Graph Clustering Based on Optimization of a Macroscopic Structure of Clusters

Yuta Taniguchi and Daisuke Ikeda

Department of Informatics, Kyushu University
{yuta.taniguchi,daisuke}@inf.kyushu-u.ac.jp

Abstract. A graph is a flexible data structure for various data, such as the Web, SNSs and molecular architectures. Not only the data expressed naturally by a graph, it is also used for data which does not have explicit graph structures by extracting implicit relationships hidden in data, e.g. co-occurrence relationships of words in text and similarity relationships of pixels of an image. By the extraction, we can make full use of many sophisticated methods for graphs to solve a wide range of problems. In analysis of graphs, the *graph clustering problem* is one of the most important problems, which is to divide all vertices of a given graph into some groups called *clusters*. Existing algorithms for the problem typically assume that the number of intra-cluster edges is large while the number of inter-cluster edges is *absolutely small*. Therefore these algorithms fail to do clustering in case of noisy graphs, and the extraction of implicit relationships tends to yield noisy ones because it is subject to a definition of a relation among vertices. Instead of such an assumption, we introduce a *macroscopic structure* (MS), which is a graph of clusters and roughly describes a structure of a given graph. This paper presents a graph clustering algorithm which, given a graph and the number of clusters, tries to find a set of clusters such that the distance between an MS induced from calculated clusters and the ideal MS for the given number of clusters is minimized. In other words, it solves the clustering problem as an optimization problem. For the m-clustering problem, the ideal MS is defined as an m-vertex graph such that each vertex has only a self-loop. To confirm the performance improvements exhaustively, we conducted experiments with artificial graphs with different amounts of noise. The results show that our method can handle very noisy graphs correctly while existing algorithms completely failed to do clustering. Furthermore, even for graphs with less noise, our algorithm treats them well if the difference between edge densities of intra-cluster edges and those of inter-cluster edges are sufficiently big. We also did experiments on graphs transformed from vector data as a more practical case. From the results we found that our algorithm, indeed, works much better on noisy graphs than the existing ones.

Keywords: graph clustering, noisy graph, macroscopic structure, optimization

1 Introduction

A graph is a flexible and expressive data structure composed of vertices and edges, which describes relationships between entities in edge connectivity of vertices. It is used in various fields to represent data such as the World Wide Web made of Web pages and hyperlinks between them [2, 3, 11, 12], social networks expressing friendships of people [10, 15], co-authors' relationships [14, 16], computer communication networks [1], biological networks [9, 17] and so on.

There exist many graphs which contain some groups of vertices. For instance, a group of Web pages in the World Wide Web corresponds to a topic of the content and a group found in an electronic circuit may correspond to a functional unit. The problem to identify such groups in a given graph is called *graph clustering problem*, and many tasks are modeled as the problem as follows: social network analysis [15], image segmentation [18], natural language processing [6], circuit layout [7] etc.

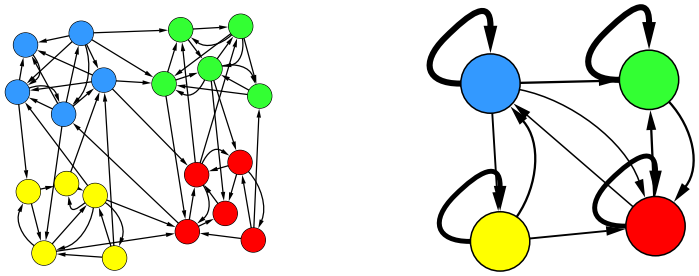
Some of the tasks handle graphs which explicitly have graph structure, e.g. social networks and molecular structures, but the other tasks, such as the image segmentation task, handle artificial graphs extracted from an original data, such as graphs representing pixels' similarity relationships on images and those describing co-occurrence relationships of words in text. Therefore, it is important for graph clustering algorithms to treat such graphs appropriately as well as natural ones.

The graph clustering problem has been extensively studied and thus there exist many algorithms for the problem. Although each paper defines its own definition of the cluster, the most of them assume that intra-cluster edges are dense while inter-cluster edges are absolutely sparse. For instance, assuming that clusters are connected by only a few edges, GN algorithm proposed by Girvan and Newman [10] uses "edge betweenness", a centrality measure of an edge, to identify edges between clusters.

This assumption makes these methods unusable especially on graphs transformed from other data because the extraction of implicit relationships is subject to a definition of a relation among vertices and tends to yield noisy ones which don't satisfy the assumption. So, to solve the tasks by exploiting such implicit relationships, it is important to develop graph clustering algorithms which don't rely on the absolute sparseness of inter-cluster edges.

Instead of such an assumption, we introduce a *macroscopic structure* (MS), which is a graph of clusters and roughly describes how a given graph is structured by the clusters (Fig. 1). In this example, an MS composed of four clusters is induced from an original graph according to a given coloring. The coloring describes correspondence relationships between vertices and clusters, and it is called a "view", we will describe precisely later.

