Data Min Knowl Disc (2009) 18:370–391
DOI 10.1007/s10618-008-0123-0

Harmony *K*-means algorithm for document clustering

Mehrdad Mahdavi · Hassan Abolhassani

Received: 21 October 2007 / Accepted: 18 November 2008 / Published online: 11 December 2008
Springer Science+Business Media, LLC 2008

Abstract Fast and high quality document clustering is a crucial task in organizing information, search engine results, enhancing web crawling, and information retrieval or filtering. Recent studies have shown that the most commonly used partition-based clustering algorithm, the *K*-means algorithm, is more suitable for large datasets. However, the *K*-means algorithm can generate a local optimal solution. In this paper we propose a novel Harmony *K*-means Algorithm (HKA) that deals with document clustering based on Harmony Search (HS) optimization method. It is proved by means of finite Markov chain theory that the HKA converges to the global optimum. To demonstrate the effectiveness and speed of HKA, we have applied HKA algorithms on some standard datasets. We also compare the HKA with other meta-heuristic and model-based document clustering approaches. Experimental results reveal that the HKA algorithm converges to the best known optimum faster than other methods and the quality of clusters are comparable.

Keywords Document clustering · Markov chain · Harmony search · Global optimization · *K*-means algorithm

Responsible editor: Charu Aggarwal.

M. Mahdavi (✉) · H. Abolhassani
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
e-mail: mahdavi@ce.sharif.edu

H. Abolhassani
School of Computer Science, Institute for Studies in Theoretical Physics and Mathematics (IPM), Tehran, Iran
e-mail: abolhassani@sharif.edu

1 Introduction

Document clustering has become an increasingly important technique for enhancing search engine results, web crawling, unsupervised document organization, and information retrieval or filtering. Clustering involves dividing a set of documents into a specified number of groups. The documents within each group should exhibit a large degree of similarity while the similarity among different clusters should be minimized. Some of the more familiar clustering methods are: partitioning algorithms based on dividing entire data into dissimilar groups, hierarchical methods, density and grid based clustering, some graph based methods and etc (Jain et al. 1999; Grira et al. 2005).

The clustering methods proposed in the literature can be classified into two major categories: discriminative (or similarity-based) approaches and generative (or model-based) approaches (Zhong and Ghosh 2003). In similarity-based approaches, one optimizes an objective function involving the pairwise document similarities, aiming to maximize the average similarities within clusters and minimize the average similarities between clusters. Model-based approaches, on the other hand, attempt to learn generative models from the documents, with each model representing one particular document group.

Model-based clustering assumes that the data were generated by a model and tries to recover the original model from the data. The model that we recover from the data then defines clusters and an assignment of documents to clusters. Model-based clustering algorithms are particularly attractive as each iteration is linear in the size of the input. Also, online algorithms can be easily constructed for model-based clustering using competitive learning techniques (Zhong and Ghosh 2005; Zhong 2006).

Clustering algorithms, from another view, can be broadly classified into two categories: hierarchal and partitional algorithms. Hierarchical algorithms represent the documents in a multi-level and tree-like structure (Zhao and Karypis 2005), while partitioning methods cluster the data in a single level (Cutting et al. 1992; Larsen and Aone 1999; Aggarwal et al. 1999; Steinbach et al. 2000a,b). Although hierarchical methods are often said to have better quality clustering results, usually they do not provide the reallocation of documents, which may have been poorly classified in the early stages of the text analysis (Jain et al. 1999; Boley et al. 1999). Moreover, the time complexity of hierarchical methods is quadratic (Steinbach et al. 2000a,b). On the other hands, in recent years the partitioning clustering methods are well suited for clustering a large document dataset due to their relatively low computational requirements (Cutting et al. 1992; Larsen and Aone 1999; Aggarwal et al. 1999; Steinbach et al. 2000a,b; Dhillon 2001; Zhao and Karypis 2004).

Partitioning methods try to partition a collection of documents into a set of groups, so as to maximize a pre-defined fitness value. The clusters can be overlapped or not. The best known partitioning algorithm is *K*-means (McQueen 1967) that, in a simple form, selects *K* documents as cluster centers and assigns each document to the nearest center. The updating and reassigning process can be kept until a convergence criterion is met. This algorithm can be performed on a large data set almost in linear time complexity.

Although K-means algorithm is easy to be implemented and works fast in most situations, it suffers from two major drawbacks that make it inappropriate for many applications (Anderberg 1973). One is sensitivity to initialization and the other is convergence to local optima. To deal with the limitations that exist in traditional partition clustering methods especially K-means, recently, new concepts and techniques have been entered into document clustering. One major approach is to use machine learning (Stumme et al. 2001; Grira et al. 2005; Stumme et al. 2006) that includes several techniques.

About the clustering of large document sets, a major part of efforts have been concerned to the learning methods such as optimization techniques. This is mostly owing to the lack of orthognality, and existing high dimension vectors. One of the advantages of partitional clustering algorithms is that they use information about the collection of documents when they partition the dataset into a certain number of clusters. So, the optimization methods can be employed for partitional clustering. Optimization techniques define a goal function and by traversing the search space, try to optimize its value. Regarding to this definition, K-means can be considered as an optimization method.

Let us stress that dividing n data into K clusters give rise to a huge number of possible partitions, which is expressed in the form of the Stirling number:

$$\frac{1}{K!}\sum_{i=1}^{K}(-1)^{K-i}\binom{K}{i}i^{n}$$

This illustrates that the clustering by examining all possible partitions of n documents of t-dimensions into K clusters is not computationally feasible. Obviously, we need to resort to some optimization techniques to reduce the search space, but then there is no guarantee that the optimal solution will be found. Methods such as Genetic Algorithm (GA) (Raghavan and Birchand 1979; Jones et al. 1995) Self-Organizing Maps (SOM) (Cui et al. 2005) and Ant Clustering (Labroche et al. 2003) have been used for document clustering. Particle Swarm Optimization (PSO) (Kennedy et al. 2001) is another computational intelligence method that has been applied to image clustering and other low dimensional datasets in Omran et al. (2002), Merwe and Engelbrecht (2003) and to document clustering in (Cui et al. 2005). Typically, these stochastic approaches take a large amount of time to converge to a globally optimal partition. Although various optimization methodologies have been developed for optimal clustering, the complexity of the task reveals the need for developing efficient algorithms to precisely locate the optimum solution. In this context, this study presents a novel stochastic approach for document clustering, aiming at a better time complexity and partitioning accuracy.

