Constraint-based Graph Clustering through Node Sequencing and Partitioning

Yu Qian[1], Kang Zhang[1], and Wei Lai[2]

[1]Department of Computer Science
The University of Texas at Dallas, Richardson, TX 75083-0688, USA
{yxq012100, kzhang}@utdallas.edu
[2]School of Information Technology
Swinburne University of Technology, Hawthorn, Victoria, 3122, Australia
wlai@it.swin.edu.au

Abstract. This paper proposes a two-step graph partitioning method to discover constrained clusters with an objective function that follows the well-known min-max clustering principle. Compared with traditional approaches, the proposed method has several advantages. Firstly, the objective function not only follows the theoretical min-max principle but also reflects certain practical requirements. Secondly, a new constraint is introduced and solved to suit more application needs while unconstrained methods can only control the number of produced clusters. Thirdly, the proposed method is general and can be used to solve other practical constraints. The experimental studies on word grouping and result visualization show very encouraging results.

1 Introduction

As a widely recognized technique for data analysis, clustering aims at gathering closely related entities together in order to identify coherent groups, i.e., clusters. Clustering methods have proven to be very useful in many application areas including data mining, image processing, graph drawing, and distributed computing. This paper presents a novel graph theoretic partitioning approach to constrained clustering, analyzes and demonstrates the advantages of such an approach.

For constrained clustering, grouping similar units into clusters has to satisfy some additional conditions. Such additional conditions come from two kinds of knowledge: background knowledge and user requirements. While there have been some works investigating the use of background knowledge in clustering process, little research, however, performs in-depth analysis on the role of user-inputs in the process of clustering. According to the role of user-input in the clustering process, clustering criteria can be classified into two categories: user-centric and data-centric. The former involves and solves practical constraints in clustering process while the latter meets only the application-independent requirements such as high cohesiveness, low coupling, less noise, and etc. A practical clustering method should be both user-centric and data-centric, e.g., in most clustering algorithms the number of the clusters to be discovered is a necessary user-input while the application-independent min-max clustering principle must hold: the similarity between two clusters is significantly less than the similarity within each cluster [3].

The number of user-input constraints varies for different applications. In spatial data clustering, user-input can be minimized because there are naturally-defined potential clusters contained in the given data and finding these clusters is exactly the

final purpose. In many other cases, however, finding natural clusters is far from the destination, which makes it not enough to meet only one constraint on the number of the clusters. Let us use a simple example in Fig. 1 to demonstrate the necessity of involving more constraints.

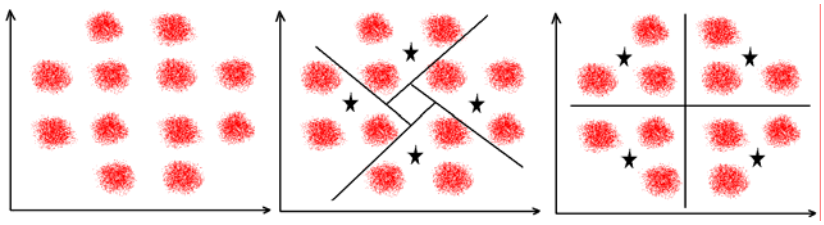


Fig. 1. (a) A population-density map with 12 communities. (b) A 4-clustering of the 12 communities. (c) A 4-clustering with a constraint on the distance between cluster centers.

Fig. 1 (a) is a map that describes the population distribution in a city. The denser the color is the more crowded the people are. A builder is planning to build some supermarkets in this city. He wants to choose four profitable locations of the supermarkets according to this map. So he uses some clustering algorithms to discover the clusters of people in order to put his supermarkets at the centers of clusters. In Fig. 1 (a), we can see that a correct and "good" data-centric clustering algorithm without any user-input would produce the twelve communities in this city as twelve clusters. Such a clustering result, unfortunately, is of little use because the builder can afford only four supermarkets. Fig. 1 (b) illustrates a feasible result after accepting the constraint on the number of clusters: the four stars indicate the best locations of the four supermarkets. The result shown in Fig. 1 (b), however, cannot satisfy the builder either. The builder needs a bigger distance between two supermarkets so that they can cover more area. After accepting one more constraint about the minimal allowable distance between two supermarkets, Fig. 1 (c) shows a desirable result. Fig. 1 has revealed an important phenomenon in many clustering applications: the clustering results that satisfy only the constraint on the number of clusters may not meet the practical requirements. It is necessary to involve more constraints into clustering process.