This idea to utilize the MS for clustering is inspired by HITS algorithm [11], a link analysis algorithm which identifies hubs and authorities within a set of Web pages related to a particular search query. The paper explains that the hub pages and the authoritative pages have a mutually reinforcing relationship "a good hub is a page that points to many good authorities; a good authority is a



(a) Original graph (unweighted graph)

(b) Macroscopic structure (weighted graph)

Fig. 1. A macroscopic structure (graph on the right hand side) induced from an original graph (one on the left hand side) according to a given coloring (called a "view" later). In the graph on the right hand side, the width of an edge denotes a weight assigned to the edge.

page that is pointed to by many good hubs." We think such relationships also exist for each graph-related problems, and we call the relationships MSs.

This paper presents a graph clustering algorithm which, given a graph for clustering and the number of clusters, tries to find a set of clusters such that the distance between an MS induced from clusters calculated from the given graph and the ideal MS for the given number of clusters is minimized. In other words, it solves the clustering problem as an optimization problem. For the m-clustering problem, the ideal MS is defined as an m-vertex graph such that each vertex has only a self-loop.

We conducted experiments on artificial graphs with various amount of noise to investigate how the accuracies of the clustering results of existing algorithms and ours change. The results present that our algorithm is more accurate on most of graphs we generated than the existing algorithms, and they failed to do clustering especially on noisy graphs. We also did experiments on graphs transformed from vector datasets for clustering as more practical cases, and our algorithm outperforms the existing algorithms on noisy graphs.

The rest of this paper is structured as follows. Section 2 explains existing methods for graph clustering problems and points out their drawback and introduces HITS algorithm from which we are inspired. Section 3 describes our algorithm in detail, and Section 4 shows results of experiments for comparing our algorithm and the existing ones. Finally we conclude the paper in Section 5.

2 Related Work

First, we describe previous works of graph clustering problems and point out their drawbacks which come from the same assumption. Then we make an introduction of HITS algorithm, which we were influenced from.

2.1 Previous Work

There exist two types of graph clustering algorithms: hierarchical graph clustering algorithms and non-hierarchical graph clustering ones. The former does clustering by iteratively removing an edges or merging clusters, and create a hierarchy of clusters. On the other hand, non-hierarchical graph clustering does not rely on such graph operations but on some computation of quantities like flow.

GN algorithm proposed by Girvan and Newman [10] is one of the most famous method for graph clustering problems. This is a hierarchical clustering algorithm, which constructs a hierarchy of clusters instead of a set of clusters. This algorithm is based on "edge betweenness" measure that evaluates an edge in a given graph with the number of shortest paths between pairs of vertices in the graph which includes the edge. If a graph is composed of several clusters, because two clusters must be connected by only few edges, most of shortest paths between different clusters may include ones of such few edges. Iteratively evaluating edges and removing the edge with the highest betweenness, the algorithm separates the clusters and finally reveals their structure. The whole process of the algorithm is as follows: 1) calculate the betweenness for each edges, 2) remove the edge with the highest betweenness, 3) recalculate the betweenness for each remaining edges, and 4) iterate 2 and 3.

Another well known algorithm for graph clustering problems is Markov Clustering (MCL) algorithm proposed by Stijn van Dongen [5], which is a non-hierarchical algorithm based on random walk on a graph. This algorithm iteratively applies the following three operations on a transition matrix M initially calculated from the adjacency matrix of a given graph until the matrix converges: *expansion*, *inflation* and *prune* in order. Let $G = (V, E)$ be an input graph, where V, E denote a vertex set and an edge set respectively. Let A be a $|V| \times |V|$ adjacency matrix of G, where $A_{ij} = 1$ if and only if $(v_i, v_j) \in E$ and $A_{ij} = 0$ otherwise. Let M be a transition matrix, where its element is defined as $M_{ij} = A_{ij} / \sum_{k=1}^{|V|} A_{kj}$. In the expansion step the random walk is performed by updating M by $M = M \times M$. Inflation step is defined as $M_{ij} = M_{ij}^r / \sum_{k=1}^{|V|} M_{kj}^r$. This step is executed for emphasizing the difference of higher probabilities and lower ones in M. At the end of inflation step, prune step is performed, in which an element of M with sufficiently small probability are removed to save memory for storing M. After M converges, we interpret the computed M as clustering by making clusters of vertices connected through transitions of M.

The most of the existing algorithms including the above focus on the difference between intra-cluster edge density and inter-cluster one. However, in fact they additionally assume that edges between different clusters are *absolutely sparse*. The assumption harms the accuracy of the algorithms on graphs extracted from original data as an implicit relationships because the extraction process tends to yield noisy graphs and such graphs don't meet the assumption. So, in this paper, we focus only on relative difference of densities between clusters by exploiting a macroscopic structure of a given graph.

2.2 HITS as a Classification Method

Our main idea is to utilize a macroscopic structure for graph clustering problems. This idea is inspired by Kleinberg's HITS algorithm [11], which identifies authoritative Web pages by analyzing hyperlink graph.

To identify the authoritative pages within a set of Web pages obtained by a query, Kleinberg assumed that hyperlink graph of the pages is structured by two types of pages: authoritative pages called *authority* and pages of collection of links called *hub*. According to the assumption, these pages have a mutually reinforcing relationship, i.e. "a good hub is a page that points to many good authorities, a good authority is a page that is pointed to by many good hubs," and HITS utilizes this structure of hyperlink graph for identifying authorities. Seeing HITS algorithm as one to divide Web pages into two groups leads us to the application of their idea to the graph clustering problems.

