Int. J. Data Mining and Bioinformatics, Vol. X, No. Y, XXXX

K-walks: clustering gene-expression data using a *K*-means clustering algorithm optimised by random walks

Min Yao

College of Animal Science,
Yangtze University,
Jingzhou, Hubei 434025, China
Email: minyao01@126.com

Qinghua Wu

College of Life Science,
Yangtze University,
Jingzhou, Hubei 434025, China
Email: wqh212@hotmail.com

Juan Li

College of Chemistry,
Xiangtan University,
Xiangtan, Hunan 411105, China
Email: juanli@xtu.edu.cn

Tinghua Huang*

College of Animal Science,
Yangtze University,
Jingzhou, Hubei 434025, China
Email: thua45@126.com
*Corresponding author

Abstract: Gene-expression data obtained from the biological experiments always have thousands of dimensions, which can be very confusing and perplexing to biologists when viewed as a whole. Clustering analysis is an explorative data-mining technique for statistical data analysis that is widely used in gene-expression data analysis. Practical approaches employed for solving the clustering problem use iterative procedures such as *K*-means, which typically converge to one of many local minima. Here, we propose a simulated annealing approximation algorithm that is optimised using random walks to solve the *K*-means clustering problem. The algorithm is verified with synthetic and real-world data sets and compared with other well-known *K*-means variants. The new algorithm is less sensitive to initial cluster centres, and the primary strength of our algorithm is its ability to produce high-quality clustering results for thousands of high-dimensional data. However, the algorithm is computationally intensive.

Keywords: gene expression; *K*-means; random walks.

Reference to this paper should be made as follows: Yao, M., Wu, Q., Li, J. and Huang, T. (XXXX) '*K*-walks: clustering gene-expression data using a *K*-means clustering algorithm optimised by random walks', *Int. J. Data Mining and Bioinformatics*, Vol. X, No. Y, pp.xxx–xxx.

Biographical notes: Min Yao is a Research Scientist in Yangtze University, China. His research includes developing of new bioinformatics tools for animal science, genetics, and molecular biology.

Qinghua Wu is an Associate Professor in Yangtze University, China. His research focuses on food and drug toxicology.

Juan Li is a Research Scientist in Xiangtan University, China. Her research includes drug metabolism, toxicology, and pharmacology.

Tinghua Huang is a Research Scientist in Yangtze University, China. His research includes developing of new bioinformatics tools for animal science, genetics, and molecular biology.

1 Introduction

Gene-expression data obtained from microarray experiments have inspired several applications, such as the identification of differentially expressed genes for molecular studies (Huang et al., 2011; Huang et al., 2008) and the creation of gene classifiers for improved diagnoses of diseases such as cancer (Fu et al., 2015; Kang et al., 2010). Traditional algorithms such as clustering have proven to be useful for identifying biologically relevant gene clusters for different biological status. The primary objective of cluster analysis is to group together similar genes and separate dissimilar genes by assigning them to different clusters. Iterative refinement methods such as *K*-means are the most effective (Jain and Dubes, 1988; Anderberg, 1973; Ball and Hall, 1967; MacQueen, 1967; Jain et al., 1999). For a given set of *N* gene data points, the *K*-means clustering method aims to solve the clustering problem by determining a set of centres to minimise the squared error distortion or the mean squared distance from each data point to its nearest centre. Lloyd's algorithm or the *K*-means algorithm, one of the most popular algorithms used for solving the *K*-means problem, is based on a simple iterative process to obtain the optimal solution (Lloyd, 1982; Forgey, 1965; MacQueen, 1967). Lloyd's algorithm first randomly initiates a number of centres in a data set. For each iteration, data objects are assigned to their nearest centres. Each centre is then updated with the centroid of its neighbourhood, which is the weighted average of the data points assigned to the centre. The iteration process repeats until the termination condition has been satisfied.

Lloyd's algorithm is computationally efficient and yields good results if the clusters are compact, spherical, or diagonal in shape, and well separated in the feature space of the data sets (Jain, 2010). However, the algorithm is particularly sensitive to initial starting conditions and always converges to a point that is the local minimum (Selim and Ismail, 1984). Lloyd's algorithm uses a random-seeding technique to initiate the centres. Each clustering run that is initiated with a different initial configuration converges to different local minima and produces different results. However, the optimal clustering solution may not be obtained.

Different global-seeding procedures have been introduced into the algorithm to improve the consistency and quantity of clustering results. Peña et al. (1999) empirically compared four *K*-means algorithm initialisation methods: random, Forgy, MacQueen, and Kaufman. These authors showed that the Kaufman initialisation method outperformed the other methods because it enhanced the effectiveness of the *K*-means algorithm and its robustness to the initial clustering configuration (Kaufman and Rousseeuw, 1990). Celebi et al. (2013) compared eight commonly used *K*-means initialisation methods on a large and diverse collection of data sets using various performance criteria. These authors demonstrated that Bradley and Fayyad's (BF98) method consistently appeared in the best-performing group (Bradley and Fayyad, 1998). In time-critical applications that involve large data sets or applications that demand determinism, the Var-Part or PCA-Part methods should be used (Ting and Jennifer, 2007). In applications that involve small data sets, the BF98 method or greedy *K*-means++ should be used (Bradley and Fayyad, 1998; Arthur and Vassilvitskii, 2007). Peterson et al. (2010) presented a detailed assessment of the performance of many commonly used well-performing initialisation methods over different data sets. The top-performing initialisation methods in terms of minimising the objective function (within group sum of squares) were the BF98 and Mirkin's method (Mik05) over both synthetic and real-world data sets (Mirkin, 2005; Bradley and Fayyad, 1998). The BF98 method employed a bootstrap-type procedure to determine the initial seeds for the centres. Several subsamples (recommended $n = 10$) of the data set were clustered using *K*-means. Each clustering operation produced a different candidate set of centroids from which a new data set was constructed. This data set was clustered using *K*-means, and the centroids were chosen as the initial seeds. The Mik05 method used a MaxMin procedure for initialisation. This procedure attempted to find data entities as the initial seeds for the clustering algorithm that were well separated from one another in the feature space of the data set. Peterson et al. demonstrated that the BF98 method was the best performing over the largest number of data sets. The Mik05 method performed better than other data sets the largest number of times. However, the results revealed that there was no *K*-means initiation (seeding) method that performed the best across all of the data sets because the true model properties of a data set cannot truly be known (Peterson et al., 2010).

In the recently studies, Ping et al. introduced random walks into a graph-based constrained clustering algorithm known as SCRAWL (He et al., 2014). This algorithm is composed of two random walks with different granularities. In the lower-level random walk, SCRAWL partitions the vertex set into the constrained and unconstrained two vertex subsets. In the higher-level random walk, SCRAWL derives the affecting scope of each pair-wise constraint, which is the edge set connecting the components around the two constrained vertices. Che et al. (2012) introduced a simulated annealing algorithm into mathematical optimisation models for clustering and the selection of suppliers. The algorithm integrates *K*-means and a simulated annealing with the Taguchi method to solve the problem of analysing supplier clusters according to customer-demand attributes, including production cost, product quality and production time. Huang et al. (2014) extended the existing *K*-means algorithms by integrating both intracluster compactness and intercluster separation methods. Furthermore, there few other software packages developed specifically for gene-expression data analysis and motif discovery such as WGCNA, which is an *R* package for weighted correlation network analysis (Langfelder and Horvath, 2008), and HHK, which is a modified Hybrid hierarchical *K*-means clustering algorithm for protein sequence analysis (Chen et al., 2010).

