A Variational Bayesian Framework for Clustering with Multiple Graphs

Motoki Shiga and Hiroshi Mamitsuka, *Member, IEEE*

Abstract—Mining patterns in graphs has become an important issue in real applications, such as bioinformatics and web mining. We address a graph clustering problem where a *cluster* is a set of densely connected nodes, under a practical setting that 1) the input is multiple graphs which share a set of nodes but have different edges and 2) a true cluster cannot be found in all given graphs. For this problem, we propose a probabilistic generative model and a robust learning scheme based on variational Bayesian estimation. A key feature of our probabilistic framework is that not only nodes but also given graphs can be clustered at the same time, allowing our model to capture clusters found in only part of all given graphs. We empirically evaluated the effectiveness of the proposed framework on not only a variety of synthetic graphs but also real gene networks, demonstrating that our proposed approach can improve the clustering performance of competing methods in both synthetic and real data.

Index Terms—Clustering, graphs, statistical machine learning, variational Bayesian learning, localized clusters.

1 Introduction

Mining patterns in graphs or networks has become an important issue in a variety of applications, such as biology, chemistry, social sciences, web and text mining. In this paper, we address a graph clustering problem, i.e. nodes being clustered over a graph, where an edge between two nodes indicates a similarity between the two nodes [1], [2]. Here a *cluster* is a set of nodes, which are densely connected with each other in one graph. Our setting for this problem is that we have multiple graphs which share the same set of nodes but have different edges from each other, meaning that clusters can be different between each pair of given graphs. We then focus on a *localized cluster*, which is commonly found in 1 to $M-1$ graphs among given M graphs but not in all M graphs. Fig. 1 (a) to (c) show three graphs which share the same nodes but have different edges from each other. Each of these three graphs has an example of localized clusters. For example, the upper-left part in (a) has nodes (colored green) tightly connected with each other which form a cluster. This cluster appears in only (a) but not in both (b) and (c), meaning that it can be a localized cluster. Similarly, red nodes form a localized cluster in (b) and blue nodes are a localized cluster in (c). Note that each of the three localized clusters cannot be found in all (a)-(c).

We emphasize that our problem setting can be found in real applications. For example, in molecular biology, gene similarity can be represented in a variety of manners, such as co-expression, sequence similarity and protein-protein interaction [3]. They can be networks which share the same set of nodes (or genes), where each network is generated by a different experimental measurement over genes. Clustering genes is useful in many practical issues, such as gene annotation. It is now recognized that clusters become more consistent if they are generated with a larger number of networks [4]. This is because experiments in molecular biology are rather unstable, possibly causing unreliable results because of measurement errors. For example, a true cluster might be hidden by noise, resulting in that a cluster cannot be necessarily captured by all measurements, naturally bringing the concept of localized clusters [5].

Fig. 1 (d) shows a graph which has all edges of Fig. 1 (a)-(c). We call this graph an *integrated graph* in which the weight of an edge is the sum over all weights of the corresponding edges in given graphs. A baseline method of the graph clustering problem with multiple graphs would be to first generate an integrated graph from multiple graphs and then run a graph clustering algorithm, such as spectral clustering [6] over the integrated graph. However, summing weights over graphs means dealing with given graphs equally, resulting in that localized clusters may be lost in the integrated graph. For example, in Fig. 1, localized clusters in (a)-(c) can no longer be found in (d). Thus we need a sophisticated clustering method which can capture and keep localized clusters from multiple graphs.

Our approach for this issue is based on a generative model and a robust parameter estimation algorithm. If only a single graph is given, our problem becomes a standard setting of graph clustering, for which we can

• *M. Shiga and H. Mamitsuka are with the Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji 611-0011, Japan. Also they are with Institute for Bioinformatics Research and Development (BIRD), Japan Science and Technology (JST).*
E-mail: {shiga, mami}@kuicr.kyoto-u.ac.jp

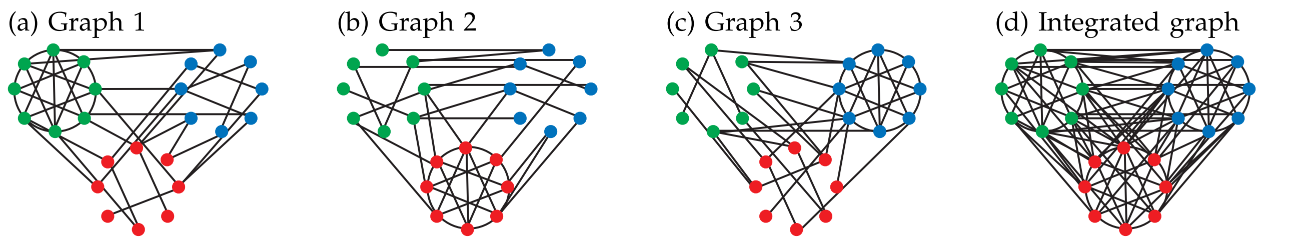

Fig. 1. Examples of (a)-(c) localized clusters and (d) the integrated graph generated from (a)-(c) by simply summing up their node weights.

consider a simple probabilistic model, which we call PMSG (Probabilistic Model for a Single Graph), with multinomial distributions for generating edges. This model is equivalent to several existing probabilistic models in the framework of statistical machine learning. For our problem setting, we extend PMSG to a model, which we call PMMG (Probabilistic Model for Multiple Graphs), in which a multinomial distribution over graphs is added, allowing to consider localized clusters. Our robust parameter learning algorithm of PMMG, based upon variational Bayesian learning, estimates the distribution of each parameter [7], [8]. We emphasize that to the best of our knowledge, there have been no approaches from statistical machine learning for our problem of clustering with multiple graphs containing localized clusters. The variational Bayesian learning is a standard learning process in the current literature of probabilistic/statistical model learning and is more robust against noise than point estimation methods such as the EM (Expectation-Maximization) algorithm, resulting in that overfitting the given data can be avoided more easily [7], [9], [8].

We empirically evaluated our method, comparing with other approaches. Our experiments started with synthetic graphs, revealing two important findings: First, the performance of PMMG was improved by using a larger number of graphs for all cases, being pronounced more when containing localized clusters more. Second, PMMG significantly outperformed other competing graph clustering methods for all cases in our experiments. This performance advantage was further clearly confirmed by the experiment of using real gene networks. Finally we analyzed clustering results on real gene networks and verified that the performance advantage of PMMG was achieved by the consideration on localized clusters.

This paper is organized as follows: In Section 2, we review related work on graph clustering, with a particular emphasis upon graph clustering by probabilistic models. In Section 3, we describe the proposed generative model and an efficient parameter estimation algorithm based on variational Bayesian learning, being followed by the explanation on related probabilistic models and the time and space complexities of the proposed learning algorithm, comparing with those of related probabilistic models. In Section 4, we extensively evaluated the performance of the proposed approach by using both synthetic and real datasets, comparing with other competing methods, mainly described in Section 2. Finally, we review the advantage of the proposed method once again in Section 5

2 Related Work

In general the input of graph clustering is one graph only. Graph clustering methods, such as probabilistic models (including PMSG) [10], [11] and spectral clustering [6], can be applied to our problem after generating an integrated graph from given multiple graphs. In the literature of graph clustering by probabilistic models, PMSG is equivalent in model structure (and complexity) to those in [12], while the most typical probabilistic model in graph clustering is stochastic block model (SBM) [13]. SBM can be trained by many approaches, including an EM algorithm [14] and Bayesian estimation by Gibbs sampling [14]. There are two recent extensions of SBM: mixed-membership SBM [15] and constrained SBM [11], which we call MSBM (mixed Membership Stochastic Block Model) and CSBM (Constrained Stochastic Block Model), respectively. Their proposed training algorithms are both based on variational Bayesian learning, and then we will explain the model structures of MSBM and CSBM in Section 3.4 and their time and space complexities in Section 3.5.

To the best of our knowledge, there are no probabilistic-model based methods for graph clustering over multiple graphs. However, Sinkkonen *et al.* treated a similar situation, in which multiple node sets are given and there can be edges both between and within node sets [16]. In our problem setting, however, their model has to be equivalent to the integrated graph approach in which an integrated graph is first generated from given graphs and then PMSG (or [12]) is applied. In other words, under our setting, [16] will have the same result as that of PMSG.

On the other hand, an extension of graph cut in spectral clustering for multiple graphs exists [17]. This was also used as a competing method in our experiments, and we call this approach MSC, standing for *Multiple graphs in Spectral Clustering*. The aim of MSC is to find consensus clusters over multiple graphs in terms of multiview learning by using the degree of each node over graphs.

3 Methodology

3.1 Notation and Preliminaries

Let $G = (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of edges. Let e be an edge and v a node. We here assume that G is an undirected graph (Note that it is easy to extend our work to directed graphs). Let $\boldsymbol{D}$ be input data, i.e. multiple graphs $G_1, \ldots, G_M$, which share N $(= |\mathcal{V}|)$ nodes[1], satisfying that $G_m = (\mathcal{V}, \mathcal{E}_m)$. Let E $(= |\mathcal{E}|)$ be the number of edges of a single graph, and $E = \max_m |\mathcal{E}_m|$ for m graphs. We note that a graph is equivalent to a weight matrix in which each element indicates a weight assigned to an edge, which connects corresponding two nodes in a given graph. Thus input data $\boldsymbol{D}$ can be given by weight matrices $\{\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(M)}\}$, where $\boldsymbol{W}^{(m)}$ is a symmetric matrix. Let $W_{ij}^{(m)}$ $(\in \mathbb{N})$ be the (i,j)-element of $\boldsymbol{W}^{(m)}$, and $W_{ij}^{(m)} > 0$ if there is an edge for nodes v_i and v_j; otherwise $W_{ij}^{(m)} = 0$. In fact, in the following sections, we will use weight matrices more than graphs. If only one graph is given, we just write it by $\boldsymbol{W}$. Let K be the number of clusters. Let $\boldsymbol{z}$ (or z) be hidden variables, corresponding to cluster labels. Let C_i $(i = 1, \ldots, N)$ be a cluster taken by each node.

