Pattern Recognition 158 (2025) 111027

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Fast adaptively balanced min-cut clustering

Feiping Nie [a,d], Fangyuan Xie [a], Jingyu Wang [b,c,*], Xuelong Li [d]

[a] School of Artificial Intelligence, OPtics and ElectroNics (iOPEN), Northwestern Polytechnical University, Xi'an 710072, PR China
[b] School of Astronautics, Northwestern Polytechnical University, Xi'an 710072, PR China
[c] National Key Laboratory of Air-based Information Perception and Fusion, Luoyang 471000, PR China
[d] Institute of Artificial Intelligence (TeleAI), China Telecom, PR China

ARTICLE INFO

Keywords:
Fast clustering
Bipartite graph
Balanced min-cut clustering
Coordinate descent method

ABSTRACT

Min-cut clustering is a typical graph clustering method, which has been extensively applied for pattern recognition, data analysis and image processing. However, min-cut has trivial solutions, which leads to skewed clustering performance. Spectral clustering (SC) alleviates this by relaxing indicator matrix into continuous embedding and then discretizes the embedding. However, there are two primary challenges in SC, which are high computational complexity and two-stage process. To resolve these issues, a fast adaptively balanced min-cut clustering model (FBMC) is proposed, which directly solves the discrete indicator matrix without any post-processing. We utilize the bipartite graph to accelerate the construction of affinity graph and process of optimization. Besides, the balance factors are added in the model, which could alleviate skewed clustering results. Two specific methods are proposed, in which one adds a balance factor to all clusters while the other assigns balance factors to each cluster separately, referred to as FBMC1 and FBMC2. What is more, a one-step optimization is proposed to solve FBMC1 and FBMC2, the complexity of which is linear of time. Finally, comprehensive experiments results highlight the superior performance of our proposed method.

1. Introduction

Clustering is an essential and crucial research area in machine learning, which measures the similarity between data with different metrics and generates labels to guide other tasks [1,2]. It has been extensively used in many real-world scenarios, for instance image classification [3,4], object segmentation [5,6] and text clustering [7,8]. Numerous researchers have introduced a variety of clustering algorithms, including partition-based, graph based and so on. Among them, KM [9] and min-cut [10] clustering are typical ones. KM clusters the data into severals partitions by minimizing the square errors between samples and their corresponding cluster centers, which is partition-based. The graph-based methods construct an affinity graph to represent the relationship between data, in which the nodes correspond to data points and the edges represent the similarity or affinity between these points. The min-cut clustering seeks to partition the graph into several disjoint subsets, minimizing the connectivity weight between clusters. However, due to the lack of constraints on the size of sub-clusters, min-cut clustering often results in imbalanced partitions where one cluster may contain a majority of the nodes while others have very few [11]. The imbalanced results are undesirable as they can lead to poor representation and analysis of the data, making the clustering less effective for practical applications. To balance the clustering results and solve the problem of min-cut effectively, there are several approaches proposed. One method is to change the graph-cut criterion and another is to add balanced regularization to avoid skewed results. Spectral clustering (SC) is proposed by adding additional weights in min-cut, which includes ratio-cut (RCUT) and normalized-cut (NCUT) [12]. RCUT improves the criterion of min-cut by normalizing the cut size with respect to the resulting partitions. While NCUT normalizes the cut by means of the internal connectivity of the partitions. Another approach to add balanced regularization for min-cut clustering, which could balance the skewed clustering partitions.

SC has drawn many attention for its satisfying clustering performance by digging out the intrinsic structure between data [13,14]. It includes several procedures. Firstly, it converts data into an undirected and weighted affinity graph. Then, it performs eigenvalue decomposition of graph Laplacian matrix to get spectral embedding and finally KM or spectral rotation (SR) are applied to obtain the final clustering assignments. Nonetheless, there exist several issues in SC which hiders its further expansion [15]. Firstly, the construction of affinity graph and eigenvalue decomposition procedure are proven time-consuming and impractical for large-scale data. The eigenvalue decomposition operation incurs high computational costs, which is $\mathcal{O}(n^2)$ and n denotes the number of samples [16]. Secondly, the subsequent KM or SR operation

* Corresponding author.
E-mail addresses: feipingnie@gmail.com (F. Nie), xiefangyuan@mail.nwpu.edu.cn (F. Xie), jywang@nwpu.edu.cn (J. Wang), li@nwpu.edu.cn (X. Li).

https://doi.org/10.1016/j.patcog.2024.111027
Received 28 April 2024; Received in revised form 17 July 2024; Accepted 15 September 2024
Available online 23 September 2024

are necessary to get the clustering labels. It is because that spectral embedding does not reflect the category information and therefore cannot be directly used as the category labels. The two-step strategy may lead to inaccurate analysis of the original data structure and makes the clustering process messy and redundant [17]. Thirdly, SC has no limit on the number of samples in each cluster, potentially leading to highly unbalanced clustering results with trivial solutions where some clusters contain very few samples. To address the aforementioned challenges, researchers have put forth numerous variants aimed at enhancing SC [18,19].

Generally, there are several ways to reduce computational cost of SC. These includes accelerating for eigenvalue decomposition [20], sampling the original data [21] and employing anchor-based technique [22,23]. Recently, anchor-based strategy have gained increasing attention due to their effectiveness, which has been widely applied in applications multi-view and subspace clustering [24,25]. Anchor generation is a crucial step in anchor-based clustering methods, typically accomplished through KM or KM-based methods, which are simple, fast and effective. After obtaining the anchor points, how to use them to accelerate the spectral clustering method is also a problem. Liu et al. [26] propose constructing a similarity graph using a bipartite graph between sample points and anchor points to speed up the construction of affinity graph. This is widely applied in subsequent anchor-based methods [22,27,28]. Chen and Cai [22] propose the landmark-based spectral clustering (LSC), which uses the bipartite graph to transform the eigenvalue decomposition problem into singular value decomposition, greatly accelerating SC. To enhance the selection of representative points or anchors, Zhu et al. [27] propose the Balanced K-means based Hierarchical K-means (BKHK) and introduce fast spectral clustering (FSC). Additionally, Huang et al. [29] propose the ultra-scalable spectral clustering (USPEC) method, which introduces a hybrid anchor selection method aiming to balance the efficiency of random selection and the effectiveness of KM selection. By means of the anchor graph, Nie et al. [28] propose a fast clustering by directly solving bipartite graph clustering problem (FDBC), which efficiently solve the problem of SC with bipartite graph and coordinate descent (CD) method. The CD method is also applied in KM, which improves the performance of Lloyd method [30]. However, the aforementioned methods still face some issues. For instance, after obtaining embeddings using bipartite graphs, subsequent discretization with K-means is necessary to obtain the final clustering results, making the entire process not end-to-end. Furthermore, the handling of trivial solutions has not been addressed.

The balanced clustering methods could avoid trivial solutions by balancing skewed clustering results. They also gain more attention due to the real world application requirement, for instance the energy load of wireless sensor networks [31]. Depending on whether there are strict limitations on the number of samples within clusters, it is categorized into hard balanced and soft balanced clustering. Our emphasis is on the latter [32,33]. The soft balanced clustering could be realized by adding exclusive lasso [34] or normalized entropy [35] as the regularization term. A self-balanced min-cut algorithm (SBMC) is proposed to balance the size of clusters and directly optimize the indicator matrix [36]. SBMC eliminates the step of eigenvalue decomposition and the subsequent discretization process by KM or SR [36]. To better dig out the discrepancy between different clusters, Chen et al. [37] put forward an enhanced balanced min-cut (EBMC) method. This approach incorporates a set of balanced parameters, forming a diagonal matrix. Later, Chen et al. [23] propose the large-scale balanced min-cut (LABIN), which improves SBMC by means of anchor strategy. Nie et al. [38] propose the fast fuzzy clustering based on anchor graph (FFCAG), in which the balanced regularization term is added in the model and the augmented Lagrangian multiplier method is used to solve the optimization problem. Liu et al. [39] improve FFCAG by adopting the form of trace ratio form. However, there still exit several issues, such as difficult to adjust hyper-parameters and two-stage optimization steps.

To address the issues of high time complexity and extremely imbalanced clustering results, we propose a fast adaptively balanced min-cut clustering method (FBMC) to ensure both efficiency and balanced representation across clusters. Our method could directly solve the discrete indicator matrix without any post-processing. We utilize the anchor to accelerate the construction of affinity graph and optimization. Besides, the balanced factors are added in the model, which could alleviate skewed clustering results. Two methods are proposed, one involves adding a balanced factor for all clusters, while the other involves adding respective balanced factors for each cluster, termed FBMC1 and FBMC2 respectively. Furthermore, the balanced factors could be updated adaptively. We propose a one-step optimization method to optimize the discrete indicator, which is solved and accelerated by coordinate descent method. Finally, the time complexity of our proposed two methods is linear with samples. Our contributions can be encapsulated as follows:

- We propose a fast adaptively balanced min-cut clustering, which aims to maximize intra-cluster similarity and could be served as an alternative to min-cut clustering. The anchor graph is adopted in the proposed method, which could reduce the computational complexity.
- By introducing the balance factor, the proposed model takes into account the balance of clustering, thereby avoiding the problem of trivial solutions in min-cut clustering. The calculation of the balance factor is adaptive and the model is parameter-free.
- A single-step optimization method is proposed to solve the optimization problem. Furthermore, in our proposed model, the discrete indicator matrix is solved directly by coordinate descent method without any post-processing operations, the complexity of which is in linear time.
- Extensive experiments are conducted to validate the effectiveness of the proposed method, including comparison with the two-step method, clustering performance, quality of anchor graph, convergence experiments and runtime analysis. The experimental results confirm the effectiveness of the proposed method.

Our paper is structured as follows. The notations and related works are introduced in Section 2. The proposed methods and optimization would be introduced in Section 3. Section 4 displays extensive experimental results on benchmark datasets. Section 5 provides the conclusion.

2. Notations and related works

2.1. Notations

Matrix is written by bold italicized capital letter and vectors are denoted by bold lowercase letters. We denote the data matrix with a total of c classes $\boldsymbol{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$, in which d is the dimension and n is the amount of samples. The label matrix is binarized and denoted by $\boldsymbol{F} = [\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_c] \in Ind$, in which c represents the number of clusters. Ind indicates that in each row of F, there is exactly one 1 and all other entries are 0. $\mathbf{f}_j$ represents the jth column and $\mathbf{f}^i$ is the ith row of $\boldsymbol{F}$. $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the affinity graph. The transpose, trace and Frobenius norm of $\boldsymbol{X}$ are $\boldsymbol{X}^T$, $\mathrm{Tr}(\boldsymbol{X})$ and $\|\boldsymbol{X}\|_F$ separately. $\boldsymbol{I}_n \in \mathbb{R}^{n\times n}$ denotes the identity matrix and $\mathbf{1}_n$ is the $\mathbb{R}^n$ vector, the elements of which are all one.