Though the idea of HITS is considered useful, we cannot apply the algorithm directly to graph clustering because it is only for classifying into hubs and authorities and the structure they assume doesn't meet graph clustering. In the next section, we propose appropriate structures for graph clustering problems and reformulate their idea as a more general optimization problem.

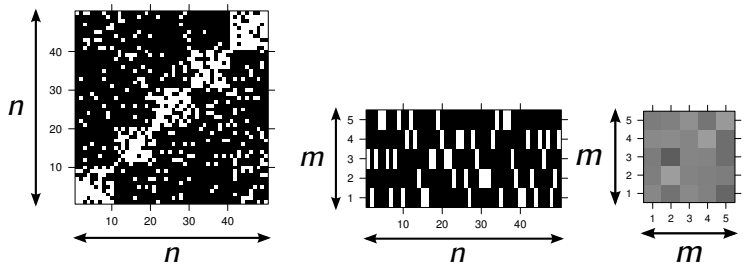
3 Our Algorithm

3.1 The Problem

This paper treats a hard graph clustering problem, which allows each vertex of a given graph to belong only a cluster. Our algorithm takes a directed graph $G = (V, E)$ and the number of clusters m as input, where $V = \{v_i\}_{i=1}^{n}$ and $E \subseteq V \times V$, and outputs a set of non-empty disjoint clusters $\mathbb{C} = \{C_i\}_{i=1}^{k}$, where $C_i \in 2^V$. In fact, as we discuss later, our algorithm does soft clustering, which allows vertices to belong several clusters probabilistically. This makes our optimization problem easier.

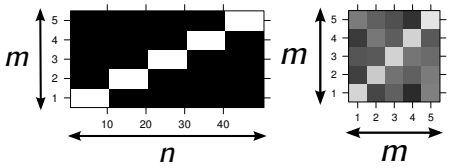
3.2 Macroscopic Structures

First, we briefly explain our idea, and then we describe our algorithm in detail. In our algorithm a macroscopic structure (MS) plays important role. Given a graph and a *view*, an MS is defined to be a graph of clusters and represents coarse structure of the original graph through the view. A view is a matrix and describes which cluster each vertex belongs, i.e. this is a clustering and what we want to obtain. Our goal is to find the most appropriate view for a given graph. Figure 2 shows an example cases of application of views to an original graph. In the figure, the adjacency matrices of an original graph and calculated MSs by using given views are drawn as bitmaps, and n, m are the number of vertices in the original graph and clusters introduced by the views respectively. Figure 2(b) shows a bad case where a randomly generated view is applied to the graph, and we can see from the MS the structure of the original graph is not



(a) Original graph (unweighted graph)

(b) Inappropriate case (left: a view; right an MS)



(c) More appropriate case (left: a view; right an MS)

Fig. 2. Applications of views to an original graph. Graphs and views are expressed in matrix and are drawn as bitmaps, where n is the number of vertices in the original graph and m is the number of clusters introduced by the views. Here two cases are shown: (b) shows an inappropriate case that a randomly generated view is applied; and (c) shows an appropriate case that a more appropriate view is applied.

correctly captured. On the other hand, in Fig. 2(c), the structure of the graph is more properly captured using a view.

Let $G = (V, E)$ be a given graph, where $V = \{v_1, v_2, \ldots, v_n\}$ and $E \subseteq V \times V$. Let $B = [b_{ij}]$ be an $n \times m$ view matrix, where m is the number of clusters, $b_{ik} \in \{0, 1\}$ and $\sum_k b_{ik} = 1$ for all $1 \leq i \leq n$. Using the view matrix, we obtain an MS $D(B)$, where $D(B) = [d(B)_{kl}]$ is an $m \times m$ matrix and its element is computed as follows:

$$d(B)_{kl} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} b_{ik} b_{jl} a_{ij}}{\sum_{i=1}^{n} \sum_{j=1}^{n} b_{ik} b_{jl}} = \frac{B^T A B}{B^T \mathbf{1} B}, \tag{1}$$

where $\mathbf{1}$ is an $n \times n$ matrix with all elements one and $A = [a_{ij}]$ is an adjacency matrix of the graph G, i.e. $a_{ij} = 1$ if $(i, j) \in E$ and $a_{ij} = 0$ otherwise.

The i-th row of a view matrix B corresponds to a vertex v_i and represents which cluster the vertex belongs to, i.e. $b_{ik} = 1$ if and only if $v_i \in C_k$. Using the relationships between vertices and clusters, Eq. (1) computes a cluster-cluster relationship, which denotes a fraction of edges between cluster k and l in G over the all possible ones between the clusters.

3.3 Optimization

We assume that an ideal MS computed by applying a proper view (clustering) looks like a graph where only vertices and self-loop edges with weight 1 exist, i.e

a graph whose adjacency matrix is an identity matrix. Our main idea is based on comparison between the ideal MS and one obtained through applying of a temporal view, and is to try to minimize the difference between them. This minimization process maximizes the edge densities within the same cluster and minimizes edge densities between different clusters.