3.2 PMSG: Probabilistic Model for a Single Graph

3.2.1 Probabilistic Structure

The input of PMSG is one graph only, i.e. $M = 1$. PMSG assumes that generating an edge is the same event as that two nodes are co-occurring from a multinomial distribution with a cluster label. The probability that two nodes of an edge e are v_i and v_j can be then modeled in a manner of finite mixture models as follows:

$$\begin{aligned} p(e=(i,j)) & \\ &= \sum_{k=1}^{K} p(z=k)p(e=(i,j)|z=k) \\ &= \sum_{k=1}^{K} p(z=k)p(v=i|z=k)p(v=j|z=k) \\ &= \sum_{k=1}^{K} \pi_k r_{ik} r_{jk}, \end{aligned}$$

where $\pi_k = p(z = k)$ and $r_{ik} = p(v = i|z = k)$, and $\sum_{k=1}^{K} \pi_k = 1$ and $\sum_{i=1}^{N} r_{ik} = 1$. For simplicity we hereafter write $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_K\}$ and $\boldsymbol{r}_{\cdot k} = \{r_{1k}, \ldots, r_{Nk}\}$. Let $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{r}\}$ for PMSG. We note that our modeling of probability $p(e = (i,j))$ is not

1. If given graphs do not share a node set, we can first take a union of nodes in given graphs to use the union as the shared node set.

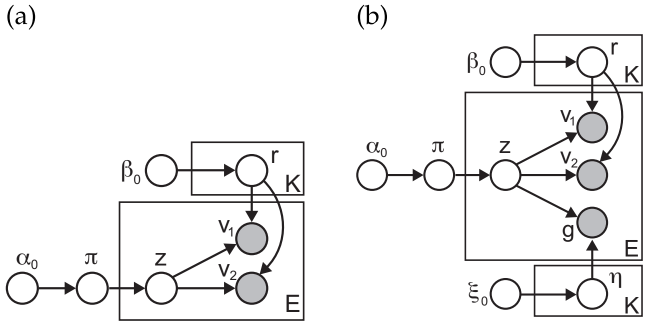

Fig. 2. Graphical model representation of (a) PMSG and (b) PMMG.

Bernoulli trials. For latent variable z, we hereafter write z_{ij}, which is assigned to edge e_{ij} between nodes v_i and v_j.

We assume the Dirichlet distribution for the prior of $\boldsymbol{\theta}$ as follows:

$$\begin{aligned} p(\boldsymbol{\pi}|\alpha_0) &= \frac{\Gamma(K\alpha_0)}{\Gamma(\alpha_0)^K} \prod_{k=1}^{K} \pi_k^{\alpha_0 - 1} \\ p(\boldsymbol{r}_{\cdot k}|\beta_0) &= \frac{\Gamma(N\beta_0)}{\Gamma(\beta_0)^N} \prod_{i=1}^{N} r_{ik}^{\beta_0 - 1} \ (k = 1, \ldots, K), \end{aligned}$$

where α_0 and β_0 are hyperparameters and Gamma function $\Gamma(\cdot)$ can be defined by $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$. The joint probability of $\boldsymbol{W}$ and $\boldsymbol{z}$ can be given by:

$$p(\boldsymbol{W}, \boldsymbol{z}|\boldsymbol{\theta}) = \prod_{k=1}^{K} \prod_{(i,j)\in\mathcal{E}} (\pi_k r_{ik} r_{jk})^{\delta(z_{ij},k) W_{ij}},$$

where $\delta(z_{ij}, k)$ is one if edge e_{ij} is in cluster k; otherwise zero. Thus we can write the joint probability of $\boldsymbol{W}$, $\boldsymbol{z}$ and $\boldsymbol{\theta}$, defined by PMSG as follows:

$$\begin{aligned} p(\boldsymbol{W}, \boldsymbol{z}, \boldsymbol{\theta}) &= p(\boldsymbol{W}, \boldsymbol{z}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) \\ &= p(\boldsymbol{W}, \boldsymbol{z}|\boldsymbol{\theta}) p(\boldsymbol{\pi}|\alpha_0) \prod_{k=1}^{K} p(\boldsymbol{r}_{\cdot k}|\beta_0). \end{aligned}$$

Here we can explain the graph generation by PMSG in the following manner:

```
1) Draw $K$-dimensional vector $\boldsymbol{\pi}$ from the Dirichlet distribution with hyperparameter $\alpha_0$.
2) For $k = 1, \ldots, K$
    a) Draw $N$-dimensional vector $\boldsymbol{r}_{\cdot k}$ from the Dirichlet distribution with hyperparameter $\beta_0$.
3) Iterate the following two steps $E$ times
    a) Draw cluster label $k$ from the multinomial distribution with parameter $\boldsymbol{\pi}$.
    b) Draw two nodes from the multinomial distribution with parameter $\boldsymbol{r}_{\cdot,k}$
```

Fig. 2 (a) shows a graphical model representation of PMSG.

3.2.2 Learning Algorithm

We apply variational Bayesian (VB) estimation to PMSG, making us assuming that posterior probability distributions are independent of each other as follows:

$$q(\boldsymbol{\theta}, \boldsymbol{z}) = q(\boldsymbol{\pi}) \prod_{k=1}^{K} q(\boldsymbol{r}_{\cdot k}) \prod_{(i,j)\in\mathcal{E}} q(z_{ij}),$$

where $q(\cdot)$ means a VB posterior distribution, for which we show the derivation below.

The VB posterior distribution is optimized by maximizing the lower bound of the marginalized likelihood $L[q]$ as follows[2]:

$$\begin{aligned} L[q] &= \sum_{\boldsymbol{z}} \int q(\boldsymbol{\theta}, \boldsymbol{z}) \log \frac{p(\boldsymbol{W}, \boldsymbol{\theta}, \boldsymbol{z})}{q(\boldsymbol{\theta}, \boldsymbol{z})} d\boldsymbol{\theta} \qquad (1)\\ &\leq \log \sum_{\boldsymbol{z}} \int p(\boldsymbol{W}, \boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta}. \end{aligned}$$

Here we write expectations to be computed, as follows for simplicity:

$$\begin{aligned} \bar{z}_{ijk} &= q(z_{ij} = k), \\ \bar{N}_k &= \sum_{(i,j)\in\mathcal{E}} \bar{z}_{ijk} W_{ij}, \qquad (2)\\ \bar{N}_{ik} &= \sum_{j|W_{ij}>0} \bar{z}_{ijk} W_{ij}. \qquad (3) \end{aligned}$$

By the variation of $L[q]$ with respect to $q(\boldsymbol{\pi})$, we can obtain the VB posterior distribution of $\boldsymbol{\pi}$ into the following:

$$\begin{aligned} q(\boldsymbol{\pi}) &= \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \pi_k^{\alpha_k - 1}, \quad \text{where} \\ \alpha_k &= \bar{N}_k + \alpha_0. \qquad (4) \end{aligned}$$

Similarly, we can obtain the VB posterior distribution of $\boldsymbol{r}_{\cdot k}$ as follows:

$$\begin{aligned} q(\boldsymbol{r}_{\cdot k}) &= \frac{\Gamma(\sum_{i=1}^{N} \beta_{ik})}{\prod_{i=1}^{N} \Gamma(\beta_{ik})} \prod_{i=1}^{N} r_{ik}^{\beta_{ik} - 1}, \quad \text{where} \\ \beta_{ik} &= 2\bar{N}_{ik} + \beta_0. \qquad (5) \end{aligned}$$

On the other hand, the VB posterior distribution of z_{ij} can be given by:

$$q(z_{ij} = k) = \frac{\exp(\gamma_{ijk})}{\sum_{k=1}^{K} \exp(\gamma_{ijk})}, \qquad (6)$$

where

$$\begin{aligned} \gamma_{ijk} &= \Psi(\alpha_k) - \Psi\Big(\sum_{k=1}^{K} \alpha_k\Big) \\ &\quad + \Psi(\beta_{ik}) + \Psi(\beta_{jk}) - 2\Psi\Big(\sum_{i=1}^{N} \beta_{ik}\Big) \qquad (7) \end{aligned}$$

2. Maximizing $L[q]$ is equivalent to minimizing the KL (Kullback-Leibler) divergence between the true posterior distribution and the variational posterior distribution [8].

```
Input : $\boldsymbol{W}$, $K$, $\alpha_0$, $\beta_0$
Output : $q(\boldsymbol{\pi})$, $q(\boldsymbol{r}_{\cdot k})$, $q(\boldsymbol{z}|v_i)$

1: Initialize $\bar{N}_k$ and $\bar{N}_{ik}$ by random numbers.
2: repeat
3:    VB E-step: Compute the posterior distribution of latent variables, $q(z_{ij} = k)$.
4:    VB M-step: Compute $\alpha_k$ and $\beta_{ik}$.
5: until Some convergence criterion is satisfied.
6: Compute $L[q]$ and $q(\boldsymbol{z}|v_i)$.
```

Fig. 3. Pseudocode of variational Bayesian estimation of PMSG.

and the digamma function $\Psi(x) = \frac{d \log(\Gamma(x))}{dx}$.

