ARTICLE IN PRESS

Information Sciences xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

Efficient stochastic algorithms for document clustering

Rana Forsati [a,*], Mehrdad Mahdavi [b], Mehrnoush Shamsfard [a], Mohammad Reza Meybodi [c,d]

[a] Faculty of Electrical and Computer Engineering, Shahid Beheshti University, G. C., Tehran, Iran
[b] Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
[c] Department of Computer Engineering and IT, Amirkabir University of Technology, Tehran, Iran
[d] Institute for Studies in Theoretical Physics and Mathematics (IPM), School of Computer Science, Tehran, Iran

ARTICLE INFO

Article history:
Received 16 January 2009
Received in revised form 18 June 2012
Accepted 20 July 2012
Available online xxxx

Keywords:
Document clustering
Stochastic optimization
Harmony search
K-means
Hybridization

ABSTRACT

Clustering has become an increasingly important and highly complicated research area for targeting useful and relevant information in modern application domains such as the World Wide Web. Recent studies have shown that the most commonly used partitioning-based clustering algorithm, the *K*-means algorithm, is more suitable for large datasets. However, the *K*-means algorithm may generate a local optimal clustering. In this paper, we present novel document clustering algorithms based on the Harmony Search (HS) optimization method. By modeling clustering as an optimization problem, we first propose a pure HS based clustering algorithm that finds near-optimal clusters within a reasonable time. Then, harmony clustering is integrated with the *K*-means algorithm in three ways to achieve better clustering by combining the explorative power of HS with the refining power of the *K*-means. Contrary to the localized searching property of *K*-means algorithm, the proposed algorithms perform a globalized search in the entire solution space. Additionally, the proposed algorithms improve *K*-means by making it less dependent on the initial parameters such as randomly chosen initial cluster centers, therefore, making it more stable. The behavior of the proposed algorithm is theoretically analyzed by modeling its population variance as a Markov chain. We also conduct an empirical study to determine the impacts of various parameters on the quality of clusters and convergence behavior of the algorithms. In the experiments, we apply the proposed algorithms along with *K*-means and a Genetic Algorithm (GA) based clustering algorithm on five different document datasets. Experimental results reveal that the proposed algorithms can find better clusters and the quality of clusters is comparable based on F-measure, Entropy, Purity, and Average Distance of Documents to the Cluster Centroid (ADDC).

1. Introduction

The continued growth of the Internet has made available an ever-growing collection of full-text digital documents and new opportunities to obtain useful information from them [3,16,45]. At the same time, acquiring useful information from such immense quantities of documents presents new challenges which has led to increasing interest in research areas such as information retrieval, information filtering and text clustering. Clustering is one of the crucial unsupervised techniques for dealing with massive amounts of heterogeneous information on the web [9,25,41], with applications in organizing information, improving search engines results, enhancing web crawling, and information retrieval or filtering. Clustering is the

* Corresponding author.
E-mail addresses: rana.forsati@gmail.com (R. Forsati), mahdavi@ce.sharif.edu (M. Mahdavi), m-shams@sbu.ac.ir (M. Shamsfard), mmeybodi@aut.ac.ir (M. Reza Meybodi).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.07.025

process of grouping a set of data objects into a set of meaningful partitions, called clusters, such that data objects within the same cluster are highly similar in comparison with one another and are very highly dissimilar to objects in other clusters.

Some of the most conventional clustering algorithms can be broadly classified into two main categories, hierarchical and partitioning algorithms [23,26]. Hierarchical clustering algorithms [22,28,38,52] create a hierarchical decomposition of the given dataset which forms dendrograma tree by splitting the dataset recursively into smaller subsets, representing the documents in a multi-level structure [14,21]. The hierarchical algorithms can be further divided into either agglomerative or divisive algorithms [51]. In agglomerative algorithms, each document is initially assigned to a different cluster. The algorithm then repeatedly merges pairs of clusters until a certain stopping criterion is met [51]. Conversely, divisive algorithms repeatedly divide the whole documents into a certain number of clusters, increasing the number of clusters at each step. Partition clustering, the second major category of algorithms, is the most practical approach for clustering large data sets [6,7]. They cluster the data in a single level rather than a hierarchical structure such as a dendrogram. Partitioning methods try to divide a collection of documents into a set of groups, so as to maximize a pre-defined objective value.

It is worth mentioning that although the hierarchical clustering methods are often said to have better quality, they generally do not provide the reallocation of documents, which could have been poorly classified in the early stages of the clustering [26]. Moreover, the time complexity of hierarchical methods is quadratic in the number of data objects [49]. Recently, it has been shown that the partitioning methods are more advantageous in applications involving large datasets due to their relatively low computational complexity [8,29,36,49,55]. The time complexity of partitioning techniques are almost linear, which makes them appealing for large scale clustering. The best known method in partitioning clustering is *K*-means algorithm [34].

Although *K*-means algorithm is straightforward, easy to implement, and works fast in most situations, it suffers from some major drawbacks that make it unsuitable for many applications. The first disadvantage is that the number of clusters *K* must be specified in advance. In addition, since the summary statistic that is maintained for each cluster by *K*-means algorithm is simply the mean of samples assigned to that cluster, the individual members of the cluster can have a high variance and hence the mean may not be a good representative for the cluster members. Further, as the number of clusters grows into the thousands, *K*-means clustering becomes untenable, approaching $O(m^2)$ comparisons where m is the number of documents. However, for relatively few clusters and a reduced set of pre-selected features, *K*-means performs well [50]. Another major drawback of the *K*-means algorithm is its sensitivity to initialization. Lastly, the *K*-means algorithm converges to local optima, potentially leading to clusters that are not globally optimal.

To alleviate the limitations of traditional partition based clustering methods discussed above, particularly the *K*-means algorithm, different techniques have been introduced in recent years. One of these techniques involves the use of optimization methods that optimize a pre-defined clustering objective function. Specifically, optimization based methods define a global objective function over the quality of clustering algorithm and traverse the search space trying to optimize its value. Any general purpose optimization method can serve as the basis of this approach such as Genetic Algorithms (GAs) [10,26,40], Ant Colony Optimization [43,46] and Particle Swarm Optimization [11,12,53], which have been used for web page and image clustering. Since stochastic optimization approaches are good at avoiding convergence to a locally optimal solution, these approaches could be used to find a global near-optimal solution [35,30,48]. However the stochastic approaches take a long time to converge to a globally optimal partition.

Harmony Search (HS) [18,32] is a new meta-heuristic optimization method imitating the music improvisation process where musicians improvise the pitches of their instruments searching for a perfect state of harmony. HS has been very successful in a wide variety of optimization problems [17–19,32], presenting several advantages over traditional optimization techniques such as: (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, and (c) the HS algorithm generates a new vector, after considering all of the existing vectors, whereas methods such as GA only consider the two parent vectors. These three features increase the flexibility of the HS algorithm.

The behavior of the *K*-means algorithm is mostly influenced by the number of specified clusters and the random choice of initial cluster centers. In this study we concentrate on tackling the latter issue, trying to develop efficient algorithms generating results which are less dependent on the chosen initial cluster centers, and hence are more stabilized. The first algorithm, called Harmony Search CLUSTtering (HSCLUST), is good at finding promising areas of the search space but not as good as *K*-means at fine-tuning within those areas. To improve the basic algorithm, we propose different hybrid algorithms using both *K*-means and HSCLUST, that differ on the stage in which we carry out the *K*-means algorithm. The hybrid methods improve the *K*-means algorithm by making it less dependent on the initial parameters such as randomly chosen initial cluster centers, and hence, are more stable. These methods combine the power of the HSCLUST with the speed of *K*-means. By combining these two algorithms into a hybrid algorithm, we hope to create an algorithm that outperforms either of its constituent parts. The advantage of these algorithms over *K*-means is that the influence of the improperly chosen initial cluster centers will be diminished by enabling the algorithm to explore the entire decision space over a number of iterations and simultaneously increasing its fine-tuning capability around the final decision. Therefore, it will be more stabilized and less dependent on the initial parameters such as randomly chosen initial cluster centers, while it is more likely to find the global solution rather than a local one. To demonstrate the effectiveness and speed of HSCLUST and hybrid algorithms, we have applied these algorithms to various standard datasets and achieved very good results compared to *K*-means and a GA based clustering algorithm [4]. The evaluation of the experimental results shows considerable improvements and demonstrate the robustness of the proposed algorithms.

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

The remainder of this paper is organized as follows. Section 2 provides a brief overreview of the vector-space model for document representation, particularly the aspects necessary to understand document clustering. Section 3 provides a general overview of *K*-means and HS algorithm. Section 4 introduces our HS-based clustering algorithm named HSCLUST, as well as theoretical analysis of its convergency and time complexity. The hybrid algorithms are explained in Section 5. The time complexity of each proposed hybrid algorithm is included after the algorithm. Section 6 presents the document sets used in our experiments, quality measures we used for comparing algorithms, empirical study of HS parameters on the convergence of HSCLUST, and finally the performance evaluation of the proposed algorithms compared to *K*-means and a GA based clustering algorithm. Finally, Section 7 concludes the paper.

2. Preliminaries

In this section we discuss some aspects that almost all clustering algorithms share.

2.1. Document representation

In document clustering, the vector-space model is usually used to represent documents and to measure the similarity among them. In the vector-space model, each document i is represented by a weight vector of n features (words, terms, or N-grams) as follows:

$$\mathbf{d}_i = (w_{i1}, w_{i2}, \ldots, w_{in}), \tag{1}$$

where the weight w_{ij} is the weight of feature j in document i and n is the total number of the unique features. The most widely used weighting schema is the combination of term frequency and inverse document frequency (TF-IDF) [16,45], which can be computed by the following formula [44]:

$$w_{ij} = \mathrm{tf}(i,j) \times \mathrm{idf}(i,j) = \mathrm{tf}(i,j) \times \log \frac{m}{\mathrm{df}(j)}, \tag{2}$$

where tf (i,j) is the term frequency, i.e. the number of occurrences of feature j in a document $\mathbf{d}_i$, and idf (i,j) is the inverse document frequency. idf (i,j) is a factor which enhances the terms which appear in fewer documents, while downgrades the terms occurring in many documents and is defined as idf $(i,j) = \log (m/\mathrm{df}\,(j))$, where m is the number of documents in the whole collection, and df (j) is the number of documents where feature j appears.

One of the major problems in text mining is that a document can contain a very large number of words. If each of these words is represented as a vector coordinate, the number of dimensions would be too high for the text mining algorithm. Hence, it is crucial to apply preprocessing methods that greatly reduce the number of dimensions (words) to be given to the text mining algorithm. As an example, the very common words (e.g. function words: a, the, in, to; pronouns: I, he, she, it) are removed completely and different forms of a word are reduced to one canonical form using Porters algorithm [29].