In this study, we propose a *K*-walks clustering algorithm that includes a simulated annealing (SA) approximation algorithm with the *K*-means algorithm as a local search procedure for minimising the mean squared distance. The proposed algorithm is an extension of the standard *K*-means algorithm, in which random walks are introduced in the iteration process of the algorithm to ensure that the clustering process is less sensitive to the initial cluster centres. The basic idea of the proposed method is that the random walks can push the centres moving out of the local optimal. The *K*-walks clustering algorithm was verified using synthetic and real data sets. The algorithm consistently produced better clustering results than Lloyd's algorithm, and it compared favourably with the BF98 and Mik05 algorithms. Application of *K*-walks in microarray data clustering demonstrated that the algorithm is still very sensitive to local minima but performs very well in recovering the true clusters and appears to be superior to its competitors (i.e. the *K*-means, BF98, and Mik05 algorithms).

The remainder of this paper is organised as follows. In the methods section, we introduce the *K*-walks clustering algorithm. In the results section, the important properties of the algorithm are investigated using synthetic data sets, and we apply the *K*-walks clustering algorithm to real-world data sets as well as gene-expression data sets. The paper concludes with a brief discussion section.

2 Methods

The proposed *K*-means clustering algorithm is an iterative process that partitions data into disjoint groups by assigning them to the nearest representative centres. In each cycle, the centres move towards their position of expectation until they converge (Peña et al., 1999). The input data sets of variables are denoted as $X = \{X_1, X_2, X_3, \ldots, X_n\}$, and the current partitions of the data are denoted as $P = \{P_1, P_2, P_3, \ldots, P_K\}$. The proposed *K*-means clustering algorithm minimises the objective function of squared error distortion (equation (1)), which is the sum of the Euclidean distance between each data object and its nearest cluster centres (centroid).

$$M = \sum_{i=1}^{K} \sum_{j=1}^{k_i} \left\| X_{ij} - C_i \right\|^2 \tag{1}$$

Here, $\|\cdot\|$ is a Euclidean norm, and *K* and K_i denote the number of clusters and objects of the cluster *i*, respectively. Further, X_{ij} is the *j*th object of the *i*th cluster, and C_i is the centroid of the *i*th cluster expressed as equation (2).

$$C_i = \frac{1}{K_i} \sum_{j=1}^{K_i} X_{ij}, i = 1, 2, 3, \ldots, K \tag{2}$$

The algorithm is initiated with the cluster centres first placed either at the centre of the data set or randomly in a multidimensional clustering space. The centres are then moved to the optimised positions to minimise the objective function. Details of the algorithm are discussed below (Figure 1).

Step 1: Initialisation: randomly assign each data object to one of the centres. Let $C = \{C_1, C_2, C_3, \ldots, C_K\}$ represent the centres, where C_K is the *K*th centre. The random walks step size (σ) is set to be zero.

Step 2: Random walk: apply random walks to the centres. Let the new centres be as in equation (3).

$$C'_i = C_i + F_G(\sigma) \tag{3}$$

The $F_G(\sigma)$ function returns a Gaussian random variant with zero mean and a standard deviation of σ.

Step 3: New means calculation: find the minimum distance between all of the cluster centres (C') and each data object (X_i). Assign the data objects to the nearest centres of U_i using equation (4) and then update the cluster centres $C' = \{C'_1, C'_2, C'_3, \ldots, C'_K\}$ using the centroid of the data objects assigned to the centre C'.

$$U_i = min_k \|X_i - C'_k\|^2 \tag{4}$$

Step 4: Distortion factor calculation: the distortion factor measures the distance by which the centres were moved, where *n* represents the number of data objects (equation (5)).

$$D = \sqrt{\frac{\sum_{i=1}^{K} \|C'_i - C_i\|^2}{K \cdot n}} \tag{5}$$

Step 5: Evaluate the candidate configuration produced by Step 3: if the squared error distortion decreases, accept the candidate configuration. Set the random walks step size to the value of the distortion factor if the distortion factor is greater than the random walks step size. Otherwise, reject the candidate configuration and decrease the step size of the random walks using equation (6).

$$\sigma = \sigma_{last}(1 - \Delta) \tag{6}$$

σ and σ_{last} represent the step size and the step size of the last iteration, respectively, and Δ represents the decreasing ratio, which is a user-determined parameter.

Step 6: Start at Step 2 and repeat until the centres no longer move (i.e. the algorithm converges).

The algorithm is developed on the basis of the general principle of random walk theorem and SA approximation procedure. During the iteration process, the centres gradually move towards the gravitational basin of the data objects. However, the random walks introduce random noise into the moving process. In each step of the random walks method, the centres randomly jump to a neighbouring position according to some probability distribution. In this simple random walks technique, the centres can only jump to the neighbouring position with the step size varying according to a Gaussian distribution (i.e. a Gaussian random walk). The probabilities of the centres jumping towards any direction in a multidimensional clustering space are the same. The step size determines the amount of noise that will be introduced into the system by the random walks. When the step size increases, the clustering can more easily move out from a local optimum. As the step size decreases, the clustering will converge more rapidly.

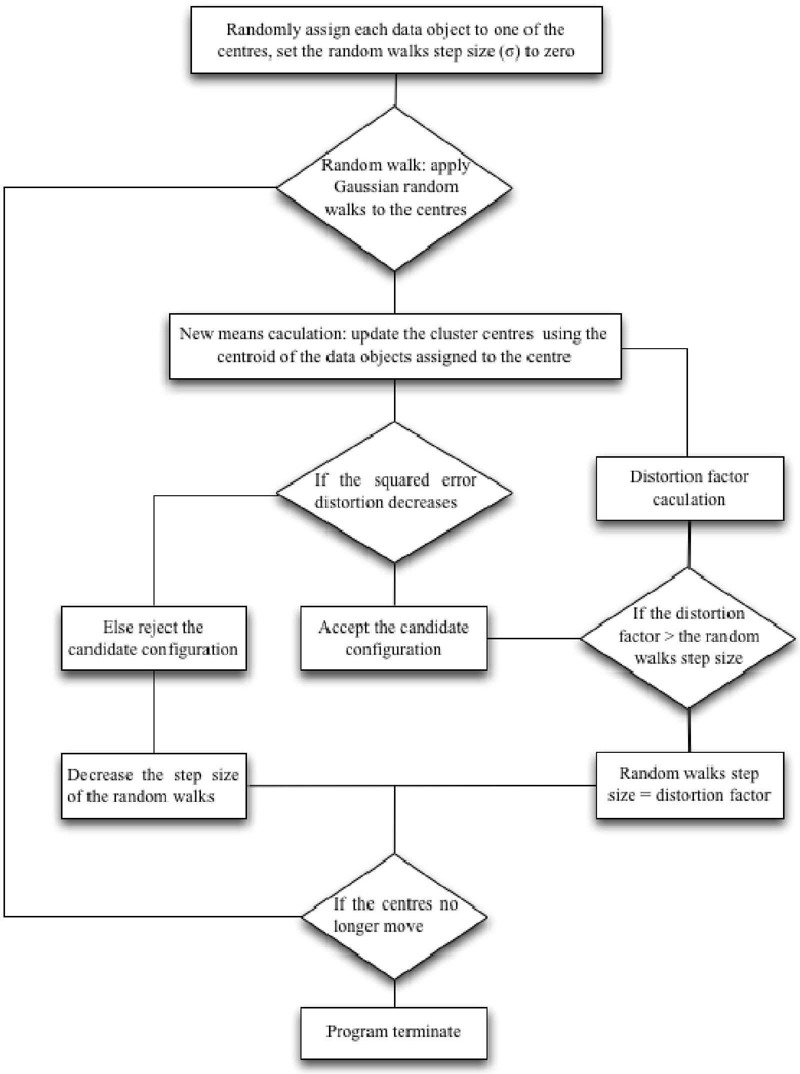
An SA technique that involves heating and controlled cooling process was used in the *K*-walks algorithm (a hybrid approach). The heating process increases the step size (σ), thereby increasing the amount of noise input by the random walks. This situation caused the centres to become more active, wandering randomly so as to be able to walk out of a local minimum. The slow cooling process gradually decreased the step size (σ), and the amount of noise input into each iteration was lower, which provided the centres with more opportunities to find the optimal position that minimises the objective function. In the SA algorithm, the acceptance or rejection of a clustering configuration depends on the cost function of the objective function gain (equation (7)).