A meta-heuristic algorithm, mimicking the improvisation process of music players, has been recently developed and named Harmony Search (HS) (Geem et al. 2002; Mahdavi et al. 2007). Harmony search algorithm had been very successful in a wide variety of optimization problems (Lee and Geem 2004; Geem et al. 2005) presenting several advantages with respect to traditional optimization techniques such as the following (Lee and Geem 2004):

(a) HS algorithm imposes fewer mathematical requirements and does not require initial value settings for decision variables;
(b) As the HS algorithm uses stochastic random searches, derivative information is also unnecessary;
(c) The HS algorithm generates a new vector, after considering all of the existing vectors, whereas the methods like genetic algorithm (GA) only considers the two parent vectors;
(d) HS does not need to encode and decode the decision variables into binary strings;
(e) HS treats continuous variables without any loss of precision

These features increase the flexibility of the HS algorithm and produce better solutions. In fact, in optimization problems, we want to search the solution space and in HS this search can be done more efficiently.

Since stochastic optimization approaches are good at avoiding convergence to a locally optimal solution, these approaches could be used to find a globally optimal solution. Typically the stochastic approaches take a large amount of time to converge to a globally optimal partition. In this paper, we propose an algorithm based on HS, Harmony *K*-means Algorithm (HKA), for clustering documents, and prove that it converges to the global optimum with probability one using theory of Markov chains. To demonstrate the effectiveness and speed of HKA, we have applied HKA algorithms on standard document sets and got very good results compared to the *K*-means. The evaluation of the experimental results shows considerable improvements and robustness of HKA.

The paper is organized as follows. Section 2 provides a brief review of vector space model for document representation, particularly the aspects necessary to understand document clustering, quality measures that will be used as the basis for our comparison of techniques, and harmony search algorithm. The proposed HKA algorithm is explained in Sect. 3. In Sect. 4, using finite Markov chain theory, we prove that the HKA converges to the global optimum. Section 5 presents document sets used in our experiments and our experimental results. Finally, Sect. 6 concludes the paper.

2 Preliminaries

2.1 Document representation and similarity computation

In most document clustering algorithms, documents are represented using vector-space model. In this model, each document d is considered to be a vector $\vec{d} = \{d_1, d_2, \ldots, d_t\}$ in term-space (set of document "words") where d_i is the weight of dimension i in vector space and t is the number of term dimensions. In text documents each weight d_i represents the term weight of term i in the document. The most widely used weighting approach for term weights is the combination of Term Frequency and Inverse Document Frequency (*TF-IDF*) (Raghavan and Birchand 1979; Everitt 1980; Salton 1989). In this approach the weight of term i in document j is defines as (1).

$$w_{ji} = tf_{ji} \times idf_{ji} = tf_{ji} \times \log_2(n/df_{ji}) \tag{1}$$

That tf_{ji} is the numbers of occurrences of term i in the document j; df_{ij} is the total term frequency in data set and n is the number of documents.

The similarity between two documents must be measured in some way if a clustering algorithm is to be used. The vector space model gives us a good opportunity for defining different metrics for similarity between two documents. The most common similarity metrics are Minkowski distances (Cios et al. 1998) and cosine measure (Raghavan and Birchand 1979; Salton and Buckley 1988; Salton 1989). Minkowski distances computes the distance of documents d and d' by (2) (for n = 2 it is converted to Euclidean distance).

$$D_n(d, d') = \left(\sum_{i=1}^{t} |d_i - d'_i|^n \right)^{1/n} \tag{2}$$

Cosine measure is defined by (3) where $d^T \cdot d'$ is the inner product (dot-product) of two vectors. Both metrics are widely used in the text document clustering literatures. But it seems that in the cases which the number of dimensions of two vectors differs largely, the cosine is more useful. In cases which two vectors have almost the same dimension, Minkowski distance can be useful.

$$\cos(d, d') = \frac{d^T \cdot d'}{|d||d'|} \tag{3}$$

2.2 Evaluation of cluster quality

Objective clustering evaluation criteria can be based on *internal* or *external* measures. Internal quality measures are used to compare different sets of clusters without reference to external knowledge. External quality measures are used to evaluate how well the clustering is working by comparing the clusters produced by the clustering techniques to known classes. The most important external methods are entropy-based methods, confusion matrix, classification accuracy, average purity (Raghavan and Birchand 1979; Selim and Ismail 1984; Omran et al. 2002), and F-measure (Larsen and Aone 1999). F-measure combines two measures, *precision* and *recall,* from information retrieval domain. If P and R show *Precision* and *Recall* respectively, this measure is defined by (4) and *precision* and *recall* are obtained by (5). In the formulas n_j shows the size of cluster j, g_i shows the size of class i and $N(i, j)$ shows the number of documents of class i in cluster j.

$$F(i\,,\,j) = \frac{2(P(i, j)^* R(i, j))}{(P(i, j) + R(i, j))},\ F = \sum_i \frac{g_t}{n} \max_j\{F(i, j)\} \tag{4}$$

$$P(i, j) = N(i, j)/n_j,\ \ R(i, j) = N(i, j)/g_i \tag{5}$$

In the absence of any external information, such as class labels, the cohesiveness of clusters can be used as a measure of cluster similarity. One method for computing the cluster cohesiveness is to use the weighted similarity of the internal cluster similarity $\frac{1}{|S|^2} \sum_{d,\, d' \in S} D(d',\, d)$, where D is the used similarity function and S is the set of documents.

2.3 Harmony search algorithm

Harmony Search (HS) (Lee and Geem 2004) is a new meta-heuristic optimization method imitating the music improvisation process where musicians improvise their instruments' pitches searching for a perfect state of harmony. The HS works as follows:

Step 1: Initialize the problem and HS parameters

The optimization problem is defined as

$$\begin{array}{ll} \text{Minimize} & f(\mathbf{x}) \quad \text{by varying decision, or design, variables } \mathbf{x} \\ \text{subject to} & \mathbf{g(x)} \geq 0 \\ & \mathbf{h(x)} = 0 \\ \text{and} & LB_i \leq x_i \leq UB_i, \ \text{ for } 1 \leq i \leq N \text{ if } x_i \text{ is continuous, or} \\ & x_i \in \mathbf{X}_i = \{X_{i,1}, X_{i,2}, \ldots, X_{i,K_i}\}, \text{ if } x_i \text{is discrete} \end{array}$$

where LB_i and UB_i are lower and upper bounds on x_i.

In addition, the parameters of the HS are specified in this step. These are the harmony memory size (*HMS*), or the number of solution vectors in the harmony memory; harmony memory considering rate (*HMCR*); pitch adjusting rate (*PAR*); and the number of improvisations (NI), or stopping criterion. The harmony memory (*HM*) is a memory location where all the solution vectors (sets of decision variables) are stored. This *HM* is similar to the genetic pool in the GA. The *HMCR*, which varies between 0 and 1, is the rate of choosing one value from the historical values stored in the *HM*, while (1-*HMCR*) is the rate of randomly selecting one value from the possible range of values.