2.2. Similarity measures

In clustering, the similarity between two documents needs to be measured. There are two prominent methods to compute the similarity between two documents $\mathbf{d}_1$ and $\mathbf{d}_2$. The first method is based on Minkowski distances [6], given two vectors, $\mathbf{d}_1 = (w_{11}, w_{12}, \ldots, w_{1n})$ and $\mathbf{d}_2 = (w_{21}, w_{22}, \ldots, w_{2n})$, their Minkowski distance is defined as:

$$D_p(\mathbf{d}_1, \mathbf{d}_2) = \left(\sum_{i=1}^{n} |w_{1i} - w_{2i}|^p \right)^{\frac{1}{p}}, \tag{3}$$

which is converted to Euclidean distance for $p = 2$. The other commonly used similarity measure in document clustering is the cosine correlation measure [45], given by:

$$\cos(\mathbf{d}_1, \mathbf{d}_2) = \frac{\mathbf{d}_1 \cdot \mathbf{d}_2}{\|\mathbf{d}_1\| \times \|\mathbf{d}_2\|}, \tag{4}$$

where $\cdot$ denotes the dot product of two vectors, and $\|\cdot\|$ denotes the length of a vector. This measure becomes 1 if the documents are identical and zero if they have nothing in common (i.e., the vectors are orthogonal to each other). Both metrics are widely used in literatures on text document clustering. However, it seems that in cases where the number of dimensions of two vectors differs greatly, the cosine is more useful. Conversely, where vectors have nearly equal dimensions, Minkowski distance can be useful. For another measure which is designed specifically for high dimensional vector spaces such as documents, we refer interested readers to [15].

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

3. Basic algorithms

3.1. The K-means algorithm

The K-means algorithm, first introduced in [34] is an unsupervised clustering algorithm which partitions a set of object into a predefined number of clusters. The K-means algorithm is based on the minimization of an objective function which is defined as the sum of the squared distances from all points in a cluster to the cluster center [33]. The K-means algorithm with its many variants is the most popular clustering method, gaining popularity because of its simplicity and intuition. Formally, the K-means clustering algorithm using matrix notation is defined as follows. Let X be the $m \times n$ data matrix associated with documents $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_m\}$. The goal of K-means algorithm is to find optimal $m \times K$ indicator matrix A^* such that

$$A^* = \arg\min_{A \in \Omega} \|X - AA^T X\|_F^2, \tag{5}$$

where Ω is the set of all $m \times K$ indicator matrices and K denotes the number of clusters. To solve (5), K-means starts with randomly selected initial cluster centroids and iteratively reassigns the data objects to clusters based on the similarity between the data object and the cluster centroid. The reassignment procedure will not stop until a convergence criterion is met (e.g., the fixed iteration number is reached, or the cluster result does not change after a certain number of iterations). This procedure is detailed in Algorithm 1.

Algorithm 1. K-means algorithm

1: **Input**: a collection of training documents $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_m\}$, number of clusters K
2: **Output**: an assignment matrix A of documents to set of K clusters
3: Randomly select K documents as the initial cluster centers
4: **repeat**
5: Randomly choose $C = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K)$ as initial centroids
6: Initialize A as zero
7: **for all** $\mathbf{d}_i$ in $\mathcal{D}$ **do**
8: let $j = \text{argmin}_{k \in \{1,2,\ldots,K\}} D(\mathbf{d}_i, \mathbf{c}_k)$
9: assign $\mathbf{d}_i$ to the cluster j, i.e. $A[i][j] = 1$
10: **end for**
11: Update the cluster means as $\mathbf{c}_k = \frac{\sum_{i=1}^{m} \left(\sum_{k=1}^{K} A[i][k] \mathbf{d}_i \right)}{\sum_{i=1}^{n} \sum_{k=1}^{K} A[i][k]}$ for $k = 1, 2, \ldots, K$
12: **until** meeting a given criterion function
13:

The K-means algorithm tends to find local minima rather that the global minimum since it is heavily influenced by the selection of the initial cluster centers and the distribution of data. Most of the time, the results become more acceptable when initial cluster centers are chosen relatively far apart since the main clusters in a given data are usually distinguished in that way. The initialization of the cluster centroids affects the main processing of the K-means as well as the quality of the final partitioning of the dataset. Therefore the quality of the result is dependent on the initial points. If the main clusters in a given data are close in characteristics, the K-means algorithm fails to recognize them when it is left unsupervised. For its improvement, the K-means algorithm needs to be associated with some optimization procedures in order to be less dependent on a given data and initialization. Notably, if good initial clustering centroids can be obtained using an alternative technique, K-means will work well in refining the clustering centroids to find the optimal clustering centers [2]. Our intuition for hybrid algorithms stems from this observation about K-means algorithm as discussed later.

3.2. The harmony search algorithm

Harmony Search (HS) [32] 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 superior performance of the HS algorithm has been demonstrated through its application to different problems. The main reason for this success is the explorative power of the HS algorithm which expresses its capability to explore the entire search space. The evolution of the expected population variance over generations provides a measure of the explorative power of the algorithm. Here, we provide a brief introduction to the main algorithm and the interested reader can refer to [13] for theoretical analysis of the exploratory power of the HS algorithm. The main steps of algorithm are described in the next five subsections.

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

3.2.1. Initialize the problem and algorithm parameters

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

Minimize $f(\mathbf{x})$ subject to:

$$\begin{aligned} &g_i(\mathbf{x}) \geqslant 0 \quad i = 1,2,\ldots,m, \\ &h_j(\mathbf{x}) = 0 \quad j = 1,2,\ldots,p, \\ &LB_k \leqslant x_k \leqslant UB_k \quad k = 1,2,\ldots,n, \end{aligned} \tag{6}$$

where $f(\mathbf{x})$ is the objective function, m is the number of inequality constraints and p is the number of equality constraints and n is the number of decision variables. The lower and upper bounds for each decision variable k are LB_k and UB_k respectively. The HS parameters are also specified in this step. These are the harmony memory size (HMS), or the number of solution vectors in the harmony memory, the probability of memory considering (HMCR), the probability of pitch adjusting (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.

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 allows:

$$HM = \left[\begin{array}{ccccc|c} x_1^1 & x_2^1 & \ldots & x_{n-1}^1 & x_n^1 & f(\mathbf{x}_1) \\ x_1^1 & x_2^1 & \ldots & x_{n-1}^1 & x_n^1 & f(\mathbf{x}_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^{HMS-1} & x_2^{HMS-1} & \ldots & x_{n-1}^{HMS-1} & x_n^{HMS-1} & f(\mathbf{x}_{HMS-1}) \\ x_1^{HMS} & x_2^{HMS} & \ldots & x_{n-1}^{HMS} & x_n^{HMS} & f(\mathbf{x}_{HMS}) \end{array}\right].$$

The initial harmony memory is generated from a uniform distribution in the ranges $[LB_i, UB_i]$, where $1 \leqslant i \leqslant n$. This is done as follows:

$$x_i^j = LB_i + r \times (UB_i - LB_i), \quad j = 1,2,\ldots,HMS, \tag{7}$$

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

3.2.3. Improvise a new harmony

Generating a new harmony is called improvisation. A new harmony vector, $\mathbf{x}' = (x'_1, x'_2, \ldots, x'_n)$, is generated based on three rules: memory consideration, pitch adjustment, and random selection. In the memory consideration, the value for a decision variable is randomly chosen from the historical values stored in the HM with the probability of HMCR. 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 probability of pitch adjustment. Variables which are not selected for memory consideration will be randomly chosen from the entire possible range with a probability equal to 1-HMCR. The pseudo code shown in Algorithm 2 describes how these rules are utilized by the HS.

Algorithm 2. Improvise a new harmony

```
1: Input: current solutions in harmony memory HM
2: Output: new harmony vector x' = (x'_1, x'_2, ..., x'_n)
3: for each i ∈ [1,n] do
4:   if U(0,1) ⩽ HMCR then
5:     x'_i = HM[j][i] where j ~ U(1,2,...,HMS)
6:     if U(0,1) ⩽ PAR then
7:       x'_i = x'_i ± r × bw, where r ~ U(0,1) and bw is an arbitrary distance bandwidth
8:     end if
9:   else
10:     x'_i = LB_i + r × (UB_i − LB_i)
11:  end if
12: end for
```

3.2.4. Update harmony memory

If the new harmony vector, $\mathbf{x}' = (x'_1, x'_2, \ldots, x'_n)$, has better fitness value than the worst harmony in the HM, the new harmony is included in the HM and the existing worst harmony is excluded from it.

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

3.2.5. Check stopping criterion

The HS is terminated when the stopping criterion (e.g., maximum number of improvisations) has been met. Otherwise, Steps 3 and 4 are repeated.

We note that in recent years, some researchers have improved the original HS algorithm. Mahdavi et al. [35] proposed an improved variant of HS by using varying parameters. The intuition behind this algorithms is as follows. Although the HMCR and PAR parameters of HS help the method in searching for globally and locally improved solutions, respectively, however PAR and *bw* parameters have a profound effect on the performance of the HS. Thus, fine tuning these two parameters is very important. Of the two parameters, *bw* is more difficult to tune because it can take any value from $(0,\infty)$. To address these shortcomings of HS, a new variant of HS, called the Improved Harmony Search (IHS), is proposed in [35]. IHS dynamically updates PAR according to the following equation,

$$PAR(t) = PAR_{min} + \frac{PAR_{max} - PAR_{min}}{NI} \times t, \tag{8}$$

where $PAR(t)$ is the pitch adjusting rate for generation t, PAR_{min} is the minimum adjusting rate, PAR_{max} is the maximum adjusting rate and NI is the maximum number of generations. In addition, bw is dynamically updated as follows:

$$bw(t) = bw_{max} \exp\left(\frac{\ln\left(\frac{bw_{min}}{bw_{max}}\right)}{NI} \times t\right), \tag{9}$$

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

In order to overcome the parameter setting problem of HS, which is very tedious and could be another daunting optimization problem, Geem et al. [20] introduced a variant of HS which eliminates tedious and difficult parameter assigning efforts.

4. HSCLUST: the basic harmony search based algorithm for document clustering

In this section we first propose our pure harmony search based clustering algorithm which is called HSCLUST. Then by modeling HSCLUST population variance as a Markov chain, its behavior is theoretically analyzed. The time complexity of HSCLUST will be analyzed in subSection 4.3.

4.1. HSCLUST algorithm

All proposed algorithms represent documents using the vector-space model discussed before. In this model, each term represents one dimension of multidimensional document space, each document $\mathbf{d}_i = (w_{i1}, w_{i2}, \ldots, w_{in})$ is considered to be a vector in the term space with n different terms, and each possible solution for clustering is a vector of centroids.

Clustering problem is casted as an optimization task in which the objective is to locate the optimal cluster centroids rather than finding an optimal partition. To this end, we chose the clustering quality as the objective function and utilize the HS algorithm to optimize the objective. The principal advantage of this approach is that the objective of the clustering is explicit, enabling us to better understand the performance of the clustering algorithm on particular types of data and to also allowing us use task-specific clustering objectives. It is also possible to consider several objectives simultaneously, as recently explored in [24].

When a general purpose optimization meta-heuristic is used for clustering, a number of important design choices have to be made. Predominantly, these are the problem representation and the objective function. Both of these have a significant effect on optimization performance and clustering quality. The following subsections describe the HSCLUST algorithm.

4.1.1. Representation of solutions

The proposed algorithm uses representations which codify the whole partition $\mathcal{P}$ of the document set in a vector of length m, where m is the number of the documents. Each element of this vector is the label where the single document belongs to; in particular if the number of clusters is K, each element of the solution vector is an integer value in the range $[K] = \{1, \ldots, K\}$. An assignment that represents K non-empty clusters is a legal assignment. Each assignment corresponds to a set of K centroids.