2.2. Anchor graph construction

Anchor graphs could accelerate the process of constructing affinity graph. KM is applied to select anchors and anchor is denoted as $\boldsymbol{U} \in \mathbb{R}^{d\times m}$, where m is the number of anchors. The effective parameter-free neighbor assignment method [40] is used to construct the sparse similarity graph $\boldsymbol{Z} \in \mathbb{R}^{n\times m}$ between samples and anchors. The calculation of $\boldsymbol{Z}$ is

$$z_{ij} = \begin{cases} \frac{e(\mathbf{x}_i,\mathbf{u}_{k+1}) - e(\mathbf{x}_i,\mathbf{u}_j)}{k e(\mathbf{x}_i,\mathbf{u}_{k+1}) - \sum_{r=1}^{k} e(\mathbf{x}_i,\mathbf{u}_r)} & j \le k \\ 0, & j > k \end{cases}. \tag{1}$$

where k is the number of nearest neighbors and $e(\mathbf{x}_i,\mathbf{u}_r) = \|\mathbf{x}_i - \mathbf{u}_r\|_2^2$, which is the square of the Euclidean distance between ith sample and rth anchor. Hence, the affinity matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is obtained by the following equation

$$\boldsymbol{A} = \boldsymbol{Z}\Lambda^{-1}\boldsymbol{Z}^T = \boldsymbol{B}^T\boldsymbol{B} \tag{2}$$

where $\Lambda = \text{diag}\left\{\sum_{j=1}^{n} z_{j1}, \sum_{j=1}^{n} z_{j2}, \ldots, \sum_{j=1}^{n} z_{jm}\right\}$. The formulation of $\boldsymbol{B} \in \mathbb{R}^{m\times n}$ is $\Lambda^{-(1/2)}\boldsymbol{Z}^T$.

2.3. SBMC, EBMC and LABIN

To balance the clustering partitions, the balanced regularization or exclusive lasso is added in min-cut clustering. The optimization problem is

$$\min_{\boldsymbol{F}\in\text{Ind}} \text{Tr}\left(\boldsymbol{F}^T\boldsymbol{L}\boldsymbol{F}\right) - \gamma\|\boldsymbol{F}\|_e \tag{3}$$

where γ is the regularization parameter and $\|\boldsymbol{F}\|_e$ is the exclusive lasso. The formulation of exclusive lasso could also be $\|\boldsymbol{F}\|_e = \text{Tr}\left(\boldsymbol{F}\boldsymbol{F}^T\boldsymbol{F}\boldsymbol{F}^T\right) = \text{Tr}\left(\boldsymbol{F}^T\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{F}\right)$ [41]. Then, problem (3) could become

$$\begin{aligned} &\min_{\boldsymbol{F}\in\text{Ind}} \text{Tr}\left(\boldsymbol{F}^T\boldsymbol{L}\boldsymbol{F}\right) - \gamma\,\text{Tr}\left(\boldsymbol{F}^T\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{F}\right) \\ \Longleftrightarrow\ &\max_{\boldsymbol{F}\in\text{Ind}} \text{Tr}\left(\boldsymbol{F}^T\boldsymbol{A}\boldsymbol{F}\right) + \gamma\,\text{Tr}\left(\boldsymbol{F}\boldsymbol{F}^T\boldsymbol{F}\boldsymbol{F}^T\right) \\ \Longleftrightarrow\ &\min_{\boldsymbol{F}\in\text{Ind}} \left\|\boldsymbol{A} - \frac{1}{2\gamma}\boldsymbol{F}\boldsymbol{F}^T\right\|_F^2. \end{aligned} \tag{4}$$

However, in problem (4), γ could not adjust automatically. To alleviate this, Chen et al. [36] introduce the balance factor to adjust the regularization parameter and propose self-balanced min-cut (SBMC) clustering. The objective function is

$$\min_{\boldsymbol{F}\in\text{Ind},s>0} \left\|\boldsymbol{A} - s\boldsymbol{F}\boldsymbol{F}^T\right\|_F^2 \tag{5}$$

where s is a scalar.

To better capture the difference among clusters, Chen et al. [37] propose the enhanced balanced min-cut (EBMC), which adds specified balanced factors for each cluster. The optimization problem of EBMC is

$$\min_{\boldsymbol{F}\in\text{Ind},\boldsymbol{S}\in Diag} \left\|\boldsymbol{A} - \boldsymbol{F}\boldsymbol{S}\boldsymbol{F}^T\right\|_F^2 \tag{6}$$

where $\boldsymbol{S}$ is a diagonal matrix where the diagonal element is the balance factor for each cluster. Problem (6) is solved by an alternative optimization algorithm by fixing one variable and updating another. By means of anchor strategy, Chen et al. [23] propose the large-scale balanced min cut method (LABIN). Furthermore, they utilize the improved spectral rotation to optimize the problem. The objective function of LABIN is

$$\min_{\boldsymbol{F}\in\text{Ind},s>0} \left\|\boldsymbol{B}^T\boldsymbol{B} - s\boldsymbol{F}\boldsymbol{F}^T\right\|_F^2. \tag{7}$$

The alternative optimization approach is also applied solve problem (7). In SBMC, EBMC and LABIN, since the indicator matrix is discrete, $\boldsymbol{F}$ is relaxed into continuous one and discrete it by spectral rotation strategy. Although they adequately addressed the issue of trivial solutions, such optimization processes are redundant and introduce additional steps or parameters. In this paper, we unify the aforementioned methods and adopt a one-stage approach to tackle these problems.

3. Proposed methodology

The proposed unified framework is illustrated as follows

$$\min_{\boldsymbol{F}\in\text{Ind},\Lambda\in Diag} \left\|\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{F}\Lambda\boldsymbol{F}^T\right\|_F^2 \tag{8}$$

where $\Lambda = diag\{\lambda_1,\lambda_2,\ldots,\lambda_c\}, \lambda_j > 0, j = 1,2,\ldots,c$. Λ is a diagonal matrix composed of balanced factors. When $\Lambda = \lambda\boldsymbol{I}_c$, the problem degrades to

$$\min_{\boldsymbol{F}\in\text{Ind},\lambda>0} \left\|\boldsymbol{B}^T\boldsymbol{B} - \lambda\boldsymbol{F}\boldsymbol{F}^T\right\|_F^2. \tag{9}$$

When $\exists i \neq j, \lambda_i \neq \lambda_j$, the optimization problem is

$$\min_{\boldsymbol{F}\in\text{Ind},\Lambda\in Diag,\exists i\neq j,\lambda_i\neq\lambda_j} \left\|\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{F}\Lambda\boldsymbol{F}^T\right\|_F^2. \tag{10}$$

Through the adaptive balanced factors in problem (9) and (10), the final clustering results would achieve high within-cluster similarity and avoid unbalanced clustering outcomes. In fact, Λ could be extended to any symmetric matrix, resulting in the following problem

$$\min_{\boldsymbol{F}\in\text{Ind},\Lambda=\Lambda^T} \left\|\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{F}\Lambda\boldsymbol{F}^T\right\|_F^2. \tag{11}$$

However, since the solving process in this case is similar to that in problem (9) and (10), which is quite complex as well. Thus, we will not discuss problem (11) in detail. In solving problem (9) and (10), we do not explicitly solve for λ or Λ. Instead, we incorporate the balanced factors back into the original problem, transforming it into a single-step method concerning $\boldsymbol{F}$ for implicit solving the balanced factors. Therefore, we name the proposed framework as fast adaptively balanced min-cut clustering. Problem (9) and (10) are referred to as FBMC1 and FBMC2 respectively.

3.1. Optimization of FBMC1

Problem (9) could be formulated as the following form and we denote the objective function as J_1

$$\max J_1(\boldsymbol{F},\lambda) = \max_{\boldsymbol{F}\in\text{Ind},\lambda>0} 2\lambda\,\text{Tr}\left(\boldsymbol{F}^T\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{F}\right) - \lambda^2\|\boldsymbol{F}\|_e. \tag{12}$$

By taking the partial derivative of λ, the expression for the optimal value of λ can be obtained as follows

$$\lambda = \frac{\text{Tr}\left(\boldsymbol{F}^T\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{F}\right)}{\text{Tr}\left(\boldsymbol{F}^T\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{F}\right)}. \tag{13}$$

From the expression of λ, it can be seen that the higher the similarity within the clusters and the more balanced the clustering results, the smaller the value of λ. From problem (4), a small value of λ indicates a big value of γ, which shows that the balancing term plays a more significant role. By substituting the expression of λ in J_1, the formulation of J_1 becomes

$$\max J_1(\boldsymbol{F}) = \max_{\boldsymbol{F}\in\text{Ind}} \frac{\left(\text{Tr}\left(\boldsymbol{F}^T\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{F}\right)\right)^2}{\text{Tr}\left(\boldsymbol{F}^T\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{F}\right)} = \max_{\boldsymbol{F}\in\text{Ind}} \frac{\left(\sum_{j=1}^{c}\mathbf{f}_j^T\boldsymbol{B}^T\boldsymbol{B}\mathbf{f}_j\right)^2}{\sum_{j=1}^{c}\mathbf{f}_j^T\mathbf{1}_n\mathbf{1}_n^T\mathbf{f}_j}. \tag{14}$$

Apparently, there is only one variable in problem (14). Therefore, our focus is concentrated on the optimization of $\boldsymbol{F}$ and there is no need to consider s. This is why we call the proposed method a one-step approach. Although formulation (14) seems complex, we utilize CD method to solve it. In each iteration, each row is seen as one coordinate and we update $\boldsymbol{F}$ row by row. Assuming that the variable waiting to be updated is the ith row of the indicator matrix, the obtained indicator matrix is denoted by $\boldsymbol{F}^{(0)}$. For ith row of $\boldsymbol{F}^{(0)}$, the pth element is 1 and other elements are all 0. That is, $f_{ip}^{(0)} = 1$. When we update ith row, there are totally c choices for $\mathbf{f}^i$, which are $\{[1,0,\ldots,0],[0,1,\ldots,0],\ldots,[0,0,\ldots,1]\}$. The corresponding indicator matrices are denoted as $\boldsymbol{F}^{(l)}, l = 1,2,\ldots,c$, in which $f_{il}^{(l)}$ in $\boldsymbol{F}^{(l)}$ is 1 and the remaining rows are the same with $\boldsymbol{F}^{(0)}$. Hence, the optimization problem of updating $\mathbf{f}^i$ is

$$\max_{\boldsymbol{F}\in\{\boldsymbol{F}^{(1)},\ldots,\boldsymbol{F}^{(l)},\ldots,\boldsymbol{F}^{(c)}\}} \frac{\left(\sum_{j=1}^{c}\mathbf{f}_j^T\boldsymbol{B}^T\boldsymbol{B}\mathbf{f}_j\right)^2}{\sum_{j=1}^{c}\mathbf{f}_j^T\mathbf{1}_n\mathbf{1}_n^T\mathbf{f}_j}. \tag{15}$$

To obtain the optimal $\boldsymbol{F}$, the value of $J_1(\boldsymbol{F}^{(l)})$ need to be calculated such that the $\boldsymbol{F}$ that maximizes J_1 is the optimal $\boldsymbol{F}$. However, if we

directly calculate the value of objective function of the c cases, the computation cost is intensive. In actual, $J_1(\boldsymbol{F}^{(l)})$ can be expressed by some elements of $\boldsymbol{F}^{(0)}$, whose values could be stored in advance. When the update of $\mathbf{f}_i$ is finished, the elements about $\boldsymbol{F}^{(0)}$ are updated as well for the update of next row. We present the process of updating $\boldsymbol{F}$ in two parts, namely the calculation of the value of the objective function and the updating of corresponding elements.

