symmetry

MDPI

Article

Optimizing MSE for Clustering with Balanced Size Constraints

Wei Tang, Yang Yang *, Lanling Zeng and Yongzhao Zhan

School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang 212013, China; 2211708027@stmail.ujs.edu.cn (W.T.); lanling73@163.com (L.Z.); yzzhan@ujs.edu.cn (Y.Z.)
* Correspondence: yyoung@ujs.edu.cn

Received: 7 February 2019; Accepted: 1 March 2019; Published: 6 March 2019

Abstract: Clustering is to group data so that the observations in the same group are more similar to each other than to those in other groups. *k*-means is a popular clustering algorithm in data mining. Its objective is to optimize the mean squared error (MSE). The traditional *k*-means algorithm is not suitable for applications where the sizes of clusters need to be balanced. Given n observations, our objective is to optimize the MSE under the constraint that the observations need to be evenly divided into k clusters. In this paper, we propose an iterative method for the task of clustering with balanced size constraints. Each iteration can be split into two steps, namely an assignment step and an update step. In the assignment step, the data are evenly assigned to each cluster. The balanced assignment task here is formulated as an integer linear program (ILP), and we prove that the constraint matrix of this ILP is totally unimodular. Thus the ILP is relaxed as a linear program (LP) which can be efficiently solved with the simplex algorithm. In the update step, the new centers are updated as the centroids of the observations in the clusters. Assuming that there are n observations and the algorithm needs m iterations to converge, we show that the average time complexity of the proposed algorithm is $O(mn^{1.65})$–$O(mn^{1.70})$. Experimental results indicate that, comparing with state-of-the-art methods, the proposed algorithm is efficient in deriving more accurate clustering.

Keywords: clustering; constrained clustering; balanced size constraints; mean squared error; linear program

1. Introduction

In the fields of data mining, machine learning, and social science, the most widely studied and fundamental method is clustering. Clustering is to partition observations into clusters so that similar observations are put together while dissimilar observations are separated [1]. There are sophisticated clustering algorithms. Unfortunately, they are not suitable for applications that impose the equally large cluster sizes as a constraint. For example, in marketing campaigns, the given customers need to be partitioned into clusters so that each cluster is allotted to a salesman. For the purpose of fairness and efficiency, each salesman should have the same workload [2]. In [3], the authors considered organizing sensor nodes into clusters of similar size so that the workloads are distributed evenly on all the master nodes. There are also other examples like circuit designing [4], document clustering [5], and image searching [6]. The balanced size constraints are introduced into clustering because of the application requirements rather than the actual distribution of the data. Moreover, according to [7], the balance of clusters is an important issue in clustering. The lack of size constraints in traditional clustering algorithms could lead to extremely unbalanced clustering.

With the demands of the balanced size-constraint clustering, many approaches have been proposed in the last decades. Most of the size-constraint clustering methods are based on soft size constraints. They could ease the problem of extremely unbalanced clustering caused by traditional clustering algorithms. However, they are not suitable for the applications where the observations are required to follow a strictly even distribution. Although there are plenty of such application requirements, only few studies have been dedicated to the hard size-constraint problem. More importantly, existing clustering methods based on hard size constraints can be hardly used due to the low accuracy or efficiency.

In this paper, we propose a novel approach to solve the problem of clustering with hard balance size constraints. Given a set of n observations, the task is to find a scheme to partition them into k clusters so that the sizes of all the clusters are approximately the same (± 1) while the mean squared error (MSE) is minimized. Similar to the k-means algorithm, our proposed method is a gradient descent solution which runs in an iterative way. Each iteration consists of two steps, namely an assignment step and an update step. In the assignment step, the observations are evenly assigned to different clusters. The balanced assignment task here is modeled as an integer linear program (ILP). We prove that the corresponding constraint matrix is totally unimodular. Thus the ILP can be relaxed as a linear program (LP) which can be efficiently solved with the simplex algorithm [8]. In the update step, the task is relatively simpler: update the new centers as the centroids of the observations assigned to the clusters. Assuming that there are n observations and the algorithm needs m iterations to converge, we show that the average time complexity of the proposed algorithm is $O(mn^{1.65})$–$O(mn^{1.70})$, which is much better than the $O(mn^3)$ method based on the Hungarian algorithm [9].

To evaluate the proposed method, experiments have been conducted. Both real and synthetic datasets are involved. The number of observations n ranges from 150 to 5000. The number of clusters k takes its value from the set of {3, 9, 21, 45, 93}, which covers both the situations when the number of observations could and could not be divisible by the number of clusters. We do not choose very large cluster numbers, because according to the rule of thumb [10], the number of clusters k for data of size n should be less than$\sqrt{n/2}$. The evaluation criteria include the cost and running time of the balanced assignment algorithm as well as the number of iterations, MSE, and running time of the balanced clustering algorithm. From the experimental results, it can be concluded that the proposed method is a practical solution for the task of clustering with balanced size constraints.

The rest of this paper is organized as follows. Section 2 reviews some related work. Section 3 covers the details about our proposed balanced clustering algorithm. In Section 4, the experimental settings and results are given. The discussion is provided in Section 5. Finally, Section 6 concludes the paper and presents future directions of research.

2. Related Work

Since the problem of clustering with balanced size constraints is a special form of the size-constraint clustering, we will briefly review some studies related to clustering with size constraints. The constraints can be divided into two categories, namely soft size constraints and hard size constraints. The former ones refer to constraints that are not guaranteed to be satisfied, while the latter ones are compulsory conditions that must be satisfied during clustering process [11].

2.1. Soft Constraints

More work has been published about soft rather than hard constraints. Ref. [7] proposed frequency sensitive competitive learning (FSCL) for balanced clustering. A multiplicative term and an additive term are introduced into the objective function in order to penalize larger clusters. Thus larger clusters are less likely to win points so that the clustering will be balanced. Similarly, Ref. [6] proposed to apply a penalty mechanism in FSCL that only includes an additive term. Ref. [12] presented a scalable clustering algorithm that satisfies the balance constraints which runs in $O(kNlogN)$ time. The algorithm first clusters the sampled data points, then populates the clusters with the unsampled data points while simultaneously keeping the balanced size constraints. The ratio-cut algorithm that solves an objective function which embodies both min-cut and equipartition was described in [13]. Ref. [14] suggested a size regularized cut (SRcut). The sizes of clusters are utilized as prior knowledge in the clustering process. A regularization term measuring the balances of clusters is added to the cost function. The results suggest that it outperforms the normalized cut which impose indirect size constraints [15]. Ref. [16] proposed to incorporate the balance condition in the cost function, and the problem is solved by finding a partition close to the given partition. In [17], exclusive lasso is applied in the balanced k-means and min-cut method to represent the size constraints. The results suggest improved performance. Clustering with soft size constraints could ease the problem of extremely unbalanced clustering, yet it is not able to deal with applications that impose hard constraints on the sizes of clusters. In this paper, our focus is on the clustering with hard size constraints.