Accordingly, the search space is the space of all permutations of size m from the set $\{1, \ldots, K\}$ that satisfies the constraint that enforces the algorithm to allocate each document to exactly one cluster and no cluster is empty. This problem is well known to be NP-hard even for $K = 2$. A natural way of encoding such permutations is to consider each row of the HM as an integer vector of m positions where the ith position represents the cluster which the ith document is assigned to. An example of representation of solutions is shown in Fig. 1. In this case, five documents $\{1,2,7,10,12\}$ are from the cluster with label 2. The cluster with label 1 has two documents $\{3,8\}$, and so on.

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

Fig. 1. Representation of clusters as an array of integers from the set $\{1, 2, \ldots, K\}$.

4.1.2. Initialization

In this step, harmony memory is filled with randomly generated feasible solution vectors. Each row of harmony memory corresponds to a specific cluster of documents in which, the value of the ith element in each row is randomly selected from the uniform distribution over the set $\{1, \ldots, K\}$. Such a randomly generated solution may not be legal since it is possible that no document is allocated to some of the clusters. This is avoided by assigning randomly chosen documents to each cluster and the rest of the documents to randomly chosen clusters. In contrast to K-means algorithm, the HS-based clustering algorithm is not sensitive to initialization of the HM but an intelligent initialization would slightly increase the convergence time of the algorithm.

4.1.3. Improvisation step

In the improvising step a new solution vector, namely NHV, is generated from the solution vectors stored in HM. The newly generated harmony vector must inherit as much information as possible from the solution vectors in HM. If the generated vector, which corresponds to a new clustering, consists mostly or entirely of assignments found in the vectors in HM, it provides good heritability.

In this algorithm each decision variable corresponds to a document and a value for a decision variable shows its cluster label. The selection of a value for the cluster of a document is as follows: the cluster number of each document in the new solution vector is selected from HM with probability HMCR and with probability 1-HMCR is randomly selected from the set $\{1,2,\ldots,K\}$. After generating the new solution, the pitch adjustment process is applied. The PAR parameter is the rate of allocating a different cluster to a document. PAR controls the fine-tuning of optimized solution vectors, thus influenc the convergence rate of the algorithm to an optimal solution.

In contrast to original HS and most of its variants where originally developed for continuous variable optimization problems, our algorithm uses a discrete representation of solutions and, consequently, we need to modify the pitch adjusting process for this type of optimization. To best lead the algorithm during improvising step we define two different PAR parameters (i.e., $PAR_1 = 0.6 \times PAR$ and $PAR_2 = 0.3 \times PAR$). For each document $\mathbf{d}_i$, whose cluster label is selected from HM, with probability of PAR_1 the current cluster of $\mathbf{d}_i$ is replaced with a new cluster for which $\mathbf{d}_i$ has the minimum distance to it according to:

$$NHV[i] = \arg\min_{j\in[K]} D(\mathbf{d}_i, \mathbf{c}_j) \tag{10}$$

and with probability PAR_2 the current cluster of $\mathbf{d}_i$ is replaced with a new cluster chosen randomly from the following distribution:

$$p_j = \Pr\{\text{cluster } j \text{ is selected as new cluster}\} = \frac{D_{max} - D(\mathbf{d}_i, \mathbf{c}_j)}{NF}\left(1 - \frac{gn}{NI}\right), \tag{11}$$

where $NF = KD_{max} - \sum_{j=1}^{K} D(\mathbf{d}_i, \mathbf{c}_j), D_{max} = \max_i D(NHV, \mathbf{c}_i), NI$ is the total number of iterations, and gn is the number of current iteration.

It should be noted that in Eq. (11) the probabilities of assignments change dynamically with generation number. In contrast to fixed probability adjusting, an adaptive probability produces a high value in the earlier generations widening the search space. In the initial stages the degree of distribution is small, helping to maintain the diversity of the population. As the search proceeds, the degree of distribution is increased, which speeds up the convergence of the algorithm. This ensures that most of the search space will be explored in the initial stages whereas the final stages will focus on fine tuning the solution.

4.1.4. Evaluation of solutions

As mentioned before, each row in HM corresponds to a clustering of documents. Let $C = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K)$ be the set of K centroids corresponding to a row in HM. The centroid of the kth cluster is $\mathbf{c}_k = (c_{k1}, \ldots, c_{kn})$ and is computed as follows:

$$c_{kj} = \frac{\sum_{i=1}^{m} a_{ki} d_{ij}}{\sum_{l=1}^{n} a_{ki}}. \tag{12}$$

The objective function is to determine the locus of the cluster centroids in order to maximize intra-cluster similarity (minimizing the intra-cluster distance) while minimizing the inter-cluster similarity (maximizing the distance between clusters). The fitness value of each row, corresponding to a potential solution, is determined by the Average Distance of Documents to the cluster Centroid (ADDC) represented by that row. ADDC is expressed as:

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

$$f = \left[\sum_{i=1}^{K}\frac{1}{n_i}\sum_{j=1}^{m_i}D(\mathbf{c}_i,\mathbf{d}_j)\right]\Big/K, \tag{13}$$

where K is the number of clusters, m_i is the numbers of documents in cluster i (e.g., $n_i = \sum_{j=1}^{n} a_{ij}$), $D(\cdot,\cdot)$ is the similarity function, and d_{ij} is the jth document of cluster i. The newly generated solution is replaced with a row in HM if the locally optimized vector has better fitness value than the solutions in HM.

4.1.5. Stopping criterion

The HSCLUST stops when either the average fitness does not change by a predefined value ϵ after a number of iterations or the maximum number of generations is reached.

4.2. Theoretical analysis of HSCLUST behavior

By modeling the changes of HM as the result of HSCLUST operations as a Markov chain, we theoretically analyze the behavior of HSCLUST for successive generations. First, a lemma is proved for better analysis of binary HSCLUST and then it will be extended to general HSCLUST.

Let the correct assignment for a clustering be defined as the assignment which assigns a document to the correct cluster in HSCLUST. So the eligibility of an algorithm is the number of correct assignments obtained. Lemma 1 shows the impact of fitness function on the explorative power of the algorithm.

Lemma 1. *For binary clustering the probability of increasing correct assignments in HM is tightly dependent on the fitness ration between clusters.*

Proof. We limit the algorithms to binary clustering as clusters 0 and 1. So each document will be allocated to one of the two clusters. Consider the HM size of HMS. To simplify the analysis, we further assume that the impact of each decision variable on overall fitness is independent from other variables. Such a limitation allows for an intuitive numbering of states. Given a fixed population of size HMS, we define the state of HM as follows: state i is the population in which exactly i documents are assigned to cluster 1 and $m - i$ documents are assigned to cluster 0. We model the changes between HM's different populations as a finite state Markov chain with transition matrix $\mathbf{P}$ between states. Matrix $\mathbf{P}$ is a (HMS + 1) by (HMS + 1) matrix where (i,j) th entry indicates the probability of going from state i to state j.

Now we compute the transition probabilities in the Markov chain. Suppose the current state has i 1 s. We want to compute the probability for having j 1s after improvisation step of HSCLUST. First, we compute the probability that improvisation step generates 1 and denote it by p_1. Considering the steps of the algorithm, the NHV is obtained as:

$$NHV = \begin{cases} x_r & \text{with probability} \quad HMCR \times (1 - PAR) \\ x_{PAR} & \text{with probability} \quad HMCR \times PAR \\ x_{new} & \text{with probability} \quad 1 - HMCR \end{cases} \tag{14}$$

where x_r is memory consideration without pitch adjusting, x_{PAR} is memory consideration with pitch adjusting, and $x_{new} \in [0, 1]$ is random assignment of the cluster number. Then the probability of choosing cluster 1 as the document cluster for NHV is:

$$p_1 = HMCR \times (1 - PAR) \times \frac{i}{HMS} + \frac{1}{2}(1 - HMCR) + HMCR \times PAR \times \frac{f_1}{f_0}, \tag{15}$$

where f_1 is the fitness of the solution which is assigned to cluster 1 and f_0 is the fitness of solution assigned to cluster 0. The probability of assigning cluster 0 to document in the improvisation step can be similarly calculated as:

$$p_0 = HMCR \times (1 - PAR) \times \frac{HMS - i}{HMS} + \frac{1}{2}(1 - HMCR) + HMCR \times PAR \times \frac{f_0}{f_1}. \tag{16}$$

Having both probabilities of p_0 and p_1, the probability of going from a state in which the number of documents which are assigned to cluster 1 is i to any other state j is computed by:

$$p_{ij} = \binom{HMS}{j} p_1^j p_0^{HMS-j}. \tag{17}$$

We note that (17) defines a complete (HMS + 1) by (HMS + 1) transition matrix for any population with size HMS. □

From Lemma 1, one can easily observe that the probabilities are tightly dependent on the fitness ratio (i.e., f_0 and f_1). Since a key characteristic of many partition-based clustering algorithms is that they use a global criterion function whose optimization drives the entire clustering process, the role of fitness function is of most importance. In Lemma 1, by assuming independent impact of documents cluster on the fitness function, it is clear that the fitness function has a major role in the

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

explorative power of the algorithm and will become more important if we skip this assumption. So, choosing a good fitness function will lead the algorithm to the optimal solution. The ADDC exactly models the correlation between the final clustering of the documents as the quality of the algorithm.

In the following theorem, we analyze the behavior of the HSCLUST in a general case and relate the quality of the solutions in successive generations.

Theorem 1. *Let X be the current solutions in HM. The expected number of following generations until Γ percentage of solutions in HM has fitness greater than the fitness of elitist solution in X is*

$$g = \gamma \log \gamma + \gamma \log\left(\frac{1}{\Gamma}\right), \tag{18}$$

where

$$\gamma = \frac{\mathrm{HMS}}{\mathrm{HMCR} \times 1 - \mathrm{PAR}}.$$

Proof. In the current population residing in HM, suppose the elitist solution is x with the fitness of $f(x)$. We call an individual x' *fit* if it has fitness of at least $f(x)$. We now estimate the number of improvising steps to find only fit individuals in the HM. Since the introduced PAR method can always improve the fitness of *NHV*, the resulting *NHV* vector without applying PAR process, will be considered here and our analysis is sound regardless of PAR process been applied on *NHV* or not. We model HM changes in different generations as a Markov chain. Let s_k, $k = 0, 1, \ldots, HMS$ represents the state where k fit individuals reside in HM.

Since the updating step of HS only replaces *NHV* with a solution in HM which has the worst fitness in HM, the number of fit individuals is an increasing function of generation number. Let $p_{ij}, 0 \leqslant i, j \leqslant HMS$ denote the probability of going from state i with i fit individuals to state j with j individuals. It is obvious from the previous discussion that p_{ij} for all $j < i$ is zero. p_{ii} is the probability of no changes in the number of fit individuals in HM.

