Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 201 (2008) 441–451
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

Novel meta-heuristic algorithms for clustering web documents

M. Mahdavi [a,*], M. Haghir Chehreghani [a], H. Abolhassani [a,b], R. Forsati [a,c]

[a] Web Intelligence Laboratory, Department of Computer Engineering, Sharif University of Technology, Azadi Avenue, Tehran, Iran
[b] School of Computer Science, Institute for Studies in Theoretical Physics and Mathematics (IPM), Niavaran, Tehran, Iran
[c] Department of Computer Engineering, Islamic Azad University, Karaj Branch, Karaj, Iran

Abstract

Clustering the web documents is one of the most important approaches for mining and extracting knowledge from the web. Recently, one of the most attractive trends in clustering the high dimensional web pages has been tilt toward the learning and optimization approaches. In this paper, we propose novel hybrid harmony search (HS) based algorithms for clustering the web documents that finds a globally optimal partition of them into a specified number of clusters. By modeling clustering as an optimization problem, first, we propose a pure harmony search-based clustering algorithm that finds near global optimal clusters within a reasonable time. Then, we hybridize *K*-means and harmony clustering in two ways to achieve better clustering. Experimental results reveal that the proposed algorithms can find better clusters when compared to similar methods and also illustrate the robustness of the hybrid clustering algorithms.

Keywords: Document clustering; Harmony search; *K*-Means; Optimization; Hybridization

1. Introduction

Clustering data is an important pre-processing task in information mining that can lead to considerable results. Clustering involves dividing a set of objects into a specified number of groups. The data objects 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, etc. [9,11].

On the web, this task has other additional roles; it can be used for enhancing search engine results, enhancing web crawling, and organizing the knowledge. In an overall categorization, we can divide the web document clustering algorithms into two categories: hierarchal and partitioning algorithms. Hierarchical algorithms represent the web pages in a multi-level and tree-like structure, while partitioning methods cluster

* Corresponding author.
E-mail addresses: mehrdad.mahdavi@gmail.com (M. Mahdavi), haghir@ce.sharif.edu (M.H. Chehreghani), abolhassani@ce.sharif.edu (H. Abolhassani), rana.forsati@gmail.com (R. Forsati).

0096-3003/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2007.12.058

the data in a single level [4,6,9,11]. Although hierarchical methods are often said to have better quality clustering results, usually they do not provide the reallocation of pages, which may have been poorly classified in the early stages of the text analysis [11]. Moreover, the time complexity of hierarchical methods is quadratic [26].

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. It seems that in recent years the partitioning clustering methods are well suited for clustering a large document dataset due to their relatively low computational requirements [26,30]. The best known partitioning algorithm is *K*-means [10] that, in a simple form, selects *K* points as cluster centers and assigns each data point 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 dataset 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. 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 web data mining, with respect to increasing need for the web knowledge extraction. One major approach is machine learning [2,9,27,28] that includes several techniques. One of these techniques is optimization methods that try to optimize a pre-defined function. So, it can be very useful in web document clustering.

About the web document clustering, a major part of efforts have been concerned to the learning methods such as optimization techniques. This is mostly owing to the lake of orthogonality, and existing high dimension vectors. Optimization techniques define a goal function and by traversing the search space, try to optimize its value. Regarding this definition, *K*-means can be considered as an optimization method. In addition to the *K*-means algorithm, several algorithms, such as genetic algorithm (GA) [12,22], self-organizing maps (SOM) [5] and ant clustering [15] have been used for document clustering. Particle swarm optimization (PSO) [13] is another computational intelligence method that has been applied to image clustering and other low dimensional datasets in [18,20] and to document clustering in [5].

A meta-heuristic algorithm, mimicking the improvisation process of music players, has been recently developed and named harmony search (HS) [16,17]. Harmony search algorithm had been very successful in a wide variety of optimization problems [8,16], presenting several advantages with respect to traditional optimization techniques such as the following [16]: (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 genetic algorithm (GA) only considers the two parent vectors. 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 efficient.

In this paper, we first model the clustering problem as an optimization problem and propose a document clustering algorithm based on HS algorithm(HClust), which is good at finding promising areas of the search space but not as good as *K*-means at fine-tuning within those areas. Then, we present a hybrid clustering approach that uses *K*-means algorithm to replace the refining stage in the HClust algorithm. The hybrid algorithm combines the power of the HClust algorithm with the speed of a *K*-means and the global searching stage and local refine stage are accomplished by those two modules, respectively. Hybridization is implemented in two different ways.