Fig. 3 shows a pseudocode of the VB learning algorithm of PMSG. This algorithm repeats the VB E- and VB M-steps alternately, where the VB posterior distribution $q(z_{ij} = k)$ is computed by Eq. (6) in the VB E-step, and α_k and β_{ik} are computed by Eqs. (2)-(5) in the VB M-step. After the convergence of the alternate iterations of these two steps, we compute the lower bound of marginalized likelihood $L[q]$, which is given by:

$$\begin{aligned} L[q] &= \log\Gamma(K\alpha_0) - K\log\Gamma(\alpha_0) + K\log\Gamma(N\beta_0) \\ &\quad - KN\log\Gamma(\beta_0) - \sum_{k=1}^{K} \sum_{(i,j)\in\mathcal{E}} \bar{z}_{ijk} \log \bar{z}_{ijk} \\ &\quad - \log\Gamma\left(\sum_{k=1}^{K} \alpha_k\right) + \sum_{k=1}^{K} \log\Gamma(\alpha_k) \\ &\quad - \sum_{k=1}^{K} \log\Gamma\left(\sum_{i=1}^{N} \beta_{ik}\right) + \sum_{k=1}^{K}\sum_{i=1}^{N} \log\Gamma(\beta_{ik}) \end{aligned}$$

We note that Fig. 3 shows a single run of the algorithm, in the sense that practically this algorithm is run a large number of times with random different initial values and the final output is given by the run which can give the maximum $L[q]$. Finally, to assign a cluster to each node, the posterior distribution of the cluster label of a node can be computed as follows:

$$\begin{aligned} q(z = k|v = i) &= \frac{\langle p(z = k)\rangle_{q(\boldsymbol{\pi})} \langle q(v = i|z = k)\rangle_{q(\boldsymbol{r})}}{\sum_{k'=1}^{K} \langle p(z = k')\rangle_{q(\boldsymbol{\pi})} \langle q(v = i|z = k')\rangle_{q(\boldsymbol{r})}} \\ &= \frac{\bar{\pi}_k \bar{r}_{ik}}{\sum_{k'=1}^{K} \bar{\pi}_{k'} \bar{r}_{ik'}}, \qquad (8) \end{aligned}$$

where,

$$\begin{aligned} \bar{\pi}_k &= \frac{\alpha_k}{\sum_{k'=1}^{K} \alpha_{k'}}, \\ \bar{r}_{ik} &= \frac{\beta_{ik}}{\sum_{i'=1}^{N} \beta_{i'k}}. \end{aligned}$$

We can then assign a cluster label C_i to node v_i in the following:

$$C_i = \arg\max_{\{1,\dots,K\}} q(z = k|v = i). \qquad (9)$$

3.3 PMMG: Probabilistic Model for Multiple Graphs

3.3.1 Probabilistic Structure

Suppose that we have M graphs, we assume that two nodes and a graph are co-occurring from a multinomial distribution with a cluster label. The probability that two nodes of edge e are v_i and v_j of graph G_m can be then modeled in a manner of finite mixture models, given in the following:

$$\begin{aligned}
&p(g=m, e=(i,j))\\
=&\sum_{k=1}^{K} p(z=k)p(g=m|z=k)p(e=(i,j)|z=k)\\
=&\sum_{k=1}^{K} p(z=k)p(g=m|z=k)p(v=i|z=k)p(v=j|z=k)\\
=&\sum_{k=1}^{K} \pi_k \eta_{mk} r_{ik} r_{jk},
\end{aligned}$$

where $\eta_{mk} = p(g=m|z=k)$ and hereafter we write $\boldsymbol{\eta}_{\cdot k} = \{\eta_{1k} \ldots \eta_{Mk}\}$. $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{r}, \boldsymbol{\eta}\}$ for PMMG. We again note that our modeling of probability $p(g = m, e = (i,j))$ is not Bernoulli trials. We hereafter write cluster label $z_{ij}^{(m)}$, which is assigned to edge e_{ij} between nodes v_i and v_j in graph G_m.

The joint probability of $\boldsymbol{D}$ and $\boldsymbol{z}$ can be then given by:

$$p(\boldsymbol{D}, \boldsymbol{z}|\boldsymbol{\theta}) = \prod_{k=1}^{K}\prod_{m=1}^{M}\prod_{(i,j)\in\mathcal{E}_m} \{\pi_k \eta_{mk} r_{ik} r_{jk}\}^{\delta(z_{ij}^{(m)},k)}$$

where $\delta(z_{ij}^{(m)}, k)$ is one if edge e_{ij} of the m-th graph is in the k-th cluster; otherwise zero.

These equations show that PMMG has parameter $\boldsymbol{\eta}_{\cdot k}$, which was not in PMSG and allows PMMG to capture localized clusters in multiple graphs. We can then assume the Dirichlet distribution for the prior of $\boldsymbol{\eta}_{\cdot k}$ as follows:

$$p(\boldsymbol{\eta}_{\cdot k}|\zeta_0) = \frac{\Gamma(M\zeta_0)}{\Gamma(\zeta_0)^M}\prod_{m=1}^{M}\eta_{mk}^{\zeta_0-1},$$

where ζ_0 is a hyperparameter.

Finally PMMG defines the joint probability of $\boldsymbol{D}$, $\boldsymbol{z}$ and $\boldsymbol{\theta}$ in the following:

$$\begin{aligned}
p(\boldsymbol{D}, \boldsymbol{z}, \boldsymbol{\theta}) &= p(\boldsymbol{D}, \boldsymbol{z}|\boldsymbol{\theta})p(\boldsymbol{\theta})\\
&= p(\boldsymbol{D}, \boldsymbol{z}|\boldsymbol{\theta})p(\boldsymbol{\pi}|\alpha_0)\prod_{k=1}^{K} p(\boldsymbol{r}_{\cdot k}|\beta_0)p(\boldsymbol{\eta}_{\cdot k}|\zeta_0)
\end{aligned}$$

We can here explain that the generation of multiple graphs by PMMG in the following way:

---

1) Draw K-dimensional vector $\boldsymbol{\pi}$ from the Dirichlet distribution with hyperparameter α_0.
2) **For** $k = 1, \ldots, K$
    a) Draw N-dimensional vector $\boldsymbol{r}_{\cdot k}$ from the Dirichlet distribution with hyperparameter β_0.
    b) Draw M-dimensional vector $\boldsymbol{\eta}_{\cdot k}$ from the Dirichlet distribution with hyperparameter ζ_0.
3) Iterate the following three steps $\sum_m |\mathcal{E}_m|$ times
    a) Draw cluster label k from the multinomial distribution with parameter $\boldsymbol{\pi}$
    b) Draw a graph from the multinomial distribution with parameter $\boldsymbol{\eta}_{\cdot,k}$
    c) Draw two nodes from the multinomial distribution with parameter $\boldsymbol{r}_{\cdot,k}$

Fig. 2 (b) shows a graphical model representation of PMMG.

3.3.2 Learning Algorithm

Similar to the derivation for PMSG, we use VB estimation by which VB posterior distributions $q(\cdot)$ are given, being independent of each other, as follows:

$$q(\boldsymbol{\theta}, \boldsymbol{z}) = q(\boldsymbol{\pi})\prod_{k=1}^{K} q(\boldsymbol{\eta}_{\cdot k})q(\boldsymbol{r}_{\cdot k})\prod_{m=1}^{M}\prod_{(i,j)\in\mathcal{E}_m} q(z_{ij}^{(m)}). \quad (10)$$

Again the VB posterior distributions can be obtained by maximizing the lower bound of the marginalized likelihood $L[q]$ of Eq.(1), after substituting Eq.(10) into Eq.(1).

Now we can write expectations to be computed for PMMG, in the following:

$$\begin{aligned}
\bar{z}_{ijk}^{(m)} &= q(z_{ij}^{(m)} = k),\\
\bar{N}_k &= \sum_{m=1}^{M}\sum_{(i,j)\in\mathcal{E}_m} \bar{z}_{ijk}^{(m)} W_{ij}^{(m)}, && (11)\\
\bar{N}_k^{(m)} &= \sum_{(i,j)\in\mathcal{E}_m} \bar{z}_{ijk}^{(m)} W_{ij}^{(m)}, && (12)\\
\bar{N}_{ik} &= \sum_{m=1}^{M}\sum_{j|W_{ij}^{(m)}>0} \bar{z}_{ijk}^{(m)} W_{ij}^{(m)}. && (13)
\end{aligned}$$

By the variation with respect to each VB posterior distribution, we can first obtain the VB posterior distribution of $\boldsymbol{\pi}$ as follows:

$$\begin{aligned}
q(\boldsymbol{\pi}) &= \frac{\Gamma(\sum_{k=1}^{K}\alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\pi_k^{\alpha_k-1}, \quad \text{where}\\
\alpha_k &= \bar{N}_k + \alpha_0. && (14)
\end{aligned}$$

Next the VB posterior distribution of $\boldsymbol{\eta}_{\cdot k}$ can be given as follows:

$$\begin{aligned}
q(\boldsymbol{\eta}_{\cdot k}) &= \frac{\Gamma(\sum_{m=1}^{M}\zeta_{mk})}{\prod_{m=1}^{M}\Gamma(\zeta_{mk})}\prod_{m=1}^{M}\eta_{mk}^{\zeta_{mk}-1}, \quad \text{where}\\
\zeta_{mk} &= \bar{N}_k^{(m)} + \zeta_0. && (15)
\end{aligned}$$

The VB posterior distribution of $\boldsymbol{r}_{\cdot k}$ can be given as follows:

$$\begin{aligned} q(\boldsymbol{r}_{\cdot k}) &= \frac{\Gamma(\sum_{i=1}^{N}\beta_{ik})}{\prod_{i=1}^{N}\Gamma(\beta_{ik})}\prod_{i=1}^{N} r_{ik}^{\beta_{ik}-1}, \quad \text{where} \\ \beta_{ik} &= 2\bar{N}_{ik}+\beta_0. \end{aligned} \tag{16}$$

In contrast, the VB posterior distribution of $z_{ij}^{(m)}$ can be given as follows:

$$q(z_{ij}^{(m)}=k) = \frac{\exp(\gamma_{ijk}^{(m)})}{\sum_{k=1}^{K}\exp(\gamma_{ijk}^{(m)})}, \tag{17}$$

where

$$\begin{aligned} \gamma_{ijk}^{(m)} &= \Psi(\alpha_k) - \Psi\Big(\sum_{k=1}^{K}\alpha_k\Big) \\ &+\Psi(\zeta_{mk}) - \Psi\Big(\sum_{m=1}^{M}\zeta_{mk}\Big) \\ &+\Psi(\beta_{ik}) + \Psi(\beta_{jk}) - 2\Psi\Big(\sum_{i=1}^{N}\beta_{ik}\Big). \end{aligned}$$