3.1.1. Calculate the objective function value

For $\boldsymbol{F}^{(0)}$, $f_{ip}^{(0)}=1$ while as for $\boldsymbol{F}^{(l)}$, $f_{il}^{(l)}=1$. Thus, when $l\neq p$, the pth and lth column are different for $\boldsymbol{F}^{(0)}$ and $\boldsymbol{F}^{(l)}$. When $l=p$, they are the same. Thus, the calculation of $J_1(\boldsymbol{F}^{(l)})$ is discussed in two cases.

Case 1: When $l\neq p$, the value of $J_1(\boldsymbol{F}^{(l)})$ is

$$J_1(\boldsymbol{F}^{(l)})=\frac{\left(\sum_{j=1}^{c}\mathbf{f}_j^{(l)T}\boldsymbol{B}^T\boldsymbol{B}\mathbf{f}_j^{(l)}\right)^2}{\sum_{j=1}^{c}\mathbf{f}_j^{(l)T}\mathbf{1}_n\mathbf{1}_n^T\mathbf{f}_j^{(l)}}. \tag{16}$$

We define $\Omega=\{j|j=1,2,\ldots,c\}$ and $\Omega^{(l)}=\{j|j=1,2,\ldots,c,j\neq l,j\neq p\}$ to express more clearly. Since the columns of $\boldsymbol{F}^{(0)}$ and $\boldsymbol{F}^{(l)}$ are the same expect lth and pth column, $J_1(\boldsymbol{F}^{(l)})$ can also be written as

$$\frac{\left(\sum_{j\in\Omega^{(l)}}\mathbf{f}_j^{(0)T}\boldsymbol{B}^T\boldsymbol{B}\mathbf{f}_j^{(0)}+\mathbf{f}_l^{(l)T}\boldsymbol{B}^T\boldsymbol{B}\mathbf{f}_l^{(l)}+\mathbf{f}_p^{(l)T}\boldsymbol{B}^T\boldsymbol{B}\mathbf{f}_p^{(l)}\right)^2}{\sum_{j\in\Omega^{(l)}}\mathbf{f}_j^{(0)T}\mathbf{1}_n\mathbf{1}_n^T\mathbf{f}_j^{(0)}+\mathbf{f}_l^{(l)T}\mathbf{1}_n\mathbf{1}_n^T\mathbf{f}_l^{(l)}+\mathbf{f}_p^{(l)T}\mathbf{1}_n\mathbf{1}_n^T\mathbf{f}_p^{(l)}}. \tag{17}$$

By observing $\mathbf{f}_l^{(l)}$ and $\mathbf{f}_l^{(0)}$, it is found that they only differ in the ith element, that is $\mathbf{f}_{il}^{(l)}=1$ and $\mathbf{f}_{il}^{(0)}=0$. Thus, $\mathbf{f}_l^{(l)}=\mathbf{f}_l^{(0)}+\mathbf{e}_i$, where $\mathbf{e}_i$ is a column vector with its ith element as 1 and others as 0. Then, for $\mathbf{f}_l^{(l)T}\boldsymbol{B}^T\boldsymbol{B}\mathbf{f}_l^{(l)}$ and $\mathbf{f}_l^{(l)T}\mathbf{1}_n\mathbf{1}_n^T\mathbf{f}_l^{(l)}$, we have

$$\begin{aligned}\mathbf{f}_l^{(l)T}\boldsymbol{B}^T\boldsymbol{B}\mathbf{f}_l^{(l)}&=\left(\mathbf{f}_l^{(0)}+\mathbf{e}_i\right)^T\boldsymbol{B}^T\boldsymbol{B}\left(\mathbf{f}_l^{(0)}+\mathbf{e}_i\right)\\&=\mathbf{f}_l^{(0)T}\boldsymbol{B}^T\boldsymbol{B}\mathbf{f}_l^{(0)}+2\mathbf{b}_i^T\boldsymbol{B}\mathbf{f}_l^{(0)}+\mathbf{b}_i^T\mathbf{b}_i.\end{aligned} \tag{18}$$

$$\begin{aligned}\mathbf{f}_l^{(l)T}\mathbf{1}_n\mathbf{1}_n^T\mathbf{f}_l^{(l)}&=\left(\mathbf{f}_l^{(0)}+\mathbf{e}_i\right)^T\mathbf{1}_n\mathbf{1}_n^T\left(\mathbf{f}_l^{(0)}+\mathbf{e}_i\right)\\&=\mathbf{f}_l^{(0)T}\mathbf{1}_n\mathbf{1}_n^T\mathbf{f}_l^{(0)}+2\mathbf{1}_n^T\mathbf{f}_l^{(0)}+1.\end{aligned} \tag{19}$$

Similarly, for $\mathbf{f}_p^{(l)}$ and $\mathbf{f}_p^{(0)}$, we have $\mathbf{f}_p^{(l)}=\mathbf{f}_p^{(0)}-\mathbf{e}_i$. Thus, for $\mathbf{f}_p^{(l)T}\boldsymbol{B}^T\boldsymbol{B}\mathbf{f}_p^{(l)}$ and $\mathbf{f}_p^{(l)T}\mathbf{1}_n\mathbf{1}_n^T\mathbf{f}_p^{(l)}$, we have

$$\begin{aligned}\mathbf{f}_p^{(l)T}\boldsymbol{B}^T\boldsymbol{B}\mathbf{f}_p^{(l)}&=\left(\mathbf{f}_p^{(0)}-\mathbf{e}_i\right)^T\boldsymbol{B}^T\boldsymbol{B}\left(\mathbf{f}_p^{(0)}-\mathbf{e}_i\right)\\&=\mathbf{f}_p^{(0)T}\boldsymbol{B}^T\boldsymbol{B}\mathbf{f}_p^{(0)}-2\mathbf{b}_i^T\boldsymbol{B}\mathbf{f}_p^{(0)}+\mathbf{b}_i^T\mathbf{b}_i.\end{aligned} \tag{20}$$

$$\begin{aligned}\mathbf{f}_p^{(l)T}\mathbf{1}_n\mathbf{1}_n^T\mathbf{f}_p^{(l)}&=\left(\mathbf{f}_p^{(0)}-\mathbf{e}_i\right)^T\mathbf{1}_n\mathbf{1}_n^T\left(\mathbf{f}_p^{(0)}-\mathbf{e}_i\right)\\&=\mathbf{f}_p^{(0)T}\mathbf{1}_n\mathbf{1}_n^T\mathbf{f}_p^{(0)}-2\mathbf{1}_n^T\mathbf{f}_p^{(0)}+1.\end{aligned} \tag{21}$$

Take Eqs. (18)–(21) into Eq. (17), $J_1(\boldsymbol{F}^{(l)})$ becomes

$$\frac{\left(\sum_{j\in\Omega}\mathbf{f}_j^{(0)T}\boldsymbol{B}^T\boldsymbol{B}\mathbf{f}_j^{(0)}+2\mathbf{b}_i^T\boldsymbol{B}\mathbf{f}_l^{(0)}-2\mathbf{b}_i^T\boldsymbol{B}\mathbf{f}_p^{(0)}+2\mathbf{b}_i^T\mathbf{b}_i\right)^2}{\sum_{j\in\Omega}\mathbf{f}_j^{(0)T}\mathbf{1}_n\mathbf{1}_n^T\mathbf{f}_j^{(0)}+2\mathbf{1}_n^T\mathbf{f}_l^{(0)}-2\mathbf{1}_n^T\mathbf{f}_p^{(0)}+2}. \tag{22}$$

Case 2: When $l=p$, the value of objective function is

$$J_1(\boldsymbol{F}^{(l)})=\frac{\left(\mathrm{Tr}\left(\boldsymbol{F}^{(0)T}\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{F}^{(0)}\right)\right)^2}{\mathrm{Tr}\left(\boldsymbol{F}^{(0)T}\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{F}^{(0)}\right)}. \tag{23}$$

We denote v_{il} as the value of objective function when we update ith row and lth column. Considering that when $l\neq p$, $f_{il}^{(0)}=0$ and when $l=p$, the value of $\bar{f}_{il}$ is 1, v_{il} can be written in a unified form:

$$\frac{\left(\mathrm{Tr}(\boldsymbol{F}^{(0)T}\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{F}^{(0)})+2\mathbf{b}_i^T(1-f_{il}^{(0)})(\boldsymbol{B}\mathbf{f}_l^{(0)}-\boldsymbol{B}\mathbf{f}_p^{(0)}+\mathbf{b}_i)\right)^2}{\mathrm{Tr}\left(\boldsymbol{F}^{(0)T}\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{F}^{(0)}\right)+2(1-f_{il}^{(0)})(\mathbf{1}_n^T\mathbf{f}_l^{(0)}-\mathbf{1}_n^T\mathbf{f}_p^{(0)}+1)}. \tag{24}$$

Several elements could be calculated and stored ahead of time, for example, $\mathrm{Tr}(\boldsymbol{F}^{(0)T}\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{F}^{(0)})$, $\mathrm{Tr}(\boldsymbol{F}^{(0)T}\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{F}^{(0)})$, $\boldsymbol{B}\boldsymbol{F}^{(0)}$, $\mathbf{1}_n^T\boldsymbol{F}^{(0)}$ and $\mathbf{b}_i^T\mathbf{b}_i, i=1,2,\ldots,n$. In this way, the calculation of v_{il} is simples, which directly takes from corresponding elements. Then $\mathbf{f}^i$ is determined by

$$f_{iq}=\left\langle q=\arg\max_{l'\in[1,c]}v_{il'}\right\rangle \tag{25}$$

where $\langle\cdot\rangle$ equals to 1 if the state is true, and 0 otherwise.