$$\Delta M = M_c - M_l \tag{7}$$

M_c and M_l denote the value of the objection of the current and last clustering configurations, respectively. Asymptotic convergence of the clustering process is guaranteed via a bidirectional progressing temperature schedule. If $\Delta M > 0$, the value of the objective function is decreasing, and the SA accepts the current clustering configuration and simultaneously adjusts the random walks step size. If $\Delta M < 0$, the value of the objective function is increasing and the SA rejects the current clustering configuration and simultaneously reduces the random walks step size. A new parameter Δ is introduced into the system to control the amount by which the walk step size must be reduced in every iteration cycle. To simplify the algorithm, we reduced the walk step size in a fixed ratio in an iterative process.

The algorithm initiates the clustering process with the random walks step size set to zero. At the end of each iteration cycle, the distortion factor is calculated and used to measure the variation in the position of centres. The random walks step size is then set to the distortion factor value (if σ is less than the distortion factor) to introduce the same amount of noise. This design avoids the use of ad hoc parameters in the clustering algorithm and also avoids improperly setting the walk step size parameters, which can result in a computationally intensive algorithm (setting the parameter too high) or, on the other hand, poor performance (setting the parameter too low).

Figure 1 Flow chart of the *K*-walks clustering algorithm


Randomly assign each data object to one of the centres, set the random walks step size (σ) to zero
Random walk: apply Gaussian random walks to the centres
New means caculation: update the cluster centres using the centroid of the data objects assigned to the centre
If the squared error distortion decreases
Distortion factor caculation
Else reject the candidate configuration
Accept the candidate configuration
If the distortion factor > the random walks step size
Decrease the step size of the random walks
Random walks step size = distortion factor
If the centres no longer move
Program terminate

3 Results

3.1 Performance of K-walks with synthetic data sets

The synthetic data, n = 10,000 points, were generated from a distribution comprising K = 36 multivariate Gaussian centres (Gaussian distribution of objects around a centre) of different sizes. All of the Gaussian centres were uniformly distributed in a two-dimensional square of side 4.0. Each cluster was drawn from a Gaussian distribution coordinate centred at the centre of the cluster and a given standard deviation $\sigma = \sqrt{2}$.

Therefore, the synthetic data sets consisted of a combination of 36 clusters centred about an even grid point. Hence, the optimal clustering solution of the synthetic data set consisted of 36 centres placed at regular grid points (Figure 2).

We conducted three experiments to investigate the ability of the random walks to move the centres and the capability of the SA algorithm to refine the clustering process. In the first experiment, the centres were initiated at the middle (centroid) of the data set, and Δ was set to 0.001 (after trying Δ with 0.1, 0.01, and 0.001). The experiment was repeated 200 times, and the clustering configuration obtained with the lowest squared error distortion is shown in Figure 2. The result shows that, after a number of iterations, the clustering algorithm converges at a configuration in which the centres are located very close to the expected positions (Figure 2). Our findings indicate that the random walks of the *K*-walks algorithm can push the centres towards the 36 Gaussian centres. Overall, the *K*-walks method yielded 98.3% classification accuracy, and the BF98 and Mik05 methods demonstrated 95.2% and 94.1% classification accuracies, respectively.

In the second experiment, we tested the bidirectional progressing temperature schedule using synthetic data sets. The centres were initiated at the middle (centroid) of the data set, and Δ was set to 0.001. The profiles of the values of the squared error distortion for each iteration were recorded. The squared error distortion profiles produced by the *K*-walks clustering algorithm were compared with the *K*-means clustering algorithm (Figure 3). Both the *K*-walks and *K*-means experiments were repeated 100 times. Our results revealed that, compared to the *K*-means clustering algorithm, the squared error distortion of the *K*-walks algorithm decreased more slowly. A plateau can be observed in which the squared error distortion values fluctuate, drop, and increase for a long time and finally stabilise at a point where the value of squared error distortion is slightly less than that of the *K*-means clustering algorithm. We also noted that the *K*-walks clustering required a larger number of iterations to converge than the *K*-means clustering. Our results revealed that the bidirectional progressing temperature schedule of the SA technique was valid.

In the third experiment, we investigated random walks. The centres were initiated at the middle (centroid) of the data set, and Δ was set to 0.001. We tested two *K*-walk configurations. The first configuration was initiated with the walk step size set to an arbitrary value of 10.0. The second configuration of the algorithm was set to start with a walk step size of zero. The σ profiles for each iteration were recorded (Figure 4). Our results revealed that the walk step size continued to drop to zero in the first configuration. In the second configuration, the random walk step size first increased to approximately 4.0 and then gradually decreased to zero. After a number of iterations, both the experiments converged at a configuration that was close to the global optimal. Our results reveal that the algorithm can automatically determine the amount of noise (from the random walk step size) that must be introduced into the system by the random walks.

3.2 *Performance of K-walks with real-world data sets*

The performance of the *K*-walks clustering algorithm was evaluated using nine standard real-world data sets proposed previously (Peterson et al., 2010). The data sets are briefly discussed as follows. The Ruspini data set is a synthetic data set containing n = 75 and p = 2 dimensions observations from K = 4 well-separated groups (Ruspini, 1970). The Image, Cloud-1, Cloud-2, and Spam data sets were obtained from the University of

California, Irvine's machine-learning repository (Bache and Lichman, 2013). The last four data sets were images obtained from the Signal and Image Processing Institute of the University of Southern California (USC) (SIPI, http://sipi.usc.edu). These data sets are colour-quantised images in which the data sets are only in $p = 3$ dimensions and with a larger number of observations ($n = 65{,}536$ or $n = 262{,}144$). All of these data sets have been used very often in studies to evaluate clustering methods.