2.2. Hard Constraints

On the other hand, there are few studies of clustering with hard size constraints, although the topic is highly relevant in practice. The method in [9] presented a new direction to deal with this problem. It translates the k-means assignment step into the average assignment problem that is to evenly assign n observations to k clusters with minimum cost. The average assignment problem is then modeled as a bipartite matching problem that is readily solved by the Hungarian algorithm. The method works well when the number of observations n is divisible by the number of clusters k. However, it is awkward when n is not divisible by k because it is not known that which clusters should be assigned to $\lceil n/k \rceil$ observations. Ref. [18] proposed a method for clustering with hard size constraints. It first applies the k-means algorithm to the data so that the partition A without any size constraints is obtained. Then the partition B with given size constraints is calculated by maximizing its agreement with partition A. The optimization problem is finally modeled as an ILP, which can be easily solved with an existing solver. The algorithm is quite convenient and efficient to implement. The experimental results also indicate that incorporating the size constraints improves the accuracy of the clustering algorithm. However, the algorithm fails to consider the similarity between observations in the mathematical model, which leads to a less optimized MSE. Refs. [19,20] presented a balanced k-means algorithm to solve the problem of partitioning areas for large scale vehicle routing. The traditional k-means algorithm is applied to divide the whole dataset into several areas at the first step. Then a heuristic algorithm is developed to adjust the clusters so that the balanced size constraints are satisfied. However, the heuristic algorithm can hardly guarantee the quality of clustering.

In case of local solutions with clusters that contain few observations or even empty clusters, a clustering method was proposed in [21] where it imposes lower bounds on the sizes of clusters. The problem is modeled as an ILP which is then transferred into a minimum cost flow (MCF) linear network optimization problem. An algorithm specifically tailored to network optimization is then applied to solve the problem. Refs. [22,23] put lower bounds and upper bounds on the sizes of clusters, respectively, and adapt heuristic algorithms to solve the size-constraint clustering. However, the objectives of these studies differ from what we are going to accomplish in this paper.

According to the above discussion, most of the current researches on clustering with size constraints are based on soft constraints. Few contributions have been dedicated to the problem of clustering with hard size constraints, and they are not competent for the problem. To fill the gap, in this paper we propose a novel method that is based on ILP to deal with the problem of clustering with hard size constraints.

3. Balanced Clustering Algorithm

3.1. Problem Formulation

Minimizing the Mean Squared Error (MSE) is one of the most common standards in clustering algorithms. In this paper, we are trying to optimize the MSE subject to hard balanced size constraints, i.e., all the clusters should have approximately the same number of observations (± 1). Given n observations, the objective is to divide them into k clusters, and each cluster contains $\lfloor n/k \rfloor$ to $\lceil n/k \rceil$ observations. So our problem can be formulated as:

$$\begin{aligned} &Minimize\ E = (1/n)\sum_{j=1}^{k}\sum_{o_i\in\mu_j}\left\|o_i - c_j\right\|^2 \\ &s.t.\ \lfloor n/k\rfloor \le \left|\mu_j\right| \le \lceil n/k\rceil \end{aligned} \tag{1}$$

Here, o_i is the i-th observation, μ_j represents the set of data in the j-th cluster, $\left|\mu_j\right|$ denotes the number of observations in the j-th cluster, c_j represents the center of the j-th cluster, and $\left\|o_i - c_j\right\|^2$ stands for the squared distance between o_i and c_j.

Let p denote the partition matrix, where $p_{i,j} = 1$ indicates that observation o_i belongs to cluster j, and $p_{i,j} = 0$ indicates that o_i does not belong to cluster j. Obviously, summing each row of p equals 1 since one observation can only be assigned to one cluster. Summing each column of p would be the number of observations in the corresponding cluster, which in the balanced case would be either $\lfloor n/k\rfloor$ or $\lceil n/k\rceil$. In this way, the problem shown in Equation (1) can be reformulated as:

$$\begin{aligned} Minimize\ E(c,p) = (1/n)\sum_{j=1}^{k}\sum_{i=1}^{n} p_{i,j}\left\|o_i - c_j\right\|^2 \\ s.t.\ \lfloor n/k\rfloor \le \sum_{i=1}^{n} p_{i,j} \le \lceil n/k\rceil,\ 1\le j\le k \\ \sum_{j=1}^{k} p_{i,j} = 1,\ 1\le i\le n \\ p_{i,j}\in\{0,1\},\ 1\le j\le k,\ 1\le i\le n \end{aligned} \tag{2}$$

3.2. Proposed Solution

Similar to the k-means algorithm, the gradient descent method, which runs in an iterative way, is applied to solve the above problem. Given n observations, we derive the initial k centers by k-means++ algorithm [24]. Then the algorithm proceeds by 2 steps, namely an assignment step and an update step.

In the assignment step, we need to evenly assign the observations to different clusters according to the distance between the observations and the cluster centers so that the MSE is minimized. The objective is to minimize $E(c,p)$ with respect to p while holding c fixed (known). Therefore, Equation (2) becomes an ILP as shown in Equation (3), where u_j, v_j and $g_{i,j}$ are slack variables used to eliminate inequalities. Z is the set of integers. In the next subsection we will prove that the integer constraints can be removed from this ILP so that the ILP becomes an LP which can be efficiently solved with the simplex algorithm [8].

$$\begin{aligned} Minimize\ E(c,p) = (1/n)\sum_{j=1}^{k}\sum_{i=1}^{n} p_{i,j}\left\|o_i - c_j\right\|^2 \\ s.t.\ \sum_{i=1}^{n} p_{i,j} + u_j = \lceil n/k\rceil,\ 1\le j\le k \\ -\sum_{i=1}^{n} p_{i,j} + v_j = -\lfloor n/k\rfloor,\ 1\le j\le k \\ \sum_{j=1}^{k} p_{i,j} = 1,\ 1\le i\le n \\ p_{i,j} + g_{i,j} = 1,\ 1\le j\le k,\ 1\le i\le n \\ u_j, v_j, g_{i,j}, p_{i,j} \ge 0,\ 1\le j\le k,\ 1\le i\le n \\ u_j, v_j, g_{i,j}, p_{i,j} \in Z,\ 1\le j\le k,\ 1\le i\le n \end{aligned} \tag{3}$$

A concrete example of the above problem can be illustrated as a bipartite graph shown in Figure 1. Five observations need to be evenly assigned to two clusters, i.e., each cluster must be assigned to two or three observations. The weight attached to each edge $w_{i,j}$ is the squared distance between observation o_i and cluster center c_j, namely $\|o_i - c_j\|^2$. The green solid lines shown in Figure 1 indicate a possible solution where o_1, o_3, and o_4 are assigned to c_1, while o_2 and o_5 are assigned to c_2.

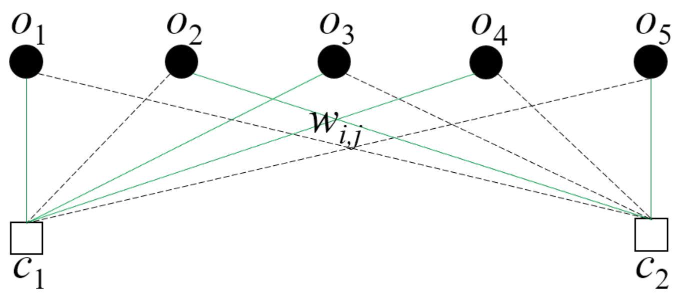


Figure 1. The bipartite graph (the green solid lines indicate a possible solution).