Given the number of clusters m, we define the ideal MS as the $m \times m$ identity matrix and denote it by I. Employing Frobenius norm of the difference $D(B) - I$ as the comparison method, we formulate a m-graph clustering problem as the following optimization problem:

$$\begin{array}{ll} \text{minimize} & f(B) = \sum_{k=1}^{m} \sum_{l=1}^{m} \left((D(B) - I)_{kl} \right)^2, \\ \text{subject to} & b_{ik} \in \{0, 1\} \text{ and} \\ & \sum_k b_{ik} = 1. \end{array}$$

This optimization problem is difficult to solve directly because this is a combinatorial optimization problem. So we relax the constraints by allowing b_{ik} to be a real number between 0 and 1, and finally we solve the following non-linear optimization problem:

$$\begin{array}{ll} \text{minimize} & f(B) = \sum_{k=1}^{m} \sum_{l=1}^{m} \left((D(B) - I)_{kl} \right)^2, \\ \text{subject to} & 0 \leq b_{ik} \leq 1 \text{ and} \\ & \sum_k b_{ik} = 1. \end{array}$$

This relaxation makes the problem a soft clustering problem, which allow a vertex to belong multiple clusters probabilistically.

Though we don't discuss any actual optimizing method for the problem in this paper, the existing non-linear optimization methods, e.g. quasi-Newton methods as we used in our experiments, can be used to solve the problem above.

4 Experiment

We did two experiments to see the difference of accuracy between our algorithm and existing algorithms on noisy graphs. First, in order to investigate how the amount of noise of an input graph impact on the accuracy of our algorithm and existing ones, we conducted experiments by applying the algorithms on artificially generated graphs with various configurations of noisiness. We generated 55 types of graphs by changing two parameters p_{in} and p_{out} which control the amount of edges within the same cluster and ones between different clusters respectively. Next, we did experiments about an application to normal clustering tasks as more practical cases by extracting graphs from clustering datasets and applying graph clustering algorithms on them. We generated ten vector datasets and transformed them into graphs in many ways by changing a parameter α of the transformation.

4.1 Evaluation

For evaluating results of graph clustering, we adopted the *normalized mutual information* (NMI), which is widely used for evaluations of graph clustering

algorithms. NMI is based on mutual information (MI), which is well known concept in the information theory field and measures the dependency between two random variables.

Let X and Y be random variables, and $p(\cdot)$ denotes the probability of them. Then the MI between the random variables is defined as follows:

$$\mathrm{MI}(X,Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}.$$

In a context of the graph clustering, $p(\cdot)$ is defined as the fraction of as follows:

$$\mathrm{MI}(\Omega,\mathbb{C}) = \sum_{k=1}^{s}\sum_{l=1}^{m} \frac{|\omega_k \cap C_l|}{N} \log \frac{|\omega_k \cap C_l| / N}{(|\omega_k| / N)(|C_l| / N)}$$

where both $\Omega = \{\omega_1, \omega_2, \ldots, \omega_s\}$ and $\mathbb{C} = \{C_1, C_2, \ldots, C_m\}$ are sets of vertices, Ω is an output from a graph clustering algorithm to be evaluated and $\mathbb{C}$ is a known correct clustering result. MI for graph clustering problems can be interpreted as the quantity of the information about the correct cluster of a vertex we will get after we know that a vertex is included in a cluster in Ω.

MI depends on the numbers of clusters in Ω and $\mathbb{C}$, and one-to-one clustering, where a single vertex forms a cluster, gets the highest score. Thus NMI, the normalized version of MI, is usually used for graph clustering algorithms to eliminate the influence of the different numbers of clusters [19]. NMI is defined as follows:

$$\mathrm{NMI}(\Omega,\mathbb{C}) = \frac{\mathrm{MI}(\Omega,\mathbb{C})}{-\frac{1}{2}\left(\sum_k \frac{|\omega_k|}{N} \log \frac{|\omega_k|}{N} + \sum_k \frac{|c_k|}{N} \log \frac{|c_k|}{N}\right)}.$$

By this normalization, NMI score takes a number from 0 (worst accuracy) to 1 (best accuracy).

4.2 Environment

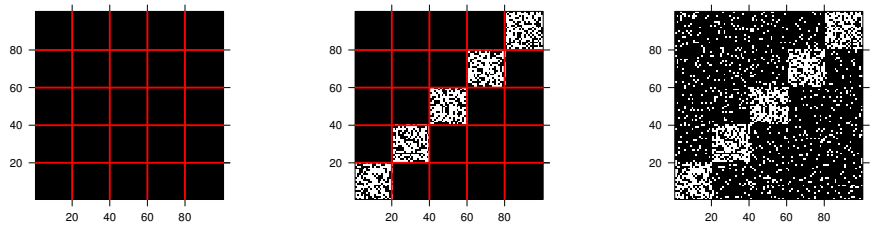
All experiments were conducted on a single Linux machine composed of Intel Core i3 2.93 GHz and 4 GB of memory. We implemented our algorithm in C++ language. As the solver for our optimization problem, we used IPOPT library [20] of version 3.8.3. This is a software package for large-scale nonlinear optimization and implements the limited-memory quasi-Newton (L-BFGS) method [13]. We compared our algorithm with GN algorithm and MCL algorithm. We used an implementation of GN algorithm included in igraph library [4] and an implementation of MCL algorithm provided by its author. All source code was compiled by GCC 4.5.2.