Figure 2 Optimal clustering solution of the synthetic data set obtained with the *K*-walks clustering algorithm. The *K*-walks clustering algorithm was initiated with all of the centres at the centre of the data set (mean vector). After a number of iterations, the clustering converged at a configuration very close to the global optimum (see online version for colours)

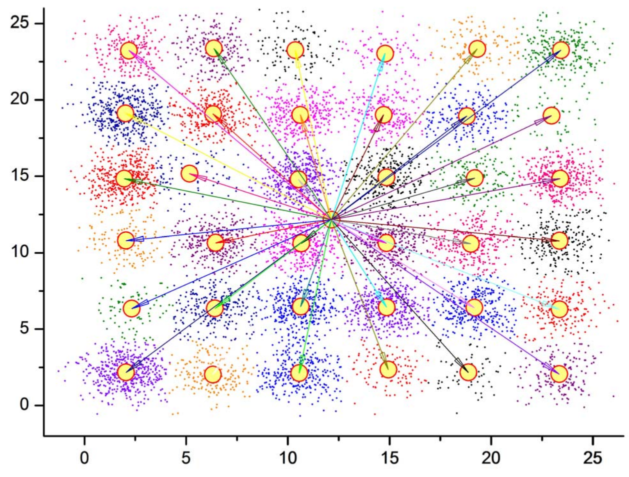


Figure 3 Comparison of the squared error distortion profiles of the *K*-walks and *K*-means clustering algorithms. The square error distortion of the *K*-walks algorithm drops more slowly and converges at a point lower than that of the *K*-means algorithm

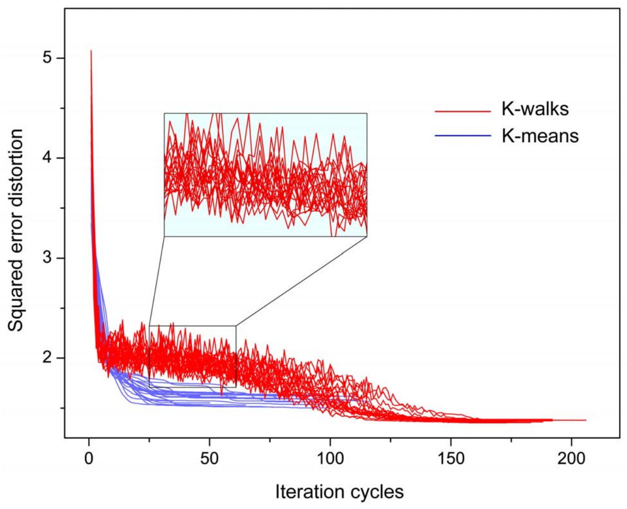
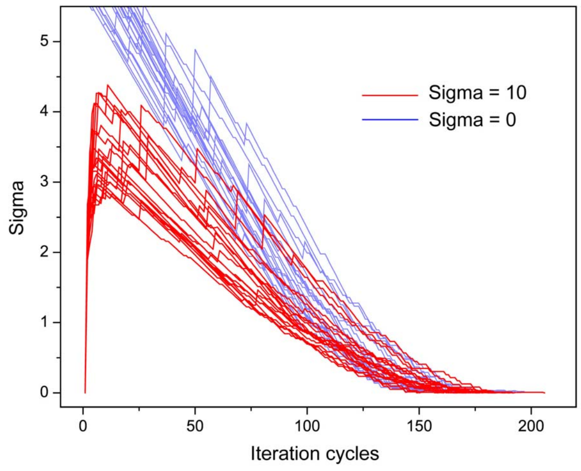


Figure 4 *K*-walks step size (σ) profiles of the *K*-walks clustering algorithm for $\sigma = 0$ and 10. Both of the clustering algorithms converge at the global optimum



Class information pertaining to the Image and Ruspini data sets is available. The information of the number of clusters of these two data sets, $K = 7$ and 4, was used to evaluate the performance of the clustering algorithm. The other data sets are truly real-world data sets, and their class information was unknown. We assumed that the number of clusters was known and evaluated the performance for $K = 10$, 25, and 50. All of the data sets were evaluated in terms of achieving the optimum objective function.

After the data sets were collected, we used the *K*-walks procedure outlined above to cluster the data. Each data set was replicated 100 times, and the achieved optimum of the objective function of the final grouping solution was recorded. The two top-performing initialisation methods (BF98 and Mik05) were also assessed and compared with the *K*-walks clustering algorithm. All of the programs were implemented as GNU Compiler Collection console applications and executed on a Linux system (source code available at http://sourceforge.net/projects/clustermx under academic free license). All of the computations were performed using Amazon Elastic Compute Cloud (Amazon web service, Tokyo).

When analysing the results, we assessed the overall performance in terms of objective function minimisation (Table 1). The *K*-walks method performs the best and is followed closely by the BF98 and Mik05 methods. The *K*-means algorithm performs the worst. The *K*-walks method is among the top performers in at least 13 of the 23 cases. In three out of the 23 clustering runs, the BF98 method produced the best configuration. In six runs, the Mik05 method produced the best configuration; in one run, the *K*-means method produced the best configuration. A duplicate run of the *K*-walks algorithm produced more stable results, and, as expected, it outperformed the BF98, Mik05, and *K*-means algorithms. Moreover, the *K*-means method was the worst performer in ten cases. The BF98 and Mik05 methods were the worst performers in four and eight cases, respectively, and the *K*-walks method was the worst performer in one case.

For the Image data set, the BF98 method was the best performer, closely followed by the *K*-walks method. The *K*-means method performed more poorly than the BF98 and *K*-walks methods. Moreover, the Mik05 method was the worst performer. For the more easily well-separated Ruspini classification data set, the *K*-walks method was able to perfectly partition the data; the other methods were poor performers in terms of identifying the correct partition. For the two cloud data sets, the *K*-walks method was the best performer in all the six cases, and the BF98 and *K*-means methods were the worst performers. For the Spam data set, the Mik05 method was the top performer in all three cases. For the four image data sets (i.e. Lena, Baboon, Peppers, and Couple), different methods performed well on different data sets and different values of *K*. It should be noted that for large *K* values ($K = 25$ and $K = 50$), the *K*-walks method performed the best in seven out of the 12 cases. For small *K* values ($K = 10$), the BF98 method performed well on two out of the three data sets in terms of achieving the lowest value for the objective function, and the Mik05 method performed well on the other data set. Based on the results of the four images, we can conclude that the *K*-walks algorithm performs better than the BF98 and Mik05 algorithms for high-dimensional data and large *K* values. The variation in the quality of solutions is smaller than that associated with the BF98 and Mik05 methods, thereby implying that fewer runs are required to guarantee a good solution.

Table 1 Performance of clustering methods using four strategies over nine data sets in terms of optimised objective function (squared error distortion)

Name	*K*	*K-walks (sed, stdev*)*	*BF98 (sed, stdev)*	*K-means (sed, stdev)*	*Mik05 (sed)*
Image	7	77.70, 1.83	**77.32, 1.82**	79.88, 9.718	101.91
Ruspini	4	**13.11, 0**	24.23, 8.24	22.74, 10.39	20.44
Cloud-1	10	**78.03, 1.31**	78.77, 6.91	87.11, 13.95	81.444
	25	47.14, 2.27	50.95, 5.86	58.04, 9.84	**47.14**
	50	**38.91, 1.05**	41.39, 4.05	45.66, 11.62	39.0
Cloud-2	10	**36.33, 0.41**	36.87, 0.97	36.92, 0.65	36.53
	25	**25.46, 0.21**	26.55, 0.79	26.40, 1.15	26.4
	50	**20.25, 0.33**	21.76, 0.88	21.08, 0.94	20.76
Spam	10	192.27, 2.73	187.95, 24.94	191.95, 0.0	**146.08**
	25	177.84, 5.10	159.35, 36.99	181.64, 0.64	**77.145**
	50	176.58, 5.63	141.39, 69.25	180.02, 2.92	**43.808**
Lena	10	18.21, 0.43	**18.20, 0.24**	18.24, 0.37	19.98
	25	**11.97, 0.07**	12.08, 0.24	11.98, 0.11	12.57
	50	**9.21, 0.03**	9.32, 0.11	9.22, 0.04	9.45
Baboon	10	30.44, 0.34	30.50, 0.35	30.66, 0.24	**30.32**
	25	**21.08, 0.01**	21.13, 0.13	21.10, 0.08	21.09
	50	**16.58, 0.04**	16.59, 0.03	16.60, 0.05	16.604
Peppers	10	25.12, 0.98	**24.79, 0.37**	24.80, 0.38	25.99
	25	16.77, 0.30	16.69, 0.18	**16.67, 0.16**	17.587
	50	**12.73, 0.16**	12.76, 0.23	12.79, 0.28	13.34
Couple	10	13.80, 0.16	13.97, 0.41	14.01, 0.38	**13.782**
	25	**9.40, 0.08**	9.44, 0.17	9.46, 0.15	9.624
	50	**7.16, 0.05**	7.31, 0.12	7.32, 0.12	7.32