3.1.2. Update elements

From formula (24), there are five elements needed to be stored in advance, which are $\mathrm{Tr}(\boldsymbol{F}^{(0)T}\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{F}^{(0)})$, $\mathrm{Tr}(\boldsymbol{F}^{(0)T}\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{F}^{(0)})$, $\boldsymbol{B}\boldsymbol{F}^{(0)}$, $\mathbf{1}_n^T\boldsymbol{F}^{(0)}$ and $\mathbf{b}_i^T\mathbf{b}_i, i=1,2,\ldots,n$. Since $\mathbf{b}_i^T\mathbf{b}_i, i=1,2,\ldots,n$ is not changed at all, there is no need to update it. After obtaining the new $\mathbf{f}^i$, other four elements do not need to update as well if $q=p$. However, if $q\neq p$, the optimal $\boldsymbol{F}^{(q)}$ would replace $\boldsymbol{F}^{(0)}$ and these four elements should be updated for the update of next row. Based on the above analysis, we do not need to recalculate the values of these elements, which can be updated by pre-stored variables. The updating formulations are exhibited below

$$\begin{aligned}\mathrm{Tr}(\boldsymbol{F}^{(0)T}\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{F}^{(0)})\leftarrow{}&\mathrm{Tr}(\boldsymbol{F}^{(0)T}\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{F}^{(0)})+2\mathbf{b}_i^T\boldsymbol{B}\mathbf{f}_q^{(0)}\\&-2\mathbf{b}_i^T\boldsymbol{B}\mathbf{f}_p^{(0)}+2\mathbf{b}_i^T\mathbf{b}_i\end{aligned} \tag{26}$$

$$\begin{aligned}\mathrm{Tr}(\boldsymbol{F}^{(0)T}\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{F}^{(0)})\leftarrow{}&\mathrm{Tr}(\boldsymbol{F}^{(0)T}\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{F}^{(0)})+2\mathbf{1}_n^T\mathbf{f}_q^{(0)}\\&-2\mathbf{1}_n^T\mathbf{f}_p^{(0)}+2\end{aligned} \tag{27}$$

$$\begin{aligned}\boldsymbol{B}\boldsymbol{F}^{(0)}(:,q)&\leftarrow\boldsymbol{B}\boldsymbol{F}^{(0)}(:,q)+\boldsymbol{B}(:,i)\\\boldsymbol{B}\boldsymbol{F}^{(0)}(:,p)&\leftarrow\boldsymbol{B}\boldsymbol{F}^{(0)}(:,p)-\boldsymbol{B}(:,i)\end{aligned} \tag{28}$$

$$\begin{aligned}\mathbf{1}_n^T\boldsymbol{F}^{(0)}(q)&\leftarrow\mathbf{1}_n^T\boldsymbol{F}^{(0)}(q)+1\\\mathbf{1}_n^T\boldsymbol{F}^{(0)}(p)&\leftarrow\mathbf{1}_n^T\boldsymbol{F}^{(0)}(p)-1\end{aligned} \tag{29}$$

It is worth noting that when updating $\mathrm{Tr}(\boldsymbol{F}^{(0)T}\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{F}^{(0)})$ and $\mathrm{Tr}(\boldsymbol{F}^{(0)T}\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{F}^{(0)})$, several elements such as $\boldsymbol{B}\mathbf{f}_l^{(0)}$, $\boldsymbol{B}\mathbf{f}_p^{(0)}$, $\mathbf{1}_n\mathbf{1}_n^T$ and $\mathbf{1}_n^T\mathbf{f}_p$ can be taken from the corresponding values in $\boldsymbol{B}\boldsymbol{F}^{(0)}$ and $\mathbf{1}_n^T\boldsymbol{F}^{(0)}$ directly.

3.2. Optimization of FBMC2

By expanding and removing extraneous items, problem (10) becomes

$$\max J_2(\boldsymbol{F},\Lambda)=\max_{\boldsymbol{F}\in\mathrm{Ind},\Lambda}2\,\mathrm{Tr}\left(\boldsymbol{F}^T\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{F}\Lambda\right)-\mathrm{Tr}\left(\boldsymbol{F}\Lambda\boldsymbol{F}^T\boldsymbol{F}\Lambda\boldsymbol{F}^T\right) \tag{30}$$

where $J_2(\boldsymbol{F},\Lambda)$ is the value of objective function. Since Λ is diagonal matrix and $\boldsymbol{F}$ is discrete indicator matrix, $\mathrm{Tr}\left(\boldsymbol{F}\Lambda\boldsymbol{F}^T\boldsymbol{F}\Lambda\boldsymbol{F}^T\right)$ can be written as

$$\begin{aligned}\mathrm{Tr}\left(\boldsymbol{F}\Lambda\boldsymbol{F}^T\boldsymbol{F}\Lambda\boldsymbol{F}^T\right)&=\sum_{j=1}^{n}\sum_{r=1}^{n}\sum_{z=1}^{c}\sum_{t=1}^{c}f_{jz}\lambda_z f_{rz}f_{jt}\lambda_t f_{rt}\\&=\sum_{j=1}^{n}\sum_{r=1}^{n}\sum_{z=1}^{c}f_{jz}\lambda_z^2 f_{rz}=\sum_{z=1}^{c}\lambda_z^2\left(\sum_{j=1}^{n}f_{jz}\right)^2\\&=\mathrm{Tr}\left(\Lambda^2\boldsymbol{F}^T\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{F}\right).\end{aligned} \tag{31}$$

By substituting expression (31) into the original problem and simplifying, we can derive the summation form of problem (30)

$$J_2(\boldsymbol{F},\Lambda)=\sum_{j=1}^{c}\mathbf{f}_j^T\left(2\lambda_j\boldsymbol{B}^T\boldsymbol{B}-\lambda_j^2\mathbf{1}\mathbf{1}^T\right)\mathbf{f}_j. \tag{32}$$

For problem 32, by taking the partial derivative of λ_j and setting its value as 0, the expression for λ_j is

$$\lambda_j=\frac{\mathbf{f}_j^T\boldsymbol{B}^T\boldsymbol{B}\mathbf{f}_j}{\mathbf{f}_j^T\mathbf{1}_n\mathbf{1}_n^T\mathbf{f}_j}. \tag{33}$$

It can be seen that the number of samples within a cluster is too large, the corresponding λ_j is smaller, so as to reduce the influence of the cluster on the clustering result. Conversely, when the number of samples within a cluster is smaller, the corresponding λ_j is larger,

increasing the importance of that clustering result. By substituting λ_j in J_2, the expression of J_2 becomes

$$\max J_2(\boldsymbol{F}) = \max_{\boldsymbol{F}\in \text{Ind}} \sum_{j=1}^{c} \frac{\left(\mathbf{f}_j^T \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_j\right)^2}{\mathbf{f}_j^T \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_j}. \tag{34}$$

Similar to the optimization of problem (14), problem (34) is solved by CD and the same variables are utilized for consistency. The obtained indicator matrix is $\boldsymbol{F}^{(0)}$ and $f_{ip}^{(0)} = 1$. The ith row is being updated and there are c alternatives. The corresponding indicator matrices are $\boldsymbol{F}^{(1)}, \boldsymbol{F}^{(2)}, \ldots, \boldsymbol{F}^{(l)}, \ldots, \boldsymbol{F}^{(c)}$. For $\boldsymbol{F}^{(l)}$, we have $f_{il}^{(0)} = 1$ and other rows are the same with $\boldsymbol{F}^{(0)}$. Thus, the optimization of updating $\mathbf{f}^i$ is

$$\max_{\boldsymbol{F}\in\{\boldsymbol{F}^{(1)},\ldots,\boldsymbol{F}^{(l)},\ldots,\boldsymbol{F}^{(c)}\}} \sum_{j=1}^{c} \frac{\left(\mathbf{f}_j^T \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_j\right)^2}{\mathbf{f}_j^T \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_j}. \tag{35}$$

To obtain the optimal $\boldsymbol{F}$, the calculation of value of objective function is necessary. Fortunately, $J_2(\boldsymbol{F}^{(l)})$ could be calculated by the previous stored elements, significantly reducing computational overhead.

3.2.1. Calculate the objective function value

Considering that the calculation of value of objective function is different when $l \neq p$ and $l = p$, two cases are discussed here.

Case 1: When $l \neq p$, the value of $J_2(\boldsymbol{F}^{(l)})$ is

$$J_2(\boldsymbol{F}^{(l)}) = \sum_{j\in\Omega^{(l)}} \frac{\left(\mathbf{f}_j^{(l)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_j^{(l)}\right)^2}{\mathbf{f}_j^{(l)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_j^{(l)}} + \frac{\left(\mathbf{f}_l^{(l)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_l^{(l)}\right)^2}{\mathbf{f}_l^{(l)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_l^{(l)}} + \frac{\left(\mathbf{f}_p^{(l)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_p^{(l)}\right)^2}{\mathbf{f}_p^{(l)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_p^{(l)}}. \tag{36}$$

For $j \in \Omega^{(l)}$, $\mathbf{f}_j^{(l)} = \mathbf{f}_j^{(0)}$. While for $\mathbf{f}_l^{(l)}$ and $\mathbf{f}_p^{(l)}$, we have $\mathbf{f}_l^{(l)} = \mathbf{f}_l^{(0)} + e_i$ and $\mathbf{f}_p^{(l)} = \mathbf{f}_p^{(0)} - e_i$. Hence, Eqs. (18)–(21) are also suited to the expression of $J_2(\boldsymbol{F}^{(l)})$. Substituting them into formulation (36), the new expression of $J_2(\boldsymbol{F}^{(l)})$ is

$$\begin{aligned} &\sum_{j\in\Omega^{(l)}} \frac{\left(\mathbf{f}_j^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_j^{(0)}\right)^2}{\mathbf{f}_j^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_j^{(0)}} + \frac{\left(\mathbf{f}_l^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_l^{(0)} + 2\mathbf{b}_i^T \boldsymbol{B}\mathbf{f}_l^{(0)} + \mathbf{b}_i^T \mathbf{b}_i\right)^2}{\mathbf{f}_l^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_l^{(0)} + 2\mathbf{1}_n^T \mathbf{f}_l^{(0)} + 1} \\ &+ \frac{\left(\mathbf{f}_p^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_p^{(0)} - 2\mathbf{b}_i^T \boldsymbol{B}\mathbf{f}_p^{(0)} + \mathbf{b}_i^T \mathbf{b}_i\right)^2}{\mathbf{f}_p^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_p^{(0)} - 2\mathbf{1}_n^T \mathbf{f}_p^{(0)} + 1}. \end{aligned} \tag{37}$$

Case 2: When $l = p$, the value of objective function is

$$\sum_{j\in\Omega} \frac{\left(\mathbf{f}_j^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_j^{(0)}\right)^2}{\mathbf{f}_j^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_j^{(0)}}. \tag{38}$$