This paper introduces a new constraint into the generic clustering problem: the upper bound of the quantified similarity between two clusters. The similarity may have different quantified values in different application domains. In the above supermarket example, the similarity between two clusters is represented by the distance between two clusters: the nearer, the more similar. The upper bound of the similarity between two clusters is the minimum allowable distance between their centers. We believe that the upper bound of similarity is a general constraint which is required by many clustering applications. To support the argument, we further provide an example on parallel task partitioning: as we know, computers in a distributed environment lack global addressing space, communication has to be inserted whenever a processor needs to access non-local data. For example, on the Intel Paragon the processor cycle time is 20 nanoseconds whereas the remote memory access time is between 10000 and 30000 nanoseconds [8], depending on the distance between communicating processors. Therefore, it is imperative that the frequency and volume of non-local accesses are reduced as much as possible. Suppose that the whole computing task is composed of n

smaller tasks. Given k machines/processors, the attempt to assign the n small tasks to the k machines/processors is a k-clustering problem, which has several concerns: firstly, there is a communication bottleneck between the machines/processors. Secondly, it may not be worth to parallelize the computing task when the ratio of the communication cost to the total cost exceeds a preset threshold. Thirdly, in most cases, the number of available machines/processors is not always fixed but flexible. The three concerns can be exactly mapped into three corresponding constraints in clustering problem: the upper bound constraint, the min-max principle, and the number of clusters.

Motivated by the above examples, this paper proposes a graph theoretic model that can represent the above three requirements (two user-input constraints and one application-independent min-max principle): 1) the desired number of clusters; 2) the objective function of clustering that reflects the min-max principle, and 3) the upper bound of the similarity between two clusters. In particular, the desired number of clusters in the proposed model is represented with a range (K_{min}, K_{max}), minimum and maximum allowable number of clusters, respectively. The objective function is defined as the ratio of the similarity within the cluster to the similarity between clusters. Maximizing the proposed objective function not only follows the min-max principle but also meets certain practical requirements, e.g., in parallel task partitioning, the ratio decides if it is worth to parallelize the computation task; in spatial data clustering, the objective function is the ratio of the density inside the cluster to the density outside the cluster when representing the spatial data sets as sparse graphs. Based on the three requirements, this paper proposes a two-step graph partitioning methodology to discover constrained clusters. The basic idea involves node sequencing and then partitioning the node sequence according to the constraints. In the sequencing process, the graph nodes are sorted using existing algorithms. In the partitioning process we find an appropriate cut point according to the objective function along the ordered node sequence so that all points on one side will be output as a cluster while all points on the other side will remain for further partitioning until all constraints are satisfied or the number of produced clusters exceeds K_{max}.

The rest of this paper is organized as follows. Related work on graph partitioning and constrained clustering is introduced in Section 2. Section 3 proposes our two-step methodology. The experimental studies on word grouping and result visualization are presented in Section 4. Section 5 concludes the paper.

2 Related Work

Since this paper focuses on a constrained graph partitioning for data clustering, the related work can be categorized into two parts: graph partitioning and constraint-based data clustering.

2.1 Graph Partitioning

Numerous graph partitioning criteria and methods have been reported in the literature. We consider matrix transformation and ordering since our method proposes similar techniques. The optimal solution to the graph partitioning problem is NP-complete due to the combinatoric nature of the problem [3, 4]. The objective of graph partitioning is to minimize the cut size [3], i.e., the similarity between the two subgraphs with the requirement that the two subgraphs have the same number of nodes. It is important to

note that minimizing the cut size for each partitioning step cannot guarantee a minimal upper bound of the cut size for the whole clustering process.