Each element of *NHV* is selected from HM with probability $HMCR\,(1 - PAR)$ and from the set $\{1, 2, \ldots, K\}$ with probability $1 - HMCR$ without considering PAR process as justified before. So the probability that improvisation step creates a clone of fit individual in state i of HM equals to $HMCR(1 - PAR) \times \frac{i}{HMS}$. It is worth mentioning that this probability is a lower bound for the probability of increasing the number of fit individuals in state s_i after improvisation step. So, the transition probabilities between states of HM are as:

$$p_{i,i} \geqslant \frac{i}{HMS} HMCR(1 - PAR), \tag{19}$$

$$p_{i,i+1} \leqslant 1 - \left(\frac{i}{HMS} HMCR(1 - PAR)\right). \tag{20}$$

To simplify our analysis, we consider the equal cases for probabilities and the final result will be an upper bound on the expected number of generations. Having transition probability matrix **P** for Markov chain we follow the method in [39] to compute the n-step transition probabilities. One can easily obtain the eigenvalues λ_k for $k = 1, 2, \ldots, HMS$ by solving the characteristic equation $\det(P - \lambda I) = 0$. Then for each λ_k obtain the $\{x_i^{(k)}\}$ and $\{y_i^{(k)}\}$ vector components from

$$\sum_{j=1}^{N} p_{ij} x_j^{(k)} = \lambda_k x_i^{(k)}, \tag{21}$$

and

$$\sum_{j=1}^{N} y_i^{(k)} p_{ij} = \lambda_k y_i^{(k)}, \tag{22}$$

and then n-step transition probabilities are computable by

$$p_{ij}^{(n)} = \sum_{k=1}^{N} c_k \lambda_k^n x_i^{(k)} y_i^{(k)}, \tag{23}$$

where

$$c_k = \frac{1}{\sum_{i=1}^{N} x_i^{(k)} y_i^{(k)}}. \tag{24}$$

Let α equals $HMCR(1 - PAR)$. From (19) and (20) we have $p_{ii} = i\alpha/HMS$ and $p_{i,i+1} = \frac{HMS - i\alpha}{HMS}$ and so that (21) reduces to

$$p_{ii} x_i^{(k)} + p_{i,i+1} x_{i+1}^{(k)} = \lambda_k x_i^{(k)}, \tag{25a}$$

$$\frac{i\alpha}{HMS} x_i^{(k)} + \frac{HMS - i\alpha}{HMS} x_{i+1}^{(k)} = \lambda_k x_i^{(k)}, \tag{25b}$$

$$(HMS - i\alpha) x_{i+1}^{(k)} = (\lambda_k HMS - i\alpha) x_i^{(k)}. \tag{25c}$$

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

For $\lambda_k = 1$ the (25) gives $x_i = 1$ for all i. Since the eigenvectors are not identically zero vectors, so there must exist an integer k such that $x_{k+1} = 0$ but $x_k \neq 0$. In that case from (25) the eigenvalues are given by:

$$\lambda_k = \frac{k\alpha}{HMS}. \tag{26}$$

The corresponding solution for (25) are given by

$$(HMS - i\alpha)x_{i+1}^{(k)} = (k\alpha - i\alpha)x_i^{(k)}$$

or

$$x_i^{(k)} = \frac{k\alpha - i\alpha + \alpha}{HMS - i\alpha + \alpha}x_{i-1}^{(k)} \tag{27a}$$

$$= \frac{\alpha(k-i+1)}{\frac{HMS}{\alpha} - i + 1}x_{i-1}^{(k)} \tag{27b}$$

$$= \frac{\left(\frac{HMS}{\alpha} - i\right)!}{HMS!}\frac{k!}{(k-1)!} \tag{27c}$$

$$= \begin{cases} \dfrac{\binom{k}{i}}{\binom{\frac{HMS}{\alpha}}{i}} & i \leqslant k \\ 0 & i > k \end{cases}. \tag{27d}$$

Similarity for $\lambda_k = \frac{k\alpha}{HMS}$, the system of equations in (22) reduces to

$$\begin{aligned} &p_{j-1,j}y_{j-1}^{(k)} + p_{jj}y_j^{(k)} = \lambda_k y_j^{(k)}, \\ &\frac{HMS - (i-1)\alpha}{HMS}y_{j-1}^{(k)} + \frac{j\alpha}{HMS}y_j^{(k)} = \frac{k\alpha}{HMS}y_j^{(k)}, \\ &(HMS - i\alpha + \alpha)y_{j-1}^{(k)} = (k\alpha - j\alpha)y_j^{(k)}, \\ &\left(\frac{HMS}{\alpha} - i + 1\right)y_{j-1}^{(k)} = (k-j)y_j^{(k)}. \end{aligned}$$

Thus

$$y_j^{(k)} = \begin{cases} (-1)^{j-k}\binom{\frac{HMS}{\alpha} - k}{j-k} & j \geqslant k \\ 0 & j < k \end{cases}. \tag{28}$$

Since for $x_i^{(k)} = 0$ for $i > k$ and $y_i^{(k)} = 0$ for $i < k$, from (24) we get

$$c_k = \frac{1}{x_i^{(k)}y_i^{(k)}} = \frac{1}{(-1)^{j-k}\dfrac{\binom{k}{i}}{\binom{\frac{HMS}{\alpha}}{i}} \times \binom{\frac{HMS}{\alpha} - k}{i-k}} = \binom{\frac{HMS}{\alpha}}{k}. \tag{29}$$

Substitution of (27)–(29) into (23) we have:

$$p_{ij}^{(n)} = \sum_{k=i}^{j} c_k x_i^{(k)} y_j^{(k)} = \alpha^n \binom{\frac{HMS}{\alpha} - i}{\frac{HMS}{\alpha} - j} \sum_{k=i}^{j} (-1)^{j-k}\left(\frac{k\alpha}{HMS}\right)^n \binom{j-i}{n-i}. \tag{30}$$

For $j < i$ we have $p_{ij}^{(n)} = 0$. Substituting 0 for i in (30) we obtain:

$$p_{0,j}^{(n)} = \alpha^{2n} \binom{\frac{HMS}{\alpha}}{\frac{HMS}{\alpha} - j} \sum_{k=i}^{j} (-1)^{j-k}\left(\frac{k}{HMS}\right)^n \binom{j}{k}. \tag{31}$$

Approximating $p_{0,j}^{(n)}$ with $p_{1,j}^{(n)}$, $p_{0,j}^{(n)}$ represents the probability of having j fit individuals after n generations on a HM with one fit individual. Computing the limiting form of (31) with $n \to \infty$ we have

$$n = \frac{HMS}{\alpha}\log\left(\frac{HMS}{\alpha}\right) + \frac{HMS}{\alpha}\log\left(\frac{1}{\Gamma}\right), \tag{32}$$

which completes the proof. □

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

4.3. Time complexity analysis

In this subsection, we determine the time complexity of the HSCLUST algorithm.

Lemma 2. *The time complexity of K-means algorithm is O(n m K I).*

Proof. The computation complexity of *K*-means is determined by the number of documents (m), the number of terms in vector-space modeling of documents (n), the desired number of clusters (K), and the number of necessary iterations to achieve convergence (I). At each iteration, the computation complexity of the calculation step is dominated by the clustering similarity function which has time complexity of $O(n)$ for cosine similarity. For the update step, recalculating the centroids needs $O(nK)$ operations. Thus, the time complexity of the whole *K*-means algorithm is $T_{K-\text{ means}} = O(m\ n\ K)$ for a single iteration. For a fixed number of iterations I, the overall complexity is therefore $O(nm\ I\ K)$. □

Lemma 3. *The time complexity of the improvisation step is $O(\beta n\ K)$ where $\beta = HMCR \times PAR$.*

Proof. In the improvisation step a new clustering solution is generated by choosing a vector of n integers from the set $[K] = \{1, 2, \ldots, K\}$ using harmony operations. Each entry of the new solution is selected independently from other entries based on two operations for considering the computational intelligence or randomness as follows: The value of entry $i, 1 \leqslant i \leqslant m$ can be randomly selected from $[K]$ with a probability of $1 - HMCR$ which takes $O(1)$ times or it can be selected from the currently selected elements in the HM with a probability of *HMCR*. After selecting the entry from HM it is the subject of applying PAR process. In harmony memory operations, the computational cost is dominated by PAR process. The PAR process is applied after selection from memory with probability $PAR = PAR_1 + PAR_2$. In this step the current cluster number of each document can be slightly adjusted by replacing it with another cluster label. The time complexity of PAR process is as computing cluster centroids from a solution takes about $O(nm)$ which is computed once for all entries and so is not considered here. The time complexity of computing the similarity of document with all of the clusters is $O(nK)$. So the time complexity of applying PAR process on one entry supposing that cluster centroids are computed is $O(\beta n \times K)$ where $\beta = HMCR \times PAR$. So the overall complexity of PAR process for all entries is $O(n\ m + n\ mK)$. It is worth mentioning that the centroids are computed once for all entries and will be considered in the time complexity of HSCLUST. In conclusion the time complexity of whole improvisation step is $O(\beta n\ K)$.The time complexity depends on the memory consideration rate and probability of the PAR process which will be discussed in the empirical study. □

Theorem 2. *The time complexity of HSCLUST is $O(\beta n\ m\ K\ I)$ where $\beta = HMCR \times PAR$.*

Proof. Each generation of HSCLUST consists of two steps: improvisation of a new solution and computation of its fitness. After generation of NHV, the PAR process is applied on NHV. Considering the time complexity of computing centroids from NHV for applying PAR process and Lemma 2, the whole time of improvisation step takes $O(\beta n\ m\ K)$. After the new solution is generated, its fitness value must be computed to check that whether it can be swapped with the worst one in the HM. The fitness of each HSCLUST according to Eq. (20) is computed in $O(n\ m\ K)$. Since the improvising and fitness evaluation steps run sequentially, the time complexity of HSCLUST for I generations is $O(\beta n\ m\ K\ I)$. □

5. The hybrid algorithms

The algorithm discussed above performs a globalize search for solution, whereas *K*-means clustering procedure performs a localized search. In localized search, the solution obtained is usually located in the proximity of the solution obtained in the previous step. As mentioned previously, the *K*-means clustering algorithm uses the randomly generated seeds as the centroids of initial clusters and refines the position of the centroids at each iteration. The refinement process of the *K*-means algorithm indicates that the algorithm often explores the very narrow proximity, surrounding the initial randomly generated centroids and its final solution depends on these initially selected centroids [11]. Moreover, it has been shown that *K*-means may fail by converging to a local minimum [47]. *K*-means algorithm is good for fine-tuning, but lack a global perspective. On the other hand, the proposed algorithm HSCLUST 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. It seems that a hybrid algorithm that combines two ideas can result in an algorithm that can outperform either one individually. To improve the algorithm, we propose three different versions of the hybrid clustering, depending on the stage when we carry out the *K*-means algorithm.

5.1. The sequential hybridization

In this subsection, we present a hybrid clustering approach that uses *K*-means algorithm to replace the refining stage in the HSCLUST algorithm. Hybrid algorithm combines the explorative power of the HSCLUST with the speed of a *K*-means algorithm

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

in refining solutions. In the hybrid HS algorithm, the algorithm includes two modules, the HSCLUST module and the *K*-means module. The HSCLUST finds the optimum region, and then the *K*-means takes over to find the optimum centroids.

We need to find the right balance between *local exploitation* and *global exploration*. The global searching stage and local refining stage are accomplished by those two modules, respectively. In the initial stage, the HSCLUST module is executed for a short period (50–100 iterations) to discover the vicinity of the optimal solution by a global search and at the same time to avoid consuming high computation. The result from the HSCLUST module is used as the initial seed of the *K*-means module. The *K*-means algorithm will be applied for refining and generating the final result.

The following Lemma which is the direct application of Lemma 2 and Theorem 2 shows the time complexity of sequential hybridization.

Lemma 4. *The time complexity of Sequential Hybridization is* $O(\beta n\ m\ K\ I_1) + O(n\ m\ K\ I_2)$.

5.2. The interleaved hybridization

In this hybrid algorithm the local method is integrated into the HSCLUST. In particular, after every predetermined I_1 iterations, 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 is repeated until stopping condition.