Fig. 4 shows a pseudocode of the VB learning algorithm of PMMG. Similar to the learning algorithm of PMSG, this algorithm iterates the VB E- and VB M-steps alternately until the convergence. That is, $q(z_{ij}^{(m)}=k)$ is computed according to Eq. (17) in the VB E-step and α_k, η_{mk} and β_{ik} are computed by using Eqs. (11)-(16) in the VB M-step. After the convergence of the iterative process, we then need compute $L[q]$, which is given as follows:

$$\begin{aligned} L[q] &= \log\Gamma(K\alpha_0) - K\log\Gamma(\alpha_0) + K\log\Gamma(N\beta_0) \\ &\quad -KN\log\Gamma(\beta_0) + K\log\Gamma(M\zeta_0) \\ &\quad -KM\log\Gamma(\zeta_0) - \sum_{k=1}^{K}\sum_{(i,j)\in\mathcal{E}_m}\bar{z}_{ij,k}^{(m)}\log\bar{z}_{ij,k}^{(m)} \\ &\quad -\log\Gamma\left(\sum_{k=1}^{K}\alpha_k\right) + \sum_{k=1}^{K}\log\Gamma(\alpha_k) \\ &\quad -\sum_{k=1}^{K}\log\Gamma\left(\sum_{i=1}^{N}\beta_{ik}\right) + \sum_{k=1}^{K}\sum_{i=1}^{N}\log\Gamma(\beta_{ik}) \\ &\quad -\sum_{k=1}^{K}\log\Gamma\left(\sum_{m=1}^{M}\zeta_{mk}\right) + \sum_{k=1}^{K}\sum_{m=1}^{M}\log\Gamma(\zeta_{mk}) \end{aligned}$$

In practice, we repeat running the algorithm in Fig. 4 many times with random different initial values and keep the output which gives the maximum $L[q]$. This final output is used to obtain $q(z=k|v_i)$ and cluster assignment C_i in the same manner as Eqs. (8)-(9) of PMSG.

3.4 Related Probabilistic Models

We here explain two probabilistic models for graph clustering: CSBM and MSBM, which are both extended from the so-called stochastic block model (SBM) [13]. These two models are related with PMSG, since the input is only one graph and their parameters can be estimated by variational Bayes learning. On the other hand, CSBM and MSBM are different from PMSG, since binomial distributions (or Bernoulli trials) are assumed for generating all node pairs, while multinomial distributions are assumed for generating (nodes of) edges in PMSG, by which PMSG can focus on connected nodes only.

```
Input : $\boldsymbol{D}$, $K$, $\alpha_0$, $\beta_0$, $\zeta_0$
Output : $q(\boldsymbol{\pi})$, $q(\boldsymbol{\eta}_{\cdot k})$, $q(\boldsymbol{r}_{\cdot k})$, $q(\boldsymbol{z}|v_i)$

1: Initialize $\bar{N}_k$, $\bar{N}_k^{(m)}$ and $\bar{N}_{ik}$ by random numbers.
2: repeat
3:    VB E-step: Compute the posterior distribution
      of latent variables, $q(z_{ij}^{(m)}=k)$.
4:    VB M-step: Compute $\alpha_k$, $\eta_{mk}$ and $\beta_{ik}$.
5: until Some convergence criterion is satisfied.
6: Compute $L[q]$ and $q(\boldsymbol{z}|v_i)$.
```

Fig. 4. Pseudocode of variational Bayesian estimation of the parameters of PMMG.

3.4.1 *Constrained Stochastic Block Model [11]*

CSBM assumes a multinomial distribution for assigning a cluster to each node, meaning that we write z_i for the cluster label of a node v_i. This model further assumes binomial distributions over node pairs with two different weights for "intra-cluster edges" and "inter-cluster edges", which are θ_{in} and $\theta_{between}$, respectively. The graph generation by CSBM can be written in the following manner:

1) Draw θ_{in} and $\theta_{between}$ from a beta distribution.
2) Draw K-dimensional vector $\boldsymbol{\pi}$ from a Dirichlet distribution.
3) **For** each node v_n $(n=1,\ldots,N)$
   a) Draw cluster label k $(=z_n)$ from the multinomial distribution with parameter $\boldsymbol{\pi}$.
4) Repeat the following step over all possible node pairs v_i and v_j $(i=1,\ldots,N;\ j>i)$.
   a) If $z_i=z_j$, draw a value from the binomial distribution with parameter θ_{in}, and connect nodes v_i and v_j with an edge if the value is one.
   b) Otherwise, draw a value from the binomial distribution with parameter $\theta_{between}$, and connect nodes v_i and v_j with an edge if the value is one.

3.4.2 *Mixed Membership Stochastic Block Model [15]*

MSBM assumes multinomial distributions for cluster labels over nodes of all possible node pairs. We then

TABLE 1
Space and time complexities of PMSG, PMMG, CSBM and MSBM for each update

	Space complexity	Time complexity
PMSG	$O(E \cdot K)$	$O(E \cdot K)$
PMMG	$O(E \cdot K)$	$O(E \cdot K)$
CSBM	$O(N \cdot K)$	$O(E \cdot K)$
MSBM	$O(N^2 \cdot K)$	$O(N^2 \cdot K^2)$

write z_{ij} for node v_i of node pair, v_i and v_j. This model further assumes the binomial distribution with parameter θ_{k_1,k_2} for a pair of nodes with cluster labels k_1 and k_2. We can explain the graph generation by MSBM in the following manner:

```
1) For each node $v_n$ $(n = 1, \ldots, N)$
    a) Draw $K$-dimensional vector $\boldsymbol{\pi}_n$ from a Dirichlet distribution.
2) Repeat the following step over all possible cluster pairs $k_1$ and $k_2$ ($k_1 = 1, \ldots, K$; $k_2 = 1, \ldots K$).
    a) Draw $\theta_{k_1,k_2}$ from a beta distribution.
3) Repeat the following three steps over all possible node pairs $v_i$ and $v_j$ ($i = 1, \ldots, N$; $j > i$).
    a) Draw cluster label $k_1$ $(= z_{ij})$ from the multinomial distribution with $\boldsymbol{\pi}_i$.
    b) Draw cluster label $k_2$ $(= z_{ji})$ from the multinomial distribution with $\boldsymbol{\pi}_j$.
    c) Draw a value from the binomial distribution with $\theta_{k_1,k_2}$ and connect an edge between nodes $v_i$ and $v_j$ if the value is one; otherwise do not connect any edge between them.
```

3.5 Computational Complexities

We first summarize the difference among PMSG, CSBM and MSBM. PMSG assumes multinomial distributions for (nodes of) edges, by which PMSG considers connected node pairs only, while CSBM and MSBM assume binomial distributions for node pairs, by which CSBM and MSBM must consider all node pairs. This difference is pronounced by hidden variables, since PMSG assign cluster labels to edges and CSBM assign labels to nodes while MSBM assign labels to nodes of all possible node pairs. This means that the number of hidden variables is linear to the number of nodes in CSBM and the number of edges (or connected two nodes) in PMSG which are both light in complexity, while it is linear to the square of the number of nodes in MSBM which is very heavy.

Table 1 shows the space and time complexities of PMSG, PMMG, CSBM and MSBM for each iteration of updating parameters. We note that PMMG keeps the same space and time complexities as those of PMSG, which are both $O(E \cdot K)$. Practically given graphs are sparse, which means that PMSG and PMMG are almost equivalent in complexity to CSBM where the space and time complexities are $O(N \cdot K)$ and $O(E \cdot K)$. On the other hand, the complexities of MSBM are remarkably high. The time complexity of MSBM reaches $O(N^2 \cdot K^2)$, which for a sparse graph, might reach the square of $O(E \cdot K)$, i.e. the time complexity of PMSG, PMMG and CSBM. The space complexity of MSBM is $O(N^2 \cdot K)$, corresponding to the number of hidden variables, which is very huge. For example, the number of hidden variables reaches 10^7 for a graph with 1,000 nodes ($|N| = 1,000$) and ten clusters. This makes the complexity of MSBM very high, resulting in that MSBM needs a very large number of trials to find good initial parameter values even for a graph of $|N| = 1,000$. Thus we can see that this high model complexity makes it hard to apply MSBM to practical graphs.

4 Experimental Results

4.1 Experimental Setting

4.1.1 Synthetic Data

We generated synthetic graphs containing localized clusters by using the probabilistic structure of PMMG, since PMMG already allows to represent localized clusters. This means that we used two probabilistic parameters of PMMG, r_{ik} and η_{mk} in the first part of Section 3.4.1., to generate each synthetic graph. More concretely, to generate a synthetic graph, we randomly generated nodes v_i and v_j (edge e_{ij}) according to r_{ik} and η_{mk} and let the value of the corresponding weight in a matrix (or a graph) one, i.e. $W_{ij} = W_{ji} = 1$.

We fixed the number of nodes in each cluster at 50, by simply assigning the same cluster label to every fifty examples (nodes). That is, for each cluster k, we set $z_i = k$ $(50(k-1)+1 \le i \le 50k)$. Note that the input is graphs only and does not contain true cluster labels, which were just used for generating synthetic data and evaluating clustering results.

We generated multiple graphs in the following manner: 1) We first randomly generated M' graphs with totally $(50 \times K)^2 \times 0.1$ edges, i.e. 10% of all possible node pairs (because $50 \times K$ is the number of all nodes). 2) We then chose M out of M' as input.

We then parameterized r_{ik} and η_{mk} by adding two types of perturbations in generating graphs, considering noise (corresponding to inter-cluster edges) and unbalanceness over multiple graphs.

For r_{ik}, we used real-valued parameter R_{out} $(\in [0, \frac{1}{2}])$, corresponding to the ratio of inter-cluster edges to all edges. We have $50 \times K$ examples (nodes). For cluster k, the number of nodes can be the sum of $50 \times (1 - R_{out})$ (i.e. the number of nodes for intra-cluster edges) and $50 \times (K-1) \times R_{out}$ (i.e. the number of nodes for inter-cluster edges). Then r_{ik} is set by:

$$r_{ik} = \begin{cases} \frac{1}{C}(1 - R_{out}) & \text{if } z_i = k \\ \frac{1}{C} R_{out} & \text{otherwise,} \end{cases}$$

where $C = 50 \times (1 - R_{out}) + 50 \times (K - 1) \times R_{out}$. We note that $\sum_{i=1}^{N} r_{ik} = 1$ for all k.