Hagen and Kahng [6] remove the requirement on the sizes of the subgraphs and show that the Fiedler vector provides a good linear search order to the ratio cut (*Rcut*) partitioning criteria, which is proposed by Cheng and Wei [2]. The definition of *Rcut* is: $Rcut=cut(A,B)/|A|+cut(A,B)/|B|$, where $G=(V,E)$ is a weighted graph with node set V and edge set E, $cut(A,B)$ is defined as the similarity between the two subgraphs A and B and $|A|$, $|B|$ denote the size of A, B, respectively.

Shi and Malik [10] propose the normalized cut by utilizing the advantages of normalized Laplacian matrix: $Ncut=cut(A,B)/deg(A)+cut(A,B)/deg(B)$, where $deg(A)$ is the sum of node degrees, which is also called the volume of subgraph A, in contrast to the size of A. Ding *et al.* [3] propose a min-max cut algorithm for graph partitioning and data clustering with a new objective function called $Mcut=cut(A,B)/W(A)+cut(A,B)/W(B)$, where $W(A)$ is defined as the sum of the weights of the edges belong to subgraph A.

All objective functions above are designed for their algorithms to find an appropriate partitioning. They are algorithm-oriented and cannot reflect the practical requirements or physical meanings, which make them infeasible to serve a constrained clustering.

2.2 Constraint-based Data Clustering

According to Tung *et al.*, constraint-based clustering [11] is defined as follows. Given a data set D with n objects, a distance function $df: D \times D \rightarrow R$, a positive integer k, and a set of constraints C, find a k-clustering $(Cl_1, Cl_2, \ldots Cl_k)$ such that

$$DISP=\sum_{i=1}^{k} disp(Cl_i, rep_i)$$

is minimized, and each cluster Cl_i satisfies the constraints C, denoted as $Cl_i|=C$, where $disp(Cl_i, rep_i)$ measures the total distance between each object in Cl_i and the representative point rep_i of Cl_i. The representative of a cluster Cl_i is chosen such that $disp(Cl_i, rep_i)$ is minimized. There are two kinds of constraint-based clustering methods, aiming at different goals: one category aims at increasing the efficiency of the clustering algorithm while the other attempts to incorporate domain knowledge using constraints. Two instance-level constraints: must-link and cannot-link constraints have been introduced by Wagstaff and Cardie [12], who have shown that the two constraints can be incorporated into COBWEB [5] to increase the clustering accuracy while decreasing runtime. Bradley *et al*. propose a constraint-based *k*-means algorithm [1] to avoid local solutions with empty clusters or clusters with very few data points that can often be seen when the value of k is bigger than 20 and the number of dimensions is bigger than 10.

The proposed method is different from the aforementioned approaches: firstly, we combine graph partitioning and constrained-based clustering together. Secondly, the proposed partitioning process is different from the traditional partitioning in that for each partitioning step we produce only one cluster instead of two subgraphs. The remaining part will be further partitioned until all constraints are satisfied or the number of produced clusters exceeds K_{max}. Thirdly, the upper bound constraint we introduce is new and its involvement does not aim at improving the efficiency of the clustering process but aim at encoding the user's requirements. Finally, we accept both

unweighted and weighted graphs. Section 3 will introduce the proposed two-step methodology.

3 A Two-Step Methodology

This section first describes the process of node sequencing, and then introduces the node partitioning method.

3.1 Node Sequencing Method (NSM)

The process of node sequencing transforms a two-dimensional graph into a one-dimensional node sequence. The sequencing method used here is proposed in our previous paper [9]. Due to the space limitation, we only provide a brief introduction. The whole sequencing includes two steps: coarse-grained sequencing, which partitions the given graph into several parts and sorts these parts, and fine-grained sequencing, which sorts the nodes inside each part produced in the coarse-grained step. As a result, we obtain a sequence of nodes in which the nodes belonging to the same cluster will be put together. Then we can apply the node partitioning method to the node sequence to find the boundary points of clusters with constraints.

