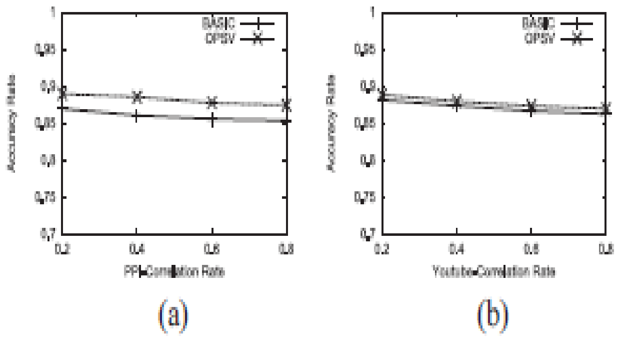


Fig. 7. *PEEDR*: effectiveness vs θ: (a) PPI-correlation rate. (b) Youtube-correlation rate.

The efficiency of *CPGS*: Fig. 8 reports the efficiency of the *CPGS* clustering algorithm and its different optimization versions by varying vertex number. Fig. 8 shows that the running time grows exponentially with the vertex number. Fig. 9 illustrates that the runtime grows linearly with the value of *K*. Furthermore, the runtime is almost not affected by the correlation rate θ, as presented in Fig. 10.

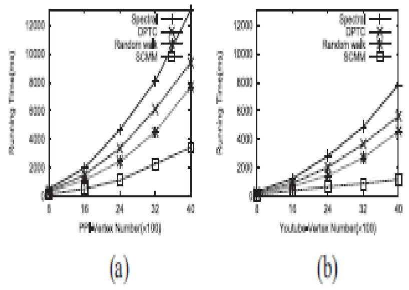


Fig. 8. *CPGS*: efficiency vs vertex number: (a) PPI-vertex number($\times$100). (b) Youtube-vertex number($\times$100).

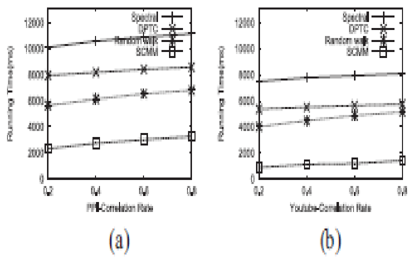


Fig. 10. *CPGS*: efficiency vs θ: (a) PPI-correlation rate. (b) Youtube-correlation rate.

6.4 Comparisons with Existing Methods

Fig. 13 reports the accuracy rate of different algorithms. The accuracy rate decreases as the vertex number increases. We can see that *CPGS* and *PEEDR* generate better cluster graphs than the *Furthest* algorithm, the Girvan-Newman algorithm and the spectral clustering algorithm do.

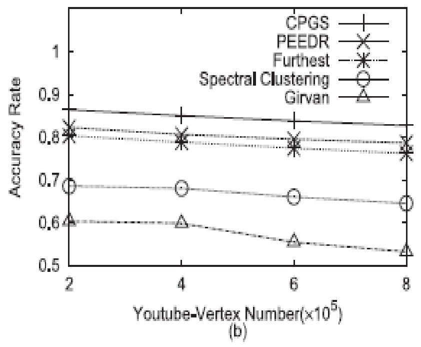


Fig. 13. Comparisons with existing methods.

7. CONCLUSION

In this paper, we have addressed the problem of clustering correlated probabilistic graphs and propose an efficient clustering algorithm named *PEEDR*. Based on the properties of joint probability, introduce several pruning methods for *PEEDR*. To achieve better effectiveness of clustering and also propose another clustering algorithm named *CPGS*. A comprehensive performance evaluation verifies the efficiency and effectiveness of our algorithms and pruning methods.

REFERENCES

[1] C. C. Aggarwal and H. Wang, Managing and Mining Graph Data,New York, NY, USA: Springer, 2010.

[2] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios, “K-nearest neighbors in uncertain graphs,” PVLDB, vol. 3, no. 1,pp. 997–1008, Sept. 2010.

[3] R. Jin, L. Liu, B. Ding, and H. Wang, “Distance-constraint reachability computation in uncertain graphs,” PVLDB, vol. 4, no. 9, pp. 551–562, Jun. 2011.

[4] Y. Yuan, G. Wang, L. Chen, and H. Wang, “Efficient subgraph similarity search on large probabilistic graph databases,” PVLDB, vol. 5, no. 9, pp. 800–811, May 2012.

[5] M. Hua and J. Pei, “Probabilistic path queries in road networks:Traffic ncertainty aware path selection,” in Proc. 13th Int. EDBT,New York, NY, USA, 2010, pp. 347–358.

[6] W. C. Wang and L. A. Demsetz, “Model for evaluating networks under correlated uncertainty-NETCOR,” J. Constr. Eng. Manage.,vol. 126, no. 6, pp. 458–466, 2000.

[7] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering:A review,” ACM Comput. Surv., vol. 31, no. 3, pp. 264–323, 1999.

[8] G. Kollios, M. Potamias, and E. Terzi, “Clustering large probabilistic graphs,” IEEE Trans. Knowl. Data Eng., vol. 25, no. 2,pp. 325–336, Feb. 2013.

[9] U. von Luxburg, “A tutorial on spectral clustering,” Statist.Comput., vol. 17, no. 4, pp. 395–416, Dec. 2007.

[10] M. R. Ackermann, J. Blömer, D. Kuntze, and C. Sohler,“Analysis of agglomerative clustering,” Algorithmica, vol. 69,no. 1, pp. 184–215, May 2014.