For η_{mk}, we used real-valued parameter λ ($\in [0, 1]$), which controls the distribution of edges (of each cluster) over graphs. In other words, parameter λ controls the cluster bias over graphs. That is, under some setting of λ, edges of a cluster can be set to generate from only one graph, resulting in that this cluster becomes a localized cluster. In reality, we first let the number of graphs be the number of clusters, i.e. $M' = K$, and then formulated η_{mk} so that when λ is one, edges of the k-th cluster are all generated from the k-th graph:

$$\eta_{mk} = \begin{cases} \lambda & \text{if} \quad m = k \\ \frac{1}{K-1}(1-\lambda) & \text{otherwise.} \end{cases}$$

In this setting, we note that $\sum_{m=1}^{K} \eta_{mk} = 1$ for each k. If $\lambda = 1$, all edges of a cluster appear only in one graph, while if $\lambda = \frac{1}{K}$, edges are generated with an equal probability $\frac{1}{K}$ for all graphs at each cluster.

In summary we have two parameters on generating synthetic data, i.e. R_{out} and λ. R_{out} controls the number of inter-cluster edges, i.e. the amount of noise, which is larger for a larger R_{out}. λ controls the generation of localized clusters, which are more likely to be generated for a larger λ. We expect that comparing to other methods, PMMG will work even for a larger R_{out} and a larger λ.

Fig. 5 shows some examples of synthetic graphs (weight matrices) for $M' = K = 3$, being shown as $\boldsymbol{W}^{(1)}$, $\boldsymbol{W}^{(2)}$ and $\boldsymbol{W}^{(3)}$. In this figure, a white pixel shows that the corresponding weight is larger than zero, meaning that there is an edge at the corresponding pixel. For all (a)-(d), $\boldsymbol{W}^{(int)}$ is a matrix in which all corresponding weights of $\boldsymbol{W}^{(1)}$, $\boldsymbol{W}^{(2)}$ and $\boldsymbol{W}^{(3)}$ are simply summed at each pixel. Fig. 5 (a) shows three weight matrices, which are generated under $\lambda = \frac{1}{3}$ and $R_{out} = 0.1$, where all $\boldsymbol{W}^{(1)}$, $\boldsymbol{W}^{(2)}$ and $\boldsymbol{W}^{(3)}$ are the same. Fig. 5 (b) shows the case of $\lambda = 0.9$ and $R_{out} = 0.1$. We note that (a) and (b) are totally different but two typical, extreme cases of λ, i.e. (a): clusters are equally distributed and (b) clusters are biased (or localized clusters). However regarding $\boldsymbol{W}^{(int)}$, (a) and (b) are almost similar and three clusters are clearly shown in $\boldsymbol{W}^{(int)}$. We can thus expect that existing clustering methods (as well as PMMG) will work well for both (a) and (b).

On the other hand, Fig. 5 (c) is by $\lambda = \frac{1}{3}$ and $R_{out} = 0.5$ and Fig. 5 (d) is by $\lambda = 0.9$ and $R_{out} = 0.5$, both of which have abundant inter-cluster edges, meaning that both are very noisy. You can easily see that in both (c) and (d), $\boldsymbol{W}^{(int)}$ shows very vague, unclear three clusters, for which a standard clustering method will not work well. However, we can expect that in (d), a localized cluster can be seen in each of $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}$ and $\boldsymbol{W}^{(3)}$, implying that our approach will work for (d) better than other competing clustering methods.

TABLE 2
Five gene networks (PI:Physical interaction, GI: Genetic interaction, PPI: Protein interaction, SS: Sequence similarity and CC: ChIP on Chip)

Name	#edges	References
$\boldsymbol{W}^{(PI)}$	569	[18]
$\boldsymbol{W}^{(GI)}$	548	[19]
$\boldsymbol{W}^{(PPI)}$	1,763	[20]
$\boldsymbol{W}^{(SS)}$	1,315	[21]
$\boldsymbol{W}^{(CC)}$	777	[22]

TABLE 3
Standard gene clusters from MIPS FunCat

ID	Name	#genes
01.01	Amino acid	202
01.03	Nucleotide	195
01.04	Phosphate	351
01.05	Carbohydrate	422
01.06	Lipid, fatty acid and isoprenoid	176
01.07	Vitamins, cofactors, etc.	138

4.1.2 Real Genomic Data

Table 2 shows five gene networks we used in our experiment. We adjusted cut-off values for $\boldsymbol{W}^{(SS)}$ and $\boldsymbol{W}^{(CC)}$ to make the number of edges in these two graphs almost equal to those of the other three networks (for which we could not control the number of edges). We focused on 1,207 metabolism-related genes, which were found in the maximal connected component (MCC) of the union of the five networks. To evaluate the clustering results, as gold standard clusters, we used six major categories in metabolism, which can be provided by FunCat in MIPS [23]. Table 3 shows the detail of the six categories.

4.1.3 Competing Methods

We compared the performance of PMMG with those of other five approaches. The first comparison method was PMSG, and the second one was MSC, which is a method for graph clustering with multiple graphs [17]. The third one was CSBM [11], for which the model generation process was described in Section 3.4. The fourth one was spectral clustering with ratio cut (SC), a standard graph clustering approach. We skip the detail of SC in this paper, and interested readers should refer [6]. The fifth one is MSBM, for which also the model generation process was already explained in Section 3.4. As shown in Section 3.5, MSBM has a high complexity which makes it hard to apply to the real data including that we used in this paper. Thus we used MSBM for synthetic data only. We note that CSBM, SC and MSBM were run on $\boldsymbol{W}^{(int)}$ and in particular, an only binary matrix, i.e. $W_{i,j}^{(int)} \in \{0, 1\}$, was used for CSBM and MSBM. For real genomic data, we further used a simple random cluster assignment method (RA), to confirm the performance advantage of each method over RA.

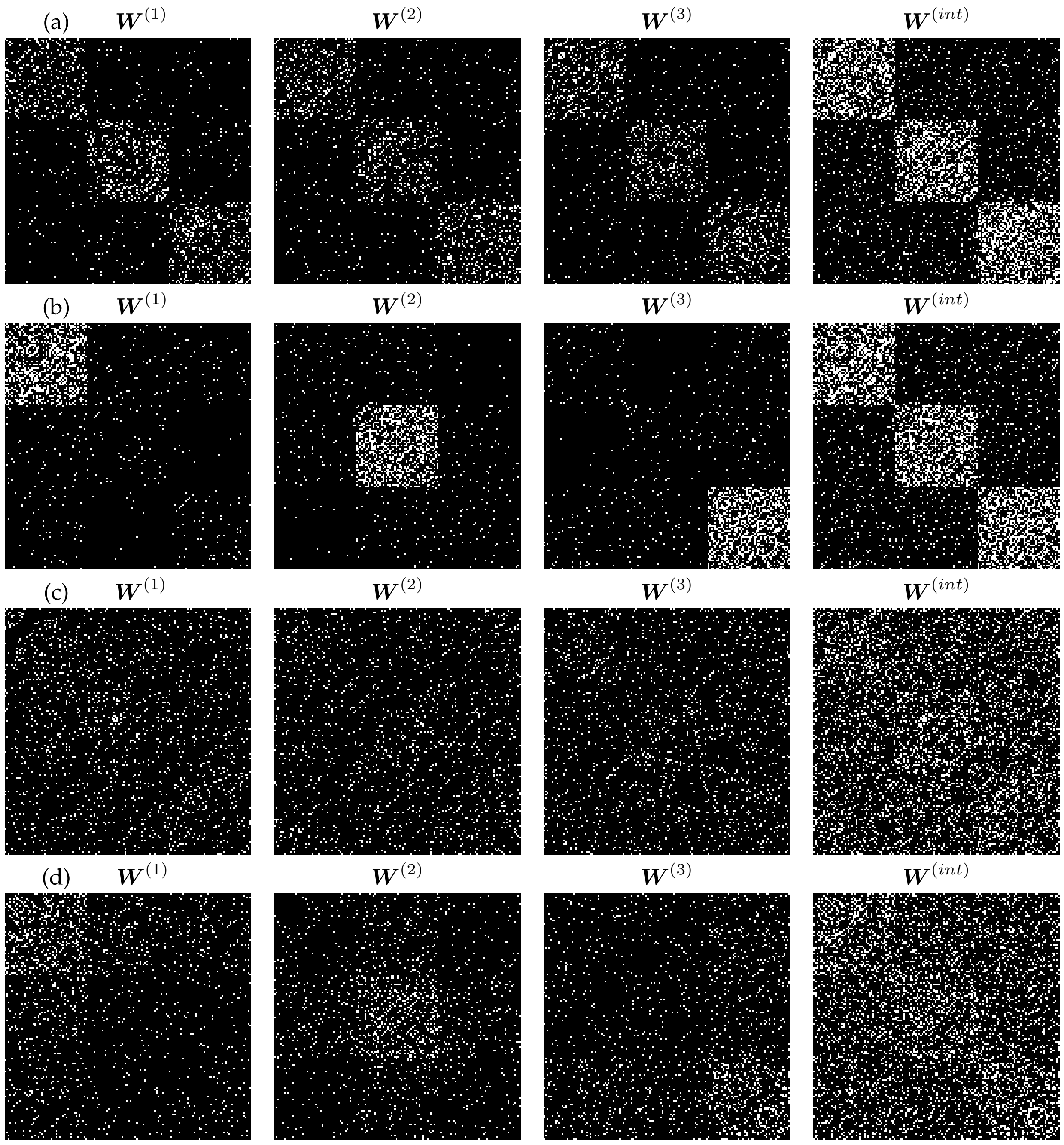

Fig. 5. Example synthetic data under (a) $\lambda = \frac{1}{3}, R_{out} = 0.1$, (b) $\lambda = 0.9, R_{out} = 0.1$, (c) $\lambda = 0.9, R_{out} = 0.5$ and (d) $\lambda = \frac{1}{3}, R_{out} = 0.5$.