4.3 Exp. 1: Exhaustive Study on Variously Noisy Graphs

Artificial Graph. To generate graphs which contain desired amount of noise, we developed Algorithm 1, which takes four parameters: the number of vertices n, the number of clusters k and the probabilities p_{in} and p_{out} of generating an edge

Algorithm 1 Generate a graph

Input: n, k, p_{in}, p_{out}
Output: graph G
  $V \leftarrow \{v_1, v_2, \ldots, v_n\}$
  $\mathbb{C} \leftarrow$ divide_into_clusters(V, k) {Fig. 3(a)}
  $E \leftarrow \emptyset$
  for $i = 1$ to k **do**
    for $j = i$ to k **do**
      if $i = j$ **then**
        $E \leftarrow E \cup$ generate_edge_random_between$(p_{\text{in}}, C_i, C_j)$ {Fig. 3(b)}
      else
        $E \leftarrow E \cup$ generate_edge_random_between$(p_{\text{out}}, C_i, C_j)$ {Fig. 3(c)}
      end if
    end for
  end for
  return (V, E)



(a) Divide vertices into k clusters

(b) Add edges between the same cluster with p_{in}

(c) Add edges between different clusters with p_{out}

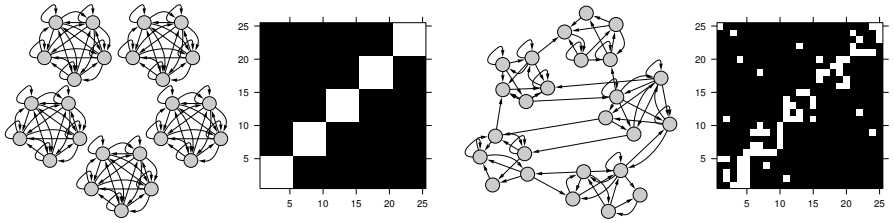
Fig. 3. Process of generating a graph of size n with k clusters. The adjacency matrices of graphs in the process are shown as bitmaps, where its horizontal and vertical axises correspond to the row and column indices of a matrix respectively and a cell on (i, j) is filled with white color if and only if there are an edge from vertex i to vertex j.

within a cluster and an edge between clusters respectively. Figure 3 shows an example of the process of generating a graph.

Figure 4 shows examples of artificial graphs generated by the algorithm. The graphs are drawn as a bitmap of its adjacency matrix, where its horizontal and vertical axises correspond to the row and column indices of a matrix respectively and a cell on (i, j) is filled with white color if and only if there are an edge from vertex i to vertex j.

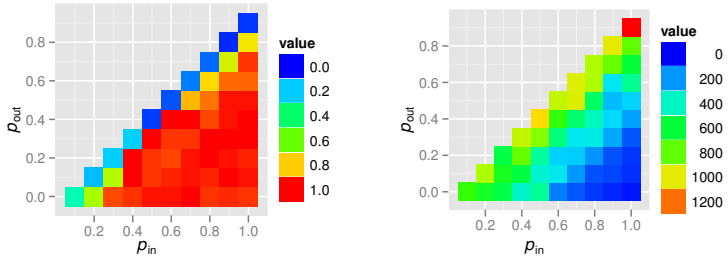
Performance of Our Algorithm. Here we show the performance of our algorithm on large graphs. We gave to Algorithm 1 the following 55 parameter value pairs that satisfy $p_{\text{in}}, p_{\text{out}} \in \{0, 0.1, 0.2, \ldots, 1\}$ and $p_{\text{in}} > p_{\text{out}}$: $(p_{\text{in}}, p_{\text{out}}) = (0.1, 0), (0.2, 0), (0.2, 0.1), \ldots, (1, 0.9)$. For each of the parameter value pairs, we randomly generated ten artificial graphs which has 1000 vertices equally divided into five clusters (i.e. 200 vertices in a cluster). We applied our algorithm on them, evaluated the results using NMI, and finally we averaged the evaluation scores every parameter value pair.

Figure 5 shows the results of these experiments, where both Fig. 5(a) and Fig. 5(b) have 55 colored cells corresponding to the pairs of $(p_{\text{in}}, p_{\text{out}})$. Hue from



(a) A graph which has five isolate cliques (b) A noisier graph which has weakly connected five clusters

Fig. 4. Two examples graphs generated by Algorithm 1 are shown in two ways: graphical representation (on the left hand side) and bitmap representation (on the right hand side). In the bitmap representation, horizontal and vertical axises correspond to the row and column indices of an adjacency matrix respectively and a cell on (i, j) is filled with white color if and only if there are an edge from vertex i to vertex j. Figure (a) is made by only adding all possible edges between vertices within the same cluster, and (b) is made by adding noise to the graph.