We have applied these 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 their robustness.

The rest of this paper is organized as follows. Section 2 provides some information on how documents are represented, how the similarity or distance between documents is computed, and how the quality of methods can be measured. Section 3 provides a general overview of the *K*-means and harmony search algorithm. The harmony clustering algorithm (HClust) is described in Section 4. Our hybrid clustering algorithms, HKClust and IHKClust, are presented in Section 5. Section 6 provides the detailed experimental evaluation of the proposed algorithms compared with *K*-means algorithm.

2. Definitions

2.1. Document representation

In most data mining algorithms, each data point (such as document) is represented as a vector $V = \{v_1, v_2, \ldots, v_t\}$, where v_i is the weight of dimension i in vector space and t is number of terms. In text documents, this is known as vector space model [7] that each weight v_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 with inverse document frequency (TF–IDF) [7,23]. In this approach the weight of term i in document j is defined as

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

where tf_{ji} is the number of occurrences of term i in the document j; df_{ij} is the total term frequency in dataset and n is the number of documents. This weighting scheme discounts the frequent words with little discriminating power. A word with a high frequency within a document and low frequency within the document collection will be assigned a high weight value.

2.2. The similarity metric

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 [3] and cosine measure [23,24]. Minkowski distances computes the distance of documents d_p and d_j by (2), where t is the dimension of documents (for $n = 2$ it is converted to Euclidean distance)

$$D_n(d_p, d_j) = \left(\sum_{i=1}^{t} | d_{i,p} - d_{i,j} |^n \right)^{1/n}. \tag{2}$$

In the vector-space model, the cosine similarity is the most commonly used method to compute the similarity between two documents. Cosine measure is defined by (3), where $d_p^t d_j$ 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

$$\cos(d_p, d_j) = \frac{d_p^t d_j}{|d_p||d_j|}. \tag{3}$$

2.3. Quality measures

Since there are many different quality and performance measures, the relative ranking of different clustering algorithms can vary substantially, depending on the measure used. However, if one clustering algorithm performs better than other clustering algorithms on many of these measures, then we can have some confidence that is truly the best clustering algorithm for the situation being evaluated. The most important methods are entropy-based [21,26] methods and F-measure [14]. In first method the entropy of a cluster is calculated by $E(j) = -\sum_i p_{ij} \log(p_{ij})$. That i shows the correct class and j is the produced cluster. Then, the total entropy will be $E_{cs} = \sum_{j=1}^{m} \frac{n_j * E(j)}{n}$. With respect to this measure, a good clustering algorithm minimizes the total entropy. F-measure combines two measures *precision* and *recall* and evaluates whether the clustering can remove the noise pages and generate clusters with high quality. If P and R show precision and recall, respectively, this measure is defined by (4) and precision and recall are obtained by (5). That n_j shows the size of cluster j, g_i shows the size of class i and $N(i, j)$ shows the number of pages of class i in cluster j

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

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

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 $\frac{1}{|S|^2}\sum_{d\in S}\sum_{d'\in S}\cos(d',d)$. This is just the square length of the cluster centroid, $\|c\|^2$.

3. Preliminary algorithms

3.1. K-Means clustering algorithm

As mentioned earlier, *K*-means is a partitioning clustering algorithm that performs clustering in almost linear time complexity. Since data clustering is an NP-complete problem, this algorithm is a heuristic solution that finds local maximum in search space. This local maximum and therefore the quality of the result are dependant on the initial points. The steps of the *K*-means have been depicted in Fig. 1.

In the algorithm of Fig. 1 the criterion that is used to distinguish the convergence and to characterize good clusters is defined as (6). This is known as minimizing *sum of squared error*.

$$E(A,C)=\sum_{i=1}^{K}\sum_{j=1}^{n}(a_{ij})D^2(c_i,d_j). \tag{6}$$

In (6) A is the assignment matrix and $C=\{c_i\}$ is the set of K cluster centers. Assignment matrix is obtained by assigning each data point to associated cluster center. This matrix includes values of 0 and 1. In the algorithm of Fig. 1, updating of centers is done according to (7).

$$c_i=\frac{\sum_{j=1}^{n}(a_{ij})d_j}{\sum_{j=1}^{n}a_{ij}},\quad 1\leqslant i\leqslant K. \tag{7}$$

K-Means suffers from several drawbacks. The main drawback is the sensitivity of result clusters to the selection of initial centroids; so that it may converge to the local optimum that is not acceptable [25]. In fact, *K*-means is an optimization algorithm that searches the local optimal solution in the vicinity of the initial solution and refines the partition result. But the same initial cluster centroids in a dataset will always generate the same cluster results and so repeating the algorithm for achieving better results must be done with different initial centroids. However, if good initial clustering centroids can be obtained using any other techniques, the *K*-means will work well in refining the clustering centroids to find the optimal clustering centers [1].