Lemma 5. *Let* I_1 *and* I_2 *denote the number of iterations for HSCLUST and K-means respectively. The time complexity of Interleaved Hybridization is* $\frac{I}{I_1+I_2}(O(\beta nmKI_1) + O(nmKI_2))$.

Proof. In each interleaved step *K*-means and HSCLUST are executed I_1 and I_2 times respectively. So each interleaved step takes $O(\beta\ n\ m\ K\ I_1) + O(n\ m\ K\ I_2)$. Considering $I/(I_1 + I_2)$ interleaved steps the total time is clear. □

5.3. Hybridizing K-means as one step of HSCLUST

To improve the algorithm a one-step *K*-means algorithm is introduced. After that a new clustering solution is generated by applying harmony operations, and the following process is applied on the new solution.

The time complexity of one step HS + *K*-means is evident from Lemma 1 and Theorem 2. In this algorithm the explorative power of HSCLUST and the fine tuning power of *K*-means algorithms are interleaved in every iteration to obtain high quality clusters.

6. Experimental results and discussions

In this section we compare the proposed algorithms according to their quality, execution time, and speed of convergence using a number of different document sets.

6.1. Document collections

To fairly compare the performance of the algorithms, we used five different datasets with different characteristics in our experiments. The first data set, Politics, contains 176 randomly selected web documents on political topics. This data set was collected in 2006. The second data set, 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). This 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). The third data set is selected from DMOZ collection and contains 697 documents that are selected from 14 topics. 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 fourth 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, 20-newsgroups dataset is selected to evaluate the performance of algorithms on large data sets. The last dataset is from the WebACE project (WAP) [6,37]. Each document corresponds to a web page listed in the subject hierarchy of Yahoo!. Description of the test datasets is given in Table 1.

6.2. Experimental setup

In the next step, the *K*-means, HSCLUST and hybrid algorithms are applied to the above mentioned data sets. The cosine correlation measure is used as the similarity measure in each algorithm. It should be emphasized at this point that

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

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

Table 1
Summary description of document sets.

Document set	Source	# of documents	# of clusters
DS1	Politics	176	6
DS2	TREC	873	8
DS3	DMOZ	697	14
DS4	20 NEWSGROUP	9249	10
DS5	WebAce	1560	20

Table 2
Scenarios for analysis of convergence behavior.

Scenario	HMCR	HMS	PAR_{min}	PAR_{max}
Standard	0.6	12	0.45	0.9
2	0.9	12	0.45	0.9
3	0.1	12	0.45	0.9
4	0.95	12	0.45	0.9
5	0.98	12	0.45	0.9
6	0.6	1	0.45	0.9
7	0.6	6	0.45	0.9
8	0.6	30	0.45	0.9
9	0.6	35	0.45	0.9
10	0.6	48	0.45	0.9
11	0.6	12	0.05	0.45
12	0.6	12	0.45	0.45
13	0.6	12	0.05	0.9

the results shown in the rest of paper are the average of over 20 runs of the algorithms (to make a fair comparison). Also, for easy comparison, the algorithms run 1000 iterations in each run since the 1000 generations are enough for convergence of algorithms. No parameter needs to be set up for the *K*-means algorithm. For HSCLUST, for each data set the HMS is set to two times the number of cluster in the data set, HMCR is set to 0.6, and PAR is set to 0.45–0.9. In the hybrid *K*-means approach, it first executes the HSCLUST algorithm for 75% of total iterations and uses the HSCLUST result as the initial seed for the *K*-means module which executes for the remaining 25% of iterations to generate the final result.

6.3. Empirical study of the impact of different HS parameters on convergence behavior

The aim of this section is to study the solution evolution of the algorithms over generations under different settings of three important parameters. These are the pitch adjusting rate (PAR), harmony memory size (HMS), and harmony memory considering rate (HMCR). Keeping that in mind, we will now show the effects of single parameter changes. Particularly, we tested the following seven different scenarios as shown in Table 2. Each scenario was tested over 30 runs and the maximum number of iterations is fixed to 10,000 for all runs. The ADDC value of solution is the value of fitness function. The algorithm that we use to evaluate is HSCLUST which was described in Section 4. In Figs. 2–4 the average ADDC for the Politics data set is shown over the number of iterations.

The effects of variation of the HMCR are demonstrated in Fig. 2. As mentioned earlier, the HMCR determines the rate of choosing one value from the historical values stored in the HM. Larger HMCR leads to less exploration and the algorithm relies on the values stored in HM which potentially lead to the algorithm to get stuck in a local optimum. On the other hand, choosing very small HMCR will decrease the algorithm efficiency and will increase its diversity which prevents the algorithm from converging to the optimal solution. In this condition the HS algorithm behaves like a pure random search. According to Fig. 2, by increasing HMCR from 0.1 to 0.6, the results improve and the best result is achieved at 0.6. A similar behavior can also be seen when HMCR increase from 0.9 to 0.98. Therefore, no single choice is superior to the others indicating the relevance to increment or decrement of HMCR.

In Fig. 3 the evolution of solution for different values of HMS is shown. We can see that decreasing the HMS leads to premature convergence and increasing the HMS leads to significant improvements in the initial phase of a run. Note that when the time or the number of iterations is finite, increasing the HMS may deteriorate the quality of the clustering. In general we can say, the larger HMS, the more time (or iterations) is needed for algorithm to find the optimal solution, but usually higher quality is achieved. In general, using a mild HMS seems to be a good and logical choice with the advantages of converging to the best result as well as reducing space requirements. In addition, empirical studies demonstrate that with a linear relation between HMS and the number of clusters, better results are reached. Specifically setting HMS, two times the number of clusters (12 in this data set) leads to the best result.

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

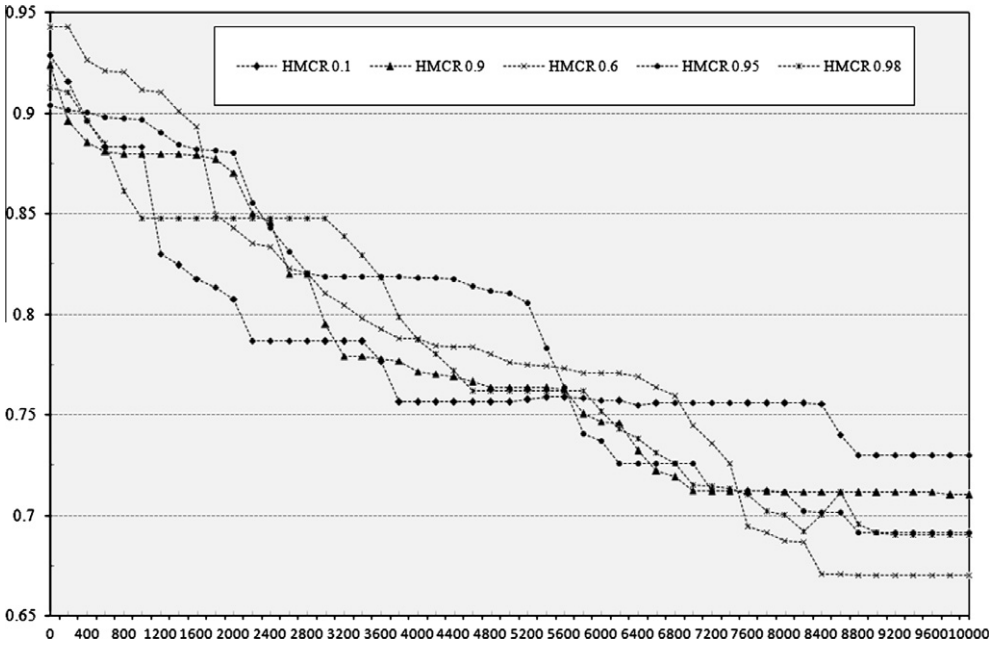

Fig. 2. Effects of the HMCR in the clustering quality.

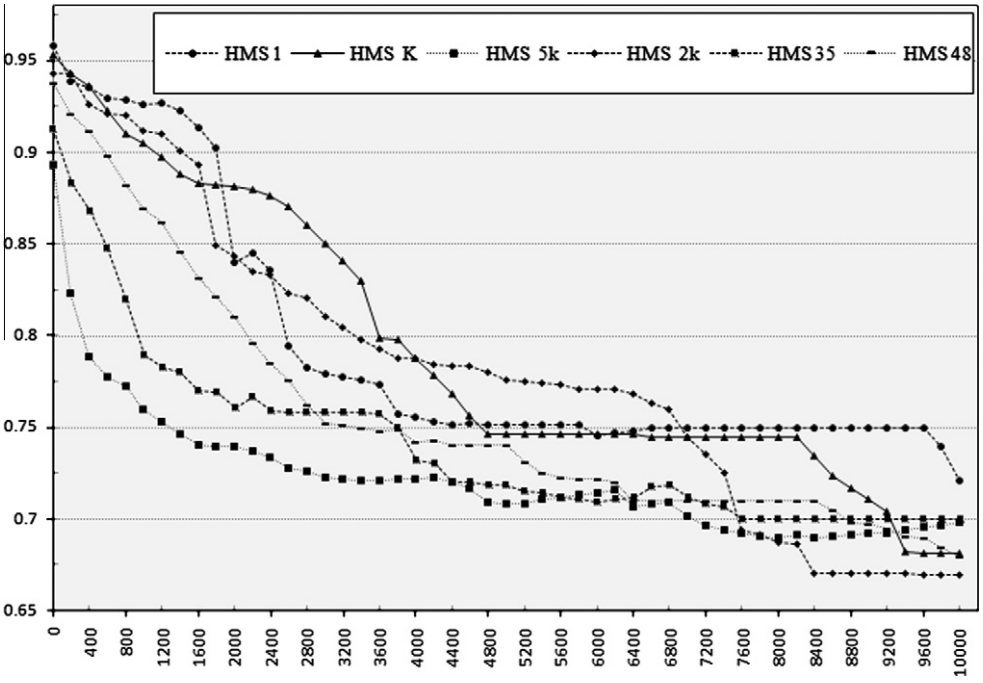

Fig. 3. Effects of the *HMS* in the clustering quality where *K* is the number of target clusters.

Finally, Fig. 4 shows the evolution of solution quality over generations for different PARs. In final generations, which algorithm converged to the optimal solution vector, large PAR values usually cause the improvement of best solutions. As seen in the standard scenario and scenario 13 that have large PAR, the best result obtained by the algorithm is better than those obtained with scenario 11 which have small PAR. Although the standard scenario and scenario 13 produce the same results, the standard scenario is preferable due to smoother convergence.