Then, the increment of the values of the objective function is calculated and we denote it as $\bar{\delta}_{il}$. When $l = p$, the value of $\bar{\delta}_{il}$ is 0. When $l \neq p$, the formulation of $\bar{\delta}_{il}$ is

$$\begin{aligned} &\frac{\left(\mathbf{f}_l^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_l^{(0)} + 2\mathbf{b}_i^T \boldsymbol{B}\mathbf{f}_l^{(0)} + \mathbf{b}_i^T \mathbf{b}_i\right)^2}{\mathbf{f}_l^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_l^{(0)} + 2\mathbf{1}_n^T \mathbf{f}_l^{(0)} + 1} \\ &+ \frac{\left(\mathbf{f}_p^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_p^{(0)} - 2\mathbf{b}_i^T \boldsymbol{B}\mathbf{f}_p^{(0)} + \mathbf{b}_i^T \mathbf{b}_i\right)^2}{\mathbf{f}_p^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_p^{(0)} - 2\mathbf{1}_n^T \mathbf{f}_p^{(0)} + 1} \\ &- \frac{\left(\mathbf{f}_l^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_l^{(0)}\right)^2}{\mathbf{f}_l^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_l^{(0)}} - \frac{\left(\mathbf{f}_p^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_p^{(0)}\right)^2}{\mathbf{f}_p^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_p^{(0)}}. \end{aligned} \tag{39}$$

Observing that the second and fourth terms of Eq. (39) are constant when we update $\mathbf{f}^i$, we subtract the constant $\frac{\left(\mathbf{f}_p^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_p^{(0)} - 2\mathbf{b}_i^T \boldsymbol{B}\mathbf{f}_p^{(0)} + \mathbf{b}_i^T \mathbf{b}_i\right)^2}{\mathbf{f}_p^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_p^{(0)} - 2\mathbf{1}_n^T \mathbf{f}_p^{(0)} + 1} - \frac{\left(\mathbf{f}_p^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_p^{(0)}\right)^2}{\mathbf{f}_p^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_p^{(0)}}$ for each $\bar{\delta}_{il}$ and there is no effect on the relative value of the increments. The new relative increment is name δ_{il} and it can be written in a uniform form by utilizing the value of $f_{il}^{(0)}$, which is

$$\begin{aligned} &\frac{\left(\mathbf{f}_l^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_l^{(0)} + 2(1 - f_{il}^{(0)})\mathbf{b}_i^T \boldsymbol{B}\mathbf{f}_l^{(0)} + (1 - f_{il}^{(0)})\mathbf{b}_i^T \mathbf{b}_i\right)^2}{\mathbf{f}_l^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_l^{(0)} + 2(1 - f_{il}^{(0)})\mathbf{1}_n^T \mathbf{f}_l^{(0)} + (1 - f_{il}^{(0)})} \\ &- \frac{\left(\mathbf{f}_l^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_l^{(0)} - 2f_{il}^{(0)}\mathbf{b}_i^T \boldsymbol{B}\mathbf{f}_l^{(0)} + f_{il}^{(0)}\mathbf{b}_i^T \mathbf{b}_i\right)^2}{\mathbf{f}_l^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_l^{(0)} - 2f_{il}^{(0)}\mathbf{1}_n^T \mathbf{f}_l^{(0)} + f_{il}^{(0)}}. \end{aligned} \tag{40}$$

After calculating δ_{il}, the $\mathbf{f}_i$ could be obtained by

$$f_{iq} = \left\langle q = \arg\max_{l'\in[1,c]} \delta_{il'} \right\rangle \tag{41}$$

where $\langle\cdot\rangle$ equals to 1 if the state is true, and 0 otherwise.

3.2.2. Update elements

From the expression of δ_{il}, it can be seen that several elements need to be calculated in advance for instance $\mathbf{f}_l^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_l^{(0)}, l = 1, 2, \ldots, c$, $\mathbf{f}_l^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_l^{(0)}, l = 1, 2, \ldots, c$, $\boldsymbol{BF}$, $\mathbf{1}_n^T \boldsymbol{F}^{(0)}$ and $\mathbf{b}_i^T \mathbf{b}_i, i = 1, 2, \ldots, n$. When $q = p$, these for elements remains the same. When $q \neq p$, the values of these elements are needed to update for the calculation of next row. The update of $\boldsymbol{BF}^{(0)}$ and $\mathbf{1}_n^T \boldsymbol{F}^{(0)}$ are Eqs. (28) and (29). The update of the others are shown below

$$\begin{aligned} &\mathbf{f}_q^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_q^{(0)} \leftarrow \mathbf{f}_q^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_q^{(0)} + 2\mathbf{b}_i^T \boldsymbol{B}\mathbf{f}_q^{(0)} + \mathbf{b}_i^T \mathbf{b}_i \\ &\mathbf{f}_p^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_p^{(0)} \leftarrow \mathbf{f}_p^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\mathbf{f}_p^{(0)} - 2\mathbf{b}_i^T \boldsymbol{B}\mathbf{f}_p^{(0)} + \mathbf{b}_i^T \mathbf{b}_i; \end{aligned} \tag{42}$$

$$\begin{aligned} &\mathbf{f}_q^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_q^{(0)} \leftarrow \mathbf{f}_q^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_q^{(0)} + 2\mathbf{1}_n^T \mathbf{f}_q^{(0)} + 1 \\ &\mathbf{f}_p^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_p^{(0)} \leftarrow \mathbf{f}_p^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \mathbf{f}_p^{(0)} - 2\mathbf{1}_n^T \mathbf{f}_p^{(0)} + 1. \end{aligned} \tag{43}$$

In all, the calculation of objective function value and update of elements can be accessed by simple addition and subtraction. We summarize the solution to problem (9) in Algorithm 1. The solution process for problem (10) is similar to that of problem (9), and we do not display it here.

Algorithm 1 Optimization algorithm for problem (9)

Input: Bipartite graph $\boldsymbol{B} \in \mathbb{R}^{m\times n}$, cluster number c.
Initialize: Label matrix $\boldsymbol{F}^{(0)}$.

1: Compute and store variables $\text{Tr}(\boldsymbol{F}^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\boldsymbol{F}^{(0)})$, $\text{Tr}(\boldsymbol{F}^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \boldsymbol{F}^{(0)})$, $\boldsymbol{BF}^{(0)}$, $\mathbf{1}_n^T \boldsymbol{F}^{(0)}$ and $\mathbf{b}_i^T \mathbf{b}_i, i = 1, 2, ..., n$.
2: **while** not convergence **do**
3: **for** $i = 1$ to n **do**
4: Calculate $v_{il}, l = 1, 2, \cdots, c$ according to Eq. (24);
5: If $\mathbf{f}^i$ is changed, update variables according to Eq. (26)-(29).
6: **end for**
7: **end while**

Output: The discrete indicator matrix $\boldsymbol{F}$.

3.3. Complexity analysis

The computational cost of FBMC is analyzed here. First, for dataset $\boldsymbol{X} \in \mathbb{R}^{d\times n}$, m anchors are selected by KM, in which the computational cost is $\mathcal{O}(ndmt_1)$ and t_1 is the number of iteration. Then, the bipartite graph is constructed by Eq. (1) and the time cost is $\mathcal{O}(nmd + nmlog(m))$. Hence, the cost of computing $\boldsymbol{B}$ is $\mathcal{O}(ndm(t_1 + 1) + nmlog(m))$.

For FBMC1, five elements are needed to be calculated and stored in advance, which is shown in Algorithm 1. For $\text{Tr}(\boldsymbol{F}^{(0)T} \boldsymbol{B}^T \boldsymbol{B}\boldsymbol{F}^{(0)})$ and $\text{Tr}(\boldsymbol{F}^{(0)T} \mathbf{1}_n \mathbf{1}_n^T \boldsymbol{F}^{(0)})$, the cost is $\mathcal{O}(c^2 m)$ and $\mathcal{O}(c^2)$ respectively. Since $\boldsymbol{F}^{(0)}$ is discrete, calculating $\boldsymbol{BF}^{(0)}$ and $\mathbf{1}_n^T \boldsymbol{F}^{(0)}$ needs mn and n additions, which could be omitted compared with multiplications. As for $\mathbf{b}_i^T \mathbf{b}_i, i = 1, 2, \ldots, n$, it costs $\mathcal{O}(mn)$. The aboves variables only needs to be calculated once. Calculating v_{il} takes $\mathcal{O}(m)$ times multiplications. As for the update of pre-stored elements, it needs additions only. All in all, the time cost of $\boldsymbol{F}$ is $\mathcal{O}(nmct_1)$, where t_1 is the number of iteration. For FBMC2, five elements need to be calculated and stored, the time cost

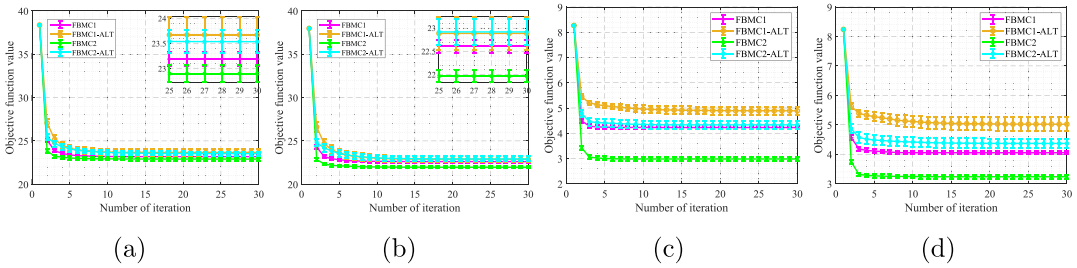

Fig. 1. The change of objective function value with the number of iteration. (a) COIL20. (b) COIL20_0.01. (c) MSRA25. (d) MSRA25_0.01.

Table 1
Comparison for SBMC, EBMC and our proposed method.

Dataset	Ratio of objective values(%)		Ratio of running time(%)		Ratio of ACC%)		Ratio of NMI(%)	
	$\frac{FBMC1}{SBMC}$	$\frac{FBMC2}{EBMC}$	$\frac{FBMC1}{SBMC}$	$\frac{FBMC2}{EBMC}$	$\frac{FBMC1}{SBMC}$	$\frac{FBMC2}{EBMC}$	$\frac{FBMC1}{SBMC}$	$\frac{FBMC2}{EBMC}$
Yale32	94.06	80.11	17.82	31.79	101.48	108.52	102.51	106.88
ORL	69.65	74.52	24.32	50.86	100.00	122.14	100.00	126.99
MSRA25	75.38	74.14	09.66	25.86	109.26	100.53	110.41	103.05
USPS20	89.13	91.25	10.77	32.60	127.42	123.66	126.57	126.26
USPST	90.88	94.90	12.15	32.09	127.61	120.78	125.86	122.67
Segment	98.80	97.19	07.24	17.26	132.35	101.77	147.12	86.30
Isolet5	79.29	93.07	25.94	20.20	136.13	102.94	158.78	99.94
Letter	80.84	93.09	29.75	10.71	206.20	102.92	231.80	109.90

of which is $\mathcal{O}(mn)$ in all. Supposing the number of iteration is t_2, the total cost of updating the discrete indicator matrix is $\mathcal{O}(mnct_2)$. Thus, the computational cost of FBMC is linear with the number of samples.