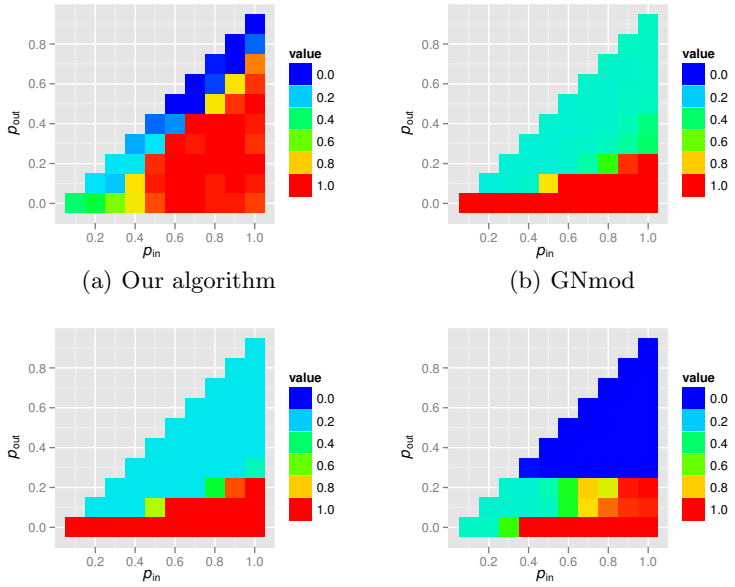


(a) NMI score (b) Computation time

Fig. 5. The results of our algorithm on graphs which has 1000 vertices equally divided into five clusters. For each parameter pair $(p_{\text{in}}, p_{\text{out}})$, corresponding cell is colored according to its NMI score and its computation time.

blue (minimum value) to red (maximum value) correspond to their NMI score or computation time. Figure 5(a) shows that our algorithm achieved high evaluation score in most of the cases. Furthermore, we can see the evaluation score and the computation time depend only on the *difference* of probabilities $(p_{\text{in}} - p_{\text{out}})$. From this observations, we can conclude that our algorithm is not affected by the *absolute amount* of noises in a given graph, and it can detect the difference of the edge density within the same cluster and one between different clusters. This property is very natural and indicates that our algorithm is suite for the tasks which treat noisy graphs extracted from original data.

Comparison with Previous Work. This section show a comparison among our algorithm, GN algorithm [10] and MCL algorithm [5]. The procedure of experiments is the same as the previous experiments. Unlike the previous ones, in this experiments, every graph has 128 vertices equally divided into four clusters (i.e. each cluster has 32 vertices).



(a) Our algorithm (b) GNmod

(c) GNnum (d) MCL (inflation = 4.0)

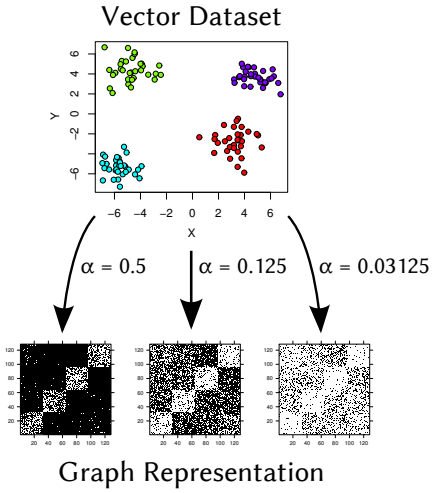
Fig. 6. Comparison of accuracy among our algorithm, GNmod, GNnum and MCL. The result of an experiment on graphs which has 128 vertices equally divided into four clusters. For each parameter pair $(p_{\text{in}}, p_{\text{out}})$ used to generate the graphs, corresponding cell is colored according to NMI score.

Since GN algorithm is a hierarchical graph clustering algorithm, to obtain an actual clustering result, we need to choose particular hierarchy. Here we used two different ways: a way to maximize the modularity measure like [15] (here we call it GNmod) and a way to choose the hierarchy which produces a result has the correct number of clusters (we call it GNnum). To execute MCL algorithm, it is necessary to properly set a parameter value called "inflation", which adjusts the granularity of the clustering and actually determines the number of the clusters output by MCL. We tried four values suggested by the author, and show the best result (inflation = 4.0) in Fig. 6.

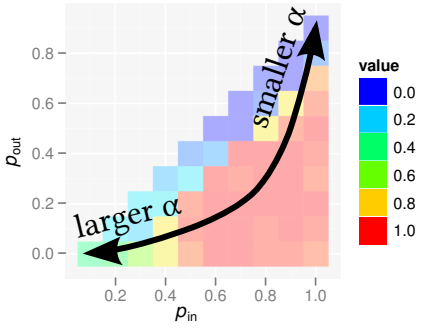
Figure 6(a) shows a result of our algorithm, and Fig. 6(b), 6(c) and 6(d) show the results of GNmod, GNnum and MCL algorithm respectively. We can see that existing algorithms fail to cluster in the upper part of the triangle and they succeed only in the lower part. This indicates that the previous works rely on the *absolute sparseness* of edges between different clusters, and that they can't detect the small differences of the edge densities within the same cluster and between different clusters.