4.1.4 Evaluation Measure

We used normalized mutual information (NMI), which is a standard measure for evaluating clustering results [24]. NMI assumes that we can have true clusters as input. For the distribution of resultant (empirical) clusters $P(X)$ and true cluster distribution $P(Y)$, NMI is given by:

$$\text{NMI} := \frac{MI(X,Y)}{\sqrt{H(X)}\sqrt{H(Y)}},$$

where $MI(X,Y) := H(X) + H(Y) - H(X,Y)$, $H(X) := -\sum_X P(X) \log P(X)$ and $H(X,Y) := -\sum_X P(X,Y) \log P(X,Y)$. NMI shows the overlap between predicted clusters and true clusters, meaning that the performance is better as NMI is larger.

4.2 Results on Synthetic Data

We generated five synthetic graphs, i.e. $M' = 5$. For PMSG and PMMG, we used $\alpha_0 = \beta_0 = \zeta_0 = 1.0$, by which uniform distributions are generated to avoid any bias. Under each values of R_{out} and λ, we generated twenty graphs randomly, and then for each graph, we run each competing method ten times with different random initial values and obtained the best.

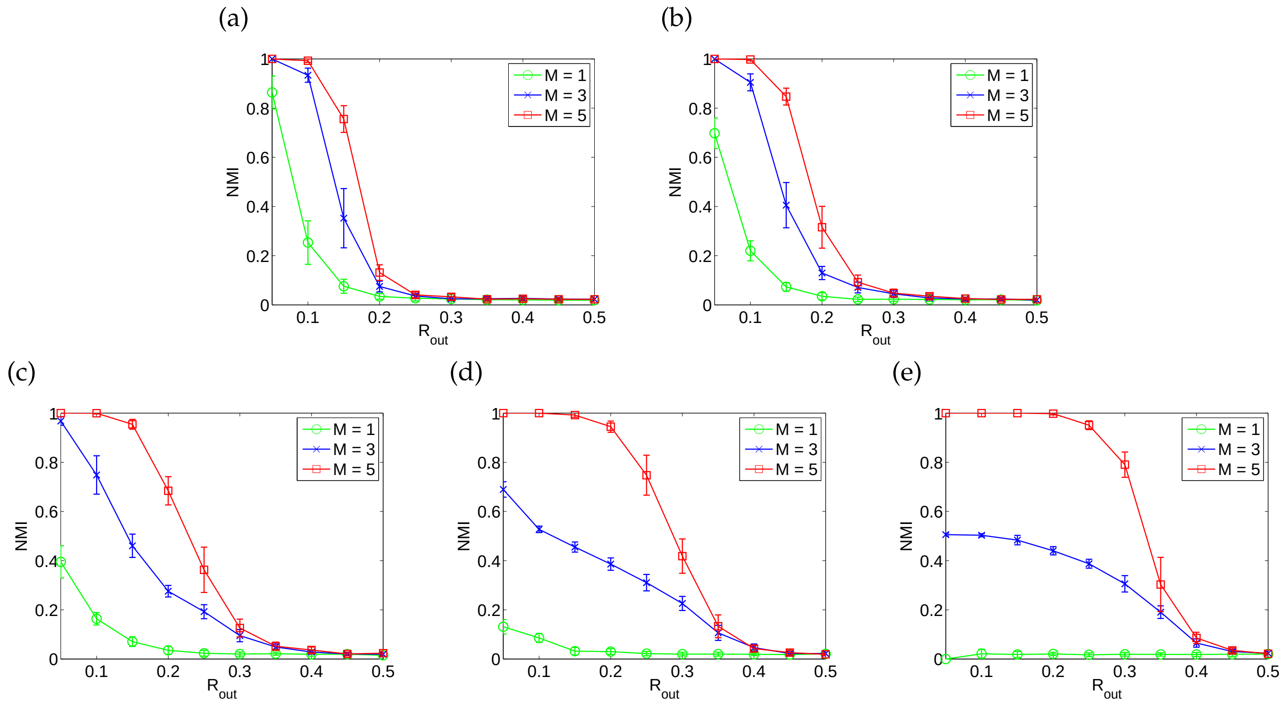

Fig. 6. NMI by PMMG for M=1, 3 and 5, when λ was (a) 0.2, (b) 0.4, (c) 0.6, (d) 0.8 and (e) 1.0.

Finally we averaged over the results of twenty best trials.

4.2.1 Effect by Increasing Graphs

We chose one, three and five graphs out of five graphs generated, meaning that M is set at one, three and five. We then checked the performance of PMMG under these three values of M. Fig. 6 shows the NMI of PMMG for M of the three values, when we changed R_{out} and λ. This figure reveals that NMI by PMMG reduced as R_{out} increased for all cases. However, for all λ, NMI reduced more slowly for a larger M. For example, in Fig. 6 (a) where $\lambda = 0.2$, at $R_{out} = 0.1$, NMI was almost 1.0 for $M = 5$ and around 0.9 for $M = 3$ while that was less than 0.3 for $M = 1$. This difference was more significant for a larger λ. For example, in Fig. 6 (e) where $\lambda = 1.0$, at $R_{out} = 0.1$, NMI was almost 1.0 for $M = 5$, while that was around 0.5 for $M = 3$ and almost 0 for $M = 1$. From this result, we can easily see that NMI of PMMG was kept higher for the case that more graphs are given. This was pronounced more for a larger R_{out} as well as a larger λ. This result indicates that PMMG captured localized clusters better by using more graphs.

4.2.2 Performance Comparison with Competing Methods

We then checked the performance advantage of PMMG over other five competing methods, i.e. PMSG, MSC, CSBM, SC and MSBM. We note that among the five competing methods, PMSG, CSBM, SC and MSBM were run on $\boldsymbol{W}^{(int)}$, while MSC was run over multiple graphs directly. Fig. 7 shows the NMI of these six methods when R_{out} and λ changed. We here focus on five methods: PMMG, PMSG, CSBM, MSC and SC only, since MSBM clearly underperformed other five methods because of its high model complexity which makes MSBM hard to estimate good parameter values for any R_{out}. As in Fig. 6, the NMI reduced as R_{out} increased for all five methods. We emphasize that the NMI of PMMG reduced most slowly among the five methods. Interestingly, this feature was pronounced more for a larger λ clearly. For example, in Fig. 7 (a) where $\lambda = 0.2$, the five curves of NMI were similar to each other, while in Fig. 7 (e) where $\lambda = 1.0$, at R_{out} of 0.2, the NMI of PMMG was still 1.0 but those of the other four were less than 0.1. Thus from this result, we can clearly say that PMMG significantly outperformed other clustering approaches: PMSG, MSC, CSBM, SC and MSBM. We emphasize that the performance advantage of PMMG was achieved by the feature of capturing localized clusters which are embedded in multiple graphs. We note again that the performance of MSBM was lowest, because of its high model complexity which made hard to find good initial values even for the size of synthetic data we used.

4.2.3 Model Selection by Using $L[q]$

We run each of PMMG, PMSG and CSBM with changing K from two to seven and computed $L[q]$ for each setting[3]. The number of true clusters is five, meaning that the true number of clusters (model) was found if

3. We were unable to run MSBM, because of its high computational burden for each K.

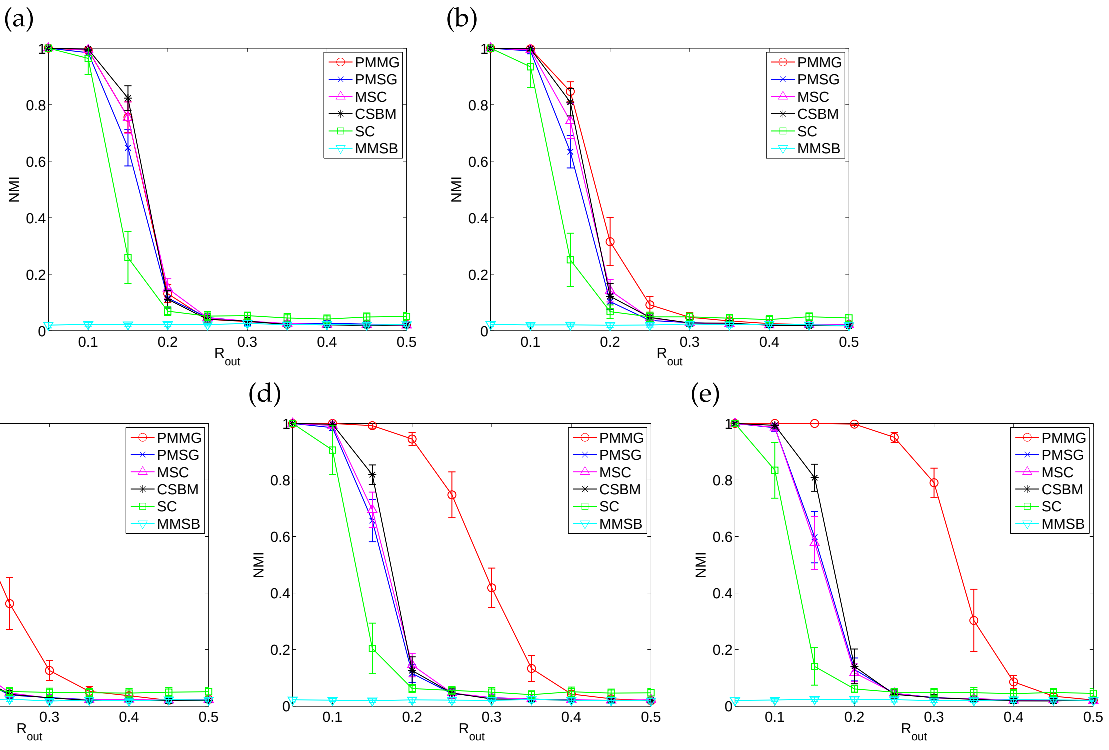

Fig. 7. NMI by six competing methods when λ was (a) 0.2, (b) 0.4, (c) 0.6, (d) 0.8 and (e) 1.0.