Step 2: Initialize the harmony memory

The algorithm maintains a store of solution vectors known as Harmony Memory HM, that is updated during the optimization process. Harmony Memory is the HMS $\times$ $(N + 1)$ augmented matrix:

$$\text{HM} = \left[\begin{array}{cccc|c} x_1^1 & x_2^1 & \ldots & x_N^1 & f(\mathbf{x^1}) \\ x_1^2 & x_2^2 & \ldots & x_N^2 & f(\mathbf{x^2}) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{\text{HMS}} & x_2^{\text{HMS}} & \ldots & x_N^{\text{HMS}} & f(\mathbf{x^{HMS}}) \end{array}\right]$$

HM consists of the decision variables only, with the objective function values stored separately. The initial harmony memory is generated from a uniform distribution in the ranges $[LB_i, UB_i]$, where $1 \leq i \leq N$. This is done as follows:

$$x_i^j = LB_i + r \times (UB_i - LB_i), \ j = 1, 2, \ldots, \text{HMS}$$

where $r \sim U(0,1)$ and U is a uniform random number generator.

Step 3: Improvise a new harmony

Generating a new harmony is called *improvisation*. The new harmony vector, $x' = (x'_1, x'_2, \ldots, x'_N)$, is generated using the following rules: memory consideration, pitch adjustment and random selection. The procedure works as follows:

```
for each $i \in [1, N]$ do
    if $U(0,1) \leq$ HMCR then    /*memory consideration*/
        begin
        $x'_i = x_i^j$, where $j \sim U(1, \ldots, \text{HMS})$.
        if $U(0,1) \leq$ PAR then    /*pitch adjustment*/
        begin
            $x'_i = x'_i \pm r \times bw$, where $r \sim U(0,1)$ and $bw$ is an arbitrary
            distance bandwidth.
        endif
    else    /*random selection*/
        $x'_i = LB_i + r \times (UB_i - LB_i)$
    endif
```

Step 4: Update harmony memory

The generated harmony vector, $x' = (x'_1, x'_2, \ldots, x'_N)$, replaces the worst harmony in the HM, only if its fitness (measured in terms of the objective function) is better than that of the worst harmony.

Step 5: Check the stopping criterion

Terminate when the maximum number of improvisations is reached.

The HMCR and PAR parameters of the HS help the method in searching for globally and locally improved solutions, respectively. PAR and *bw* have a profound effect on the performance of the HS. Thus, fine tuning these two parameters is very important. From these two parameters, *bw* is more difficult to tune because it can take any value from $(0, \infty)$.

To address the shortcomings of the HS, Mahdavi et al. (2007) proposed a new variant of the HS, called the Improved Harmony Search (IHS). The IHS dynamically updates PAR according to the following equation,

$$\text{PAR}(t) = \text{PAR}_{\min} + \frac{(\text{PAR}_{\max} - \text{PAR}_{\min})}{NI} \times t \tag{6}$$

where PAR(t) is the pitch adjusting rate for generation t, $\mathrm{PAR}_{\min}$ is the minimum adjusting rate, $\mathrm{PAR}_{\max}$ is the maximum adjusting rate, NI is the number of generations, and t is the generation number.

In addition, bw is dynamically updated as follows:

$$bw\,(t) = bw_{\max} e^{\left(\frac{\ln\left(\frac{bw_{\min}}{bw_{\max}}\right)}{NI} \times t\right)} \tag{7}$$

where $bw\,(t)$ is the bandwidth for generation t, $bw_{\min}$ is the minimum bandwidth and $bw_{\max}$ is the maximum bandwidth.

3 Harmony *K*-means algorithm

For our clustering algorithms, documents are represented using the vector-space model. In this model, each term represents one dimension of multidimensional document space and, each document $d_i = (d_{i1}, d_{i2}, \ldots, d_{it})$ is considered to be a vector in the term-space and each possible solution for clustering document collection is a vector of centroids.

In order to clustering documents using harmony search algorithm, we must first model the clustering problem as an optimization problem that locates the optimal centroids of the clusters rather than to find an optimal partition. This model enables us to apply HS optimization algorithm on the optimal clustering of a collection of documents.

Since our goal is to optimize an objective function, clustering is essentially a search problem. Viewing the clustering problem as an optimization problem of such an objective function formalizes the problem to some extent. However, we are not aware of any function that optimally captures the notion of a ‘good’ cluster, since for any function one can exhibit cases for which it fails. Furthermore, not surprisingly, no polynomial-time algorithm for optimizing such cost functions is known. The brute force solution would be to enumerate all possible clusterings and pick the best. However, there are exponentially many partitions, so this approach is not feasible. The key design challenge in objective function-based clustering is the formulation of an objective function capable of reflecting the nature of the problem so that its minimization reveals meaningful structure (clusters) in the data. The following subsections describe our modeling and harmony operators’ accord to this model for clustering purpose.

3.1 Coding

Let $\{d_i, i = 1, 2, \ldots, n\}$ be the set of documents. Let d_{ij}denote the weight of jth feature of document d_i. Also, define a_{ij} for $i = 1, 2, \ldots, K$ and $j = 1, 2, \ldots, n$,

$$a_{ij} = \begin{cases} 1\,, & \text{if jth document belongs to ith cluster,} \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Then, the assignment matrix A $= [a_{ij}]$ has the properties that each $a_{ij} \in \{0, 1\}$ and each document must assigned exactly to one cluster (e.g. $\sum_{i=1}^{K} a_{ij} = 1$ for $j = 1, 2, \ldots, n$). An assignment that represents K nonempty clusters is a *legal* assignment. Each assignment matrix corresponds to a set of K centroids $C = (c_1, c_2, \ldots, c_i, \ldots, c_K)$. So, the search space is the space of all A matrices that satisfy constraint in which each document must be allocated to exactly one cluster and there is no cluster that is empty. A natural way of encoding such A into a string, s, is to consider each row of HM of length n and allow each element to take the values from $\{1, 2, \ldots, K\}$. In this encoding, each element corresponds to a document and its value represents the cluster number to which the corresponding document belongs.

3.2 Initialization of harmony memory

Harmony memory must be initialized with randomly generated feasible solutions. Each row of harmony memory corresponds to a specific clustering of documents in which, the value of the ith element in each row is randomly selected from the uniform distribution over the set $\{1, 2, \ldots, K\}$ and indicates the cluster number of ith document. Such randomly generated solutions may not be legal in which no document is allocated to some clusters. This is avoided by assigning $\lfloor \frac{n}{K} \rfloor$ randomly chosen documents to each cluster and the rest of documents to randomly chosen clusters. This step takes $O(n)$ where n is the number of documents.

3.3 Improvise a new clustering

In improvising step, we need a technique which generates one solution vector, *NHV*, from all HMS solution vectors exists in HM. The new generated harmony vector must inherit as much information as possible from the solution vectors that are in the HM. If the generated new harmony vector, which is corresponds to a new clustering, consists mostly or entirely of assignments found in the vectors in HM, it provides good heritability.