4. Experimental results

4.1. Comparison with two step approach

To verify the one-step approach yields a better solution, the two-stage optimization method is compared. We name the two stage methods as FBMC1-ALT and FBMC2-ALT respectively, in which two variables are updated alternatively. λ and Λ are solved by taking derivatives of the objective function, while $\boldsymbol{F}$ is solved by CD method. Two image datasets are applied in the experiment, which are COIL20 and MSRA25. We also add two different type of Gaussian noise on the image. The Gaussian noise has a mean of zero and a variance of 0.01. To reduce randomness, each method is repeated for 100 times with each experiment differing only in its initialization. Through this experiment, the bipartite graph $\boldsymbol{B}$ remains the same and the stopping condition for iterations is set at a maximum of 30 iterations. The mean and variance of the objective function values for each number of iterations are visualized, which is displayed in Fig. 1. From Fig. 1, it is evident that FBMC1 produces solutions with smaller objective function values compared to FBMC-ALT. Similarly, FBMC2 outperforms FBMC2-ALT in terms of minimizing the objective function, as illustrated in Figs. 1(a)–1(d). Furthermore, it is worth noting that both FBMC2 and FBMC2-ALT achieve lower objective function values than FBMC1 and FBMC1-ALT respectively, especially in Figs. 1(c) and 1(d). This indicates that increasing balance factors allows indicator matrix to better approximate similarity graph.

4.2. Comparison with SBMC and EBMC

SBMC and EBMC are two models that are closely related to FBMC1 and FBMC2. The difference is the optimization method. In this section, we adopt the eight datasets in EBMC [37]. For each dataset, 100 times experiments are conducted and convergence criteria set to a relative error of less than 10^{-8}. The convergence objective values, running time, average clustering accuracy (ACC) and normalized mutual information (NMI) are recorded, the ratio of which are shown in Table 1. For these four ratio values, the smaller ratio signifies a lower objective function value achieved by our proposed method, along with less time consumption. While a larger ratio for ACC and NMI indicates better clustering results. As shown in Table 1, our proposed algorithm achieves lower objective function values, shorter runtimes, and higher clustering results in nearly all cases. These suggest superior clustering performance of our method.

4.3. Experiment setup

To demonstrate the effectiveness of our proposed methods, experiments are carried out on eight benchmark datasets, which are caltech101-silhouettes-16, caltech101-silhouettes-28,[1] COIL100, Connect4,[2] Isolet5, PalmData25,[3] Protein[4] and extended Yale face database B (YaleB) datasets. The COIL100, Isolte5 and YaleB datasets could be find here.[5] For convenience, the name of caltech101-silhouettes-16, caltech101-silhouettes-28 and PalmData25 are abbreviated as CAL16, CAL28 and PAL25. The features are all normalized to mitigate the impact of scale differences. The detailed information is displayed in Table 2. The compared methods are KM [9], NCUT [12], LSC [22], Nystrom [42], FSC [27], LABIN [23], USPEC [29] and FDBC [28]. For NCUT, the affinity is constructed by Gaussian function, in which σ is set as the average Euclidean distance of the data points and k is varied in range of $\{10, 20, \ldots, 60\}$. The other methods are based on anchors or representative points. For these methods, the generation method of anchor points and parameters setting are set according to the corresponding paper. The number of anchor points is selected from the range $\{2^7, 2^8, 2^9, 2^{10}\}$. For our method, k is selected from $\{5, 10, \ldots, 50\}$. Each experiment run 10 times in order to avoid randomness. Grid search is employed to seek the optimal parameters, and the clustering results along with their corresponding standard deviations are recorded. We analyze the sensitivity regarding the number of nearest neighbors and anchors. Subsequently, we compare the clustering performance across eight datasets. The convergence and running time are also evaluated to provide a comprehensive assessment.

[1] https://data.caltech.edu/records/mzrjq-6wc02.
[2] https://archive.ics.uci.edu/dataset/26/connect+4.
[3] https://www.scholat.com/xjchensz.
[4] https://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/.
[5] http://www.cad.zju.edu.cn/home/dengcai/Data/data.html.

Table 2
Data sets description for Number of instances, dimensions and clusters.

Data sets	Instances	Dimensions	Clusters	Data sets	Instances	Dimensions	Clusters
CAL16	8641	256	101	CAL28	8641	784	101
COIL100	7200	1024	100	Connect4	67 557	126	3
Isolet5	7797	617	26	PAL25	2000	256	100
Protein	24 387	357	3	YaleB	2414	1024	38

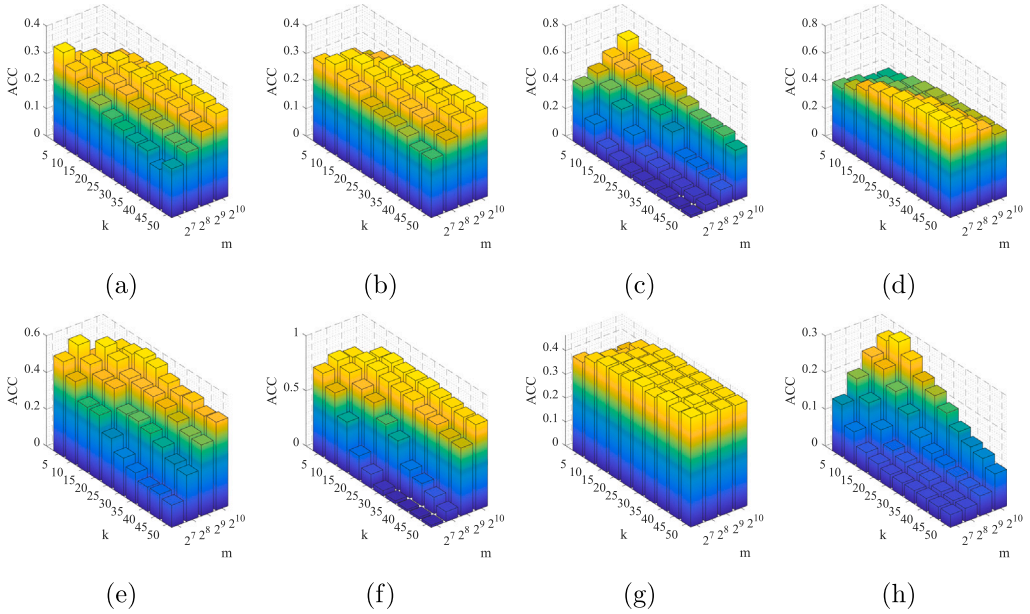

Fig. 2. ACC varies with No. of anchor points and neighbors for FBMC1. (a) CAL16. (b) CAL28. (c) COIL100. (d) Connect4. (e) ISO5. (f) PAL25. (g) Protein. (h) YaleB.

4.4. Sensitivity analysis

In the constructing of bipartite graph, two parameters which are number of nearest neighbor k and anchor points m are included. We analyze the variation of clustering accuracy of the two methods with these two parameters. Figs. 2 and 3 show the sensitivity results for FBMC1 and FBMC2 respectively. For all datasets, k is varied in $\{5, 10, 20, 30, \ldots, 50\}$ and m is in the range of $\{2^7, 2^8, 2^9, 2^{10}\}$. Form Figs. 2 and 3, it can be observed that on the same dataset, the influence of k and m on clustering accuracy is similar for both FBMC1 and FBMC2. However, there are some differences in details. For instance, on YaleB dataset, FBMC1 achieves the maximum accuracy when the value of k is 10 and the number of anchors is 2^{10}. While for FBMC2, the maximum accuracy is achieved when the number of neighbors is 5. For CAL16, CAL28, Isolet5, and PALM25 datasets, the variation of clustering accuracy with k and m is similar. On Protein dataset, m and k show less sensitivity to clustering performance. When the number of anchors is small, the clustering accuracy achieves better with a small k. Conversely, when the number of anchors is large, a higher value of k is needed to achieve a higher clustering accuracy. In reality, for datasets with fewer than 2000 samples, the number of anchors can be set to half the number of samples. For datasets larger than 2000 samples, the number of anchors can be defined as 1024. The choice of the number of nearest neighbors can range from 5 to 50, with grid search needed for optimal results.

4.5. Clustering performance analysis

The average clustering performance is displayed in Tables 3 and 4. For each dataset, the highest value of ACC or NMI value is bolded, and the second highest is underlined. As can be observed from the clustering results, on most datasets, most graph-based methods outperform KM, which suggests that by utilizing affinity graph, clustering methods can better capture the structure of the data. Furthermore, the NCUT method, utilizing the full-connected affinity graph, achieves optimal clustering accuracy on CAL16, PAL25 and YaleB datasets. While on other datasets, some anchor-based methods could surpass the performance of NCUT. The performance of LSC and Nystrom is inferior to that of NCUT and our proposed methods. FDBC utilizes a bipartite graph to construct the full graph and solves the discrete indicator matrix directly using coordinate descent. Its performance is decent, but still not as good as the proposed approach. In general, the proposed methods have better clustering performance on most datasets.

4.6. Convergence analysis

The convergence behavior is explored in this subsection. We plot the objective function values of problem (14) and (34) for FBMC1 and FBMC2 respectively. The trend of objective function values with the number of iterations is shown in Fig. 4. The iteration stops when the relative error of the objective function value being less than 10^{-8} and the maximum number of iterations is set to 30. From the convergence curve, it can be seen that on most datasets, our methods converge within 30 steps. On the PAL25 and Protein datasets, even both methods can achieve convergence within 10 steps. For FBMC1 and FBMC2, the initialization of indicator matrix is the same and it can be seen that the converge value of FBMC2 is higher than that of FBMC1.

4.7. Time analysis

An examination of the running time for comparison approaches on benchmark datasets is conducted. The number of nearest neighbors and anchor points are set to 10 and 1024 respectively. The stopping criterion for our methods and LABIN is that the absolute error of the

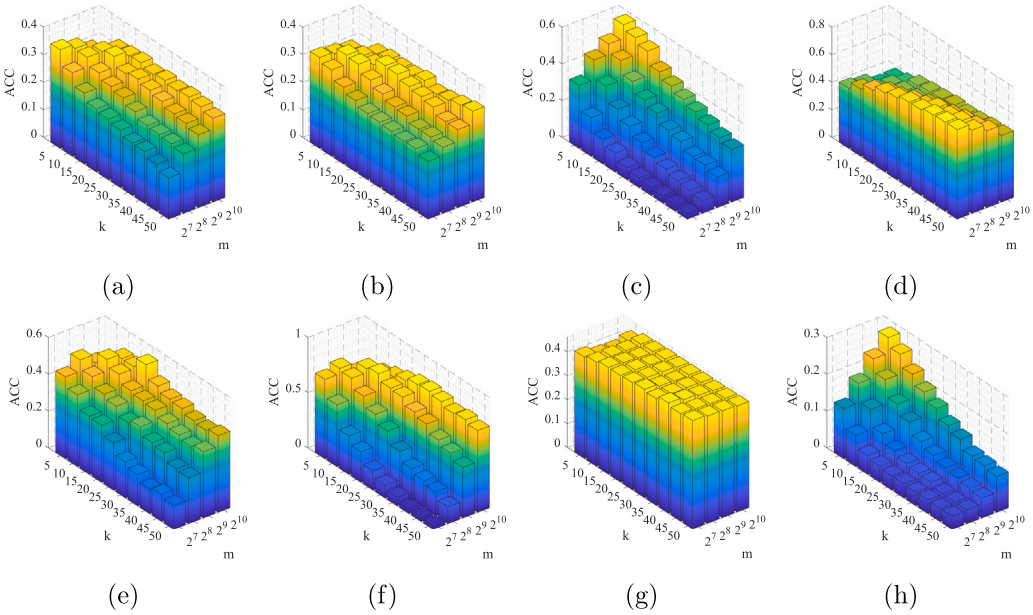

Fig. 3. ACC varies with No. of anchor points and neighbors for FBMC2. (a) CAL16. (b) CAL28. (c) COIL100. (d) Connect4. (e) ISO5. (f) PAL25. (g) Protein. (h) YaleB.