Note: **sed* is the optimised value of the squared error distortion function, and *stdev* is the standard deviation of the squared error distortion of the duplicated clustering run.

3.3 Classification tests for gene-expression data sets

We tested the *K*-walks clustering algorithms on two microarray data sets. The first data set was a subset of gene-expression profiles over 11 time points collected during the process of bacterial cell division (Laub et al., 2000) and contained 340 gene-expression profiles with a standard deviation greater than 0.5 and no missing data points. The second data set was a subset of gene-expression profiles over 7 time points collected during the developmental program of sporulation in budding (Chu et al., 1998) and contained 495 gene-expression profiles with a standard deviation greater than 1.0 and no missing data points. These two data sets have been used to evaluate *K*-means clustering (Wu, 2008) and were originally publicly available in the Stanford microarray database (Sherlock et al., 2001). The two data sets are denoted 'BAC' and 'SPO' in this paper, respectively.

In our experiments, the number of clusters was set to 2, 4, 6, 8, and 10. The centres were initiated randomly distributed in the clustering space of the data set, and Δ was set to 0.001. We recorded the running time, obtained lowest squared error distortion, and memory usage. A comparison of the average running times and memory usage of the four clustering algorithms for the two data sets is shown in Table 2. For the BAC data set, the running time of *K*-walks is significantly higher than that of the BF98, *K*-means, and Mik05 methods for all five of the runs ($p < 0.05$). Based on our results, in two out of five runs the BF98 method required the least amount of time, and the *K*-means method required the least time for one of the five runs. In the remaining two clustering runs, the Mik05 method required the least amount of time. For the SPO data set, the running time of *K*-walks was again significantly higher than that of the BF98, *K*-means, and Mik05 methods for all five of the runs ($p < 0.05$). These results demonstrate that for three out of five runs the BF98 method required the least amount of time; the *K*-means method required the least amount of time for the two remaining clustering runs. The *K*-walks method performed the worst in terms of running time for both the BAC and SPO data sets, followed by the Mik05, *K*-means, and BF98 algorithms. Furthermore, the *K*-means algorithm used the least memory, followed closely by Mik05 and *K*-walks algorithms. The BF98 method used the largest amount of memory.

A comparison of the optimised values of the squared error distortion function of the four clustering algorithms for the two data sets is shown in Table 3. For the BAC data set, the *K*-walks method produced the smallest squared error distortion for four out of five runs. For two out of five runs, the BF98 method produced the best results. For $K = 2$, the *K*-walks and BF98 methods performed equally well. For $K = 4$, the BF98 method produced the best result. For K values larger than four, the *K*-walks method performed the best across all of the clustering runs. For four out of five runs with the SPO data set, the *K*-walks algorithm produced the smallest squared error distortion. For $K = 2$, the *K*-walks, BF98, and Mik05 clustering algorithms performed equally well. For $K = 4$, the Mik05 method produced the best results. When K was greater than 4, the *K*-walks and BF98 methods performed the best (the *K*-walks method performed the best for two cases, and the *K*-walks and BF98 methods performed equally well in one case).

A comparison of the average adjusted Rand index (ARI) of the four clustering algorithms for the two data sets is shown in Figure 5. In our experiments, the average

ARIs were computed over all of the clusters for the results of a number of repeated running of *K*-walks, BF98, and *K*-means clustering algorithms. There have been several attempts to evaluate clustering methods to recover true clusters in a data set (Dudoit and Fridlyand, 2002). A clustering problem can be considered to be a partition of objects into a number of groups; to evaluate a clustering result it is necessary to define a measure of agreement between two clustering partitions of the same data set. In this study, we used the ARI (Rand, 1971); its expected value is 1 when the clustering results match perfectly and 0 when the two partitions are selected at random. A large ARI value indicates that the two strongly partitions agree. To investigate the sensitivity of the clustering methods to initial partitions, the clustering method was run with numerous different random initial configurations. Then, the average ARI of all pair-wised clustering result was calculated. This average ARI indicates the sensitivity of the clustering method to initial partitions. The larger the average ARI, the more insensitive (better) the clustering method is to initial partitions.

The average ARI of the *K*-walks, BF98, *K*-means, and GWKMA methods for $K = 2 – 10$ were calculated from the results of ten clustering runs of both the BAC data set (upper panel of Figure 5) and the SPO data set (lower panel of Figure 5). The results showed that the average ARI values for the *K*-walks were greater than 0.75 for the BAC data set and greater than 0.65 for the SPO data set. For $K = 2$ and 4, there was no significant difference in the average ARI values for the three clustering algorithms tested. When *K* is greater than 4, the average ARI value of the *K*-walks method is significantly higher than that of the BF98 and *K*-means clustering methods ($p < 0.01$). The GWKMA algorithm performed almost as well as the *K*-walks algorithm on the BAC data sets. However, for the SPO data sets the average ARI was not very stable.

Table 2 Performance of the clustering methods using four strategies over two gene-expression data sets in terms of running time (seconds)

Name	*K*	*K-walks (time, stdev*, mem*)*	*BF98 (time, stdev, mem)*	*K-means (time, stdev, mem)*	*Mik05 (time, mem)*
BAC	2	0.035, 0.045, 0.62	0.405, 0.005, 0.83	**0.33**, 0.015, 0.61	0.56, 0.63
	4	1.29, 0.057, 0.63	**0.32**, 0.01, 0.882	0.695, 0.027, 0.62	0.535, 0.635
	6	1.685, 0.008, 0.636	**0.515**, 0.027, 0.91	0.53, 0.008, 0.628	0.525, 0.64
	8	2.47, 0.107, 0.64	0.8, 0.026, 0.955	1.02, 0.017, 0.63	**0.76**, 0.645
	10	2.88, 0.059, 0.656	1.155, 0.037, 1.11	1.265, 0.051, 0.64	**1.09**, 0.65
SPO	2	0.63, 0.007, 0.64	**0.08**, 0.004, 0.80	0.115, 0.007, 0.63	0.33, 0.63
	4	0.74, 0.027, 0.648	**0.255**, 0.005, 0.83	0.855, 0.036, 0.64	0.46, 0.64
	6	2.375, 0.041, 0.668	1.345, 0.022, 0.87	**1.02**, 0.025, 0.651	1.14, 0.65
	8	3.235, 0.049, 0.680	**0.845**, 0.058, 0.92	1.465, 0.081, 0.67	1.655, 0.672
	10	3.48, 0.140, 0.696	1.93, 0.082, 1.02	**1.78**, 0.094, 0.68	1.98, 0.68