Each value is selected from harmony memory with probability HMCR and with probability (1 − HMCR) is randomly selected from set $\{1, 2, \ldots, K\}$. After generating the new solution, the PAR process is applied. PAR is originally the rate of allocating a different cluster to a document. To apply pitch adjusting process to document d_i the algorithm proceeds as follow. The current cluster of d_i is replaced with a new cluster chosen randomly from the following distribution:

$$p_j = \Pr\{\text{cluster } j \text{ is selected as new cluster}\} = \frac{D_{\max} - D(NHV, c_j)}{\sum_{j=1}^{K} (D_{\max} - D(NHV, c_i))} \quad (9)$$

where $D_{\max} = \max_i\{D(NHV, c_i)\}$ and *NHV* is the recently improvised clustering.

3.4 Hybridization

The algorithm with the above processes performs a globalize searching for solutions, whereas *K*-means clustering procedure performs a localized searching. In localized searching, the solution obtained is usually located in the proximity of the solution obtained in the previous step. The refining process of the *K*-means algorithm indicates that the algorithm only explores the very narrow proximity, surrounding the initial randomly generated centroids and its final solution depends on these initially selected centroids. So the proposed algorithm is good at finding promising areas of the search space, but not as good as *K*-means at fine-tuning within those areas, so it may take more time to converge. On the other hand, *K*-means algorithm is good at fine-tuning, but lack a global perspective. It seems a hybrid algorithm that combines two ideas can results in an algorithm that can outperform either one individually.

To improve the algorithm a one-step *K*-means algorithm is introduced. After that a new clustering solution is generated with applying harmony operations, the following process is applied on new solution. First, the cluster centroids are calculated using Eq. 10 for the new solution. Then each document is reassigned to the cluster with the nearest centroid. The resulting assignment may represent an illegal partitioning. The illegal assignments are converted to legal one by placing in each empty cluster a document from the cluster with the maximum within-cluster variation (Klein and Dubes 1989).

3.5 Evaluation of solutions

Each row in HM corresponds to a clustering with assignment matrix A. Let $C = (c_1, c_2, \ldots, c_i, \ldots, c_K)$ is set of K centroids for assignment matrix A. The centroid of the kth cluster is $c_k = (c_{k1}, c_{k2}, \ldots, c_{kt})$ and is computed as follows:

$$c_{kj} = \frac{\sum_{i=1}^{n} (a_{ki})\, d_{ij}}{\sum_{i=1}^{n} a_{ki}} \tag{10}$$

where t is the number of terms in all documents. Fitness value of each row is determined by Average Distance of Documents to the Cluster centroid (ADDC) represented by that row. This value is measured by equation:

$$f = \frac{\sum_{i=1}^{k} \left\{ \frac{\sum_{j=1}^{n_i} D(c_i, d_{ij})}{n_i} \right\}}{K} \tag{11}$$

where K is the number of clusters, n_i is the numbers of documents in cluster i $\left(\text{e.g. } n_i = \sum_{j=1}^{n} a_{ij}\right)$, D is distance function, and d_{ij} is the jth document of cluster i.

The new generated solution is replaced with a row in harmony memory, if the locally optimized vector has better fitness value than those in HM.

3.6 Time complexity analysis

In this section the time complexity of HKA is computed. For ease of analysis, we define the following symbols to describe the parameters:

- g: number of algorithm generations
- P_{Random}: probability of random selection equals to $(1 - HMCR)$
- P_{Memory}: probability of memory consideration equals to $HMCR$
- P_{Pitch}: probability of pitch adjusting process equals to $HMCR \times PAR$
- K: number of clusters
- n: number of documents
- t: the dimension of each document in vector-space model

Since we add the K-means algorithm as a one step in HKA, we first analyze the time complexity of K-means algorithm.

Lemma 1 *The time complexity of K-means algorithm is* $O(g^* K^* n^* t)$.

Proof In K-means algorithm, most of the time is spent on computing vector distances. One such operation costs $O(t)$. The reassignment step computes the similarity between each document and all the clusters, so $O(K^* n)$ distances must be computed in each reassignment. Therefore, the overall complexity of each iteration is $O(K^* n^* t)$. In the reassignment step, each vector gets added to a centroid once, so the complexity of this step is $O(n^* t)$. For a fixed number of iterations g, the overall complexity is therefore $O(g^* K^* n^* t)$. □

The following lemma shows the time complexity of improvising a new clustering in HKA algorithm.

Lemma 2 *The time complexity of generating a new clustering in improvising step is* $O(n^* t)$.

Proof Each clustering solution in our representation is represented by a vector of size n. To improvise a new clustering, each entry in the new vector must be assigned a value from the set $\{1, 2, \ldots, K\}$ based on the memory consideration and PAR process. Each document d_i of the new clustering independent from other documents is assigned to a cluster based on two operations for considering the computational intelligence or randomness as follows: randomly selected from the set $\{1, 2, \ldots, K\}$ with probability P_{Random} and from the memory with probability P_{Memory}. The PAR process is applied after selection from memory with probability P_{PAR} .The complexity of this step is outperformed by the time complexity of PAR process. Each PAR process takes $O(K^* t)$ since the similarity of document is computed with all of the clusters. Considering the probability of applying PAR process, the total complexity of this step is $O(K^* t^* n^* P_{\text{PAR}}{}^* P_{\text{Memory}})$. Note that the complexity of similarity function is $O(t)$. □

Lemma 3 *Each evaluation phase has the time complexity of* $\theta(n^*t)$

Proof From the Eq. 11 it is clear that the similarity must be computed for each document and the cluster which it is assigned. Totally, the similarity function utilized n times and each computation takes $\theta(t)$. Therefore the evaluation phase costs $\theta(n^*t)$. □

The following theorem shows the time complexity of HKA algorithm.

Theorem 1 *The HKA algorithm with the mentioned steps has the time complexity of* $\theta(g^*n^*t^*K)$.