3.2 Node Partitioning Method (NPM)

This section proposes a novel node partitioning method. We first introduce the algorithm parameters used in the algorithm.

3.2.1 Algorithm Parameters

The algorithm parameters used in NPM include *alpha1*, *alpha2*, *beta*, and E_{inter}, which are defined as follows. Given a node sequence S of n nodes, and a cut at ith node separates S into two sub-sequences, say S_1, and S_2 where S_1 contains the nodes from 1 to i, and S_2 contains the nodes from $i+1$ to n. Let E_1 denote the number of edges inside S_1 and E_2 the number of edges inside S_2, and E_{inter} the number of edges between S_1 and S_2, we have the following algorithm parameters, as intuitively shown in Fig. 2.

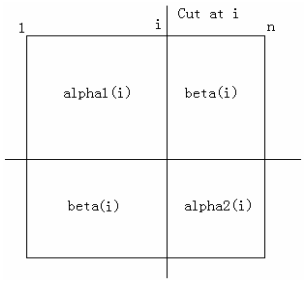


Fig. 2. The physical meanings of the algorithm parameters

$$alpha1(i) = E_1 / (i(i-1)/2) \quad (1)$$

$$alpha2(i) = E_2 / ((n-i)(n-i-1)/2) \quad (2)$$

$$beta(i) = E_{inter} / (((n-i) \cdot i)/2) \quad (3)$$

As Fig. 2 shows, the big square represents an adjacency matrix of the given graph, and the cut at i separates the node sequence $1...n$ into $1...i$, and $i+1...n$. *alpha1* represents the density of the upper left square, *alpha2* represents the density of the lower right square, and *beta* represents the density of the upper right or lower left square.

Given a node sequence S of n nodes, for every node i, $1<i<n$, "*cut at i*" means partitioning the sequence at the ith node. A cut at i is *acceptable* only if its corresponding objective function *cutvalue(i)* is a peak value. A *cutvalue(i)* is a peak value means that *cutvalue(i)* is bigger than any other *cutvalue* bwteen $i-\varepsilon$ and $i+\varepsilon$ where ε is a threshold defined as $n/2k$ and k is the number of desired clusters. *Cutvalue* measures the ratio of the similarity within the cluster and the similarity between clusters and helps discovering where we should separate the node sequence. Its definition is:

$$cutvalue(i)=\begin{cases} alpha\ \ 1(i) / beta\ (i) & beta\ (i) <> 0 \\ MAX\ \ + alpha\ \ 1(i) & beta\ (i) = 0 \end{cases} \quad (4)$$

The *MAX* in formula (4) is a very big constant used to distinguish the nodes when *beta(i)* is zero. According to the definitions, *beta(i)* represents the density of inter-cluster area while *alpha1(i)* is the density of intra-cluster area. The physical meaning of using the ratio of *alpha1(i)* to *beta(i)* is to effectively reflect the relative density. If *cutvalue* increases significantly, the corresponding cut point is more possibly located at the boundary of two clusters.

```
Algorithm ComputeCut(Graph g, Integer n)
begin
  for each i from 2 to n-2
     compute alpha1(i), beta(i), and
     cutvalue according to (4)
   for each node i from 1 to n do
     cut←getFirstPeak(cutvalue[i]);
   return cut;
end

Algorithm Npm (NodeSequence seq) {seq is
the result after node sequencing}
begin
   t←0, i←0;
   while (i<Kmin) do
    begin {the partitioning procedure}
      Remove the nodes before Node[t] from
      seq; n←n-t;
      Create the residual graph;
      t←ComputeCut(g,n); i←i+1;
    end
   while (Uinter>U) && i<Kmax) do
      repeat the partitioning procedure;
   if (Uinter>U) return clustering result;
   else return ("No such kind of clustering");
end
```