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

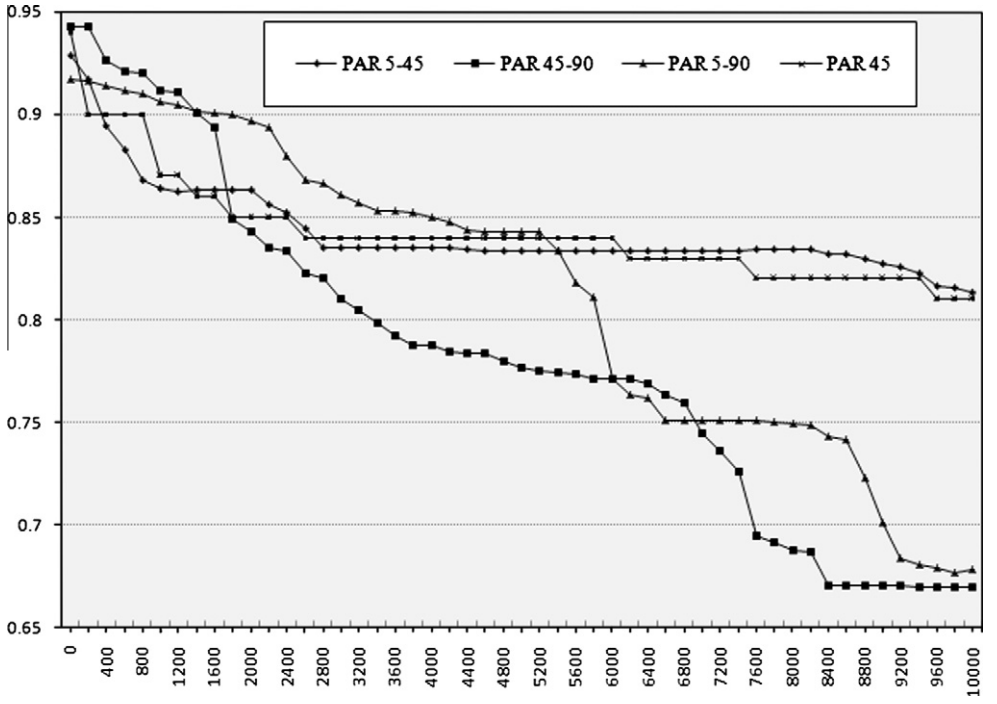

Fig. 4. Effects of the *PAR* in the clustering quality.

6.4. Performance measures for clustering

Certain measures are required to evaluate the performance of different clustering algorithms. The performance of a clustering algorithm can be analyzed with *external*, *internal*, or *relative* measures [27]. External measures use statistical tests in order to quantify how well a clustering matches the underlying structure of the data. An external quality measure evaluates the clustering by comparing the groups produced by clustering technique to the known ground-truth clusters. The most important external measures are Entropy [54], F-measure [5] and Purity [42] which are used to measure the quality of the produced clusters of different algorithms. In absence of an external judgment, internal clustering quality measures must be used to quantify the validity of a clustering. Relative measures can be derived from internal measures by evaluating different clusterings and comparing their scores. However, if one clustering algorithm performs better than the others on many of these measures, then we can have some confidence that it is the best clustering algorithm for the situation being evaluated.

The F-measure tries to capture how well the groups of the investigated partition are at the best match of the groups of the reference partition. In other words, the F-measure quantifies how well a clustering matches a reference partitioning of the same data; it is hence an external validity measure. The F-measure combines the precision and recall ideas from information retrieval [31] and evaluates whether the clustering can remove the noisy pages and generates clusters with high quality and constitutes a well-accepted and commonly used quality measure for automatically generated document clustering. Precision (P) and recall (R) are common measures used in information retrieval for evaluation. The precision, $\mathsf{P}(i,j)$, is the fraction of the documents in the cluster i that are also in the class j. Whereas the recall, $\mathsf{R}(i,j)$, is the fraction of the documents in the class j that are in the cluster i. Precision and recall are defined as follows:

$$\mathsf{P}(i,j) = \frac{n_{ij}}{n_i} \text{ and } \mathsf{R}(i,j) = \frac{n_{ij}}{n_j}, \tag{33}$$

where n_{ij} is the number of members of class j in cluster i (the number of the overlapping member), n_i is the number of members of cluster i and n_j is the number of members in class j. $\mathsf{P}(i,j)$ and $\mathsf{R}(i,j)$ take values between 0 and 1 and, $\mathsf{P}(i,j)$ intuitively measures the accuracy with which cluster i reproduces class j, while $\mathsf{R}(i,j)$ measures the completeness with which i reproduces class j. The F-measure for a cluster i and class j combines precision and recall with equal weight on each as follows:

$$\mathsf{F}(i,j) = \frac{2\mathsf{P}(i,j)\mathsf{R}(i,j)}{\mathsf{P}(i,j) + \mathsf{R}(i,j)}. \tag{34}$$

The F-measure of the whole clustering is:

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

$$F = \sum_j \frac{n_j}{n} \max_i F(i,j). \tag{35}$$

The F-measure tries to capture how well the groups of the investigated partition best match the groups of the reference. A perfect clustering exactly matches the given partitioning and leads to an F-measure value of 1.

The second evaluation measure is Entropy measure, which analyzes the distribution of categories in each cluster. The entropy measure looks at how various classes of documents are distributed within each cluster. First, the class distribution is calculated for each cluster and then this class distribution is used to calculate the entropy for each cluster. The entropy of a cluster c_i, $E(c_i)$ is defined as:

$$E(c_i) = -\sum_j p_{ij} \log p_{ij}, \tag{36}$$

where p_{ij} is the probability that a member of cluster j belongs to class i and then the summation of all classes is taken. After the entropy is calculated, the summation of entropy for each cluster is calculated using the size of each cluster as weight. In other words, the entropy of all produced clusters is calculated as the sum of the individual cluster entropies weighted according to the cluster size, i.e.,

$$E = -\sum_{i=1}^{K} \frac{n_i}{n} E(c_i), \tag{37}$$

where n_i is the size of cluster i, n is the total number of documents, and K is the number of clusters. The best clustering solution will be the one that leads to clusters which contain documents from only a single class, in this case the entropy will be zero. As the entropy measures the amount of disorder in a system, the smaller the entropy, the better the clustering solution [55].

The Purity measure evaluates the degree to which each cluster contains documents from primarily one class. In other words, it measures the largest class for each cluster. Purity tries to capture on average how well the groups match the reference. In general, the larger the value of purity, the better the clustering solution. Note that each cluster may contain documents from different classes [1]. The purity gives the ratio of the dominant class size in the cluster to the cluster size itself. The value of the purity is always in the interval $\left[\frac{1}{K^+}, 1\right]$. A large purity value implies that the cluster is a pure subset of the dominant class. The purity of each cluster c_i is calculated as:

$$P(c_i) = \frac{1}{n_i} \max_j n_{ij}. \tag{38}$$

The purity of all produced clusters is computed as a weighted sum of the individual cluster purities and is defined as:

$$P = \sum_{i=1}^{K} \frac{n_i}{n} P(c_i). \tag{39}$$

While entropy and the precision measures compare flat partitions (which may be a single level of a hierarchy) with another flat partition, the F-measure compares an entire hierarchy with a flat partition.

6.5. Results and discussions

6.5.1. Quality of clustering

In this part of experiments we compare the proposed algorithms according to their quality of generated clusters with *K*-mean and a GA based clustering algorithms [4]. For evaluation of the clustering results quality, we use four metrics, namely F-measure, Purity, Entropy, and ADDC where the first three measures have been chosen from external quality measures and ADDD has been selected from internal measures. F-measure, Purity, and Entropy expresses the clustering results from an external expert view, while ADDC examines how much the clustering satisfies the optimization constraints.

6.5.1.1. ADDC. Table 3 demonstrates the normalized ADDC of algorithms for cosine and Euclidian similarity measures applied to the mentioned document sets and Fig. 5 shows the ADDC values for cosine similarity for five datasets. The smaller the ADDC value, the more compact the clustering solution is. Looking at the Fig. 5, we can see that the results obtained by Hybrid HS + *K*-means algorithm are comparable to those obtained by *K*-means. It is clear from Table 3 that for datasets with low dimension both similarity measures have comparable result but for datasets with high dimension Euclidean measure seems to be a good choice.

6.5.1.2. F-measure. In order to make a better evaluation of clustering, as a primary measure of quality, we used the widely adopted F-measure [5]; the harmonic means of *precision* and *recall* from information retrieval. We treat each cluster as if it were the result of a query and each class as if it is the desired set of documents for a query. We then calculate the recall and precision of that cluster for each given class. For a given cluster of documents C, to evaluate the quality of C with respect to an ideal cluster C^*(categorization by human) we first compute precision and recall as usual:

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

Table 3
Performance comparison of algorithms considering ADDC values of generated clusters.

		ADDC				
		K-means	HSCLUST	Sequential hybridization	Interleaved hybridization	Hybrid HS + *K*-means
Politis	Euclidian	0.73254	0.6524	0.6241	0.5562	0.5074
	Cosine	0.76909	0.6732	0.6510	0.5815	0.5109
Newspaper	Euclidian	0.6815	0.3682	0.6152	0.5145	0.4890
	Cosine	0.7263	0.6723	0.6210	0.5745	0.4924
DMOZ	Euclidian	0.4587	0.3952	0.3315	0.3210	0.3154
	Cosine	0.4821	0.4092	0.3619	0.3243	0.3140
Message	Euclidian	0.8325	0.7612	0.7842	0.7785	0.7425
	Cosine	0.9206	0.8843	0.8315	0.7910	0.7543
Wap	Euclidian	0.8652	0.8147	0.7852	0.7421	0.7415
	Cosine	0.8753	0.8200	0.8101	0.7840	0.7615

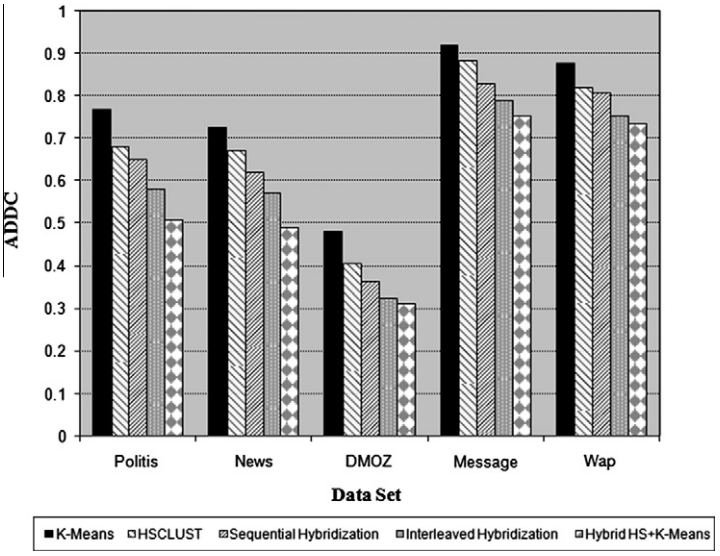

Fig. 5. Comparison of the ADDC for algorithms.

$$P(C, C^*) = \frac{|C \cap C^*|}{|C|} \text{ and } R(C, C^*) = \frac{|C \cap C^*|}{|C^*|}. \tag{40}$$

Then we define:

$$F(C, C^*) = \frac{2 * P * R}{P + R}. \tag{41}$$

The performances of the algorithms in the document collections considering F-measure are shown in Fig. 6. In comparison, the results for different algorithms, it is seen that Hybrid HS + *K*-means has the best F-measure among the other algorithms from Fig. 6. This issue is due to the high quality of produced clusters by this algorithm. HSCLUST outperforms *K*-means algorithm in all of datasets and the lowest value between all algorithms is for *K*-means. The reason is that it converges to the nearest local maximum having the values of *K* centroids. As can be noticed, the accuracy obtained using our proposed algorithm is in all the datasets comparable with that obtained from the other investigated methods in all data sets. Another important point in Fig. 6 is that Hybrid HS + *K*-means outperforms the other two hybrid algorithms. HS + *K*-means efficiently utilizes the HSCLUST and *K*-means strong points in each generation whereas other hybrid algorithms apply *K*-means after a large number of generations of HSCLUST. In other words, in HS + *K*-means, *K*-means fine-tunes the result of HSCLUST at each generation. The number of times that fine-tuning process is applied will have an effect on both the execution time and quality of clustering. More fine-tuning increases the execution runtime and accuracy and this is the reason why HS + *K*-means has larger execution time but the best quality compared to the other algorithms. As it is clear form Table 4 HSCLUST outperforms the GA clustering algorithm in all datasets.