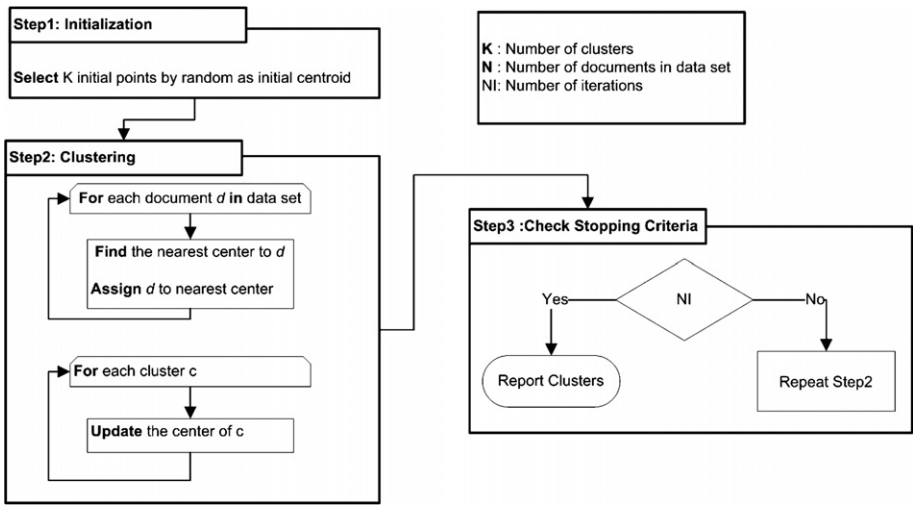

Fig. 1. The *K*-means algorithm.

3.2. Harmony search optimization algorithm

Mimicking the improvisation process of music players, harmony search (HS) algorithm, which is a nature-inspired algorithm, has been recently developed by Geem et al. [16]. The HS algorithm is simple in concept, few in parameters, and easy in implementation, it has been successfully applied to various benchmarking and real-world problems including traveling salesman problem [4], and power economic dispatch [29]. The steps in the procedure of harmony search are as follows [17]:

Step 1: Initialize the problem and algorithm parameters.
Step 2: Initialize the harmony memory.
Step 3: Improvise a new harmony.
Step 4: Update the harmony memory.
Step 5: Check the stopping criterion.

These steps are described in the next five subsections.

3.2.1. Initialize the problem and algorithm parameters

In Step 1, the optimization problem is specified as follows:

$$\begin{aligned}&\text{Minimize} \quad f(x)\\ &\text{subject to}: x_i \in X, \quad i = 1, 2, \ldots, N,\end{aligned}$$

where $f(x)$ is an objective function; $\mathbf{x}$ is the set of each decision variable x_i; N is the number of decision variables, X_i is the set of the possible range of values for each decision variable, that is ${}_Lx_i \leqslant X_i \leqslant {}_Ux_i$, and ${}_Lx_i$ and ${}_Ux_i$ are the lower and upper bounds for each decision variable. The HS algorithm parameters are also 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 [6]. Here, HMCR and PAR are parameters that are used to improve the solution vector. Both are defined in Step 3.

3.2.2. Initialize the harmony memory

In Step 2, the HM matrix is filled with as many randomly generated solution vectors as the HMS

$$\mathrm{HM} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_{N-1}^1 & x_N^1 \\ x_1^2 & x_2^2 & \cdots & x_{N-1}^2 & x_N^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^{\mathrm{HMS}-1} & x_2^{\mathrm{HMS}-1} & \cdots & x_{N-1}^{\mathrm{HMS}-1} & x_N^{\mathrm{HMS}-1} \\ x_1^{\mathrm{HMS}} & x_2^{\mathrm{HMS}} & \cdots & x_{N-1}^{\mathrm{HMS}} & x_N^{\mathrm{HMS}} \end{bmatrix}$$

3.2.3. Improvise a new harmony

A new harmony vector, $x' = (x'_1, x'_2, \ldots, x'_N)$, is generated based on three rules: (1) memory consideration, (2) pitch adjustment and (3) random selection. Generating a new harmony is called 'improvisation' [16].