Fig. 3. The *ComputeCut* Algorithm and the whole *NPM* algorithm

Now let us define the upper bound constraint for the similarity between two clusters. For clusters C_i and C_j, and nodes $u \in C_i$, $v \in C_j$, if $(u,v) \in E$, we say (u,v) is an edge between clusters C_i and C_j. Let *inter(i,j)* denote the number of edges between clusters C_i and C_j and *sum_inter(i,j)* the sum of the weights of the edges between clusters C_i and C_j. We have the following definitions:

Definition 3.1 (Coupling Bound, Bound Constraint)

Coupling Bound is the biggest number of inter-cluster edges for a clustering result, represented by an integer U_{inter}: $U_{inter} = Max(inter(i,j)), \forall i,j \in \{1,2,....,k\}, i \neq j$. For weighted graphs, *coupling bound* is the maximal sum of the weights of inter-cluster edges, denoted by an integer $U_{inter\text{-}w}$: $U_{inter\text{-}w} = Max(sum\_inter(i,j)), \forall i,j \in \{1,2,...,k\}, i \neq j$. *Bound Constraint*, denoted by U, is a user-input constraint on the maximum allowable coupling bound. For a satisfactory clustering result, the formula $Uinter < U$ must hold (for weighted graph, the formula is $U_{inter\text{-}w} < U$).

Definition 3.2 (Granularity, G-constraint)

Granularity is defined as the number clusters for a clustering result, denoted by an integer k. G-constraint is a pair of integers (K_{min}, K_{max}) input by the user. For a satisfactory clustering result, the formula $K_{max} \geq k \geq K_{min}$ must hold.

Fig. 3 describes the algorithms that computes the *cutvalue* for each cut at i and returns the cut with the first peak value and the whole NPM algorithm.

4 Experimental Studies

Our experimental studies consist of three parts: the first part evaluates min-max principle and demonstrates that the parameter values represent the data distribution precisely. The second part of our experiments evaluates the ability of our approach on

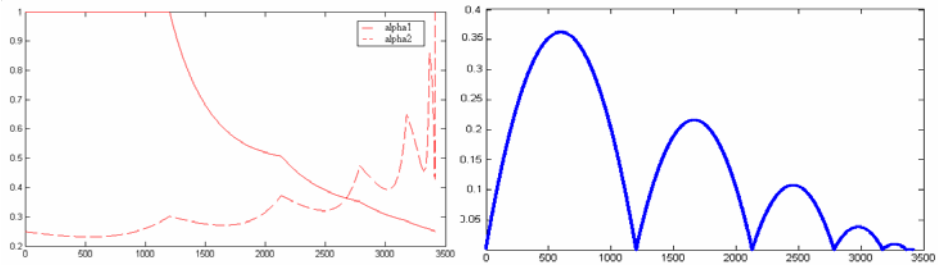
constraint satisfaction. Since there are two kinds of constraints involved in our algorithm: the upper bound constraint and the range of the desired number of clusters, the second part uses a set of synthetic constraints to evaluate if our algorithm can find the constrained clustering results for the given data set. The third part of our experiments visualizes the clustering results intuitively and compares our results with the one produced by Kamada and Kawai's method, a well-known force-directed (spring) graph clustering algorithm.

The clustering task in our experiments is word grouping, a basic technique of text mining and document clustering. The testing data are generated from the subsets of an English dictionary. In the first part of our experiments, two data sets: DS1 and DS2 are used. In the second part of our experiments, three data sets: DS3, DS4, and DS5 are used. The properties of their corresponding graphs are shown in Table 1. The similarity between two words is defined on their edit distance except for DS1. Each word is regarded as a graph node and if the edit distance between two words is less than a fixed threshold, there is an edge between the two corresponding nodes. For DS1, there is an edge between the two nodes if and only if the two corresponding words have the same length.