Once the observations are assigned, the new centers need to be updated so that the MSE is minimized. The objective is to minimize $E(c,p)$ with respect to c while holding p fixed (known). Since p is fixed, we can remove all the constraints in Equation (2). Then the task of the update step becomes an optimization problem without any constraints as shown in Equation (4).

$$\begin{aligned} Minimize\ E(c,p) &= (1/n)\sum_{j=1}^{k}\sum_{i=1}^{n} p_{i,j}\left\|o_i - c_j\right\|^2 \\ &= (1/n)\sum_{j=1}^{k}\sum_{i=1}^{n} p_{i,j}(o_i - c_j)(o_i - c_j)^{\mathrm{T}} \end{aligned} \tag{4}$$

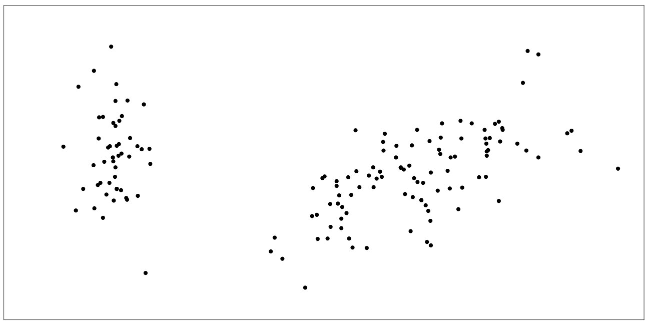
The minimum value of E can be achieved when

$$\begin{aligned} d(E(c,p))/d(c) = 0 &\Rightarrow \sum_{i=1}^{n} p_{i,j}c_j - \sum_{i=1}^{n} p_{i,j}o_i = 0 \\ &\Rightarrow c_j = (1/\sum_{i=1}^{n} p_{i,j})\sum_{i=1}^{n} p_{i,j}o_i \end{aligned} \tag{5}$$

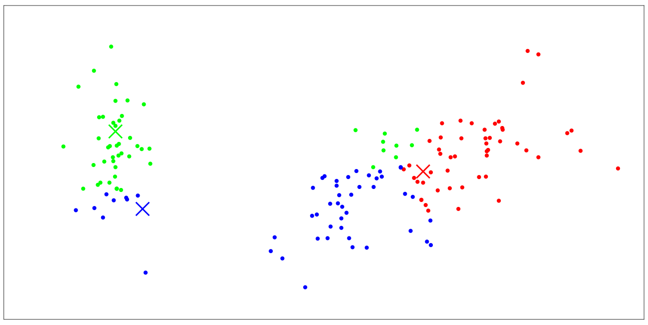
In summary, the proposed method can be detailed as Algorithm 1. A concrete example of our proposed algorithm can be found in Figure 2.

Algorithm 1 Clustering with balanced size constraints

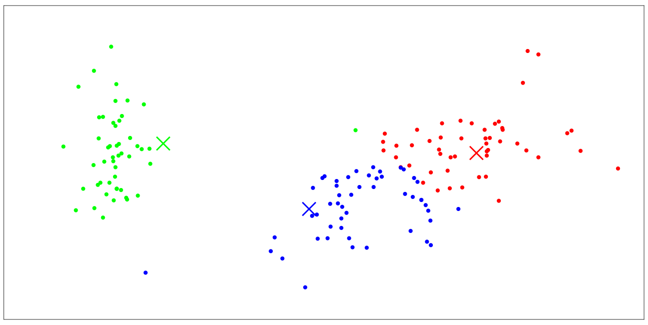
```
Input:   Observations $O = \{o_1, o_2, ..., o_n\}$, number of clusters $k$
Output:   Labels of the observations.
1: Initialize $k$ centers using the $k$-means++ algorithm;
2: repeat
3:     Assignment step: assign the observations to $k$ clusters by solving Equation (3);
4:     Update step: update the new centers of the clusters by solving Equation (5);
5: until There is no more change to the partition matrix;
6: return Labels of the observations.
```



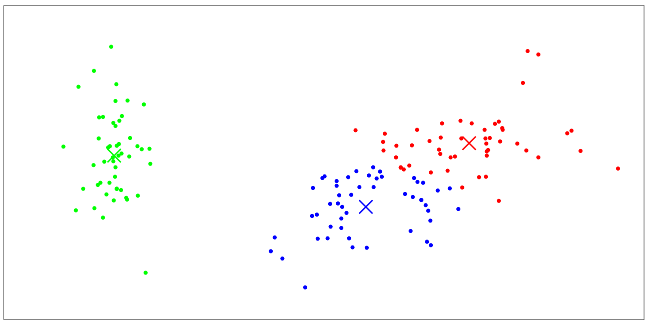
(**a**)



(**b**)



(**c**)



(**d**)

Figure 2. An example of our proposed method: (**a**) The data points. (**b**) The result after the first iteration. (**c**) The result after the second iteration. (**d**) The result after the last iteration.

Our algorithm is guaranteed to converge, but not always to the global optimum due to the non-convexity of the objective function. Assuming that $p^{(t)}$, $c^{(t)}$ are the partition matrix and the centers at the end of the t-th iteration, $p^{(t)}$ must satisfy the constraints in Equation (3). In the assignment step of $(t+1)$-th iteration, we evenly assign all the observations to different clusters so that $E(c^{(t)}, p)$ is minimized, i.e., we optimize $E(c^{(t)}, p)$ with respect to p under the constraints in Equation (3) to get $p^{(t+1)}$. Thus we have $E(c^{(t)}, p^{(t+1)}) \leq E(c^{(t)}, p^{(t)})$. In the update step, the new centers are updated, which further optimize the objective function. Thus we have $E(c^{(t+1)}, p^{(t+1)}) \leq E(c^{(t)}, p^{(t+1)})$. The value of E monotonically decreases during the course of our algorithm, so it converges.

3.3. Time Complexity

Each iteration of the proposed algorithm consists of two steps, namely an assignment step and an update step. According to Equation (5), the update step takes $O(n)$ time, where n is the number of observations. Solving the ILP in the assignment step on the other hand is relatively more difficult. Typically, to handle the general ILP shown in Equation (6), the simplex algorithm needs to be executed multiple (at most exponential) times in a branch-and-bound way. However, according to [25], if the vector b is integral and the constraint matrix A is totally unimodular (a matrix is totally unimodular if and only if the determinant of every square sub-matrix is 0, 1, or -1), the solution to the relaxed LP (without integer constraints) is guaranteed to be integral, i.e., the integer constraints can be simply removed so that the ILP can be transformed into an LP which can be efficiently solved by the simplex algorithm.

$$\begin{aligned} Minimize\ E(l, x) &= lx^{\mathrm{T}} \\ s.t.\ Ax^{\mathrm{T}} &= b \\ x &\geq 0 \\ x &\in Z^{t} \end{aligned} \tag{6}$$

Theorem 1. *The constraint matrix of the ILP for the balanced assignment problem shown in Equation (3) is totally unimodular.*

Proof. Putting Equation (3) into the form of Equation (6), we have the variables.

$$x = \begin{pmatrix} x' & x'' \end{pmatrix}$$

x' and x'' represent the decision variables and slack variables, respectively.

$$x' = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,k} & p_{2,1} & p_{2,2} & \cdots & p_{2,k} & \cdots & p_{n,1} & p_{n,2} & \cdots & p_{n,k} \end{pmatrix}$$

$$x'' = \begin{pmatrix} u_1 & u_2 & \cdots & u_k & v_1 & v_2 & \cdots & v_k & g_{1,1} & g_{1,2} & \cdots & g_{n,k} \end{pmatrix}$$