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

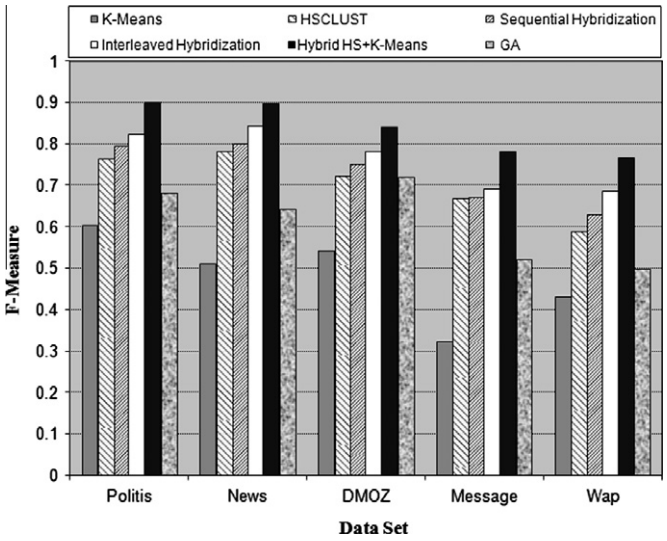

Fig. 6. Comparison of the F-measure for algorithms.

6.5.1.3. Entropy. The second external quality measure which is used to compare the quality of the proposed clustering algorithms is Entropy. The best clustering solution will be the one that leads to clusters that contain documents from only a single class, in which case the entropy will be zero. In general, the smaller the entropy values, the better the clustering solution is. Fig. 7 presents the Entropy of clusters obtained by applying algorithms on different datasets. The most important observation from the experimental results is that Hybrid HS + *K*-means performed better than other algorithms. Although the *K*-means is examined with various random initializations in different runs, it has the worst quality based on Entropy. High Entropy for clusters obtained by *K*-means reveals that documents in generated clusters belong to different classes of documents. It should be mentioned here that the results, shown in Fig. 7, are specifically for the mentioned datasets and it is possible that results would change slightly for other datasets.This is a major drawback of all document clustering algorithms where they are very sensitive to datasets and their performance varies from dataset to dataset. The results of Fig. 7 also demonstrate that the quality of hybrid algorithms lies between *K*-means and Hybrid HS + *K*-means. The Entropy of clusters obtained by HSCLUST is significantly better than *K*-means and it is due to its explorative power. HSCLUST avoids premature convergence in the successive generations and its stochastic behavior ensures that most of the regions in the search space have been explored.

6.5.1.4. Purity. Our last measure to evaluate the algorithms is Purity. Purity measures to what extend each cluster contained documents from a single class. In other words, it measures the largest class for each cluster. In general, the larger the value of purity, the better the clustering solution is. Fig. 8 summarizes the Purity of clusters for different datasets applying proposed algorithms. It is noticeable from Table 4 that GA algorithm outperforms the HSCLUST algorithm for dataset Message but in other datasets HSCLUST has better Purity than GA.

6.5.2. Comparison of the time performance

Next we compare the execution time of the clusters that are created using the six algorithms with different documents. Fig. 9 shows the average execution time of all algorithms using the dataset DMOZ. The evaluations were conducted for the document numbers varying from 500 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 all tests. In Fig. 9, it can be seen that the HSCLUST and GA algorithms yield competitive execution times. In general, the execution time of HSCLUST and GA are approximately the same especially when the number of documents is less than 6000. According to Fig. 9, by increasing the number of documents, the execution time of GA becomes slightly better than HSCLUST, but as it is clear in Fig. 9 the average performance of HSCLUST algorithm in comparison with other algorithms differs tremendously. Also, it is evident from Fig. 9 that the *K*-means algorithm has the worst runtime and also the running time of *K*-means increases linearly as the number of documents increases. The reason for *K*-means algorithm running slower than HSCLUST + *K*-means is that it may get stuck in local optimum solution and should be restarted several times. Therefore, multi-run of *K*-means is slower than the other algorithms.

6.5.3. Convergence analysis

We present experiments in this section to demonstrate the effectiveness of the different algorithms. The criterion for evaluating algorithms is their convergence rate to optimal solution. Fig. 10 illustrates the convergence behaviors of HSCLUST

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

Table 4
Performance comparison of six clustering algorithms.

	K-means			HSCLUST			Sequential hybrid			Interleaved hybrid			Hybrid HS + *K*-means			GA		
	Purity	F-measure	Entropy	Purity	F-measure	Entropy	Purity	F-measure	Entropy	Purity	F-measure	Entropy	Purity	F-measure	Entropy	Purity	F-measure	Entropy
News	0.5823	0.5117	0.7313	0.7278	0.7824	0.5195	0.7312	0.8012	0.4914	0.7578	0.8435	0.4508	0.7611	0.8968	0.4401	0.7023	0.6427	0.6352
Improvement				24.98%	52.90%	28.96%	25.57%	56.57%	32.80%	30.13%	64.84%	38.35%	30.70%	75.25%	39.81%	20.60%	25.60%	13.14%
Politics	0.6305	0.6035	0.5722	0.7635	0.7655	0.4312	0.7976	0.7952	0.3878	0.80	0.8216	0.3625	0.8123	0.8994	0.2444	0.7805	0.6822	0.4233
Improvement				21.09%	26.84%	24.64%	26.50%	31.16%	32.22%	26.88%	36.13%	36.64%	28.83%	49.03%	57.28%	23.19%	13.04%	26.02%
Wap	0.4745	0.4302	0.7254	0.5041	0.58963	0.6093	0.5231	0.63125	0.5713	0.5456	0.6856	0.5355	0.5647	0.7662	0.4701	0.4945	0.4982	0.6281
Improvement				6.23%	37.05%	16.00%	10.24%	46.73	21.24	14.98	59.36%	26.17%	19.00%	18.10%	35.19%	4.21%	15.80%	13.41%
	0.5334	0.3236	0.8057	0.6270	0.6692	0.6943	0.6570	0.67231	0.5910	0.6612	0.6912	0.5590	0.6712	0.7805	0.5083	0.6402	0.5213	0.7142
Message improvement				17.54%	106.19	13.82	23.17%	101.75%	26.64%	23.95%	113.59%	30.61%	25.83%	141.19%	36.912%	20.02%	61.09%	11.35%
DMOZ	0.5556	0.5423	0.8334	0.6519	0.72445	0.7305	0.6832	0.75123	0.4138	0.7078	0.7816	0.3726	0.73283	0.83977	0.3312	0.6433	0.7194	0.7400
Improvement				17.33%	33.58%	12.34%	22.96%	38.52%	50.34%	27.39%	44.12%	55.29%	37.89%	54.85%	60.25%	15.78%	32.65%	11.20%

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

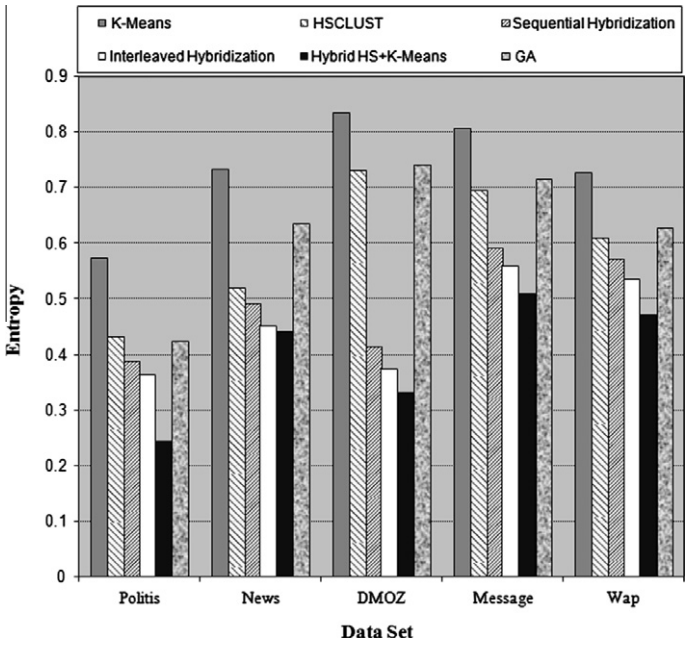

Fig. 7. Comparison of the Entropy for algorithms.

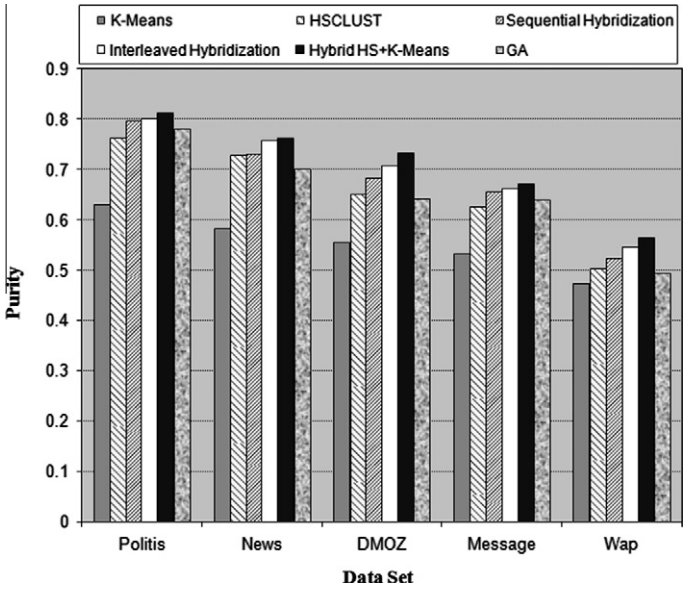

Fig. 8. Comparison of the Purity for algorithms.

and *K*-means algorithms on the document dataset DMOZ. For each data set we have conducted 20 independent trials with randomly generated initialization and the average value is recorded to account for the stochastic nature of the algorithm. It is obvious from Fig. 10 that HSCLUST took more time to reach the optimal solution and *K*-means converges more quickly. This is because the *K*-means algorithm may be trapped in local optima. Although the *K*-means algorithm is more efficient than HSCLUST with respect to execution time, the HSCLUST generates much better clustering than the *K*-means algorithm. In Fig. 11 performance of HSCLUST and hybridized algorithms are compared using document dataset DMOZ. Fig. 10 illustrates that the reduction of ADDC value in HSCLUST follows a smooth curve from its initial vectors to final optimum solution with no sharp moves. Another noteworthy point in Fig. 11 is that ADDC has the lowest final value for hybrid HS + *K*-means among the other algorithms. The sequence of other algorithms with respect to their ADDC values are: Interleaved Hybridization, Sequential Hybridization and HSCLUST. This issue shows that the cluster produced by Hybrid HS + *K*-means has best quality

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

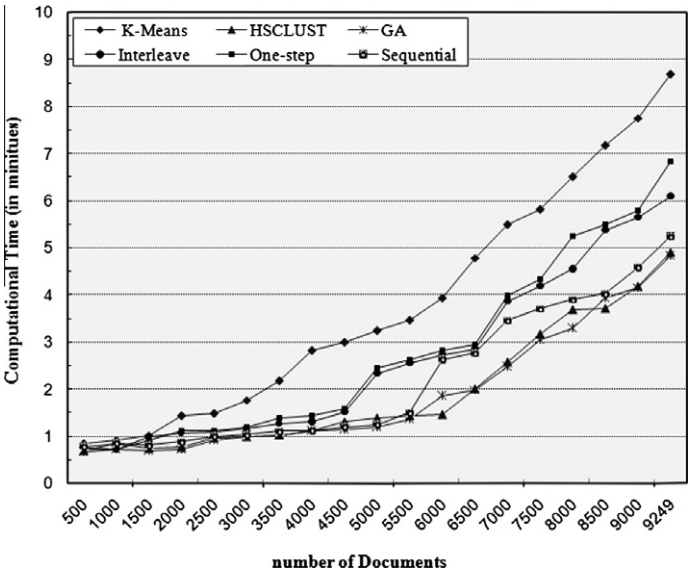

Fig. 9. Comparison of the execution time of the algorithms.