In the memory consideration, the value of the first decision variable (x'_1) for the new vector is chosen from any of the values in the specified HM range $(x_1'^1 - x_1'^{\mathrm{HMS}})$. Values of the other decision variables $(x'_2, x'_3, \ldots, x'_N)$ are chosen in the same manner. 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

$$x'_i \leftarrow \begin{cases} x'_i \in \{x_i^1, x_i^2, \ldots, x_i^{\mathrm{HMS}}\} & \text{with probability HMCR,} \\ x'_i \in X_i & \text{with probability}(1 - \mathrm{HMCR}). \end{cases} \tag{8}$$

Every component obtained by the memory consideration is examined to determine whether it should be pitch-adjusted. This operation uses the PAR parameter, which is the rate of pitch adjustment as follows (9):

$$\text{Pitch adjusting decision for } x'_i \leftarrow \begin{cases} \text{Yes} & \text{with probability PAR,} \\ \text{No} & \text{with probability}(1 - \text{PAR}). \end{cases} \tag{9}$$

The value of $(1 - \text{PAR})$ sets the rate of doing nothing. If the pitch adjustment decision for x'_i is YES, x'_i is replaced as follow:

$$x'_i \leftarrow x'_i \pm \text{rand}() * \text{bw},$$

where bw is an arbitrary distance bandwidth; rand(): is a random number between 0 and 1.

In Step 3, HM consideration, pitch adjustment or random selection is applied to each variable of the new harmony vector in turn.

To improve the performance of the HS algorithm and eliminate the drawbacks lie with fixed values of HMCR and PAR, Mahdavi et al. [17] proposed an improved harmony search (IHS) algorithm that uses variable PAR and bw in improvisation step. Also, Omran and Mahdavi [19] proposed a new variant of harmony search, called the global best harmony search (*GHS*), in which concepts from swarm intelligence are borrowed to enhance the performance of *HSA* such that the new harmony can mimic the best harmony in the ***HM***.

3.2.4. Update harmony memory

If the new harmony vector, $x' = (x'_1, x'_2, \ldots, x'_N)$ is better than the worst harmony in the HM, judged in terms of the objective function value, the new harmony is included in HM and the existing worst harmony is excluded from the HM.

3.2.5. Check stopping criterion

If the stopping criterion (i.e. maximum number of improvisations) is satisfied, computation is terminated. Otherwise, Steps 3 and 4 are repeated.

4. HClust: harmony clustering

In the vector space model the content of a document is formalized as a point in the multi-dimensional space represented by a vector *w,* such that $w = (w_1, w_2, \ldots, w_n)$, where $w_i (i = 1, 2, \ldots, n)$ is the term weight of the term t_i in one document. In this model, each term represents one dimension of multidimensional document space and, each document d 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 cluster 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.

Let consider each cluster centroid as a decision variable; so each row of harmony memory, which contains k decision variables, represents one possible solution for clustering. In the other hand, each row contains a number of candidate centroids that represents each cluster. In this case each solution contains k vectors and constitutes a row of matrix, $C = (C_1, C_2, \ldots, C_K)$ where $c_i (i = 1, 2, \ldots, K)$ is the ith cluster centroid vector and **k** is the number of clusters.

A key characteristic of most partitional clustering algorithms is that they use a global criterion function whose optimization drives the entire clustering process. For those partitional clustering algorithms, the clustering problem can be stated as computing a clustering solution such that the value of a particular objective function is optimized. Our objective function is to minimize intra-cluster similarity while maximizing the inter-cluster similarity.

According to improvising step in harmony search algorithm, the new vector is generated in each iteration. Fitness value of each row, which corresponds to one potential solution, is determined by average distance of documents to the cluster centroid (ADDC) represented by each solution. This value is measured by equation:

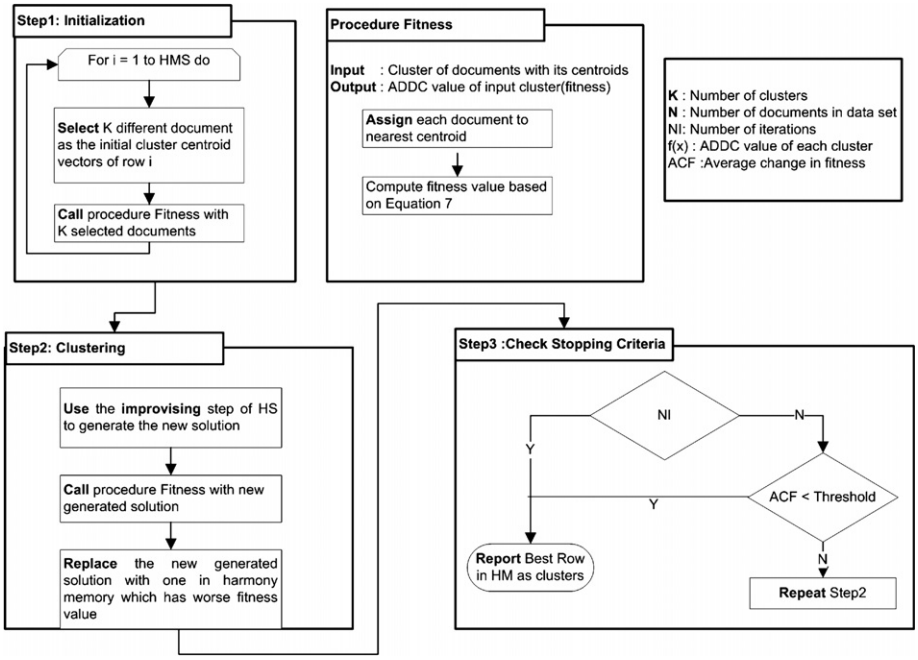

Fig. 2. Steps of the harmony clustering algorithm (HClust).

$$f = \frac{\sum_{i=1}^{k} \left\{ \frac{\sum_{j=1}^{n_i} D(c_i, d_{ij})}{n_i} \right\}}{k}, \tag{10}$$

where k is the number of clusters, n_i is the numbers of documents in cluster i, 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**. The HClust algorithm can be summarized as shown in Fig. 2.

5. Harmony *K*-means clustering

The HClust algorithm 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. For example, the *K*-means clustering algorithm uses the randomly generated seeds as the initial clusters' centroids and refines the position of the centroids at each iteration. 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, HClust and *K*-means algorithms have complementary strong and weak points, HClust is good at finding promising areas of the search space, but not as good as *K*-means at fine-tuning within those areas. On the other hand, *K*-means algorithm is good at fine-tuning, but lack a global perspective. It seems a hybrid algorithm that combines HClust with *K*-means can results in an algorithm that can outperform either one individually.

For this reason, we present a hybrid clustering approach that uses *K*-means algorithm to replace the refining stage in the HClust algorithm. The hybrid algorithm combines the power of the HClust algorithm with the speed of a *K*-means and the global searching stage and local refine stage are accomplished by those two modules, respectively.

Hybrid algorithm combines the power of the HClust with the speed of a *K*-means algorithm. The HClust finds the region of the optimum, and then the *K*-means takes over to find the optimum centroids. We need to