Correspondingly, the constraint matrix can be derived.

$$A = \begin{pmatrix} A'_1 & A''_1 \\ A'_2 & A''_2 \\ A'_3 & A''_3 \\ A'_4 & A''_4 \end{pmatrix}$$

Each row of matrix A contains the coefficients for each set of constraints shown in Equation (3). $(A'_1 \quad A''_1)$ are the coefficients for the first set of constraints $\sum_{i=1}^{n} p_{i,j} + u_j = \lceil n/k \rceil, 1 \leq j \leq k$. $(A'_2 \quad A''_2)$ are the coefficients for the second set of constraints $-\sum_{i=1}^{n} p_{i,j} + v_j = -\lfloor n/k \rfloor$, $1 \leq j \leq k$. $(A'_3 \quad A''_3)$ are the coefficients for the third set of constraints $\sum_{j=1}^{k} p_{i,j} = 1$, $1 \leq i \leq n$. $(A'_4 \quad A''_4)$ are the coefficients for the final set of constraints $p_{i,j} + g_{i,j} = 1$, $1 \leq j \leq k$, $1 \leq i \leq n$. $(A'_1 \quad A'_2 \quad A'_3 \quad A'_4)^{\mathrm{T}}$ and $(A''_1 \quad A''_2 \quad A''_3 \quad A''_4)^{\mathrm{T}}$ correspond to the coefficients for the decision and slack variables, respectively.

$$A_1' = \begin{pmatrix} I_{k\times k} & I_{k\times k} & \cdots & I_{k\times k} \end{pmatrix} \qquad A_1'' = \begin{pmatrix} I_{k\times k} & 0_{k\times k} & 0_{k\times nk} \end{pmatrix}$$

$$A_2' = -A_1' \qquad A_2'' = \begin{pmatrix} 0_{k\times k} & I_{k\times k} & 0_{k\times nk} \end{pmatrix}$$

$$A_3' = \begin{pmatrix} 1\,1 \cdots 1\,0\,0 \cdots 0 \cdots 0\,0 \cdots 0 \\ 0\,0 \cdots 0\,1\,1 \cdots 1 \cdots 0\,0 \cdots 0 \\ \vdots\ \vdots\ \ddots\ \vdots\ \vdots\ \vdots\ \ddots\ \vdots\ \ \vdots\ \ \vdots\ \vdots\ \ddots\ \vdots \\ 0\,0 \cdots 0\,0\,0 \cdots 0 \cdots 1\,1 \cdots 1 \end{pmatrix} \qquad A_3'' = \begin{pmatrix} 0_{n\times 2k} & 0_{n\times nk} \end{pmatrix}$$

$$A_4' = I_{nk\times nk} \qquad A_4'' = \begin{pmatrix} 0_{nk\times k} & 0_{nk\times k} & I_{nk\times nk} \end{pmatrix}$$

Here, $I_{k\times k}$ is the identity matrix of size $k \times k$. $0_{k\times k}$ is the zero matrix of size $k \times k$.

According to Lemma 4, $(A_2' \quad A_3')^{\mathrm{T}}$ is a totally unimodular matrix, as every column has two non-zero entries being ± 1, and the sum of each column equals 0.

According to Lemma 2 and Lemma 3, $(A_1' \quad A_2' \quad A_3')^{\mathrm{T}}$ is totally unimodular, because multiplying A_1' with-1 results in a duplicate of A_2', and duplication does not affect total unimodularity.

According to Lemma 1, $(A_1' \quad A_2' \quad A_3' \quad A_4')^{\mathrm{T}}$ is totally unimodular, because every row of A_4' contains exactly one non-zero entry which is 1. Inserting A_4' to $(A_1' \quad A_2' \quad A_3')^{\mathrm{T}}$ would still result in a totally unimodular matrix.

According to Lemma 1, A is totally unimodular, because every column of $(A_1'' \quad A_2'' \quad A_3'' \quad A_4'')^{\mathrm{T}}$ contains exactly one non-zero entry which is 1. Inserting $(A_1'' \quad A_2'' \quad A_3'' \quad A_4'')^{\mathrm{T}}$ to $(A_1' \quad A_2' \quad A_3' \quad A_4')^{\mathrm{T}}$ leads to a totally unimodular matrix.

All the lemmas can be found in [26].

According to the above description, the constraint matrix A for the ILP shown in Equation (3) is a totally unimodular matrix. □

Lemma 1. *The constraint matrix A remains totally unimodular if inserting a column (row) with only one non-zero entry being ± 1.*

Lemma 2. *The constraint matrix A remains totally unimodular if multiplying a column (row) with -1.*

Lemma 3. *The constraint matrix A remains totally unimodular if duplicating a column (row).*

Lemma 4. *The constraint matrix A is totally unimodular if it has at most two non-zero entries being ± 1 in each column (row), and, for every column (row) with two non-zero entries, the sum of the column (row) is 0.*

The constraint matrix A in our problem is totally unimodular. In addition, the vector b is integral so that the linear program shown in Equation (3) can be simply solved by the simplex algorithm. The worst time complexity of the simplex algorithm is exponential. Fortunately, the simplex algorithm is remarkably efficient in practice, and it has polynomial time complexity in the average case [27,28]. More specifically, the average time complexity of the simplex algorithm is $T = \beta s^{\alpha} t d^{0.33}$ [29]. Here, s and t are the numbers of constraints and variables in the linear program, respectively. d is the percentage of non-zero entries in the constraint matrix. α and β are the two unknowns we need to find out.

To evaluate α and β, we collected 240 different sets of data for regression of the form $T/(td^{0.33}) = \beta s^{\alpha}$. All the data are randomly generated. The number of observations n varies from 100 to 100,000. The number of clusters k varies from 3 to 100. It is found that α is in the range $(0.43, 0.46)$, β is in the range $(2.09 \times 10^{-6}, 3.46 \times 10^{-5})$. The parameters related to the goodness of fit are $SSE = 8.38 \times 10^{-6}$, $R - square = 0.95$, $Adjusted\ R - square = 0.95$. Figure 3 illustrates the result of the regression.

Furthermore, in our case, we have $s = n + 2k + nk$, $t = 2nk + 2k$, and $d = (5nk + 2k)/(st)$. By placing everything together into $T = \beta s^{\alpha} t d^{0.33}$, we have $T = O((nk)^{1.1})$–$O((nk)^{1.13})$. Considering the maximum value of k is $\sqrt{n/2}$ [10], the average time complexity of each assignment step is approximately $O(n^{1.65})$–$O(n^{1.70})$. Assuming that there are m iterations, the average time complexity of the balanced clustering algorithm is $O(mn^{1.65})$–$O(mn^{1.70})$.