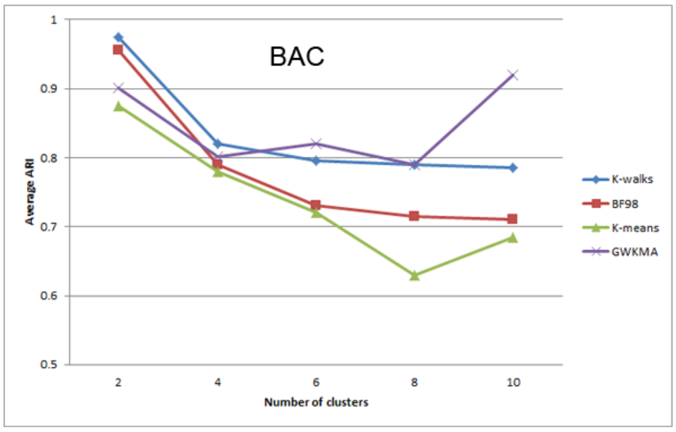
Note: **time* is the averaged value of the running time, *stdev* is the standard deviation of the running time of duplicated clustering run, and **mem* is the averaged value of the memory usage (Mb).

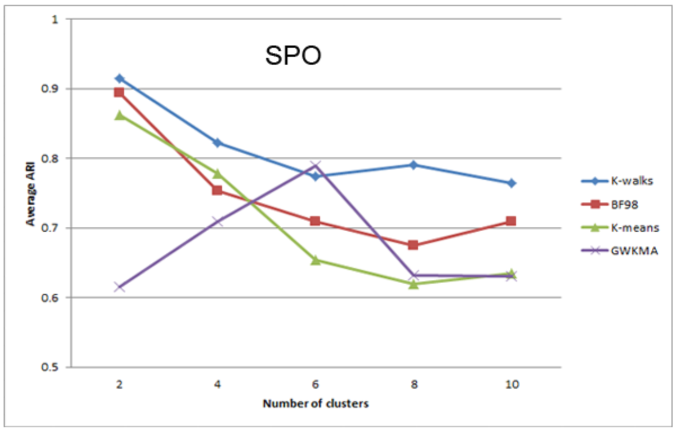
Table 3 Performance of clustering methods using four strategies over two gene-expression data sets in terms of optimised objective function (squared error distortion)

Name	*K*	*K-walks (sed, stdev*)*	*BF98 (sed, stdev)*	*K-means (sed, stdev)*	*Mik05 (sed)*
BAC	2	**5.871**, 0.001	**5.871**, 0.001	5.874, 0.015	5.866
	4	4.367, 0.017	**4.365**, 0.017	4.367, 0.008	4.367
	6	**3.858**, 0.027	3.892, 0.023	3.883, 0.069	3.883
	8	**3.636**, 0.032	3.658, 0.065	3.658, 0.043	3.652
	10	**3.5**, 0.028	3.512, 0.056	3.513, 0.067	3.515
SPO	2	**3.123**, 0.009	**3.123**, 0.037	3.135, 0.062	**3.132**
	4	2.458, 0.007	2.463, 0.049	2.46, 0.034	**2.453**
	6	**2.085**, 0.004	2.088, 0.033	2.089, 0.037	2.087
	8	**1.832**, 0.018	**1.832**, 0.007	1.835, 0.037	1.839
	10	**1.713**, 0.002	1.723, 0.028	1.721, 0.014	1.723

**sed* is the optimised value of the squared error distortion function and *stdev* is the standard deviation of the squared error distortion of the duplicated clustering run.

Figure 5 Comparison of the *K*-walks, BF98, and *K*-means algorithms in terms of the average ARI of clustering over all of the gene clusters in the BAC and SPO data sets





3.4 Clustering real-life gene-expression data

Finally, we tested the *K*-walks clustering algorithms on a real-life porcine microarray data set. This data set derives from a microarray expression profiling experiment of 16 different tissues from four animals of two extreme pig breeds, Large White and Iberian (Ferraz et al., 2008). To replicate and extend the initial analysis presented in Ferraz's research, we set out to examine which genes share similar expression pattern across different tissues using the computational analytical method called *K*-walks developed in this study.

The raw data were downloaded from the Gene-Expression Omnibus database (GSE10898) (Barrett et al., 2013). Quality control (QC) of these data using the array QualityMetrics package in Bioconductor (Kauffmann et al., 2009) showed that all of data sets were of high quality. To identify cohorts of tightly co-expressed genes across different tissues, the data were clustered using $K = 30$ and $\Delta = 0.001$. 28 clusters with a cluster size higher than ten co-expressing genes were generated. The expression patterns of all 30 clusters of co-expressed genes are shown in Figure 6.

For these clusters, Cluster-1, Cluster-2, Cluster-4, Cluster-7, and Cluster-10 consist large number of genes (3777, 2502, 441, 164, and 72, respectively). However, these clusters do not show obvious expression patterns due to the low expression level or high variation across different tissue samples. Cluster-23, Cluster-29, and Cluster-30 contain only a small number of genes (seven for Cluster-23, six for Cluster-29, and five for Cluster-30). With a Pearson correlation coefficient less than 0.75, 23 clusters have gene numbers exceeding 10. Table 4 lists the top 15 co-expressed gene clusters with the largest number of genes and distinctive expression patterns.

The genes in the clusters were mapped to GO terms (GO slim terms) to assign known biological functions to the identified tissue differences in expression. The GO terms mapped to the genes were subjected to an enrichment test to compare tissue expression with known biological functions (Yao et al., 2016). The significantly over-represented GO terms of the top 15 clusters of co-expressed genes are listed in Table 4, and a full list is available in Supplemental document 2. As expected, we were able to find numerous expected matches between the observed tissue-specific gene expression and the GO biological terms that directly associate with specific tissues. For example, in Cluster-8, which consists of genes highly expressed in diaphragm muscles and the biceps femoris muscle, we found 'muscle contraction' to be a significantly enriched GO term. Similarly, in Cluster-6, which consists of genes highly expressed in the blood and ileum, we found 'immune response' and 'cytokine-mediated signalling' to be significantly enriched GO terms. Increased expression of genes assigned with 'nervous system development' was observed in Cluster-5, which is highly expressed in the olfactory bulb, hypothalamus, pineal gland, adenohypophysis, and neurohypophysis tissues. Genes assigned to processes 'steroid metabolism' and 'xenobiotic metabolism' were observed in Cluster-9, which is exclusively expressed in liver tissue.