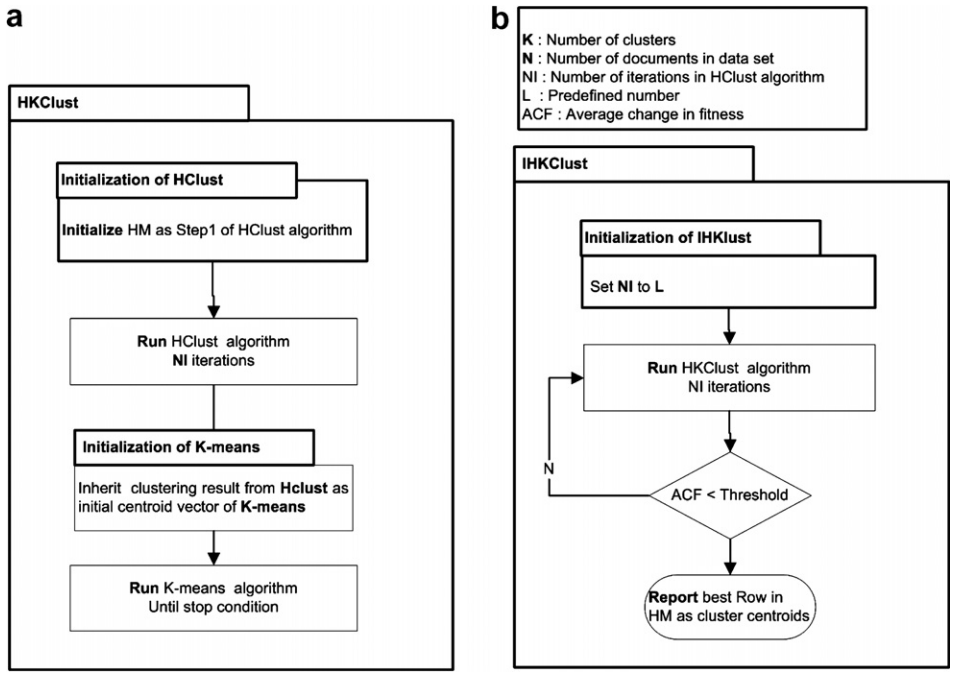

Fig. 3. Steps of the HKClust and IHKClust algorithms.

find the right balance between local exploitation and global exploration. We have two different versions of the hybrid clustering, depending on the stage when we carry out the *K*-means algorithm:

1. The HClust performs coarse search first. When the HClust is completed or shows a negligible trend of improvement after many iterations, the *K*-means begins its job and uses the best vector from the harmony memory (HM) as the starting point. The method (HKClust) is shown in Fig. 3a.
2. The local method is integrated into the HClust. For instance, after every L iteration, the *K*-means uses the best vector from the harmony memory (HM) as its starting point. HM is updated if the locally optimized vectors have better fitness value than those in HM and this procedure repeated until stop condition. The method (IHKClust) is shown in Fig. 3b.

By assigning different values to HMCR and PAR parameters in the harmony search algorithm, we can control the shift time from the global searching stage to the local refining stage. To improve the performance of the hybrid algorithm, the computation time spent by the global search and local search should be adjusted. In many problems, it is not efficient to carry out a local optimization algorithm for each iteration due to the dimensions of the search space and limits on computational resources. This is especially important when the local search algorithm have a high computational cost. If we apply the optimization algorithm to every new improvised vector, we will obtain many similar solutions with a considerable waste of time. Therefore, in our approach we apply the local search method to only a few solution vectors.

6. Experimental results

6.1. Document collections

To compare the quality and the speed of the *K*-means and the three proposed algorithms in this paper, we used three different web page collections. 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.

First dataset is collected from *Politics* area and contains 176 web documents that are selected randomly in some topics of Politics. We have collected this dataset in 2006. Second dataset is collected from News sites and contains 424 different news texts. This dataset is collected in 2006, as well. Third dataset 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 dataset. In next step, we apply the *K*-Means and the proposed algorithms on these datasets and then evaluate the results.

6.2. Experimental methodology and metrics

For each one of the different datasets, we apply the *K*-means and HClust, HKClust and IHKClust described in Sections 4 and 5. The cosine correlation measure is used as the similarity metric in each algorithm.

In each algorithm, two types of parameters must be set, when implementing optimization problems. First type includes determining the algorithm execution time for converging to optimal solution, while second type includes the parameters which defined in optimization method and determines its behavior. In all of algorithms the number of clusters must be specified before. Of course, this problem exists in other clustering methods. The harmony search algorithm parameters for all datasets presented in this paper are shown in Table 1.

In the HKClust approach, it first executes the HClust algorithm for 80% of total iterations and uses the HClust result as the initial seed for the *K*-means module and the *K*-means module executes for 20% remaining iterations to generate the final result. In all algorithms, we increase iteration numbers from 10 to 2000.

We compare the algorithms according to their quality and speed of convergence. 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. Therefore, in a real clustering, both of these metrics are seemed to be evaluated necessarily. Each metric is described in Section 2.3 with enough details. For evaluation of convergence speed, different algorithms are compared with respect to the number of repetitions that they need to converge.

6.3. Results and discussions

As mentioned before, the evaluation of result clusters includes two parts: quality and speed. For evaluation of quality, at first we have used the *F*-measure. The results of calculating this measure for different clusters of different algorithms are brought in Table 2. In comparison, the results for different algorithms, IHKClust presents the highest values. This issue is due to the high quality of produced clusters by this algorithm. The lowest value is for *K*-means. This is due to the fact that it converges to the nearest local maximum having the values of *K* centriods.