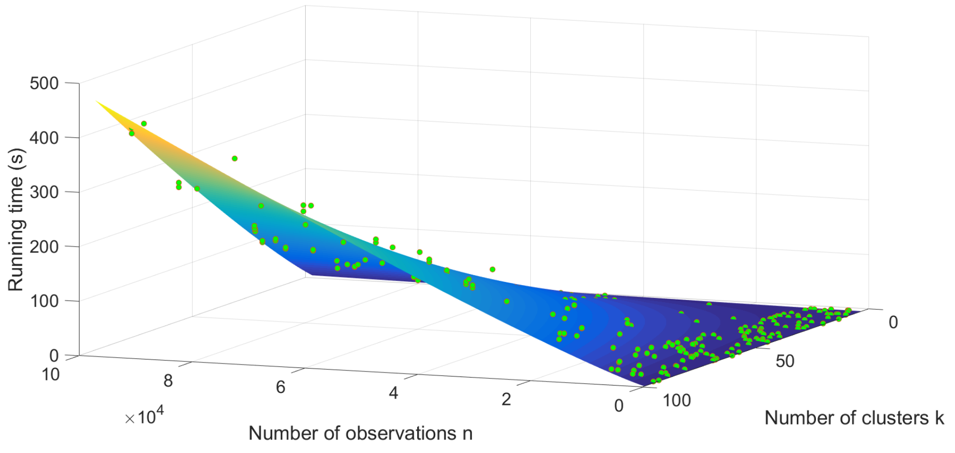


Figure 3. The result of the regression.

4. Experiments

4.1. Experimental Settings

In order to evaluate the performance of our proposed approach, both synthetic and real datasets are explored in our experiments. The synthetic datasets are generated from the standard uniform distribution on the interval (0, 100). The real datasets are from the UCI machine learning repository [30]. Table 1 shows the details of the datasets in our experiments.

Table 1. A brief summary of experimental datasets.

Category	Data	Size	Dimension
Random	R1	200	2
	R2	600	2
	R3	1000	2
	R4	2000	2
	R5	5000	2
Real	Iris	150	4
	Wine	178	14
	Thyroid	215	5
	Hill Valley (HV)	606	100
	Anuran Calls Subset (ACS)	1500	22

We compare the balanced clustering performances between our proposed method and state-of-the-art methods which include the balanced clustering method presented by [9] and the size-constraint clustering method proposed in [18]. Notice that for each run, all the algorithms start with the same set of initial centers derived from the k-means++ algorithm so that the comparison is objective. In addition, since [9] and we both model each iteration as a balanced assignment problem, the performances of the balanced assignment algorithms are also compared to provide a deeper insight. Notice that the approach in [18] is not an iterative approach, so it is not included in this comparison.

For the evaluation of the balanced assignment algorithm, we consider the cost and running time (measured in seconds). For the evaluation of the balanced clustering algorithm, we record the number of iterations, MSE, and running time (measured in seconds). It should be noted that we do not record the number of iterations for the method in [18] as it is not an iterative method. Furthermore, to evaluate the stability of the clustering algorithms, the coefficient of variation for the number of iterations, MSE and running time are collected. All the metrics mentioned above are calculated and averaged over 10 runs. For the sake of compactness, we round all the metrics to two decimal places. In addition, all trailing zeros are omitted. The best results are highlighted with the bold format in all the tables.

The number of clusters for all the data in our experiments takes its value from the set of {3, 9, 21, 45, 93}. We do not choose very large cluster number, as according to the rule of thumb [10], the number of clusters k for data of size n should be less than $\sqrt{n/2}$. For the datasets involved in our experiments, the largest number of observations is 5000, and $\sqrt{5000/2} = 50$ so that the range of k is enough for us to evaluate the clustering algorithm.

All the algorithms involved in this paper are implemented in MATLAB which ran on an Intel i7-7700HQ 2.8 GHz processor with 16 GB memory. The code can be downloaded from https://github.com/IGGIUJS/BalancedClustering.

4.2. Experimental Results

4.2.1. Balanced Assignment

From Tables 2 and 3, we can observe two differences. (1) Although occasionally the assignment method in [9] produces equal cost, more frequently, the proposed method achieves a better cost for the balanced assignment problem. (2) Our proposed assignment method takes much less time than the method in [9].

Table 2. Cost and running time for balanced assignment (synthetic data).

Data	K	Cost Ours/BK [1]	Time Ours/BK [1]
R1	3	**2.87**/2.88 (10^5)	0.37/**0.21**
	9	**1.11**/1.12 (10^5)	**0.02**/0.11
	21	**4.76**/5.10 (10^4)	**0.03**/0.1
	45	**2.52**/3.17 (10^4)	**0.03**/0.13
	93	**9.23**/13.25 (10^3)	**0.07**/0.11
R2	3	**1.15/1.15** (10^6)	**0.02**/14.54
	9	**4.22**/4.25 (10^5)	**0.03**/3.19
	21	**1.84**/1.9 (10^5)	**0.05**/1.94
	45	**8.01**/8.66 (10^4)	**0.09**/1.35
	93	**3.66**/4.6 (10^4)	**0.17**/1.26
R3	3	**1.6/1.6** (10^6)	**0.05**/84.7
	9	**5.92**/5.94 (10^5)	**0.04**/14.95
	21	**2.55**/2.6 (10^5)	**0.08**/6.77
	45	**1.7**/1.77 (10^5)	**0.18**/6.01
	93	**5.92**/6.4 (10^4)	**0.41**/4.58
R4	3	**2.86/2.86** (10^6)	**0.14**/1081.9
	9	**1.23/1.23** (10^6)	**0.12**/151.8
	21	**6.53**/6.58 (10^5)	**0.2**/79.91
	45	**4.27**/4.38 (10^5)	**0.47**/48.86
	93	**1.4**/1.48 (10^5)	**1.19**/29.07
R5	3	**6.81/6.81** (10^6)	**0.68**/59946
	9	**2.93/2.93** (10^6)	**0.43**/7582.1
	21	**1.52**/1.53 (10^6)	**0.75**/2618.4
	45	**6.82**/6.85 (10^5)	**1.78**/786.55
	93	**4.67**/4.76 (10^5)	**4.91**/620.3

[1] The method in [9].

Table 3. Cost and running time for balanced assignment (real data).

Data	**K**	**Cost Ours/BK** [1]	**Time Ours/BK** [1]
Iris	3	**4.95/4.95** (10^{14})	**0.01**/0.08
	9	**2.28**/2.34 (10^{14})	**0.02**/0.06
	21	**1.31**/1.45 (10^{14})	**0.03**/0.07
	45	**5.11**/8.85 (10^{13})	**0.03**/0.07
	93	**1.16**/3.92 (10^{13})	**0.05**/0.06
Wine	3	**5.12**/5.13 (10^{11})	**0.02**/0.13
	9	**3.74**/3.75 (10^{11})	**0.02**/0.1
	21	**2.45**/2.54 (10^{11})	**0.04**/0.1
	45	**1.68**/1.71 (10^{11})	**0.04**/0.15
	93	**8.62**/10 (10^{10})	**0.06**/0.08
Thyroid	3	**1.31**/1.32 (10^{15})	**0.02**/0.43
	9	**8.75**/8.86 (10^{14})	**0.02**/0.17
	21	**4.79**/4.91 (10^{14})	**0.03**/0.16
	45	**2.31**/2.73 (10^{14})	**0.04**/0.15
	93	**1.03**/1.3 (10^{14})	**0.07**/0.13
HV	3	**2.77/2.77** (10^{13})	**0.02**/3.57
	9	**1.4**/1.41 (10^{13})	**0.04**/2.78
	21	**1.15**/1.2 (10^{13})	**0.07**/2.72
	45	**3.66**/4.01 (10^{12})	**0.11**/3.9
	93	**1.63**/1.89 (10^{12})	**0.22**/4.27
ACS	3	**1.31/1.31** (10^{3})	**0.04**/198.75
	9	**654.78**/656.54	**0.08**/36.16
	21	**478.67**/482	**0.13**/32.07
	45	**299.23**/307.21	**0.25**/41.05
	93	**212.48**/221.01	**0.7**/64.48

[1] The method in [9].