Table 1. The properties of the five testing data sets

Testing Data	Number of Nodes	Number of Edges	Threshold of Edit Distance	Result shown in
DS1	3419	146284	N/A	Fig. 4
DS2	472	428	1	Fig. 5
DS3	300	953	2	Fig. 6 (a)
DS4	1000	4908	2	Fig. 6 (b)
DS5	5000	312834	2	Fig. 6 (c)

4.1 Parameter Effectiveness



(a) (b)

Fig. 4. (a) Values of alpha1 and alpha2 for DS1 in the first partitioning step (b) Values of beta for DS1 in the first partitioning step

Fig. 4 (a) shows the computed values of the two algorithm parameters: *alpha1* and *alpha2* for DS1. According to the similarity definition of DS1, the words with the same length form a complete graph; the whole graph contains 6 complete subgraphs that are isolated from each other. The boundaries of the 6 subgraphs/clusters are clearly shown where the values of alpha1 drops dramatically.

Fig. 4 (b) shows the computed values of beta. We can see that the values of beta reach minimum at the boundaries of clusters while the values of *alpha1* reach maximum at the boundaries. According to the definition of *cutvalue*, the value of *alpha1(i)/beta(i)* would be significantly bigger when *i* is the index of a boundary point, i.e., the cutting point can be correctly found.

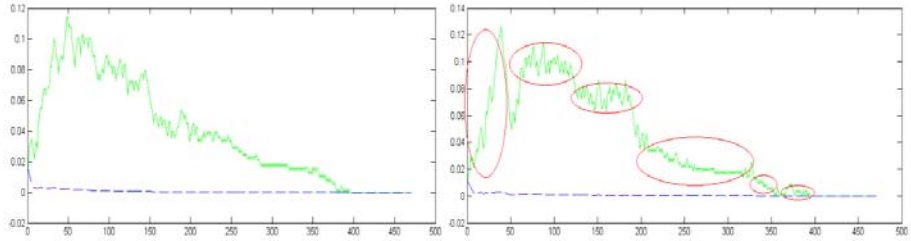


Fig. 5. The values of beta (a) before applying BEA and (b) after applying BEA for DS2.