Proof First we consider the time complexity of hybridization. In this phase, one step of K-means is applied on the clustering generated by the harmony operators. This step takes $\theta(K^*n^*t)$ due to each document must be assigned to closest cluster which needs the similarity of each document to all clusters is computed based on lemma 1. Considering the lemmas 2 and 3 and the hybridization at each generation the total time complexity of HKA is $\theta(g^*n^*t^*K)$. □

4 Convergence analysis of HKA algorithm

4.1 Finite Markov chains

Markov chains provide very flexible, powerful, and efficient means for the analysis of probabilistic processes over a state space S. Each Markov chain associated with a *transition* matrix $\mathbf{P} = (p_{ij})$ in which the $p_{ij} \in [0,\ 1]$ is the probability of transitioning from state $i \in S$ to state $j \in S$ at step t and $\sum_{j=1}^{|S|} p_{ij} = 1$ for all $i \in S$. In this case the matrix $\mathbf{P}$ is called *stochastic.* In *homogeneous* Markov chains, the transition probability is independent of parameter t. Markov chains depend on the structure of the transition matrix $\mathbf{P}$ can be classified as following (Rudolph 1994):

Definition 1 (a) A square matrix $\mathbf{A}$**:** $n \times n$ is said to be *nonnegative* if $a_{ij} \geq 0$ for all $i, j \in \{1, \ldots, n\}$.
(b) A square matrix $\mathbf{A}$**:** $n \times n$ is said to be *positive* if $a_{ij} > 0$ for all $i, j \in \{1, \ldots, n\}$.
(c) A nonnegative matrix $\mathbf{A}$**:** $n \times n$ is said to be *primitive* if there exists a $k \in \mathrm{N}$ such that $\mathbf{A}^k$ is positive.
(d) A nonnegative matrix $\mathbf{A}$**:** $n \times n$ is said to be *column-allowable* if $\sum_{j=1}^{|S|} p_{ij} = 1$ for all $i \in S$ and it has at least one positive entry in each column.

It is clear from the definition that every positive matrix is also primitive. Also, the product of stochastic matrices yields a stochastic matrix. The following proposition (Rudolph 1994) is the basis of proving convergence of HKA algorithm in the next subsection.

Proposition 1 *Let* $\mathbf{C}$, $\mathbf{M}$ *and* $\mathbf{S}$ *be stochastic matrices, where* $\mathbf{M}$ *is positive and* $\mathbf{S}$ *is column-allowable. Then the product* $\mathbf{C} \cdot \mathbf{M} \cdot \mathbf{S}$ *is positive.*

4.2 Markov chain analysis of HKA algorithm

In this subsection the global convergence of proposed algorithm without hybridization with K-means is proved using finite Mrkov chain theory by deriving conditions on the parameters of proposed algorithm that ensures the global convergence. Let $H(t)$ represents the harmony memory state by the algorithm at generation t. The state space of process $H(t)_{t\geq 0}$ is the space of all possible harmony memories $S = \{1, 2, \ldots, K\}^{HMS\cdot K}$ and the states can be numbered from 1 to $|S|$. The state space contains only legal solutions that representing partitions with K nonempty clusters. From the definition of the algorithm each state $H(t)$ can be completely defined by state $H(t-1)$, so the process $\{H(t)_{t\geq 0}\}$ is a Markov chain. Let $p_{ij}(t)$ represents the probability of moving from harmony memory state $i \in S$ to state $j \in S$ at time step t. Transition probabilities are independent of the time instant, therefore, $\{H(t)_{t\geq 0}\}$ is *homogeneous.*

The transition probabilities of a homogeneous finite Markov chain can be gathered in a stochastic transition matrix $\mathbf{P} = (p_{ij})$ where $p_{ij} \in [0, 1]$ and $\sum_{j=1}^{|S|} p_{ij} = 1$ for all $i \in S$.

Harmony memory state changes when the new generated solution has fitness better than the fitness of worst solution in the memory. New solution is generated considering three processes: Memory Consideration with probability $p_{\text{Memory}} = \text{HMCR} \times$ (1-PAR), Pitch Adjusting with probability $p_{\text{Pitch}} = \text{HMCR} \times \text{PAR}$,and Random Choosing with probability $p_{\text{Random}} = (1 - \text{HMCR})$. So, the probabilistic changes of the solutions within the harmony memory caused by the harmony operations are captured by the transition matrix $\mathbf{P}$, which can be decomposed in a natural way into a product of stochastic matrices $\mathbf{P} = \mathbf{MC} \cdot \mathbf{PA} \cdot \mathbf{RC}$, where **MC**, **PA** and **RC** describe the intermediate transition matrices caused by memory consideration, pitch adjusting and random choosing processes described in the HKA algorithm.

Theorem 2 *The transition matrix* **P** *of the HKA algorithm with memory consideration probability* $p_{Memory} \in (0, 1)$*, random choosing probability* $p_{Random} \in (0, 1)$*, and pitch adjusting probability* $p_{Pitch} \in (0, 1)$ *is* **primitive**.

Proof According definition of **P** as product of three matrices **MC**, **PA** and **RC** and considering the Lemma 1, in order to prove the theorem, it is sufficient prove that the three matrices are stochastic, **PA** is positive, and **RC** is column-allowable.

Each of the three processes applied to generate new harmony solution may change the state of harmony memory by replacing the worst solution in memory with the new generated one. So, each of them is a total function that probabilistically maps each state of S to another state. Therefore, the matrices are stochastic.

The matrix **PA** is positive if any solution can be obtained from any other solution on application of pitch adjusting process. Due to the PAR process is applied independently to each document cluster number, the probability that state i becomes state j after PAR process is positive. So, the PAR process described in the HKA algorithm can change any legal clustering solution to any other legal solution with nonzero probability. Thus, the **PA** is positive.

Random choosing process changes the state of harmony memory if randomly generated solution is replaced with another solution in memory. If the new randomly generated solution by random choosing process is same as one of solutions in harmony memory or all of the solutions in harmony memory have better fitness than the new generated solution, then the state of harmony memory remains in the same state. So, the probability that random choosing does not alter the state of harmony memory is bounded by:

$$rc_{ii} \geq \Pr\{\text{the new solution is same as one of solutions in HM}\} = \frac{1}{HMS^{K-1}} \geq 0 \tag{12}$$

for all $i \in S$, so that **RC** is column-allowable.

It follows by Proposition 1 that $\mathbf{P} = \mathbf{MC} \cdot \mathbf{PA} \cdot \mathbf{RC}$ is positive. Since every positive matrix is primitive, the proof is completed. □

Theorem 3 *The HKA converges to global optimum.*

Proof In (Rudolph 1994) it is proved that a meta-heuristic algorithm whose transition matrix **P** is primitive and which maintains the best solution over the generations, converges to the global optimum. The transition matrix **P** of HKA algorithm is primitive as proved in Theorem 2. The best solution in HM is the best solution found until the current generation that is evident from definition of HS. So, it follows that the probability of being in any globally optimal state convergence to one and the theorem is proved. □

5 Experimental results

We compare the algorithms according to their quality and speed of convergence using a number of different document sets. In this section the datasets is described and the proposed algorithm is compared with *K*-means algorithm considering speed of convergence and quality of clustering.

5.1 Document collections

We conducted experiments using HKA on five data sets. In these document datasets, the very common words (stop words) are stripped out completely and different forms of a word are reduced to one canonical form by using Porter's algorithm.