4.2.2. Balanced Clustering

From Tables 4 and 5, we can observe three differences. (1) Compared with the method in [18], the proposed method achieves much lower MSE with bearable loss of efficiency. (2) Compared with the method in [9], the proposed method leads to a better MSE, while at the same time, it is much faster. (3) The proposed method generally converges with fewer iterations than the method in [9].

The coefficient of variation (C.V. = standard deviation/mean) is explored to evaluate the stability of the proposed algorithm. From Tables 6 and 7, we can observe two differences. (1) Our method is as stable as the method in [9]. In fact, the standard deviations of these measures in our method are mostly lower than that in the method of [9], and the means of these measures in our method are even lower (better). (2) Compared with the method in [18], the running time of the proposed method is less stable while the MSE is more stable.

Table 4. Mean squared error (MSE), running time, and iteration count for balanced clustering (synthetic data).

Data	K	MSE Ours/BK [1]/BSCK [2]	Time Ours/BK [1]/BSCK [2]	Iterations Ours/BK [1]
R1	3	**629.81**/630.97/794.71	0.2/0.57/**0.06**	**5.6**/6.3
	9	**168.88**/169.1/440.24	0.11/0.53/**0.03**	**5**/6.1
	21	**71.7**/74.71/791.17	0.17/0.58/**0.04**	**4.4**/5.7
	45	**28.13**/31.9/711.57	0.24/0.51/**0.06**	**3.2**/4.8
	93	**9.93**/12.64/669.44	0.36/0.35/**0.12**	**2**/3
R2	3	**645.78/645.78**/721.07	0.13/5.69/**0.03**	**4.1**/5.1
	9	183.54/**180.61**/382.93	0.83/12.73/**0.06**	**11.3**/16
	21	**78.29**/78.48/422.43	1.06/7.42/**0.13**	**6.7**/8.3
	45	**34.06**/34.68/492.25	1.81/7.74/**0.29**	**5.5**/7.6
	93	**14.86**/15.71/594.6	3.62/7.88/**0.65**	**5.1**/6.5
R3	3	**635.91**/636.01/710.53	1.24/62.07/**0.06**	**9.7**/10.3
	9	**190.61**/190.67/382.46	1.64/26.72/**0.13**	**7.4**/8.8
	21	**78.08**/78.17/416.13	2.93/30.18/**0.31**	**8.1**/11.2
	45	**35.26**/35.6/437.29	7.62/34.15/**0.75**	10.4/**9.6**
	93	**15.9**/16.28/459.43	13.2/40.71/**1.66**	**8.3**/9.2
R4	3	**664.25**/664.3/807.32	2.32/783.43/**0.21**	**9**/9.5
	9	**190.78**/190.86/343.39	4.68/223.61/**0.55**	15.1/**12.9**
	21	**79.59**/79.8/376.08	4.81/210.67/**1.27**	15.2/**14.2**
	45	**34.82**/34.92/360.01	8.5/272.93/**2.89**	**16.6**/17
	93	**16.65**/16.76/443.89	15.01/318.73/**7.83**	13.6/**13.2**
R5	3	**664.54**/664.58/800.04	22.32/67500/**0.37**	12.8/**11.8**
	9	**184.39**/184.40/283.19	7.88/1900/**0.42**	7.9/**7.7**
	21	**79.84**/79.84/252.66	49.95/3100/**1.37**	**17.7**/20
	45	36.53/**36.45**/366.76	90.64/5100/**5.38**	**26.3**/30.2
	93	**17.32**/17.37/327.91	88.69/5100/**22.95**	**19.7**/20.9

[1] The method in [9]. [2] The method in [18].

Table 5. MSE, running time, and iteration count for balanced clustering (real data).

Data	K	MSE Ours/BK [1]/BSCK [2]	Time Ours/BK [1]/BSCK [2]	Iterations Ours/BK [1]
Iris	3	**9.35/9.35**/23.5 (10^{11})	0.03/0.15/**0.01**	**1.6**/2.6
	9	**4.08**/4.2/28.3 (10^{11})	0.11/0.36/**0.02**	6.2/**6.1**
	21	**2.18**/2.32/25.4 (10^{11})	0.15/0.37/**0.03**	**5.2**/5.9
	45	**1.09**/1.26/26.6 (10^{11})	0.17/0.29/**0.04**	**3**/4.1
	93	**6.07**/19.4/212 (10^{10})	0.18/0.41/**0.09**	**1**/4.9
Wine	3	**1.25/1.25**/1.36 (10^{9})	0.03/0.17/**0.01**	**1**/2.2
	9	9.14/**9.10**/18.4 (10^{8})	0.11/0.6/**0.02**	**5.5**/7.6
	21	**6.77**/6.83/18.6 (10^{8})	0.16/0.42/**0.03**	**4.6**/4.8
	45	**4.66**/4.72/18.6 (10^{8})	0.16/0.3/**0.05**	**2**/2.7
	93	3.23/**3.18**/14 (10^{8})	0.33/0.33/**0.11**	**1.9**/3.1
Thyroid	3	**3.46**/3.47/11.9 (10^{12})	0.1/2.71/**0.02**	**6.2**/7.6
	9	**1.88**/1.89/16 (10^{12})	0.12/1.48/**0.02**	**5.3**/7.6
	21	**9.15**/9.46/106 (10^{11})	0.24/1.16/**0.04**	**6.1**/6.4
	45	**3.97**/4.35/86 (10^{11})	0.31/1.02/**0.07**	**3.6**/4.8
	93	**1.94**/2.35/55.3 (10^{11})	0.48/0.81/**0.13**	**2.5**/4.4
HV	3	**2.75/2.75**/24.89 (10^{10})	0.21/13.56/**0.11**	**2/2**
	9	**8.44**/8.5/349.29 (10^{9})	0.12/8.16/**0.06**	**2**/2.3
	21	**2.04**/2.18/359.52 (10^{9})	0.25/12.49/**0.12**	3.4/**3.1**
	45	**6.59**/6.98/3203.1 (10^{8})	0.65/20.54/**0.25**	**4.3**/4.4
	93	**2.43**/2.94/2449.2 (10^{8})	1.57/32.64/**0.54**	**4.9**/5.3
ACS	3	**0.31/0.31**/0.51	0.35/1008.9/**0.07**	**7/7**
	9	**0.14/0.14**/0.47	1.11/268.08/**0.14**	12.2/**11.3**
	21	**0.08/0.08**/0.54	2.9/259.55/**0.42**	17.6/**13.6**
	45	**0.06/0.06**/0.54	4.56/287.23/**1**	**14.6**/14.7
	93	**0.04**/0.05/0.54	9.71/265.17/**3.27**	**11**/12

[1] The method in [9]. [2] The method in [18].

Table 6. Coefficient of variation for MSE, running time, and iteration count (synthetic data).