4.4 Exp. 2: Application to Vector Clustering

Finally, we compare the accuracies of the algorithms in the so-called "clustering" application task by giving the algorithms graphs which are extracted



(a) Examples of transforming a vector dataset into graphs. Three different values of a parameter α generate graphs which have various amount of edges (noise).



(b) Approximate characteristic curve of generated graphs by changes in the transformation parameter α. A small value of α generates graphs with a lot of edges (the top right of the triangle, a large value of α generates graphs with a few edges (the bottom left of the triangle

Fig. 7. Transformation of a vector dataset.

from datasets consisting of vectors and reflect Euclidean distance relationships of them. We employed artificially generated datasets and Fisher's well known Iris dataset [8] for this experiment.

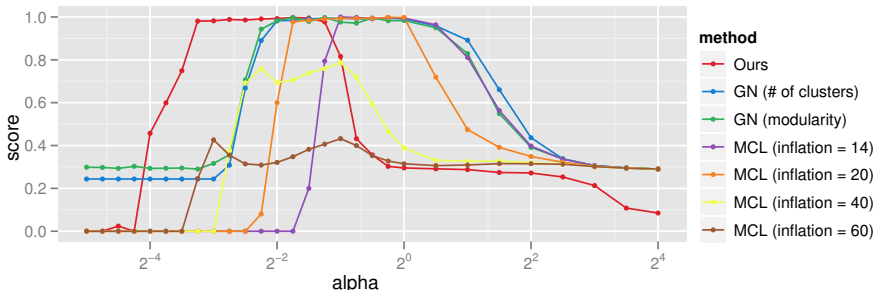
For the artificial datasets, a d-dimensional vector dataset of size nk for k-clustering problem was generated as follows:

1. uniformly choose k centers of clusters from $[0, 1]^d$; and
2. for each cluster, sample n vectors from a d-dimensional multivariate normal distribution with mean vector of its center vector and covariance matrix of the identity matrix of size d.

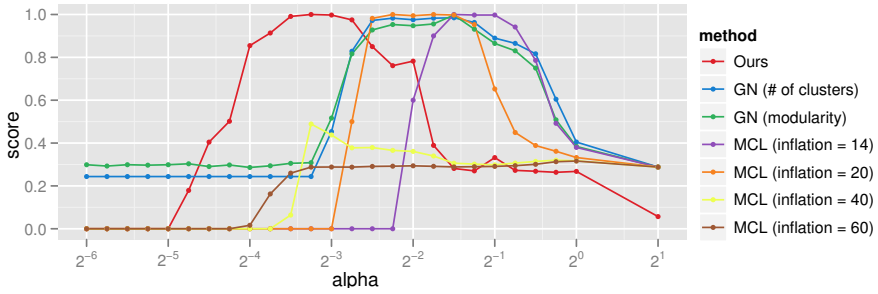
We transformed a dataset into a graph by expressing the strength of the relationship between two vectors in a Euclidean space as the probability of linking the corresponding nodes in a graph, An adjacency matrix $A = [a_{ij}]$ for a given dataset $D = \{v_i\}$ is constructed as the following:

$$a_{ij} = \begin{cases} 1 & \text{if rand}() < \exp(-\alpha \left\| v_i - v_j \right\|) \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

where v_i, v_j is vectors in the dataset, rand() is a call of random number generator which return a real value in $[0, 1]$, $\alpha \in (0, \infty)$ is a parameter which controls the number of edges to be generated and $\|\cdot\|$ is the Euclidean metric.



(a) two-dimensional vector datasets



(b) ten-dimensional vector datasets

Fig. 8. Result of the experiment on artificial vector datasets. For each value of α, an average NMI scores of ten datasets is plotted.

An example of a generated vector dataset and three example graphs obtained by the transformation with different α values is shown in Fig. 7(a). According to the value of α, characteristics of generated graphs approximately vary along a curve shown in Fig. 7(b).

Figure 8 shows the result of the experiments on artificial datasets. We did experiments on the two-dimensional vector datasets and ten-dimensional ones with $n = 32$ and $k = 4$. In order to minimize the effect of random generation of datasets, we generated ten datasets and plotted average NMI scores of them. We can see that our algorithm got higher score in the left side of the figure, where the generated graphs are very noisy because of many edges, and the existing algorithms failed to do clustering on the graphs. This observation indicates the advantage of our algorithm against the noisy graphs. On the other hand, our algorithm failed to in the right side of the figure though the existing ones successfully did clustering. This is because, according to Fig. 7(b), the difference of edge densities within the same cluster and between different clusters is too small for our algorithm to detect.

Figure 9 shows the result of the experiment on the Iris dataset. Although the Iris dataset consists of 150 samples of three species of Iris flowers, as a clustering task, it is said to be difficult for clustering algorithms to distinguish two species of them. So we gave $k = 2$ (2-clustering problem) to our algorithm and GNnum.