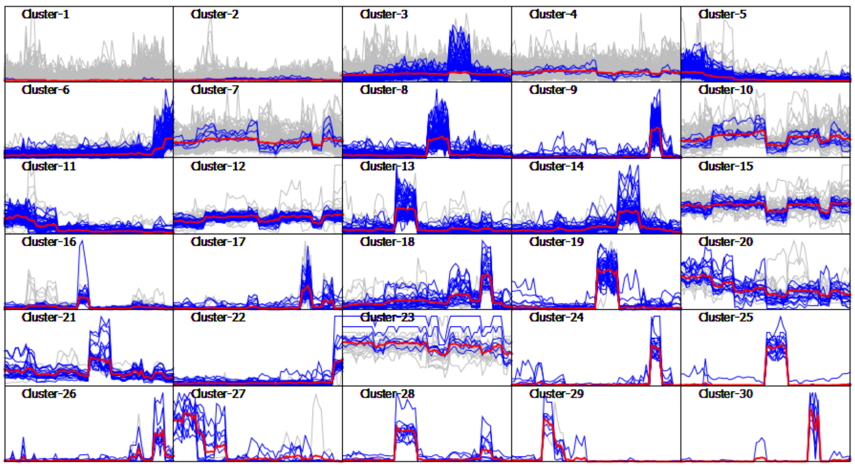
Table 4 Co-expressed gene clusters with distinctive expression pattern in porcine tissues

Cluster name	*Expression*	*GO annotation*
Cluster-3	Highly expressed in back fat and abdominal fat	Extracellular matrix, extracellular region
Cluster-5	Highly expressed in the olfactory bulb, hypothalamus, pineal gland, adenohypophysis, and neurohypophysis	Nervous system development
Cluster-6	Highly expressed in the ileum and blood	Type I interferon-mediated signalling, regulation of immune response

Table 4 Co-expressed gene clusters with distinctive expression pattern in porcine tissues (continued)

Cluster name	*Expression*	*GO annotation*
Cluster-8	Highly expressed in diaphragm muscles and the biceps femoris muscle	Respiratory chain, mitochondrial inner membrane, muscle contraction, gluconeogenesis
Cluster-9	Highly expressed in the liver	Steroid and xenobiotic metabolism, microsome, oxidoreductase activity, lipid metabolic process, peptidase activity
Cluster-11	Highly expressed in the olfactory bulb, hypothalamus, pineal gland, adenohypophysis, and neurohypophysis	Cytoplasm
Cluster-12	Minimally expressed in the olfactory bulb, hypothalamus, pineal gland, diaphragm muscle, and biceps femoris muscle	Nucleolus, cytoplasm
Cluster-13	Highly expressed in the adrenal cortex and adrenal medulla	Endoplasmic reticulum, mitochondrion
Cluster-14	Highly expressed in back fat and abdominal fat	Mitochondrion, plasma membrane
Cluster-16	Highly expressed in the thyroid gland	Integral to membrane, plasma membrane
Cluster-17	Highly expressed in the stomach and ileum	
Cluster-18	Highly expressed in the liver, back fat tissue, and abdominal fat tissue	Mitochondrion
Cluster-19	Highly expressed in the diaphragm muscles and the biceps femoris muscle	
Cluster-21	Highly expressed in diaphragm muscles and the biceps femoris muscle	Membrane
Cluster-22	Highly expressed in blood	Metal ion binding

Figure 6 Clustering of porcine gene-expression data using the *K*-walks clustering algorithm



4 Discussion

We have proposed the *K*-walks clustering algorithm, which involves a random walks optimised *K*-means clustering method that yields excellent results in terms of minimising the objective function of the mean squared distance. The *K*-walks clustering algorithm differs from most other *K*-means implementations because it introduces heuristic random walks to the process of moving centres. The rationale behind the proposed method is based on the following assumptions: (1) random walks can push the centres out of the local optima; and (2) in each iteration, only the correct configuration will be accepted. The algorithm is less sensitive to starting conditions and compares favourably with the *K*-means algorithm and other *K*-means variations in terms of minimising global distortion.

The new algorithm introduced a new parameter: the decreasing factor (Δ), which determines the performance of the algorithm. The decreasing factor determines the amount by which the walk step size will be reduced in each iteration. If the decreasing ratio increases, the step size will drop more rapidly; if the decreasing ratio decreases, there is a higher probability that the iteration process will find a better configuration. For a particular data set, the *K*-walks method is initiated with a reasonable decreasing ratio factor (Δ), and the maximum number of iterations of the *K*-walks algorithm is considered to be fixed. Therefore, the complexity of the clustering system can be evaluated, and Δ can be adjusted without having to consider trade-offs between computational load and performance of the clustering method employed.

In each iteration of the clustering algorithm, the acceptance or rejection of a clustering configuration depends on the cost function of *M*, which is a measure of the global distortion of the configuration. The cost function gradually decreases during the clustering process but increasingly selects the better or downhill solution (for a minimisation problem) as the walk step size approaches zero. Allowing for uphill moves of the walk step size potentially saves the method from becoming trapped at local optima, which is an advantage of the *K*-walks method. During the iteration process, the objective function gain gradually decreases to zero when the system becomes equivalent. The random noise introduced by random walks also decreases gradually as long as the step size becomes zero and there are no more perturbations of the centres introduced by the random walk. The clustering was then degraded via Lloyd's iteration process, and the termination condition was assumed to be attained. In practice, the state of the random walk system was found to halt immediately.

The application of *K*-walks clustering in well-annotated gene-expression data sets indicates that the advantages and disadvantages of the *K*-walks clustering algorithm compared with the BF98, Mik05, and *K*-means clustering algorithms are clear. Based on our implementation, the processing time of the *K*-walks method is significantly longer than that of its three competitors. However, the quantity of the obtained clustering solutions in terms of the mean squared difference is better. Furthermore, the *K*-walks method is less sensitive (better) to the initial partitions. Most importantly, the *K*-walks clustering algorithm is more efficient for complex clustering problems (large *K* values). When clustering real-life gene-expression data sets, the *K*-walks method successfully discovered interesting gene clusters. The GO annotation results of these gene clusters demonstrated that tissue-specific gene-expression patterns corresponded to the major tissue0specific GO terms, as expected. The integration of *K*-walks clustering and functional gene annotations in real-life gene-expression clustering analysis confirmed that the *K*-walks clustering was able to detect genes sharing similar expression patterns across tissues; most importantly, those clusters were associated with known tissue-specific functions.

Acknowledgements

This project was funded by the National Natural Science Foundation of China (NSFC Grant No. 31402055), the Yangtze Youth Talents Fund (Grant No. 2015cqr12), the Yangtze Youth Fund (Grant No. 2015cqn39), and the Scientific Research Starting Foundation for the Returned Overseas Chinese Scholars of the Ministry of Education of China.

References

Anderberg, M.R. (1973) *Cluster Analysis for Applications*, 1st ed., Academic Press, New York.

Arthur, D. and Vassilvitskii, S. (2007) 'k-means++: the advantages of careful seeding', *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA.

Bache, K. and Lichman, M. (2013) *UCI Machine Learning Repository*, University of California, School of Information and Computer Science, Irvine, CA.

Ball, G.H. and Hall, D.J. (1967) 'A clustering technique for summarizing multivariate data', *Behavioral Science*, Vol. 12, No. 3, pp.153–155.

Barrett, T., Wilhite, S.E., Ledoux, P., Evangelista, C., Kim, I.F., Tomashevsky, M., Marshall, K.A., Phillippy, K.H., Sherman, P.M., Holko, M., Yefanov, A., Lee, H., Zhang, N., Robertson, C.L., Serova, N., Davis, S. and Soboleva, A. (2013) 'NCBI GEO: archive for functional genomics data sets – update', *Nucleic Acids Research*, Vol. 41, pp.D991–D995.

Bradley, P.S. and Fayyad, U.M. (1998) 'Refining initial points for K-means clustering', *Proceedings of the 15th International Conference on Machine Learning*, Vol. 1, pp.91–99.