Data	K	MSE C.V. Ours/BK [1]/BSCK [2]	Time C.V. Ours/BK [1]/BSCK [2]	Iterations C.V. Ours/BK [1]
R1	3	**0.03/0.03**/0.07	1.57/**0.77**/2.43	0.95/**0.8**
	9	**0.04/0.04**/0.39	**0.49**/0.55/0.59	0.57/**0.53**
	21	**0.04**/0.05/0.24	**0.28**/0.38/**0.28**	**0.33**/0.41
	45	0.08/**0.07**/0.12	0.22/0.27/**0.04**	0.29/**0.27**
	93	**0.1**/0.11/0.18	**0.03**/0.11/0.05	**0**/0.16
R2	3	**0.03/0.03**/0.05	0.76/0.63/**0.07**	0.94/**0.75**
	9	0.06/**0.04**/0.33	0.63/0.78/**0.03**	**0.68**/0.77
	21	0.03/**0.02**/0.17	0.36/0.24/**0.03**	0.42/**0.23**
	45	0.05/**0.03**/0.13	0.24/0.4/**0.05**	**0.27**/0.47
	93	**0.03/0.03**/0.11	0.23/0.28/**0.05**	**0.28**/0.29
R3	3	**0/0**/0.04	0.29/0.29/**0.05**	0.32/**0.3**
	9	**0.03/0.03**/0.11	0.78/0.62/**0.04**	0.87/**0.61**
	21	**0.02/0.02**/0.17	0.41/0.53/**0.05**	**0.47**/0.55
	45	0.03/**0.01**/0.23	0.21/0.19/**0.08**	0.23/**0.16**
	93	**0.02/0.02**/0.1	0.38/0.21/**0.05**	0.43/**0.26**
R4	3	**0.01/0.01**/0.04	1.52/1.51/**0.05**	1.55/**1.52**
	9	**0/0**/0.11	0.78/0.95/**0.05**	**0.82**/0.88
	21	**0.01/0.01**/0.21	0.58/0.41/**0.05**	0.59/**0.43**
	45	**0.03/0.03**/0.19	0.4/0.32/**0.07**	0.38/**0.34**
	93	**0.01/0.01**/0.17	0.22/0.24/**0.05**	**0.23**/0.3
R5	3	**0.01/0.01**/0.04	**0.98**/1.07/1	**0.99**/1.12
	9	**0.01/0.01**/0.11	0.45/**0.34**/0.35	0.45/**0.4**
	21	**0.01/0.01**/0.16	0.52/0.57/**0.29**	**0.51**/0.52
	45	0.02/**0.01**/0.22	0.35/**0.23**/0.27	0.33/**0.23**
	93	**0.01/0.01**/0.21	**0.14**/0.24/0.22	**0.14**/0.25

[1] The method in [9]. [2] The method in [18].

Table 7. Coefficient of variation for MSE, running time, and iteration count (real data).

Data	K	MSE C.V. Ours/BK [1]/BSCK [2]	Time C.V. Ours/BK [2]/BSCK [2]	Iterations C.V. Ours/BK [1]
Iris	3	**0/0**/0.89	0.37/0.7/**0.07**	0.6/**0.37**
	9	**0.03**/0.04/0.33	0.51/0.44/**0.06**	0.6/**0.45**
	21	**0.03**/0.07/0.26	0.24/0.25/**0.05**	0.31/**0.27**
	45	**0.07**/0.17/0.18	0.2/0.17/**0.04**	0.27/**0.18**
	93	0.4/**0.09**/0.12	**0.02**/0.17/0.03	**0**/0.2
Wine	3	**0/0**/0.02	**0.03**/0.22/0.11	**0**/0.19
	9	0.03/**0.02**/0.15	0.29/0.35/**0.05**	0.36/**0.33**
	21	0.03/**0.02**/0.18	0.36/0.24/**0.09**	0.45/**0.26**
	45	0.04/**0.03**/0.13	0.24/**0.22**/0.03	0.33/**0.31**
	93	0.07/**0.05**/0.09	0.32/0.13/**0.11**	0.39/**0.1**
Thyroid	3	**0/0**/0.37	0.07/0.09/**0.05**	**0.07/0.07**
	9	**0.09/0.09**/0.18	0.31/0.35/**0.06**	0.37/**0.27**
	21	**0.01/0.01**/0.2	0.24/0.13/**0.06**	0.25/**0.13**
	45	**0.03/0.03**/0.12	**0.11/0.11**/0.13	0.14/**0.13**
	93	0.07/0.08/**0.06**	0.21/0.12/**0.04**	**0.28**/0.29
HV	3	**0/0**/0.06	2.61/**0.02**/2.37	**0/0**
	9	**0**/0.01/0.25	1.08/**0.07**/0.35	**0**/0.21
	21	**0**/0.03/0.16	0.15/**0.08**/0.11	0.15/**0.1**
	45	**0**/0.04/0.1	0.19/**0.09**/0.12	**0.11**/0.12
	93	0.06/0.09/**0.02**	0.07/**0.06**/0.11	**0.12**/0.13
ACS	3	**0/0**/0.04	0.96/**0.03**/0.98	**0/0**
	9	**0.02/0.02**/0.28	0.4/0.33/**0.25**	**0.43**/0.48
	21	**0.01/0.01**/0.21	0.3/0.29/**0.24**	0.31/**0.26**
	45	**0.01**/0.02/0.19	0.2/0.3/**0.19**	**0.19**/0.31
	93	**0.01/0.01**/0.04	0.15/0.15/**0.08**	**0.15**/0.19

[1] The method in [9]. [2] The method in [18].

5. Discussion

As we can see in Tables 4 and 5, in most cases, our method produces a lower MSE than the method based on the Hungarian algorithm [9]. However, there are situations when the method in [9] achieves equal or even better MSE, such as the cases when dividing the dataset R2 into 3 clusters, dividing the dataset Iris into 3 clusters, and dividing R5 into 45 clusters. There are two main reasons for this. (1) When the number of observations n is divisible by the number of clusters k, the Hungarian algorithm is more likely to get the same result as our assignment method and this will lead to an equal MSE. (2) From Tables 2 and 3, we can see that the proposed method always produces a balanced assignment with equal or lower cost. However, it is an iterative method and does not always converge to the global optimum.

When the number of observations n is not divisible by the number of clusters k, the sizes of clusters vary from $\lfloor n/k \rfloor$ to $\lceil n/k \rceil$. The proposed method could adaptively determine the size of each cluster during the process of optimization, while for the method based on the Hungarian algorithm, the sizes of clusters must be pre-settled. The awkward situation is that it is not known which clusters should have size $\lfloor n/k \rfloor$, and which clusters should have size $\lceil n/k \rceil$ in advance. Thus, theoretically, in each assignment step, the Hungarian algorithm needs to be executed $C^{n}_{n\%k}$ times in order to get the optimal assignment, where % is the modulus operator.

Our proposed method is suitable for applications which impose upper and (or) lower bounds on the sizes of clusters. The balanced size-constraint clustering in this paper is actually a special case when the upper bounds are $\lceil n/k \rceil$ and the lower bounds are $\lfloor n/k \rfloor$. It is straightforward to see that the constraint matrix is totally unimodular when there are both upper bounds and lower bounds on the sizes of clusters. In situations when only upper (lower) bounds are available, the constraint matrix would still be totally unimodular. We will skip the proof here, as it is similar to the proof to Theorem 1.