Data sets DS1 and DS2 are from TREC-5, TREC-6, and TREC-7 (TREC 1999). The class labels of DS1 and DS2 came from the relevance judgments provided by particular queries (TRECQ 1999). The data set DS3 reviews datasets were derived from the San Jose Mercury newspaper articles that are distributed as part of the TREC collection (TIPSTER). This dataset was constructed by selecting documents that are part of certain topics in which the various articles were categorized (based on the DESCRIPT tag). The DS3 dataset contains documents about computers, electronics, health, medical, research, and technology. In selecting these documents we ensured that no two documents share the same DESCRIPT tag (which can contain multiple categories).

Table 1 Summary of text document datasets

Document set	Source	# of documents	# of clusters
DS1	TREC	414	6
DS2	TREC	204	9
DS3	TREC	873	8
DS4	DMOZ	697	14
DS5	20 NEWSGROUP	9249	10

The data set DS4 is selected from DMOZ collection and contains 697 documents that are selected among 14 topics including art, business, computer, game, health, home, recreation, reference, science, shopping, society, sport, news and regional. From each topic some web pages are selected and are included in data set. In this case, the clusters produced by the algorithm were compared with the original DMOZ categories. The 20-newsgroups data[1] is used for constructing the final data set.

The DS5 dataset is a collection of 10,000 messages, collected from 10 different usenet newsgroups, 1000 messages from each. After preprocessing there are a total of 9249 documents in this data set. In addition to remove stop words in preprocessing phase, words that occur in less than three documents are removed. Description of the test datasets is given in Table 1.

5.2 Experimental setup

The *K*-means and HKA clustering algorithms are applied on the above mentioned data sets. The Euclidian distance measure and cosine correlation measure are used as the similarity metrics in each algorithm. It is to be emphasized at this point that the results shown in the rest of paper is the average over 30 runs of both algorithms. Also, for an easy comparison, the algorithms run 500 iterations in each run since the 500 generations is enough to convergence of both algorithms. No parameter needs to be set up for the *K*-means algorithm. For HKA, for each data set the HMS is set 2 times the number of cluster in the data set, HMCR[2] is set to 0.9, $PAR_{\min} = 0.09$, and $PAR_{\max} = 0.99$.

5.3 Significance of hybridization

In the first set of experiments, HKA with and without of *K*-means was applied on the DS1 data set to evaluate the significance of hybridization on the proposed algorithm. Figure 1 shows the ADDC value over generations corresponding to this

[1] http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html.

[2] The value of the HMCR parameter was suggested by Dr. Zong Geem in a private communication.

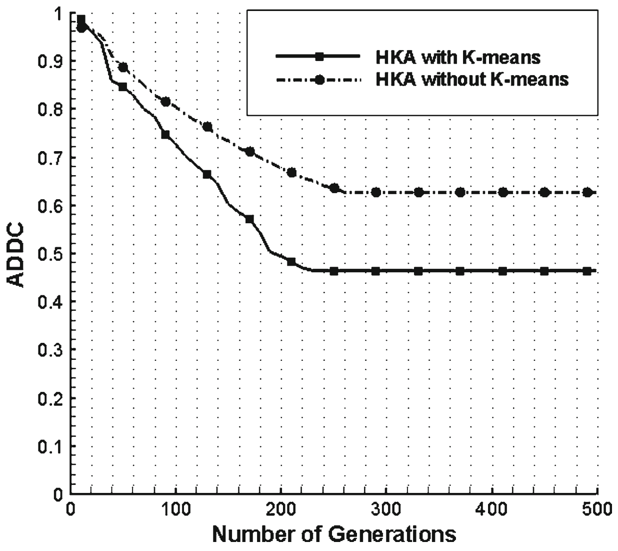

Fig. 1 Performance of HKA algorithm with and without *K*-means on document set DS1

experiment. Although it is shown analytically that HKA without *K*-means converges to the global optimum, the algorithm has better convergence when is hybridized with *K*-means. It can be observed from Fig. 1 that hybridization of HKA with *K*-means significantly increases the quality of clusters considering ADDC values as quality measure.

5.4 Comparison with *K*-means algorithm

In the next experiment, HKA is compared with *K*-means algorithm on the document dataset DS1. In this experiment, the initial configuration of *K*-means algorithm is included in harmony memory in its initialization phase for better comparison. Figure 2 illustrates the convergence behaviors of these algorithms. It is obvious from Fig. 2 that the *K*-means algorithm implementation need much less computing time and iterations to reach a stable clustering result than HKA algorithm and HKA took more time to reach the optimal solution and *K*-means converges more quickly. This is because the *K*-means algorithm can be trapped in local optima. Although the *K*-means algorithm is more efficient than HKA considering execution time, the HKA generates much better clustering than the *K*-means algorithm.

Figure 2 illustrates that the reduction of ADDC value in *K*-means is sharp, but HKA follows a smooth curve from its initial vectors to final optimum solution and has not a sharp move like *K*-means. From Fig. 2 we can infer that *K*-means is not assured to reach the global optimum. The situation becomes worse when the search space is large and there are many local optima.

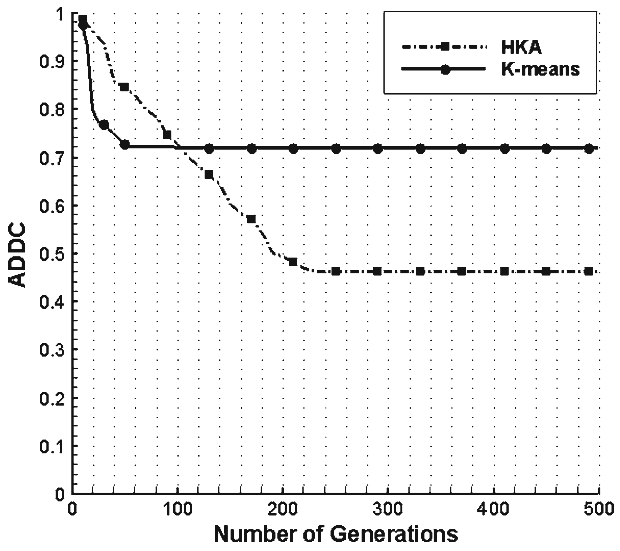

Fig. 2 The convergence behaviors of HKA and *K*-means algorithms on document set DS1

5.5 Quality of clustering

Our third set of experiments was focused on evaluating the quality of the clustering solutions produced by the *K*-means and HKA algorithms. For evaluation of the clustering results quality, we use two metrics, namely F-measure and ADDC. F-measure expresses the clustering results from an external export view, while ADDC examines how much the clustering satisfies the optimization constraints. The smaller the ADDC value, the more compact the clustering solution is. Table 2 demonstrates the normalized ADDC of algorithm for two similarity measures applied on five document sets. Looking at the results in Table 2 we can see that the results obtained by HKA are significantly comparable by results obtained by *K*-means.