Table 3
ACC ± Standard deviation (%) of compared methods on benchmark datasets.

Methods	CAL16	CAL28	COIL100	Connect4	ISO5	PAL25	Protein	YaleB
KM	29.13 ± 0.46	26.42 ± 0.65	43.01 ± 2.20	53.82 ± 9.02	51.35 ± 2.84	71.04 ± 2.72	44.88 ± 0.90	10.00 ± 0.36
NCUT	25.91 ± 0.56	27.21 ± 0.69	60.89 ± 1.20	60.83 ± 0.53	53.93 ± 2.18	**87.05** ± 1.48	45.29 ± 0.00	**29.46** ± 1.27
LSC	25.37 ± 0.46	26.42 ± 0.65	54.97 ± 1.33	61.47 ± 7.25	55.27 ± 2.76	73.71 ± 1.03	43.37 ± 1.05	21.88 ± 0.90
Nystrom	22.43 ± 0.45	23.42 ± 0.60	44.18 ± 1.59	46.27 ± 11.14	54.98 ± 2.30	74.22 ± 2.36	43.83 ± 0.02	16.03 ± 1.07
FSC	31.81 ± 1.55	31.53 ± 0.57	55.62 ± 1.69	65.03 ± 0.31	54.05 ± 3.11	76.44 ± 1.36	43.94 ± 0.03	15.77 ± 1.11
LABIN	33.92 ± 1.82	30.00 ± 1.12	45.62 ± 1.44	59.62 ± 2.55	55.80 ± 3.38	73.91 ± 2.07	43.16 ± 1.17	23.15 ± 1.13
USPEC	26.53 ± 0.73	26.54 ± 0.74	57.48 ± 1.13	**65.75** ± 0.25	55.93 ± 2.30	81.49 ± 2.22	44.73 ± 0.56	22.95 ± 1.48
FDBC	34.15 ± 0.41	33.35 ± 0.12	47.15 ± 1.30	55.03 ± 2.06	54.94 ± 2.00	76.56 ± 2.70	44.65 ± 0.65	26.31 ± 0.62
FBMC1	34.25 ± 0.68	33.77 ± 0.93	**64.73** ± 0.76	65.23 ± 0.24	**57.84** ± 4.51	83.80 ± 0.59	**46.36** ± 0.01	28.74 ± 1.13
FBMC2	**35.74** ± 0.58	**34.58** ± 0.48	57.91 ± 1.49	64.81 ± 0.38	54.61 ± 3.51	81.01 ± 2.15	46.20 ± 0.37	28.31 ± 1.78

Table 4
NMI ± Standard deviation (%) of compared methods on benchmark datasets.

Methods	CAL16	CAL28	COIL100	Connect4	ISO5	PAL25	Protein	YaleB
KM	53.10 ± 0.42	53.02 ± 0.24	71.85 ± 0.81	0.07 ± 0.05	68.75 ± 0.97	89.66 ± 0.96	0.23 ± 0.17	13.94 ± 0.43
NCUT	50.39 ± 0.28	51.62 ± 0.33	**83.18** ± 0.41	0.15 ± 0.05	72.94 ± 1.16	**95.70** ± 0.32	**0.81** ± 0.03	37.20 ± 0.49
LSC	50.34 ± 0.33	51.17 ± 0.19	76.67 ± 0.40	0.13 ± 0.05	72.38 ± 0.62	89.67 ± 1.39	0.41 ± 0.06	30.81 ± 0.54
Nystrom	48.18 ± 0.22	49.25 ± 0.38	72.50 ± 0.65	0.03 ± 0.02	68.47 ± 1.04	92.38 ± 0.93	0.72 ± 0.77	26.28 ± 1.35
FSC	50.71 ± 0.40	51.31 ± 0.22	78.26 ± 0.73	**0.21** ± 0.05	72.22 ± 1.38	92.00 ± 0.35	0.34 ± 0.01	23.41 ± 0.77
LABIN	52.52 ± 3.10	52.28 ± 1.72	71.42 ± 2.04	0.18 ± 0.05	71.38 ± 2.22	90.47 ± 0.78	0.75 ± 0.29	29.48 ± 1.97
USPEC	50.90 ± 0.40	51.54 ± 0.37	77.93 ± 0.50	0.11 ± 0.07	72.69 ± 1.01	93.94 ± 0.69	0.45 ± 0.09	30.92 ± 1.14
FDBC	54.55 ± 0.12	55.35 ± 0.17	73.17 ± 1.27	**0.21** ± 0.04	71.70 ± 2.20	92.34 ± 1.18	0.66 ± 0.01	34.11 ± 0.61
FBMC1	53.09 ± 0.28	54.34 ± 0.14	82.57 ± 0.35	**0.21** ± 0.06	**73.02** ± 1.50	94.64 ± 0.30	0.49 ± 0.18	**37.99** ± 0.54
FBMC2	**54.75** ± 0.45	**55.97** ± 0.45	79.28 ± 0.68	0.17 ± 0.02	70.88 ± 2.46	94.01 ± 2.44	0.50 ± 0.26	35.35 ± 1.00

objective function is less than 10^{-8}. The maximum number of iteration is set as 30. Each algorithm runs 10 times and the average running time is recorded in Table 5. For each dataset, except for KM, the method with lowest time in the table is bolded, and the second lowest is underlined. KM shows shorter runtime because it is based on partitioning and does not require affinity graph. Compared with other graph-based methods, the proposed methods demonstrate higher time efficiency on the majority of datasets. FSC improves computational efficiency through anchor graph construction and singular value decomposition, which is still time-consuming. In comparison to LABIN and NCUT, FBMC1 and FBMC2 is computationally less time-consuming. Although FDBC exhibits better time performance than the proposed method on some datasets, it still falls short on large-scale datasets compared to our proposed approach.

4.8. Discussion

From the experimental results above, we can derive some useful key points. Firstly, when the objective function contains multiple variables, we can seek to transform these multiple variables into a single variable for solving. Comparisons between single-variable optimization and alternative optimization show that solving the optimization problem with a single variable could yield better solutions. Some works related to KM have also demonstrated this. Specifically, Nie *et al.* [30] represent the cluster center matrix using indicator matrix, transforming the problem into a single-variable issue regarding the indicator matrix, which improves the performance of the original method. This provides us with another approach for solving optimization problems in the

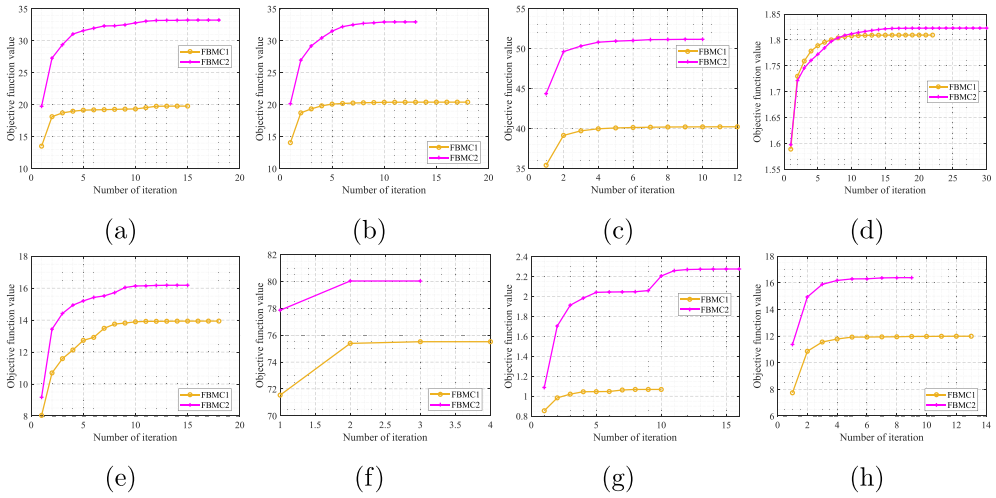

Fig. 4. The change of objective function value with the number of iteration. (a) CAL16. (b) CAL28. (c) COIL100. (d) Connect4. (e) ISO5. (f) PAL25. (g) Protein. (h) YaleB.

Table 5
Running time (s) of different methods. # means out of memory error.

Methods	CAL16	CAL28	COIL100	Connect4	ISO5	PAL25	Protein	YaleB
KM	0.54	1.66	1.32	0.99	0.53	0.05	0.88	0.22
NCUT	152.92	306.10	237.57	#	195.13	3.48	2190.40	23.76
FSC	17.21	29.66	25.22	292.66	22.70	1.40	51.36	4.52
LABIN	45.47	54.15	44.79	1216.90	31.83	4.40	102.83	8.47
FDBC	**6.64**	**9.07**	**7.54**	77.58	6.40	1.10	17.83	2.69
FBMC1	9.12	11.09	9.55	**39.24**	6.21	0.93	**12.43**	**1.91**
FBMC2	7.77	11.81	9.41	43.80	**5.27**	**0.82**	12.92	1.99

future. Additionally, for solving discrete constraints, coordinate descent method and acceleration strategies are quite effective, generally exhibiting a linear time complexity with respect to the sample size. This also results in a significant improvement in the computational speed of the proposed algorithms. Although not all metrics achieved optimal results, the performance of FBMC1 and FBMC2 is satisfying. Therefore, the proposed method strikes a balance between effectiveness and efficiency.