$L[q]$ was maximized at $K = 5$. As a parameter to be used in this subsection, we first selected $R_{out} = 0.1$ where NMI was high for all three methods in Fig. 7. We further selected $R_{out} = 0.2$ and two λ values, i.e. $\lambda = 0.2$ and 0.6, since NMI was very low for all cases under $\lambda = 0.2$ while NMI of PMMG was improved under $\lambda = 0.6$. Fig. 8 shows $L[q]$ with changing K under these three cases. The results showed that $L[q]$ was maximized at $K = 5$ for all methods under $R_{out} = 0.1$. On the other hand, under $R_{out} = 0.2$, $L[q]$ was not maximized under λ=0.2 for all methods, while $L[q]$ was maximized at $K = 5$ only for PMMG under λ=0.6. This result was consistent with the results of NMI, showing that PMMG selected the best model correctly the most among the three methods.

TABLE 4
NMI of six competing methods with Z-scores (in parentheses) against RA.

K	PMMG	PMSG	MSC	CSBM	SC	RA
6	**0.1015**	0.0815	0.0767	0.0870	0.0622	0.0073
	(41.56)	(32.70)	(30.56)	(35.12)	(24.13)	-

TABLE 5
#edges in MIPS FunCat ID of 01.04 (PI:Physical interaction

Name	#edges in 01.04
$\boldsymbol{W}^{(PI)}$	90
$\boldsymbol{W}^{(GI)}$	110
$\boldsymbol{W}^{(PPI)}$	166
$\boldsymbol{W}^{(SS)}$	906
$\boldsymbol{W}^{(CC)}$	40

4.3 Results on Real Genomic Data

We first run PMMG, PMSG and CSBM with changing K and computed $L[q]$ to find the number of clusters which maximizes $L[q]$[4]. Fig. 9 shows the optimized $L[q]$ with changing K of the three methods, showing that the optimized number of clusters was six, four and 27 for PMMG, PMSG and CSBM, respectively. In fact, the true number of clusters given by MIPS for this dataset was six, meaning that the true number of clusters was selected by PMMG only. We then focused on the case of $K = 6$ and run all competing methods under this setting. Table 4 shows the resultant NMI of competing methods, where RA is a random assignment and the NMI of RA was averaged over 10,000 trials of RA. Each NMI value was relatively low, i.e. 0.05 - 0.1, mainly because gold standard clusters were overlapped with each other, while hard assignment was done in each clustering method. However these values of NMI were significant, since their Z-scores against RA were 24.13 to 41.56. More importantly, this table shows that PMMG clearly outperformed other competing methods in both NMI and Z-scores. In addition, Z-scores of PMSG and CSBM under K=4 and 27 were 35.28 and 38.12, respectively, being significantly large but lower than 41.56 which was obtained by PMMG under $K = 6$.

4. At each K, we run each method one hundred times with different random initial values and chose the best case, which maximizes $L[q]$. For PMMG (and PMSG), we kept $\alpha_0 = \beta_0 = \zeta_0 = 1.0$.

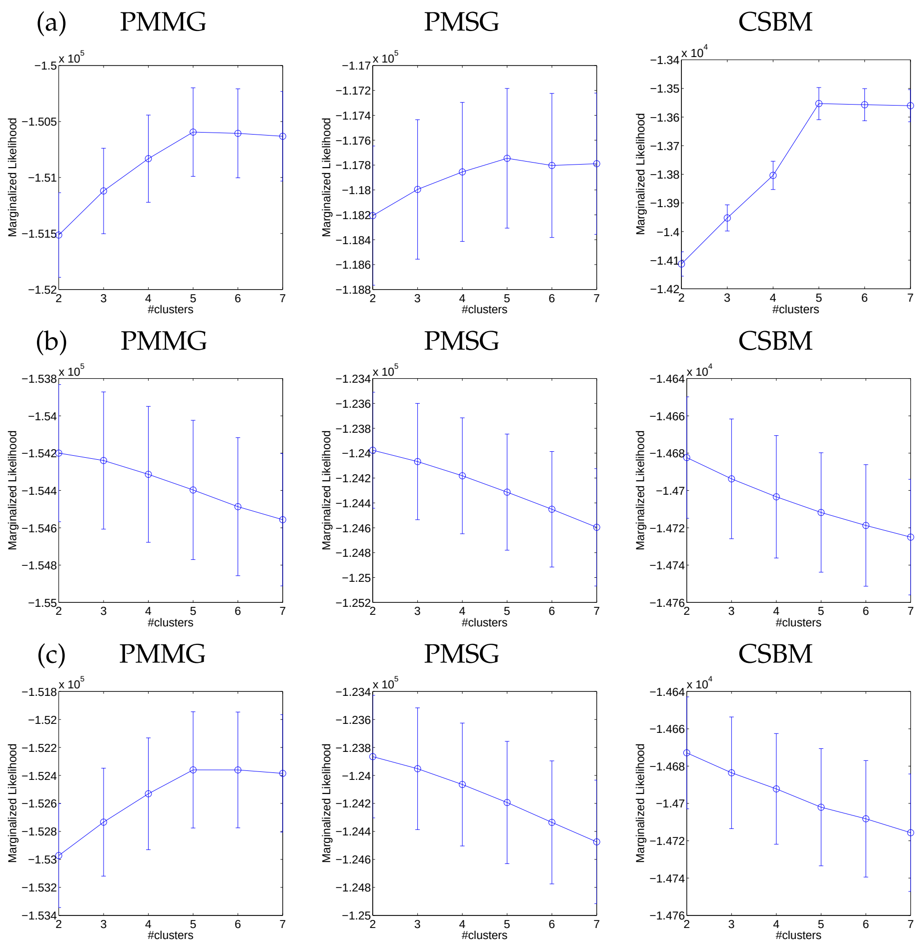

Fig. 8. Optimized values of $L[q]$ for PMMG, PMSG and CSBM under different K (# clusters) and (a) $R_{out} = 0.1$, $\lambda = 0.2$, (b) $R_{out} = 0.2$, $\lambda = 0.2$, (c) $R_{out} = 0.2$, $\lambda = 0.6$.

We went into the detail of clustering results by PMMG. We first checked the distribution of edges generated by gold standard clusters over five gene networks and then found that edges by genes in FunCatID:01.04 (i.e. phosphate metabolism) were mainly found in $\boldsymbol{W}^{(SS)}$. In fact, Table 5 shows the number of edges in 01.04 for five gene networks, and this table shows that $\boldsymbol{W}^{(SS)}$ had 906 edges in 01.04, while each of other networks had only around 200 or less edges. This means that there is a localized cluster in $\boldsymbol{W}^{(SS)}$, corresponding to 01.04. Fig. 10 (a), (b) and (c) are $\boldsymbol{W}^{(SS)}$, $\boldsymbol{W}^{(GI)}$ and $\boldsymbol{W}^{(int)}$, respectively, where $\boldsymbol{W}^{(int)}$ is the integrated graph over five gene networks and in each of the three graphs, genes (nodes) are sorted according to the gold standard clusters (01.01: green, 01.03: blue, 01.04:light blue, 01.05:pink, 01.06:yellow and 01.07: white). We can see that many edges fall into the third rectangle, i.e. 01.04, in (a), while this feature cannot be seen in (b). Fig. 10 (c) shows that the density of dots in the third rectangle is not so much higher than the rest of all areas, implying that this rectangle might not be captured as a cluster by using only (c). We then checked clustering results of PMMG and PMSG. Fig. 11 (a) and (b) show weight matrices with resultant clusters by PMSG and PMMG, respectively, where genes (nodes) are first sorted according to the clustering results of each method and then gold standard clusters. The dotted lines show the clustering results and colored lines show the gold standard clusters where colors are consistent with those specified by Fig. 10. From the comparison of $\boldsymbol{W}^{(SS)}$ between PMMG and PMSG, we can first see that dots were well gathered in a light blued rectangle for PMMG but not for PMSG, meaning that PMMG captured the localized cluster well while PMSG not. Furthermore this light blued rectangle occupies the fourth rectangle (from the left) of $\boldsymbol{W}^{(SS)}$ by PMMG (Fig. 11 (b)), meaning that this cluster corresponds to 01.04. Similarly, the sixth rectangle (from the left) of $\boldsymbol{W}^{(GI)}$ by PMMG contained a pink colored rectangle with many dots, which however was not shown in $\boldsymbol{W}^{(GI)}$ by PMSG. This also shows a result that a pink colored localized cluster in $\boldsymbol{W}^{(GI)}$ (See Fig. 10 (b)) was (partly) captured by PMMG but not by PMSG. Finally these results also confirm the effectiveness of PMMG on real applications including localized clusters.