Celebi, M.E., Kingravi, H.A. and Vela, P.A. (2013) 'A comparative study of efficient initialization methods for the k-means clustering algorithm', *Expert Systems with Applications*, Vol. 40, No. 1, pp.200–210.

Che, Z.H. (2012) 'Clustering and selecting suppliers based on simulated annealing algorithms', *Computers and Mathematics with Applications*, Vol. 63, No. 1, pp.228–238.

Chen, B., He, J., Pellicer, S. and Pan, Y. (2010) 'Using hybrid hierarchical K-means (HHK) clustering algorithm for protein sequence motif super-rule-tree (SRT) structure construction', *International Journal of Data Mining and Bioinformatics*, Vol. 4, pp.316–330.

Chu, S., Derisi, J., Eisen, M., Mulholland, J., Botstein, D., Brown, P.O. and Herskowitz, I. (1998) 'The transcriptional program of sporulation in budding yeast', *Science*, Vol. 282, pp.699–705.

Dudoit, S. and Fridlyand, J. (2002) 'A prediction-based resampling method for estimating the number of clusters in a dataset', *Genome Biology*, Vol. 3, doi:10.1186/gb-2002-3-7-research0036.

Ferraz, A.L., Ojeda, A., Lopez-Bejar, M., Fernandes, L.T., Castello, A., Folch, J.M. and Perez-Enciso, M. (2008) 'Transcriptome architecture across tissues in the pig', *BMC Genomics*, Vol. 9, p.173.

Forgey, E. (1965) 'Cluster analysis of multivariate data: efficiency vs. interpretability of classification', *Biometrics*, Vol. 21, No. 1, pp.768–780.

Fu, J., Khaybullin, R., Zhang, Y., Xia, A. and Qi, X. (2015) 'Gene expression profiling leads to discovery of correlation of matrix metalloproteinase 11 and heparanase 2 in breast cancer progression', *BMC Cancer*, Vol. 15, p.473.

He, P., Xu, X., Hu, K. and Chen, L. (2014) 'Semi-supervised clustering via multi-level random walk', *Pattern Recognition*, Vol. 47, pp.820–832.

Huang, T.H., Uthe, J.J., Bearson, S.M., Demirkale, C.Y., Nettleton, D., Knetter, S., Christian, C., Ramer-Tait, A.E., Wannemuehler, M.J. and Tuggle, C.K. (2011) 'Distinct peripheral blood RNA responses to Salmonella in pigs differing in Salmonella shedding levels: intersection of IFNG, TLR and miRNA pathways', *PLoS One*, Vol. 6, e28768.

Huang, X., Ye, Y. and Zhang, H. (2014) 'Extensions of kmeans-type algorithms: a new clustering framework by integrating intracluster compactness and intercluster separation', *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 25, pp.1433–1446.

Huang, T.H., Zhu, M.J., Li, X.Y. and Zhao, S.H. (2008) 'Discovery of porcine microRNAs and profiling from skeletal muscle tissues during development', *PLoS One*, Vol. 3, e3225.

Jain, A.K. (2010) 'Data clustering: 50 years beyond K-means', *Pattern Recognition Letters*, Vol. 31, pp.651–666.

Jain, A.K. and Dubes, R.C. (1988) *Algorithms for Clustering Data*, 1st ed., Prentice Hall, Englewood Cliffs, NJ.

Jain, A.K., Murty, M.N. and Flynn, P.J. (1999) 'Data clustering: a review', *ACM Computing Surveys*, Vol. 31, pp.264–323.

Kang, H., Chen, I.M., Wilson, C.S., Bedrick, E.J., Harvey, R.C., Atlas, S.R., Devidas, M., Mullighan, C.G., Wang, X., Murphy, M., Ar, K., Wharton, W., Borowitz, M.J., Bowman, W.P., Bhojwani, D., Carroll, W.L., Camitta, B.M., Reaman, G.H., Smith, M.A., Downing, J.R., Hunger, S.P. and Willman, C.L. (2010) 'Gene expression classifiers for relapse-free survival and minimal residual disease improve risk classification and outcome prediction in pediatric B-precursor acute lymphoblastic leukemia', *Blood*, Vol. 115, pp.1394–1405.

Kauffmann, A., Gentleman, R. and Huber, W. (2009) 'arrayQualityMetrics – a bioconductor package for quality assessment of microarray data', *Bioinformatics*, Vol. 25, pp.415–416.

Kaufman, L. and Rousseeuw, P.J. (1990) 'Finding groups in data : an introduction to cluster analysis', *Wiley Series in Probability and Mathematical Statistics Applied Probability and Statistics*, John Wiley & Sons, New York, 342p.

Langfelder, P. and Horvath, S. (2008) 'WGCNA: an R package for weighted correlation network analysis', *BMC Bioinformatics*, Vol. 9, p.559.

Laub, M.T., Mcadams, H.H., Feldblyum, T., Fraser, C.M. and Shapiro, L. (2000) 'Global analysis of the genetic network controlling a bacterial cell cycle', *Science*, Vol. 290, pp.2144–2148.

Lloyd, S.P. (1982) 'Least squares quantization in PCM', *IEEE Transactions on Information Theory*, Vol. 28, No. 2, pp.129–137.

Macqueen, J. (1967) 'Some methods for classification and analysis of multivariate observations', *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, pp.281–297.

Mirkin, B. (2005) *Clustering for Data Mining: A Data Recovery Approach*, 1st ed., Chapman and Hall, London.

Peña, J.M., Lozano, J.A. and Larrañaga, P. (1999) 'An empirical comparison of four initialization methods for the K-means algorithm', *Pattern Recognition Letters*, Vol. 20, No. 10, pp.1027–1040.

Peterson, A.D., Ghosh, A.P. and Maitra, R. (2010) 'A systematic evaluation of different methods for initializing the k-means clustering algorithm', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, pp.522–537.

Rand, W.M. (1971) 'Objective criteria for the evaluation of clustering methods', *Journal of the American statistical Association*, Vol. 66, No. 336, pp.846–850.

Ruspini, E.H. (1970) 'Numerical methods for fuzzy clustering', *Information Sciences*, Vol. 2, pp.319–350.

Selim, S.Z. and Ismail, M.A. (1984) 'K-means-type algorithms: a generalized convergence theorem and characterization of local optimality', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 6, No. 1, pp.81–87.

Sherlock, G., Hernandez-Boussard, T., Kasarskis, A., Binkley, G., Matese, J.C., Dwight, S.S., Kaloper, M., Weng, S., Jin, H., Ball, C.A., Eisen, M.B., Spellman, P.T., Brown, P.O., Botstein, D. and Cherry, J.M. (2001) 'The stanford microarray database', *Nucleic Acids Research*, Vol. 29, pp.152–155.

Ting, S. and Jennifer, G.D. (2007) 'In search of deterministic methods for initializing K-means and Gaussian mixture clustering', *Intelligent Data Analysis*, Vol. 11, No. 4, pp.319–338.

Wu, F.X. (2008) 'Genetic weighted k-means algorithm for clustering large-scale gene expression data', *BMC Bioinformatics*, Vol. 9, p.S12.

Yao, M., Gao, W., Tao, H., Yang, J., Liu, G. and Huang, T. (2016) 'Regulation signature of miR-143 and miR-26 in porcine Salmonella infection identified by binding site enrichment analysis', *Molecular Genetics and Genomics*, Vol. 291, pp.789–799.