Table 2 Comparison of the Euclidian and Cosine similarity measures of HKA and *K*-means algorithms considering ADDC values of generated clusters

Document set	ADDC			
	K-means		HKA	
	Euclidian	Cosine	Euclidian	Cosine
DS1	0.7184	0.7849	0.4638	0.5462
DS2	0.6415	0.7032	0.4517	0.5053
DS3	0.7425	0.7925	0.5287	0.5841
DS4	0.4153	0.4419	0.2510	0.3554
DS5	0.8425	0.9236	0.6170	0.7523

Table 3 Comparison of the F-measure for HKA and *K*-means algorithms

Document set	F-measure	
	K-means	HKA
DS1	0.5632	0.7662
DS2	0.5202	0.7824
DS3	0.6117	0.8968
DS4	0.7236	0.8692
DS5	0.4236	0.6805

As a primary measure of quality we use the F-measure; the harmonic means of precision and recall. For a given cluster of documents C, to evaluate the quality of C respect to an ideal cluster C^*(categorization by human) we first compute precision and recall as usual:

$$P(C, C^*) = \frac{|C \cap C^*|}{|C|} \quad \text{and} \quad R(C, C^*) = \frac{|C \cap C^*|}{|C^*|} \tag{13}$$

Then we define:

$$F(C, C^*) = \frac{2 * P * R}{P + R} \tag{14}$$

Table 3 shows the comparison of F-measure values for *K*-means and HKA algorithms. The results in Table 3 reveal that HKA outperforms *K*-means algorithm in all of datasets.

5.6 Comparison with other algorithms

To demonstrate how our method improves the document clustering accuracy in comparison to the best contemporary methods, we implemented three known partitioning algorithms namely Genetic *K*-means (GA), Particle Swarm Optimization based clustering (PSO) (Cui et al. 2005) and a Mises-Fisher Generative Model based algorithm (GM)[3] (Zhong and Ghosh 2005). For the GA we used the same representation as used for HKA. We evaluated the clustering methods over data sets representing distinct clustering difficulties in the same experimental conditions in order to better appreciate the performance of each clustering algorithm.

To compare the algorithms fairly, we employed the following termination rule .In all of the mentioned algorithms, the current optimal solution is always recorded. For the current optimal solution, we record the number of continuous iterations without improving it. Then we calculate the ratio of this number to the total iteration number. If the ratio exceeds the given upper bound ratio, we believe that the continuous running of the algorithm will not contribute any improvement to the solution. Hence, we end

[3] An implementation of this algorithm is available at http://www.cse.fau.edu/~zhong/software/index.htm.

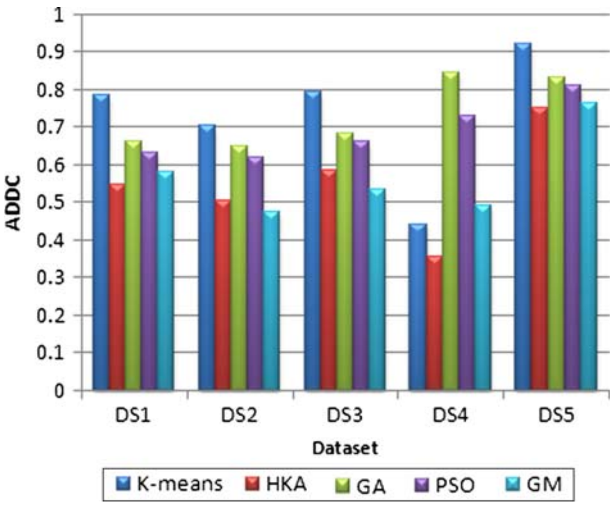

Fig. 3 The quality of clustering generated by HKA, GM, K-means, GA, and PSO algorithms on different datasets

the search. In addition, the maximum number of iterations is also given to guarantee that the algorithm will end after a certain number of solutions have been searched.

Figure 3 shows the results of applying mentioned algorithms on five datasets considering the normalized ADDC of algorithm. From the results, it is easy to know that our proposed method outperforms GA, *K*-means, and PSO in all datasets. Surprisingly, for datasets DS2 and DS3 the GM algorithm generates high quality clusters than HKA.

5.7 Runtime analysis

This set of experiments is about the runtimes of HKA, *K*-means, GA, PSO, and GM algorithms with different documents. Figure 4 shows the evaluation results using the DS5. The evaluations were conducted for the document numbers ranging from 1000 to approximately 10,000. For each given document number, 10 test runs were conducted on different randomly chosen documents, and the final performance scores were obtained by averaging the scores from the all tests. Because *K*-means algorithm is not guaranteed to find the global optimum, it is beneficial to perform *K*-means algorithm a few times with different initial values and choose the trial with minimal ADDC. The GM algorithm has the lowest runtime in comparison to all of other algorithms due to the model-based partitional clustering algorithms often have a complexity of $O(n)$ where n is the number of documents. In contrast, in the other algorithms calculating the pairwise similarities is more time-consuming.

For small number of documents the runtime of algorithms is approximately same, but by increasing the number of documents the difference beings significant. The *K*-means algorithm has the worst runtime. The runtime of PSO and GA is nearly the same. HKA behaves better than other algorithms except the GM. Specially; the runtime of the HKA is comparable with the GM algorithm.

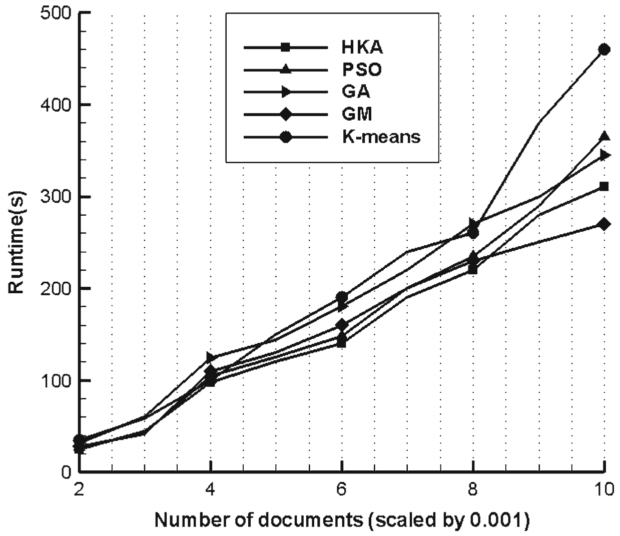

Fig. 4 Execution time of the HKA algorithm in comparison with GM, *K*-means, GA, and PSO algorithms on different sub collections of DS5 dataset with different sizes

6 Conclusion

In this paper the problem of finding a globally optimal partition of a given set of documents into a specified number of clusters is considered and a novel algorithm, named HKA, by modeling clustering problem as an optimization of an objective function is proposed. In the proposed algorithm the harmony search algorithm is employed for global optimization. Also, we add a one step of *K*-means algorithm to PAR process of harmony search to better fine-tuning of algorithm which significantly improved the speed of convergence of HKA. The convergence of proposed algorithm is studied by using the theory of Markov chain and it is proved that the HKA with probability one convergence to the global optima. Our experimental results on five different document sets showed that HKA algorithm produces better solutions with high quality considering ADDC and F-measure quality measures in comparison with *K*-means and other three known algorithms and the difference is tremendous.