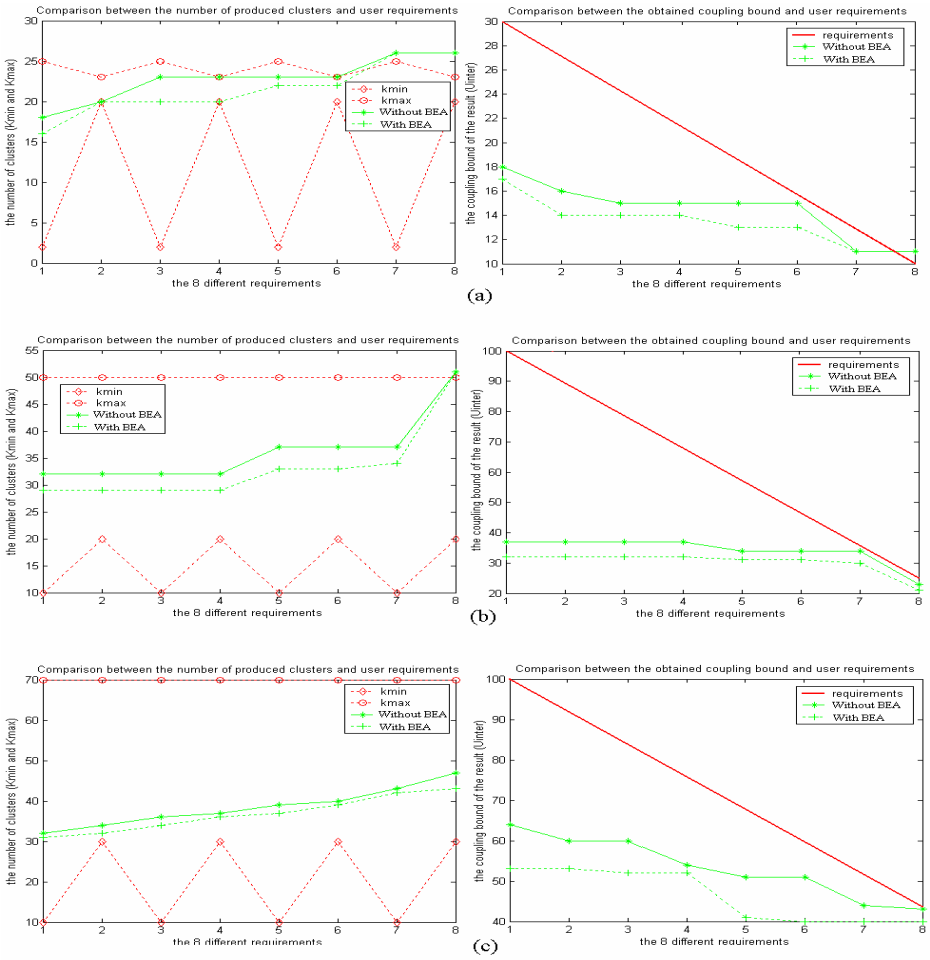


Fig. 6. the constraint satisfaction process for (a) DS3 (b) DS4 and (c) DS5

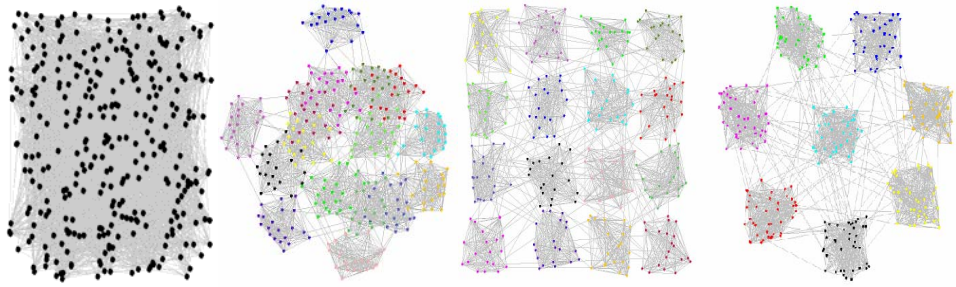
Fig. 5 shows the difference on the values of beta before and after applying the BEA. Fig. 5(a) shows that before applying the BEA the couplings between the data points are not very different and we cannot find a cut point to partition the node sequence. After applying BEA to the same graph, we find that the difference of couplings of different clusters is sharpened while the distribution of the coupling values inside the cluster becomes smoother. As clearly shown in Fig 5 (b), the graph contains six clusters.

4.2 Constraint Satisfaction

This part of experiments evaluates whether the proposed partitioning algorithm can produce satisfactory results according to different user-input constraints. Each testing graph is evaluated against 8 different clustering requirements and each requirement contains two kinds of constraints, as defined in Section 3. Each of Fig. 7 (a), Fig. 7 (b), and Fig. 7 (c), contains two sub-figures, corresponding to the two kinds of constraints respectively. The 8 requirements are designed from loose to strict, i.e., the first requirement is the easiest to be satisfied and the last is the hardest. If the produced granularity is between the minimum and maximum requirement, the corresponding clustering result is satisfactory on granularity; if the coupling bound of the produced result is below the upper bound constraint, the corresponding clustering result is satisfactory on coupling. If both kinds of constraints are satisfied, the clustering result is satisfactory.