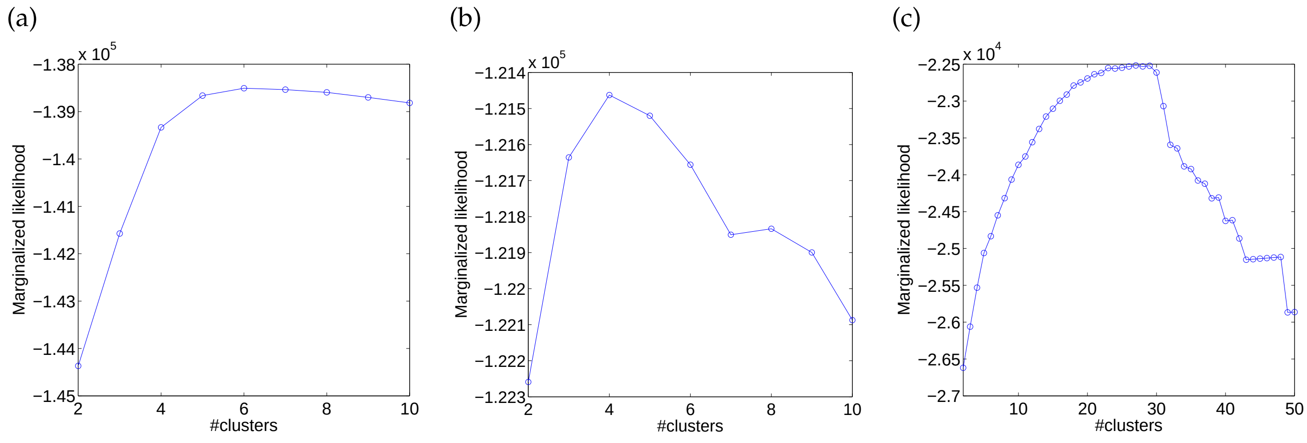

Fig. 9. Optimized $L[q]$ with changing K for (a) PMMG, (b) PMSG and (c) CSBM

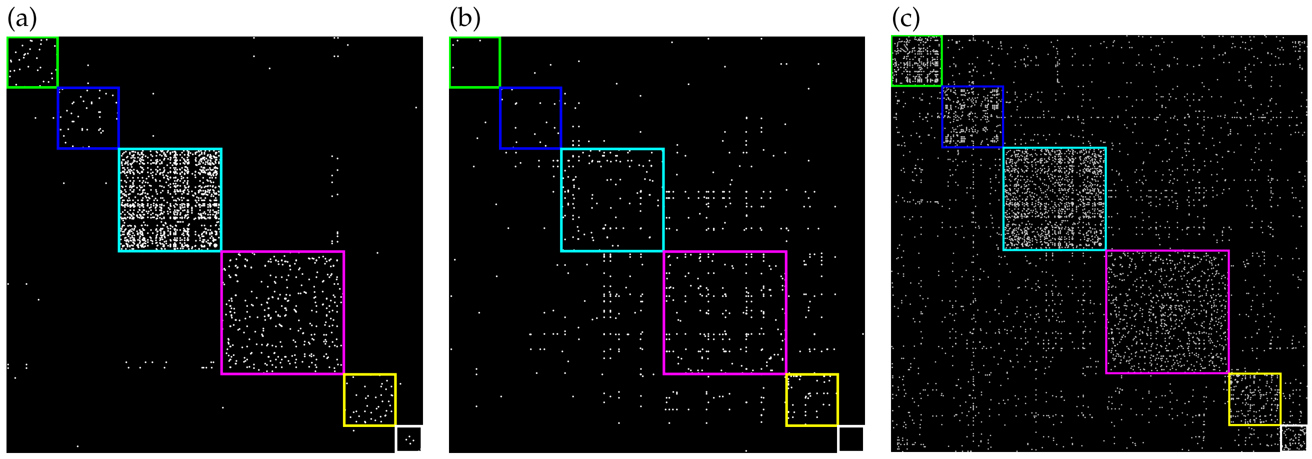

Fig. 10. Three weight matrices, (a) $\boldsymbol{W}^{(SS)}$, (b) $\boldsymbol{W}^{(GI)}$, and (c) $\boldsymbol{W}^{(int)}$, where genes are ordered, according to the MIPS (gold standard) clusters. Edges in FunCatID of 01.04 (phosphate metabolism) are in the third rectangle from the left.

5 Concluding Remarks

We have proposed a probabilistic model-based approach, PMMG (Probabilistic Model for Multiple Graphs), for clustering with multiple graphs. The emphasis of our approach is placed upon localized clusters, which are not found in all given graphs and well-accepted in real applications. The complexity analysis of PMMG and related probabilistic models shows that the space and time complexities of PMMG are kept the same as those of the most efficient probabilistic models including PMSG. Experimental results showed that PMMG clearly outperformed competing graph clustering approaches in both synthetic and real data, showing the effectiveness of PMMG for the problem of graph clustering with multiple graphs. Furthermore the results from real genomic data of using five gene networks showed that PMMG captured localized clusters, which were uniquely found in the network on sequence similarity.

Frequent pattern mining is the most major approach in knowledge discovery and data mining [25], and there already exist efficient methods for mining from graphs, e.g. [26]. Frequent patterns (subgraphs) do not have to appear in all given graphs, and instead they need to appear a larger number of times than the user-specified value, called support. Frequent patterns can be *localized* patterns, meaning that our problem shares some concept with frequent pattern mining over graphs. However, a clear difference between these two is that we focus on ‘clusters’ but not patterns. In fact, clusters are more flexible and concise than frequent patterns, which are rigid but usually redundant. This point indicates that our approach based on statistical machine learning would be a right direction to solve our issue of clustering under multiple graphs with localized clusters.

Acknowledgments

M.S. has been supported in part by KAKENHI, MEXT of Japan, 20700269.

References

[1] S. E. Schaeffer, “Graph clustering,” *Computer Scinece Review*, vol. 1, pp. 27–64, 2007.

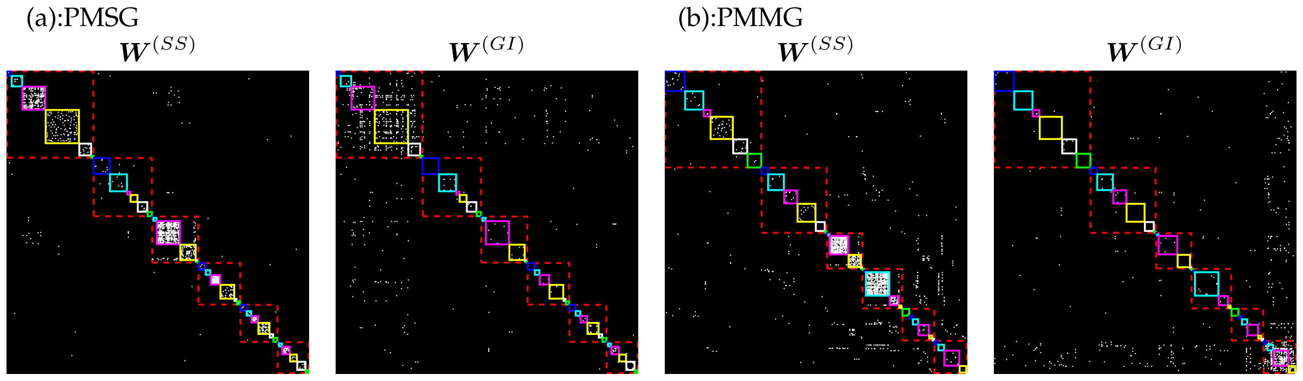

Fig. 11. Weight matrices, $\boldsymbol{W}^{(SS)}$ and $\boldsymbol{W}^{(GI)}$, with resultant clusters by (a) PMMG and (b) PMSG.

[2] S. Arora, S. Rao, and U. Vazirani, "Geometry, flows, and graph-partitioning algorithms," *Communications of the ACM*, vol. 51, no. 10, pp. 96–105, 2008.

[3] E. M. Marcotte *et al.*, "A combined algorithm for genome-wide prediction of protein function." *Nature*, vol. 402, no. 6757, pp. 25–26, 1999.

[4] R. Sharan *et al.*, "Network-based prediction of protein function," *Molecular Systems Biology*, vol. 3, no. 1, p. 88, 2007.

[5] K. Maciag *et al.*, "Systems-level analyses identify extensive coupling among gene expression machines," *Molecular Systems Biology*, vol. 2, no. 1, 2006.

[6] U. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[7] H. Attias, "A variational Bayesian framework for graphical models," in *NIPS*, 2000, pp. 209–215.

[8] C. M. Bishop, "Approximate inference," in *Pattern Recognition and Machine Learning*. Springer, 2006, ch. 10, pp. 461–522.

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, pp. 993–1022, 2003.

[10] M. E. J. Newman and E. A. Leicht, "Mixture models and exploratory analysis in networks," *PNAS*, vol. 104, no. 23, pp. 9564–69, 2007.

[11] J. M. Hofman and C. H. Wiggins, "Bayesian approach to network modularity," *Phys. Rev. Lett.*, vol. 100, no. 25, p. 258701, 2008.

[12] J. Sinkkonen, J. Aukia, and S. Kaski, "Component models for large networks," 2008, arXiv e-prints, arXiv:0803.1628.

[13] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social Networks*, vol. 5, no. 2, pp. 109–137, 1983.

[14] T. Snijders and K. Nowicki, "Estimation and prediction for stochastic blockmodels for graphs with latent block structure," *Journal of Classification*, vol. 14, no. 1, pp. 75–100, 1997.

[15] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," *JMLR*, vol. 9, pp. 1981–2014, 2008.

[16] J. Sinkkonen, J. Parkkinen, J. Aukia, and S. Kaski, "A simple infinite topic mixture for rich graphs and relational data," in *NIPS 2008 Workshop on Analyzing Graphs: Theory and Applications*, 2008.

[17] D. Zhou and C. J. C. Burges, "Spectral clustering and transductive learning with multiple views," in *ICML*. New York, NY, USA: ACM, 2007, pp. 1159–1166.

[18] T. Ito *et al.*, "A comprehensive two-hybrid analysis to explore the yeast protein interactome," *PNAS*, vol. 98, no. 8, pp. 4569–74, 2001.

[19] A. H. Tong *et al.*, "Global mapping of the yeast genetic interaction network," *Science*, vol. 303, no. 5569, pp. 803–13, 2004.

[20] C. von Mering *et al.*, "Comparative assessment of large-scale data sets of protein-protein interactions," *Nature*, vol. 417, no. 6887, pp. 399–403, 2002.

[21] J. M. Cherry *et al.*, "Genetic and physical maps of saccharomyces cerevisiae," *Nature*, vol. 387, no. 6632 Suppl., pp. 67–73, 1997.

[22] C. T. Harbison *et al.*, "Transcriptional regulatory code of a eukaryotic genome," *Nature*, vol. 431, no. 7004, pp. 99–104, 2004.

[23] A. Ruepp *et al.*, "The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes," *NAR*, vol. 32, no. 18, pp. 5539–5545, 2004.

[24] A. Strehl and J. Ghosh, "Relationship-based clustering and visualization for high-dimensional data mining," *INFORMS Journal on Computing*, vol. 15, no. 2, pp. 208–230, 2003.

[25] J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent pattern mining: Current status and future directions," *Data Mining and Knowledge Discovery*, vol. 15, pp. 55–86, 2007.

[26] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," in *ICDM*, 2002, pp. 721–724.

Motoki Shiga received his B.E., M.E. and Ph.D. in Information Science from Gifu University in 2001, 2003 and 2006, respectively. He has been working on data mining, machine learning and their applications to bioinformatics.

Hiroshi Mamitsuka received B.S. in Biophysics and Biochemistry, M.E. in Information Engineering and PhD in Information Sciences all from the University of Tokyo in 1988, 1991 and 1999, respectively. He has been working in machine learning, data mining and bioinformatics. His current research interests are in mining from graphs and networks in biology and chemistry.