Table 1
Parameters of different algorithms

Collection	*K*	HMCR	HMS	PAR	bw
Dataset 1	6	0.95	4	0.35	5e−4
Dataset 2	10	0.95	8	0.45	1e−5
Dataset 3	14	0.95	15	0.45	5e−4

Table 2
F-measure values of different algorithms

Collection	*K*-means	HClust	HKClust	IHKClust
Dataset 1	0.65348	0.72250	0.75947	0.80212
Dataset 2	0.70639	0.76560	0.80541	0.83719
Dataset 3	0.62143	0.70465	0.73977	0.79493

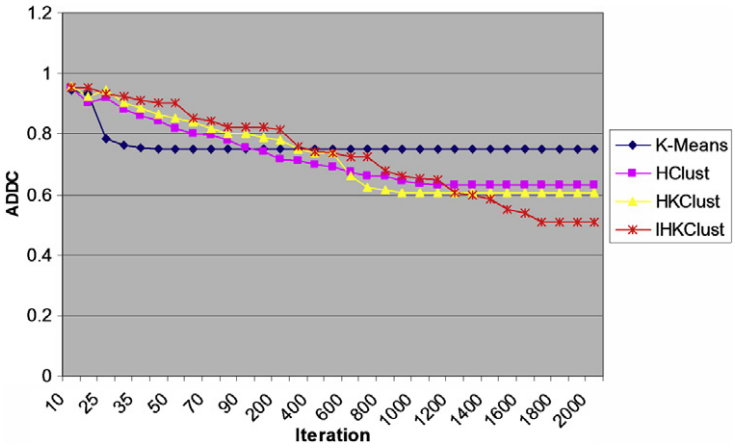

Fig. 4. The convergence behavior of the proposed algorithms versus *K*-means.

Table 3
The ADDC results of different algorithms

Collection	*K*-means	HClust	HKClust	IHKClust
Dataset 1	0.729091	0.633645	0.604732	0.508787
Dataset 2	0.87653	0.84114	0.84016	0.790240
Dataset 3	0.92046	0.897300	0.85473	0.81024

The second criterion for evaluating algorithms is their convergence rate to optimal solution. As shown in Fig. 4, *K*-means algorithm has the highest convergence rate, but it converges to local optima in few iterations and shows no further improvement.

HClust follows a smooth curve from its initial vectors to final optimum solution and has not a sharp move like *K*-means. The inability to fine tune near optimal solution is the disadvantage of this algorithm.

HKClust algorithm overcomes HClust disadvantage by incorporating a two-step hybrid algorithms. In the first step, the algorithm uses harmony search to get close to optimal solution, but since it does not fine-tune this result, thus the obtained result is passed as the initial vector to *K*-means algorithm and then, *K*-means fine tunes that.

IHKClust has a two-step iterative behavior. As shown in Fig. 4, in the first step of each iteration, algorithm smoothly approaches near the optimal solution and in the second step of each iteration, *K*-means fine tunes the result. Actually this algorithm uses HKClust in an iterative procedure for each iteration step.

Another noteworthy point in Fig. 4 is that the final value of ADDC for IHKClust is the lowest among other algorithms. The sequence of other algorithms with respect to their ADDC values are: HKClust, HClust, and *K*-means. This issue shows that the clusters produced by IHKClust have higher quality. Table 3 shows the results obtained for different datasets based on ADDC metric.

7. Conclusion

In this paper, we proposed new methods for clustering the web documents by modeling clustering problem as an optimization problem. First, we proposed an algorithm based on harmony search optimization (HClust) that is appropriate for finding near global regions within a reasonable time, but not as good as *K*-means at fine-tuning within those regions. So, second improvement was hybridization of *K*-means and harmony search-based clustering sequentially (HKClust). This method uses *K*-means algorithm to replace the refining stage in the HClust algorithm. The hybrid algorithm combines the power of the HClust algorithm with the speed of a *K*-means and the global searching stage and local refine stage are accomplished by these two modules, respectively. Alternative hybrid algorithm was accomplished by integrating *K*-means in HClust

(IHKClust). Our experimental results illustrate that using these hybrid algorithms can generate higher quality clusters than using either the HClust or the *K*-means alone.

Acknowledgements

This research was in part supported by a grant from IPM. (No. CS1386-4-06).