5. Conclusion and future work

In this paper, we have proposed a fast adaptively balanced min-cut (FBMC) clustering method, which aims to maximize the intra-cluster similarity and balanced clustering results. Two specific approaches are proposed name FBMC1 and FBMC2 motivated by SBMC and EBMC. Compared with min-cut clustering, FBMC has no trivial solutions and the complexity is linear with time. Besides, the optimization algorithms of FBMC1 and FBMC2 is one-step, in which there is no additional hyper-parameters. What is more, the discrete indicator matrix could be solved directly by CD method without any post-processing. The comprehensive experimental results demonstrate the superiority of FBMC1 and FBMC2. However, when constructing bipartite graphs in FBMC1 and FBMC2, the number of neighbors is required as a known parameter. Determining this parameter quickly and accurately is a direction for future research. Additionally, in both methods, after constructing the bipartite graph, it is still necessary to construct a fully connected graph, which may be unnecessary. In the future, we will focus on directly digging out clustering information from the bipartite graph. Additionally, the proposed method has a significant advantage in running time and the next step we will focus on achieving optimal performance as well. Last, we will evaluate other performance factors, such as fairness or robustness, in comparison to state-of-the-art approaches.

CRediT authorship contribution statement

Feiping Nie: Supervision, Project administration, Methodology, Conceptualization. **Fangyuan Xie:** Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Conceptualization. **Jingyu Wang:** Supervision, Methodology, Funding acquisition, Conceptualization. **Xuelong Li:** Supervision, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

This work was supported in part by the “Ye Qisun” Science Foundation of National Natural Science Foundation of China under Grant U2341224, in part by the Aeronautical Science Foundation of China under Grant 20230001053007 and in part by the National Natural Science Foundation of China under Grant 62236001.

References

[1] Z. Chen, Z. Fan, Y. Chen, Y. Zhu, Camera-aware cluster-instance joint online learning for unsupervised person re-identification, Pattern Recognit. 151 (2024) 110359.

[2] S. Chakraborty, S. Das, Detecting meaningful clusters from high-dimensional data: A strongly consistent sparse center-based clustering approach, IEEE Trans. Pattern Anal. Mach. Intell. 44 (6) (2022) 2894–2908.

[3] J. Liang, J. Yang, M.-M. Cheng, P.L. Rosin, L. Wang, Simultaneous subspace clustering and cluster number estimating based on triplet relationship, IEEE Trans. Image Process. 28 (8) (2019) 3973–3985.

[4] H. Zhao, F. Zhou, L. Bruzzone, R. Guan, C. Yang, Superpixel-level global and local similarity graph-based clustering for large hyperspectral images, IEEE Trans. Geosci. Remote Sens. 60 (2022) 1–16.

[5] Z. Yang, H. Niu, X. Wang, L. Fan, A segmentation method based on the deep fuzzy segmentation model in combined with SCANDLE clustering, Pattern Recognit. 146 (2024) 110027.

[6] H. Chen, T. Xie, M. Liang, W. Liu, P.X. Liu, A local tangent plane distance-based approach to 3D point cloud segmentation via clustering, Pattern Recognit. 137 (2023) 109307.

[7] L. Zhang, J. Xu, Y. Gong, L. Yu, J. Zhang, J. Shen, Unsupervised image and text fusion for travel information enhancement, IEEE Trans. Multimedia 24 (2022) 1415–1425.

[8] X. Tang, C. Dong, W. Zhang, Contrastive author-aware text clustering, Pattern Recognit. 130 (2022) 108787.

[9] J.A.H.A. Wong, Algorithm AS 136: A K-means clustering algorithm, J. R. Stat. Soc. 28 (1) (1979) 100–108.

[10] E.L. Johnson, A. Mehrotra, G.L. Nemhauser, Min-cut clustering, Math. Programm. 62 (1) (1993) 133–151.

[11] F. Nie, F. Xie, W. Yu, X. Li, Parameter-insensitive min cut clustering with flexible size constrains, IEEE Trans. Pattern Anal. Mach. Intell. (2024) 1–14.

[12] U. von Luxburg, A tutorial on spectral clustering, Stat. Comput. 17 (4) (2007) 395–416.

[13] G. Yang, S. Deng, X. Chen, C. Chen, Y. Yang, Z. Gong, Z. Hao, RESKM: A general framework to accelerate large-scale spectral clustering, Pattern Recognit. 137 (2023) 109275.

[14] L. Ding, C. Li, D. Jin, S. Ding, Survey of spectral clustering based on graph theory, Pattern Recognit. 151 (2024) 110366.

[15] C. Gao, W. Chen, F. Nie, W. Yu, Z. Wang, Spectral clustering with linear embedding: A discrete clustering method for large-scale data, Pattern Recognit. 151 (2024) 110396.

[16] J. Wang, Z. Ma, F. Nie, X. Li, Progressive self-supervised clustering with novel category discovery, IEEE Trans. Cybern. (2021) 1–14.

[17] G. Zhong, C.-M. Pun, Improved normalized cut for multi-view clustering, IEEE Trans. Pattern Anal. Mach. Intell. (2021) 1.

[18] Z. Wang, Z. Li, R. Wang, F. Nie, X. Li, Large graph clustering with simultaneous spectral embedding and discretization, IEEE Trans. Pattern Anal. Mach. Intell. 43 (12) (2021) 4426–4440.

[19] P. Zhou, J. Chen, L. Du, X. Li, Balanced spectral feature selection, IEEE Trans. Cybern. (2022) 1–13.

[20] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, E.Y. Chang, Parallel spectral clustering in distributed systems, IEEE Trans. Pattern Anal. Mach. Intell. 33 (3) (2011) 568–586.

[21] H. Jia, Q. Ren, L. Huang, Q. Mao, L. Wang, H. Song, Large-scale non-negative subspace clustering based on Nyström approximation, Inform. Sci. 638 (2023) 118981.

[22] X. Chen, D. Cai, Large scale spectral clustering with landmark-based representation, in: W. Burgard, D. Roth (Eds.), Proceedings of the AAAI Conference on Artificial Intelligence, AAAI Press, 2011.

[23] X. Chen, R. Chen, Q. Wu, Y. Fang, F. Nie, J.Z. Huang, LABIN: Balanced min cut for large-scale data, IEEE Trans. Neural Netw. Learn. Syst. 31 (3) (2020) 725–736.

[24] Z. Kang, Z. Lin, X. Zhu, W. Xu, Structured graph learning for scalable subspace clustering: From single view to multiview, IEEE Trans. Cybern. 52 (9) (2022) 8976–8986.

[25] Y. Mi, H. Chen, Z. Yuan, C. Luo, S.-J. Horng, T. Li, Fast multi-view subspace clustering with balance anchors guidance, Pattern Recognit. 145 (2024) 109895.

[26] W. Liu, J. He, S.-F. Chang, Large graph construction for scalable semi-supervised learning, in: Proceedings of the International Conference on Machine Learning, 2010, pp. 679–686.

[27] W. Zhu, F. Nie, X. Li, Fast spectral clustering with efficient large graph construction, in: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 2017, pp. 2492–2496.

[28] F. Nie, J. Xue, R. Wang, L. Zhang, X. Li, Fast clustering by directly solving bipartite graph clustering problem, IEEE Trans. Neural Netw. Learn. Syst. (2022) 1–12.

[29] D. Huang, C.-D. Wang, J.-S. Wu, J.-H. Lai, C.-K. Kwoh, Ultra-scalable spectral clustering and ensemble clustering, IEEE Trans. Pattern Anal. Mach. Intell. 32 (6) (2020) 1212–1226.

[30] F. Nie, J. Xue, D. Wu, R. Wang, H. Li, X. Li, Coordinate descent method for kk-means, IEEE Trans. Pattern Anal. Mach. Intell. 44 (5) (2022) 2371–2385.

[31] P. Zhou, J. Chen, M. Fan, L. Du, Y. Shen, X. Li, Unsupervised feature selection for balanced clustering, Knowl.-Based Syst. 193 (2020) 105417.

[32] P.S. Bradley, K.P. Bennett, A. Demiriz, Constrained K-Means Clustering, Microsoft Research Technical Report, 2000.

[33] H. Chen, Q. Zhang, R. Wang, F. Nie, X. Li, A general soft-balanced clustering framework based on a novel balance regularizer, Signal Process. 198 (2022) 108572.

[34] Z. Li, F. Nie, X. Chang, Z. Ma, Y. Yang, Balanced clustering via exclusive lasso: A pragmatic approach., in: Proceedings of the AAAI Conference on Artificial Intelligence, 2018, pp. 2492–2496.

[35] W. Lin, Z. He, M. Xiao, Balanced clustering: A uniform model and fast algorithm, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2019.

[36] X. Chen, J.Z. Haung, F. Nie, R. Chen, Q. Wu, A self-balanced min-cut algorithm for image clustering, in: Proceedings of IEEE International Conference on Computer Vision, 2017, pp. 2080–2088.

[37] X. Chen, W. Hong, F. Nie, J.Z. Huang, L. Shen, Enhanced balanced min cut, Proc. Int. J. Comput. Vis. 128 (7) (2020) 1982–1995.

[38] F. Nie, C. Liu, R. Wang, Z. Wang, X. Li, Fast fuzzy clustering based on anchor graph, IEEE Trans. Fuzzy Syst. 30 (7) (2022) 2375–2387.

[39] C. Liu, F. Nie, R. Wang, X. Li, Scalable fuzzy clustering with anchor graph, IEEE Trans. Knowl. Data Eng. 35 (8) (2023) 8503–8514.

[40] F. Nie, X. Wang, M.I. Jordan, H. Huang, The constrained Laplacian rank algorithm for graph-based clustering, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI Press, 2016, pp. 1969–1976.

[41] Z. Li, F. Nie, X. Chang, Z. Ma, Y. Yang, Balanced clustering via exclusive lasso: A pragmatic approach, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.

[42] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, E.Y. Chang, Parallel spectral clustering in distributed systems, IEEE Trans. Pattern Anal. Mach. Intell. 33 (3) (2011) 568–586.

Feiping Nie received the Ph.D. degree in Computer Science from Tsinghua University, China in 2009, and is currently a full professor in Northwestern Polytechnical University, China. His research interests are machine learning and its applications, such as pattern recognition, data mining, computer vision, image processing and information retrieval. He has published more than 100 papers in the following journals and conferences: TPAMI, IJCV, TIP, TNNLS, TKDE, ICML, NIPS, KDD, IJCAI, AAAI, ICCV, CVPR, ACM MM. His papers have been cited more than 20000 times and the H-index is 84. He is now serving as Associate Editor or PC member for several prestigious journals and conferences in the related fields.

Fangyuan Xie is currently pursing the Ph.D. degree with the School of Artificial Intelligence, OPtics and ElectroNics (iOPEN), Northwestern Polytechnical University, Xi'an. Her research interests include machine learning and its applications, such as pattern recognition and data mining.

Jingyu Wang received the Ph.D. degree in signal, image and automatic from the Université Paris-Est, Paris, France, in 2015. He is currently a full professor with the School of Astronautics, School of Artificial Intelligence, OPtics and ElectroNics (iOPEN), Northwestern Polytechnical University, Xi'an, China.

Xuelong Li is with the Institute of Artificial Intelligence (TeleAI), China Telecom, P. R. China.