4.3 Result Visualization

Another way to evaluate our approach is to visualize the results. Fig. 8(a) shows the original graph with 320 nodes. The graph nodes belong to the same cluster are in the same gray level. We apply a popular force-directed graph clustering algorithm: Kamada and Kawai's method [7] to the graph and its result is shown in Fig. 8 (b) while our results are shown in Fig. 8 (c) and (d). Although our approach does not compete with Kamada and Kawai's method on the quality of graph layout, it separates clusters clearly. In Fig. 8 (b) many graph nodes belong to different clusters are mixed up while our method can discover all clusters correctly. Fig. 8 (c) and (d) show that our method can produce different numbers of clusters according to the user input. Apart from the advantage on effectiveness, our method is faster than Kamada and Kawai's method. It takes only several minutes for our program to generate the results in Fig. 8 (c) and (d) while Kamada and Kawai's method needs more than 1 hour to reach a stable layout.



(a) (b) (c) (d)

Fig. 8. (a) the original graph before clustering (b) the same graph after applying Kamada and Kawai's method**.** The same graph after applying our algorithm with (c) 16 clusters (d) 8 clusters

5 Conclusions

This paper has presented a novel graph partitioning method for constrained data clustering. A new constraint: upper bound of the similarity between two clusters is introduced and solved with the proposed graph partitioning method. The method consists of two steps: sequencing the given set of graph nodes, and then partition the node sequence into final clusters. This method has at least two advantages: first, the objective function not only follows the theoretical min-max principle but also reflects certain practical requirements. Second, new constraints from practical clustering problems are introduced and solved so that the clustering results can be tailored to more application needs while unconstrained methods can only control the number of produced clusters. Our experimental studies have visualized the clustering results intuitively and demonstrated that the combination of graph partitioning and constrained data clustering is successful. Future work will explore whether it is possible to locate the feasible range of the constraints for a given clustering task so that the user can be guided on constraint input.

References

1. Bradley, P. S., Bennett, K. P., and Demiriz, A. Constrained *K*-Means Clustering, In MSR-TR-2000-65, Microsoft Research. (2000)
2. Cheng, C-K., and Wei, Y. A. An improved two-way partitioning algorithm with stable performance. IEEE. Trans. on Computed Aided Design. 10 (1991), pp. 1502-1511.
3. Ding, H. Q. C., He, X., Zha, H., Gu, M., and Simon, H. A Min-Max Cut Algorithm for Graph Partitioning and Data Clustering. Proc. of International Conf on Data Mining, (2001), pp. 107-114.
4. Donath, W. E. and Hoffman, A. J. Lower bounds for partitioning of graphs. IBM J. Res. Develop., 17 (1973), pp. 420-425.
5. Fisher, D. Knowledge acquisition via incremental conceptual clustering, Machine Learning, 2, (1987), pp. 139-172.
6. Hagen, L. and Kahng, A. B. New spectral methods for ratio cut partitioning and clustering. IEEE Trans. on Computed Aided Design, 11(1992), pp. 1074-1085.
7. Kamada, T. and Kawai, S. An algorithm for drawing general undirected graphs. Information Processing Letters, 31(1989), pp. 7-15.
8. Kandemir, M., Banerjee, P., Ramanujam, J., and Shenoy, N. A global communication optimization technique based on data-flow analysis and linear algebra. ACM Transactions on Programming Languages and Systems, Vol. 21, No. 6, (2000) pp.1251-1297.
9. Qian, Y. and Zhang, K.: A Customizable Hybrid Approach to Data Clustering. Proc. of the 2003 ACM Symposium on Applied Computing, (2003) 485-489.
10. Shi, J. and Malik, J.: Normalized cuts and image segmentation. IEEE Trans. on Pattern Analysis and Machine Intelligence. Vol. 22, No. 8, (2000), pp. 888-905.
11. Tung, A. K. H., Han, J., Lakshmanan, L. V. S., and Ng, R. T. Constrained-based clustering in large databases, Proc. 8th Intl. Conf. on Database Theory (ICDT'01), London, UK, (2001), pp. 405-419.
12. Wagstaff, K. and Cardie, C. Clustering with instance-level constraints, Proc. of the 17th Intl. Conf. on Machine Learning, (2000), pp. 1103-1110.