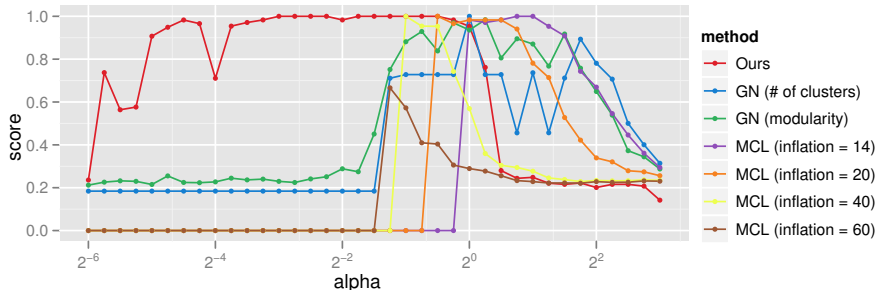


Fig. 9. Result of the experiment on the Iris dataset as 2-clustering problem. For each value of α, an average NMI scores of three experiments on randomly generated graph using the value is plotted.

The result of the existing algorithms is similar to the previous one, and they did clustering well only on a small range of values of α which generates cleaner graphs. By contrast, our algorithm successfully did clustering in a very wide range of values of α, where the scores of the existing ones are significantly low.

5 Conclusion

This paper proposes an algorithm for graph clustering problems focusing on a macroscopic structure of a given graph, which is a graph of clusters and describes the coarse structure of the graph through a view (clustering). Leveraging macroscopic structures, the problems are formulated as an optimization problems to find the best view that minimizes the difference between an ideal MS and an MS obtained by applying a temporal view. We conducted experiments and the results of them show that our algorithm outperforms existing algorithms on noisy graphs.

As future work, it can be considered to extend our algorithm to *graph labeling problems*. Graph labeling is assignment of labels to vertices of a given graph, the labels have different meanings, and thus they should be distinguished unlike graph clustering problems. We think that an appropriate definition of the ideal MS enables us to apply our idea, to utilize an MS of a graph, on a wide range of the problems. In fact, our algorithm can be seen as a graph labeling algorithm which labels vertices with the names of cluster like “cluster1” and “cluster2”. So we will investigate further application of our idea.

References

1. Ammann, P., Wijesekera, D., Kaushik, S.: Scalable, graph-based network vulnerability analysis. In: Proceedings of the 9th ACM Conference on Computer and Communications Security. pp. 217–224. ACM (2002)
2. Angelova, R., Weikum, G.: Graph-based text classification: learn from your neighbors. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 485–492. ACM (2006)

3. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Computer networks and ISDN systems 30(1-7), 107–117 (1998)
4. Csárdi, G., Nepusz, T.: The igraph software package for complex network research. InterJournal Complex Systems 1695 (2006), http://cneurocvs.rmki.kfki.hu/igraph
5. van Dongen, S.: Graph clustering by flow simulation. Ph.D. thesis, University of Utrecht (May 2000)
6. Dorow, B., Widdows, D., Ling, K., Eckmann, J.P., Sergi, D., Moses, E.: Using curvature and markov clustering in graphs for lexical acquisition and word sense discrimination. Arxiv preprint cond-mat/0403693 (2004)
7. Dutt, S., Deng, W.: Cluster-aware iterative improvement techniques for partitioning large VLSI circuits. ACM Transactions on Design Automation of Electronic Systems (TODAES) 7(1), 91–121 (2002)
8. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annals of Human Genetics 7(2), 179–188 (1936), http://dx.doi.org/10.1111/j.1469-1809.1936.tb02137.x
9. Gerhardt, G.J.L., Lemke, N., Corso, G.: Network clustering coefficient approach to DNA sequence analysis. Chaos, Solitons & Fractals 28(4), 1037–1045 (2006)
10. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proceedings of the National Academy of Sciences of the United States of America 99(12), 7821–7826 (2002)
11. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. Journal of the ACM (JACM) 46(5), 604–632 (1999)
12. Kleinberg, J.M., Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A.S.: The web as a graph: Measurements, models, and methods. In: Proceedings of the 5th annual international conference on Computing and combinatorics. pp. 1–17. Springer-Verlag (1999)
13. Liu, D.C., Nocedal, J.: On the limited memory bfgs method for large scale optimization. Mathematical Programming 45(1), 503–528 (1989)
14. Liu, X., Bollen, J., Nelson, M.L., Van de Sompel, H.: Co-authorship networks in the digital library research community. Information Processing & Management 41(6), 1462–1480 (2005)
15. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical review E 69(2), 26113 (2004)
16. Otte, E., Rousseau, R.: Social network analysis: a powerful strategy, also for the information sciences. Journal of Information Science 28(6), 441 (2002)
17. Rual, J.F., Venkatesan, K., Hao, T., Hirozane-Kishikawa, T., Dricot, A., Li, N., Berriz, G.F., Gibbons, F.D., Dreze, M., Ayivi-Guedehoussou, N., et al.: Towards a proteome-scale map of the human protein–protein interaction network. Nature 437(7062), 1173–1178 (2005)
18. Sharon, E., Brandt, A., Basri, R.: Fast multiscale image segmentation. In: Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on. vol. 1, pp. 70–77. IEEE (2000)
19. Strehl, A., Strehl, E., Ghosh, J., Mooney, R.: Impact of similarity measures on Web-page clustering. In: In Workshop on Artificial Intelligence for Web Search (AAAI 2000 (2000)
20. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming 106(1), 25–57 (2006)