6. Conclusions

In this paper, we propose a novel approach to address the issue of balanced clustering which has many applications in practice. Each iteration of the proposed algorithm consists of two steps, namely an assignment step and an update step. In the assignment step, observations are evenly assigned to different clusters, and the problem is modeled as an ILP. We prove that the constraint matrix of the ILP is totally unimodular so that the ILP can be efficiently solved with the simplex algorithm. In the update step, the new centers are updated as the centroids of the observations in the clusters. The regression result suggests that the average time complexity of the proposed method is $O(mn^{1.65})$–$O(mn^{1.70})$, which is much faster than the $O(mn^3)$ method based on the Hungarian algorithm. Experiments on both synthetic and real data validate that the proposed method achieves both high quality and efficiency in the task of clustering with balanced size constraints.

There are several issues we can consider in our future work. For instance, the proposed method could be adapted into a general size-constraint clustering algorithm, and the clustering accuracy could be evaluated. Moreover, the proposed algorithm is a variant of the k-means algorithm. The hard size constraints are incorporated into the objective function of the k-means algorithm. Similarly, the hard constraints could be applied to other clustering algorithms, such as spectral clustering. Furthermore, in addition to the cluster-level constraints considered in this paper, instance-level constraints could be studied. For example, the observations o_i and o_j must (or must not) be in the same cluster. Finally, it is interesting to investigate the applications of the proposed method.

Author Contributions: Conceptualization, Y.Y. and W.T.; methodology, Y.Y., W.T. and L.Z.; writing original draft preparation, W.T. and Y.Y.; writing-review and editing, L.Z. and Y.Z.; supervision, Y.Z.; funding acquisition, Y.Y. and W.T.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 61402205), China Postdoctoral Science Foundation (Grant No. 2015M571688), and Postgraduate Research & Practice Innovation Program of Jiangsu Province (Grant No. KYCX18_2258).

Acknowledgments: The authors are deeply thankful to the editors and reviews for their valuable suggestions to improve the quality of this manuscript.

Conflicts of Interest: We declare that we have no conflict of interests. And this article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Xu, R.; Wunsch, D.C. Survey of clustering algorithms. *IEEE Trans. Neural Netw.* **2005**, *16*, 645–678.
2. Yang, Y.; Padmanabhan, B. Segmenting customer transactions using a pattern-based clustering approach. In Proceedings of the International Conference on Data Mining, Melbourne, FL, USA, 19–22 November, 2003; pp. 411–418.
3. Liao, Y.; Qi, H.; Li, W. Load-Balanced Clustering Algorithm With Distributed Self-Organization for Wireless Sensor Networks. *IEEE Sens. J.* **2013**, *13*, 1498–1506.
4. Hagen, L.; Kahng, A. Fast spectral methods for ratio cut partitioning and clustering. In Proceedings of the IEEE International Conference on Computer-Aided Design, Santa Clara, CA, USA, 11–14 November, 1991; pp. 10–13.
5. Issal, C.; Ebbesson, M. Document Clustering. *IEEE Swarm Intel. Symp.* **2010**, *38*, 185–191.
6. Dengel, A.; Althoff, T.; Ulges, A. Balanced Clustering for Content-Based Image Browsing. *Gi-Informatiktage* **2008**, 27–30.
7. Banerjee, A.; Ghosh, J. Frequency-sensitive competitive learning for scalable balanced clustering on high-dimensional hyperspheres. *IEEE Trans. Neural Netw.* **2004**, *15*, 702–719.
8. Koberstein, A. Progress in the dual simplex algorithm for solving large scale LP problems: techniques for a fast and stable implementation. *Comput. Optim. Appl.* **2008**, *41*, 185–204.
9. Malinen, M.I.; Fränti, P. Balanced *k*-means for Clustering. In Proceedings of Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Joensuu, Finland, 20–22 August, 2014; pp. 32–41.
10. Mardia, K.V.; Kent, J.T.; Bibby, J.M. Multivariate analysis. *Math. Gazette* **1979**, *37*, 123–131.
11. Grossi, V.; Romei, A.; Turini, F. Survey on using constraints in data mining. *Data Mining Knowl. Discov.* **2017**, *31*, 424–464.
12. Banerjee, A.; Ghosh, J. Scalable Clustering Algorithms with Balancing Constraints. *Data Mining Knowl. Discov.* **2006**, *13*, 365–395.
13. Luxburg, U.V. A tutorial on spectral clustering. *Stat. Comput.* **2007**, *17*, 395–416.
14. Chen, Y.; Zhang, Y.; Ji, X. Size Regularized Cut for Data Clustering. In Proceedings of the Advances in Neural Information Processing Systems 18, Vancouver, British Columbia, Canada, 5–8 December, 2005; pp. 211–218.
15. Shi, J.; Malik, J. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **1997**, *22*, 888–905.
16. Kawahara, Y.; Nagano, K.; Okamoto, Y. Submodular fractional programming for balanced clustering. *Pattern Recognit. Lett.* **2011**, *32*, 235–243.
17. Chang, X.; Nie, F.; Ma, Z.; Yang, Y. Balanced *k*-means and Min-Cut Clustering. *arXiv* **2014** *arXiv:1411.6235* . Available online: https://arxiv.org/abs/1411.6235 (Accesed on 5 March 2019).
18. Zhu, S.; Wang, D.; Li, T. Data clustering with size constraints. *Knowl.-Based Syst.* **2010**, *23*, 883–889.
19. He, R.; Xu, W.; Sun, J.; Zu, B. Balanced *k*-means Algorithm for Partitioning Areas in Large-Scale Vehicle Routing Problem. In Proceedings of the International Symposium on Intelligent Information Technology Application, NanChang, China, 21–22 November, 2009; pp. 87–90.
20. Tai, C.L.; Wang, C.S. Balanced *k*-means. In *Intelligent Information and Database Systems*; Nguyen, N.T., Tojo, S., Nguyen, L.M., Trawiński, B., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 75–82.
21. Bennett, K.; Bradley, P.; Demiriz, A. *Constrained k-Means Clustering*; Technical report; Microsoft Research: Redmond, WA, USA, 2000.
22. Yuepeng, S.; Min, L.; Cheng, W. A Modified *k*-means Algorithm for Clustering Problem with Balancing Constraints. In Proceedings of the International Conference on Measuring Technology and Mechatronics Automation, Shanghai, China, 6–7 January, 2011; pp. 127–130.

23. Ganganath, N.; Cheng, C.T.; Chi, K.T. Data Clustering with Cluster Size Constraints Using a Modified *k*-means Algorithm. In Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Shanghai, China, 13–15 October, 2014; pp. 158–161.
24. Arthur, D.; Vassilvitskii, S. *k*-means++: The advantages of careful seeding. In Proceedings of the Eighteenth ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, USA, 7–9 January, 2007; pp. 1027–1035.
25. Papadimitriou, C.H.; Steiglitz, K. *Combinatorial Optimization: Algorithms and Complexity*; Prentice Hall: Mineola, NY, USA, 1998.
26. Schrijver, A. *Theory of Linear and Integer Programming*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1986.
27. Spielman, D.A.; Teng, S. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. *J. ACM* **2004**, *51*, 385–463.
28. Borgwardt, K.H. *The Simplex Method: A Probabilistic Analysis*; Springer Science & Business Media: Berlin, Heidelberg, Germany, 1987.
29. Fang, S.C.; Puthenpura, S. *Linear Optimization and Extensions: Theory and Algorithms*; Prentice-Hall: Upper Saddle River, NJ, USA, 1993.
30. Dheeru, D.; Taniskidou, E.K. *UCI Machine Learning Repository*; University of California: Irvine, CA, USA, 2019.

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).