References

[1] M.R. Anderberg, Cluster Analysis for Applications, Academic Press, Inc., New York, NY, 1973.
[2] E.M. Bozsak, S. Ehrig, A. Handschuh, A. Hotho, B. Maedche, D. Motik, C. Oberle, S. Schmitz, L. Staab, N. Stojanovic, R. Stojanovic, G. Studer, Y. Stumme, J. Sure, R. Tane, V. Volz, Zacharias, Kaon – towards a large scale semantic Web, in: K. Bauknecht, A. Min Tjoa, G. Quirchmayr (Eds.), E-Commerce and Web Technologies, Third International Conference Proceedings, EC-Web, LNCS, vol. 2455, Springer, 2002, pp. 304–313.
[3] K. Cios, W. Pedrycs, R. Swiniarski, Data Mining-Methods for Knowledge Discovery, Kluwer Academic Publishers, 1998.
[4] C.A.C. Coello, Constraint-handling using an evolutionary multi objective optimization technique, Civil. Eng. Environ. Syst. 17 (2000) 319–346.
[5] X. Cui, T.E. Potok, P. Palathingal, Document Clustering using Particle Swarm Optimization, IEEE Swarm Intell. Symp. (2005) 185–191.
[6] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Meth. Appl. Mech. Eng. 186 (2000) 311–338.
[7] B. Everitt, Cluster Analysis, second ed., Halsted Press, New York, 1980.
[8] Z.W. Geem, C. Tseng, Y. Park, Harmony search for generalized orienteering problem: best touring in China, Springer Lect. Notes Comput. Sci. 3412 (2005) 741–750.
[9] N. Grira, M. Crucianu, N. Boujemaa, Unsupervised and semi-supervised clustering: a brief survey, in: 7th ACM SIGMM International Workshop on Multimedia Information Retrieval, 2005, pp. 9–16.
[10] J. McQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297.
[11] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. (CSUR) 31 (3) (1999) 264–323.
[12] Jones Gareth, A.M. Robertson, Santimetvirul Chawchat, P. Willett, Non-hierarchic document clustering using a genetic algorithm, Informat. Res. 1 (1) (1995).
[13] J. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence, Morgan Kaufman, New York, 2001.
[14] B. Larsen, C. Aone, Fast and effective text mining using linear-time document clustering, in: Proceedings of SIGKDD'99, CA, 1999, pp. 16–22.
[15] N. Labroche, N. Monmarche', G. Venturini, AntClust: ant clustering and web usage mining, Genet. Evolut. Comput. Conf. (2003) 25–36.
[16] K.S. Lee, Z.W. Geem, A new meta-heuristic algorithm for continues engineering optimization: harmony search theory and practice, Comput. Meth. Appl. Mech. Eng. 194 (2004) 3902–3933.
[17] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Appl. Math. Comput. 188 (2) (2007) 1567–1579.
[18] V.D. Merwe, A.P. Engelbrecht, Data clustering using particle swarm optimization, in: Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003), Australia, 2003, pp. 215–220.
[19] M.G.H. Omran, M. Mahdavi, Global-best harmony search, Appl. Math. Comput. 198 (2008) 643–656.
[20] M. Omran, A. Salman, A.P. Engelbrecht, Image classification using particle swarm optimization, in: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002 (SEAL 2002), Singapore, 2002, pp. 370–374.
[21] R.J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, 1993.
[22] V.V. Raghavan, K. Birchand, 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, 1979, pp. 10–22.
[23] G. Salton, Automatic Text Processing, Addison-Wesley, 1989.
[24] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Informat. Process. Manage. 24 (5) (1988) 513–523.
[25] S.Z. Selim, M.A. Ismail, *K*-means type algorithms: a generalized convergence theorem and characterization of local optimality, IEEE Trans. Pattern Anal. Mach. Intell. 6 (1984) 81–87.
[26] M. Steinbach, G. Karypis, V. Kumar, A comparison of document clustering techniques, KDD'2000, Technical Report of University of Minnesota, 2000.
[27] G. Stumme, A. Hotho, B. Berendt, Semantic Web Mining State of the art and future directions journal of Web Semantics, Sci. Serv. Agents World Wide Web 4 (2) (2006) 124–143.
[28] G. Stumme, A. Hotho, B. Berendt, Semantic web mining, Freiburg, September 3rd, in: 12th European Conference on Machine Learning (ECML'01)/5th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'01), 2001.
[29] A. Vasebi, M. Fesanghary, S.M.T. Bathaeea, Combined heat and power economic dispatch by harmony search algorithm, Int. J. Elec. Power 29 (2007) 713–719.
[30] Y. Zhao, G. Karypis, Empirical and theoretical comparisons of selected criterion functions for document clustering, Mach. Learn. 55 (3) (2004) 311–331.