Acknowledgements The authors would like to thank the anonymous referees for their valuable comments. Their careful reading and valuable comments greatly enhanced the richness and readability of this paper. Also, we would like to thank the Prof. Zhong for providing the implementation of his proposed algorithms.

References

Aggarwal CC, Gates SC, Yu PS (1999) On the merits of building categorization systems by supervised clustering. In: Proceedings of the fifth ACM SIGKDD Int'l conference on knowledge discovery and data mining, pp 352–356

Anderberg MR (1973) Cluster analysis for applications. Academic Press Inc, New York, NY

Boley D, Gini M, Gross R, Han EH, Hastings K, Karypis G et al (1999) Partitioning-based clustering for web document categorization. Decis Support Syst 27(3):329–341. doi:10.1016/S0167-9236(99)00055-X

Cios K, Pedrycs W, Swiniarski R (1998) Data mining methods for knowledge discovery. Kluwer Academic Publishers

Coello CAC (2000) Constraint-handling using an evolutionary multi objective optimization technique. Civ Eng Environ Syst 17:319–346. doi:10.1080/02630250008970288

Cui X, Potok TE, Palathingal P (2005) Document clustering using particle swarm optimization. In: IEEE swarm intelligence symposium, pp 185–191

Cutting DR, Pedersen JO, Karger DR, Tukey JW (1992) Scatter/gather: a cluster-based approach to browsing large document collections. In: Proceedings of the ACM SIGIR. Copenhagen, pp 318–329

Dhillon IS (2001) Co-clustering documents and words using bipartite spectral graph partitioning. In: Knowledge discovery and data mining, pp 269–274

Everitt B (1980) Cluster analysis, 2nd edn. Halsted Press, New York

Geem ZW, Kim JH, Loganathan GV (2002) Harmony search optimization: application to pipe network design. Int J Model Simul 22:125–133

Geem ZW, Tseng C, Park Y (2005) Harmony search for generalized orienteering problem: best touring in China, Springer. Lect Notes Comput Sci 3412: 741–750

Grira N, Crucianu M, Boujemaa N (2005) Unsupervised and semi-supervised clustering: a brief survey. In: 7th ACM SIGMM international workshop on multimedia information retrieval, pp 9–16

Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 264–323. CSUR. doi:10.1145/331499.331504

Jones G, Robertson AM, Santimetvirul C, Willett P (1995) Non-hierarchic document clustering using a genetic algorithm. Inform Res 1(1)

Kennedy J, Eberhart RC, Shi Y (2001) Swarm intelligence. Morgan Kaufmann, New York

Klein RW, Dubes RC (1989) Experiments in projection and clustering by simulated annealing. Pattern Recognit 22:213–220. doi:10.1016/0031-3203(89)90067-8

Labroche N, Monmarche' N, Venturini G (2003) AntClust: ant clustering and web usage mining. In: Genetic and evolutionary computation conference, pp 25–36

Larsen B, Aone C (1999) Fast and effective text mining using linear-time document clustering. In: Proceedings of the fifth ACM SIGKDD Int'l conference on knowledge discovery and data mining, pp 16–22

Lee KS, Geem ZW (2004) A new meta-heuristic algorithm for continues engineering optimization: harmony search theory and practice. Comput Method Appl Mech Eng 194:3902–3933. doi:10.1016/j.cma.2004.09.007

Mahdavi M, Fesanghary M, Damangir E (2007) An improved harmony search algorithm for solving optimization problems. Appl Math Comput 188:1567–1579

McQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth berkeley symposium on mathematical statistics and probability, pp 281–297

Merwe VD, Engelbrecht AP (2003) Data clustering using particle swarm optimization. In: Proceedings of IEEE congress on evolutionary computation (CEC 2003), Australia, pp 215–220

Omran M, Salman A, Engelbrecht AP (2002) Image classification using particle swarm optimization. In: Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning (SEAL 2002), Singapore, pp 370–374

Quinlan RJ (1993) C4.5: programs for machine learning. Morgan Kaufmann

Raghavan VV, Birchand K (1979) A clustering strategy based on a formalism of the reproductive process in a natural system. In: Proceedings of the second international conference on information storage and retrieval, pp 10–22

Rudolph G (1994) Convergence analysis of canonical genetic algorithms. IEEE Trans Neural Netw 5(1):96–101. doi:10.1109/72.265964

Salton G (1989) Automatic text processing: the transformation, analysis, and retrieval of information by computer. Addison-Wesley

Salton G, Buckley C (1988) Term-weighting approaches in automatic text retrieval. Inf Process Manage 24:513–523. doi:10.1016/0306-4573(88)90021-0

Selim SZ, Ismail MA (1984) K-means type algorithms: a generalized convergence theorem and characterization of local optimality. IEEE Trans Pattern Anal Mach Intell 6:81–87

Steinbach M, Karypis G, Kumar V (2000a) A comparison of document clustering techniques. KDD'2000. Technical report of University of Minnesota

Steinbach M, Karypis G, Kumar V (2000b) A comparison of document clustering techniques. In: KDD workshop on text mining

Stumme G, Hotho A, Berendt B (2001) Semantic Web Mining. Freiburg, September 3rd, 12th Europ. Conf. on Machine Learning (ECML'01)/5th Europ. Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'01)

Stumme G, Hotho A, Berendt B (2006) Semantic Web Mining State of the art and future directions. J Web Semantics: Sci Serv Agents World Wide Web 124–143

TREC (1999) Text REtrieval conference. http://trec.nist.gov

TRECQ (1999) Text REtrieval conference relevance judgments. http://trec.nist.gov/data/qrels_eng/index.html

Zhao Y, Karypis G (2004) Empirical and theoretical comparisons of selected riterion functions for document clustering. Mach Learn 55(3):311–331. doi:10.1023/B:MACH.0000027785.44527.d6

Zhao Y, Karypis G (2005) Hierarchical clustering algorithms for document datasets. Data Min Knowl Discov 10:141–168. doi:10.1007/s10618-005-0361-3

Zhong S (2006) Semi-supervised model-based document clustering: a comparative study. Mach Learn 65(1). doi:10.1007/s10994-006-6540-7

Zhong S, Ghosh J (2003) A unified framework for model-based lustering. J Mach Learn Res 4:1001–1037. JMLR. doi:10.1162/jmlr.2003.4.6.1001

Zhong S, Ghosh J (2005) Generative model-based clustering of documents: a comparative study. Knowl Inf Syst 8:374–384. KAIS. doi:10.1007/s10115-004-0194-1