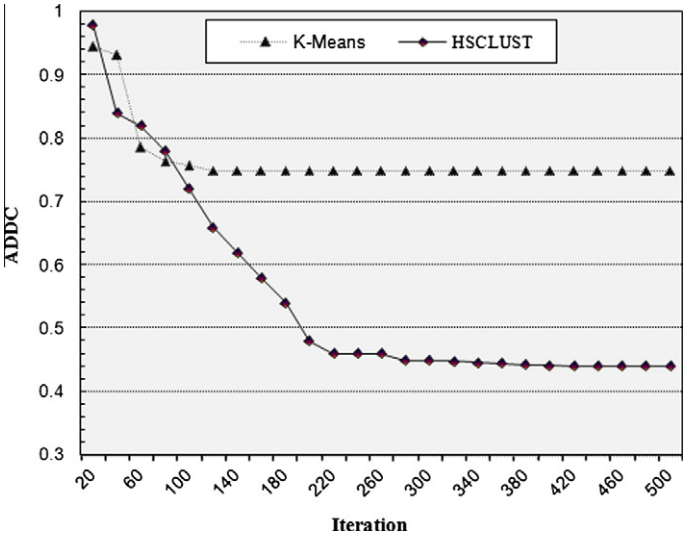

Fig. 10. The convergence behaviors of HSCLUST and *K*-means algorithms on document set DMOZ.

and results produced by hybrid algorithms have higher quality than HSCLUST. It can be inferred from Fig. 11 that hybrid algorithms overcome HSCLUST disadvantage by incorporating 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, the obtained result is passed as the initial vector to *K*-means algorithm and then, *K*-means fine tunes that. The results show that the hybrid approaches outperform the component algorithms (*K*-means and harmony search) in terms of cluster quality.

7. Conclusion

In this paper we have considered the problem of finding a near optimal partition, optimum with respect to ADDC criterion, of a given set of documents into a specified number of clusters. We have proposed four algorithms for this problem by modeling the partitioning problem as an optimization problem. First an algorithm based on HS, namely HSCLUST, has been developed aimed at optimizing the objective function associated with optimal clustering. Then the harmony search based algorithm has been extended using the *K*-means algorithm to devise three different hybrid methods. Furthermore, we have theoretically analyzed the behavior of the proposed algorithm by modeling its population variance as a Markov chain. We

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025

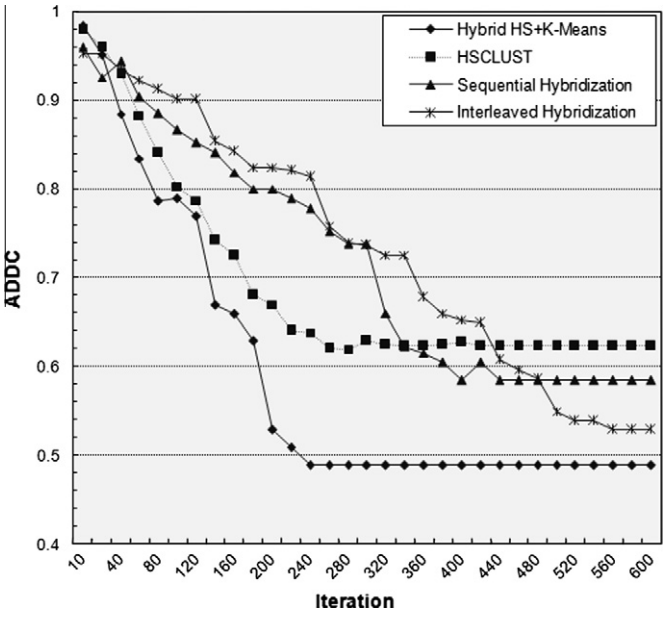

Fig. 11. Performance of hybridization algorithms on document set DMOZ.

have validated our theoretical results by conducting experiments on five datasets with varying characteristics to evaluate the performance of the hybrid algorithms compared with the other algorithms. Results shows that the proposed hybrid algorithms are better than *K*-means, GA based clustering, and HSCLUST based on the resulting clusters at the cost of increased time complexity.

Acknowledgments

The authors thank anonymous reviewers and editor in chief for their constructive comments which led to the overall improvement of this paper. Also, the authors are indebted to Belmond Yoberd and Nicholas Slocum for their literary contributions.

References

[1] R.M. Aliguliyev, Performance evaluation of density-based clustering methods, Information Sciences 179 (20) (2009) 3583–3602.
[2] M.R. Anderberg, Cluster Analysis for Applications, Academic Press, Inc., New York, NY, 1973.
[3] J. Aslam, K. Pelekhov, D. Rus, Using star clusters for filtering, in: Proceedings of the Ninth International Conference on Information and Knowledge Management, USA, 2000.
[4] S. Bandyopadhyay, U. Maulik, An evolutionary technique based on *K*-means algorithm for optimal clustering, Information Sciences 146 (2002) 221–237.
[5] A. Banerjee, C. Krumpelman, S. Basu, R. Mooney, J. Ghosh, Model based overlapping clustering, in: Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD), 2005.
[6] D. Boley, M. Gini, R. Gross, et al, Document categorization and query generation on the World Wide Web using WebACE, Artificial Intelligence Review 13 (5/6) (1999) 365–391.
[7] P.S. Bradley, U.M. Fayyad, C.A. Reina. Scaling EM (Expectation Maximization) clustering to large databases, Microsoft Research Technical Report, November 1998.
[8] M.C. Chiang, C.W. Tsai, C.S. Yang, A time-efficient pattern reduction algorithm for *K*-means clustering, Information Sciences 181 (4) (2011) 716–731.
[9] K. Cios, W. Pedrycz, R. Swiniarski, Data Mining-Methods for Knowledge Discovery, Kluwer Academic Publishers.
[10] R.M. Cole, Clustering with Genetic Algorithms, Masters thesis, Department of Computer Science, University of Western Australia, Australia, 1998.
[11] X. Cui, T.E. Potok, P. Palathingal, Document clustering using particle swarm optimization, in: Proceedings of the IEEE swarm intelligence symposium, SIS 2005, Springer, Piscataway, IEEE Press, pp. 185191.
[12] X. Cui, T.E. Potok, Document clustering analysis based on hybrid PSO+*K*-means algorithm, Journal of Computer Sciences (2005) 27–33.
[13] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, B.K. Panigrahi, Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 41 (1) (2011) 89–106.
[14] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering 186 (2000) 311–338.
[15] J. D'hondt, J. Vertommen, P.A. Verhaegen, D. Cattrysse, J.R. Duflou, Pairwise-adaptive dissimilarity measure for document clustering, Information Sciences 180 (2) (2010) 2341–2358.
[16] B. Everitt, Cluster analysis, second ed., Halsted Press, New York, 1980.
[17] Z.W. Geem, C. Tseng, Y. Park, Harmony search for generalized orienteering problem: best touring in China, in: Lecture Notes in Computer Science, 3612, Advances in Natural Computation, 2005, pp. 741–750.
[18] Z.W. Geem, Novel derivative of harmony search algorithm for discrete design variables, Applied Mathematics and Computation 199 (1) (2008) 223–230.
[19] Z.W. Geem, Music-Inspired Harmony Search Algorithms: Theory and Applications, Springer-Verlag, Germany, 2009.

[20] Z.W. Geem, K-B Sim, Parameter-setting-free harmony search algorithm, Applied Mathematics and Computation 217 (8) (2010) 3881–3889.
[21] 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.
[22] S. Guha, R. Rastogi, K. Shim, CURE: An efficient clustering algorithm for large databases, in: Proc. of ACM-SIGMOD Int. Conf. Management of Data (SIGMOD98), 1998, pp. 73–84.
[23] J. Han, M. Kamber, A.K.H. Tung, Spatial clustering methods in data mining: a survey, in: Geographic Data Mining and Knowledge Discovery, New York, 2001.
[24] J. Handl, J. Knowles, An evolutionary approach to multiobjective clustering, IEEE Transactions on Evolutionary Computation 11 (1) (2007) 56–76.
[25] D. Hsek, J. Pokorn, H. Rezankov, V. Snsel, Web data clustering, in: Studies in Computational Intelligence, Springer, 2009, pp. 325–353.
[26] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Computing Surveys (CSUR) 31 (3) (1999) 264–323.
[27] A.K. Jain, C.D. Richard, Algorithm for Clustering in Data, Prentice Hall, Englewood Cliffs, NJ, 1990. ISBN:0-13-022278-X.
[28] G. Karypis, E.H. Han, V. Kumar, CHAMELEON: a hierarchical clustering algorithm using dynamic modeling, Computer 32 (1999) 68–75.
[29] J. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence, Morgan Kaufmann, New York, 2001.
[30] K. Krishna, M. Narasimha Murty, Genetic *K*-means algorithm, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 29 (3) (1999) 433–439.
[31] B. Larsen, C. Aone, Fast and effective text mining using linear-time document clustering, in: Proceedings of the KDD-99 Workshop, San Diego, USA, 1999.
[32] K.S. Lee, Z.W. Geem, A new meta-heuristic algorithm for continues engineering optimization: harmony search theory and practice, Computer Methods in Applied Mechanics and Engineering 194 (2004) 3902–3933.
[33] Y. Li, Z. Xu, An ant colony optimization heuristic for solving maximum independent set problems, in: Proceedings of the 5th International Conference on Computational Intelligence and Multimedia Applications, Xi'an, China, 2003, pp. 206–211.
[34] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistic and Probability, University of California Press, Berkley, CA, 1967, pp. 281–297.
[35] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Applied Mathematics and Computation 188 (2) (2007) 1567–1579.
[36] M. Mahdavi, H. Abolhassani, Harmony *K*-means Algorithm for Document Clustering, 18 (3), Springer, 2009, pp. 370–391.
[37] J. Moore, E. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, Web page categorization and feature selection using association rule and principal component clustering, in: 7th Workshop on Information Technologies and Systems, 1997.
[38] C.F. Olson, Parallel algorithms for hierarchical clustering, Parallel Computing 21 (1995) 1313–1325.
[39] A. Papoulis, S. Unnikrishna, Probability, Random Variables and Stochastic Process, fourth ed., McGraw-Hill, 2002.
[40] V.J. Rayward-Smith, Metaheuristics for clustering in KDD, Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 3, IEEE Press, Piscataway, 2005, pp. 2380–2387.
[41] V. Rijsbergen, Information Retrieval, second ed., Buttersworth, London, 1979.
[42] A.M. Rubinov, N.V. Soukhorokova, J. Ugon, Classes and clusters in data analysis, European Journal of Operational Research 173 (2006) 849–865.
[43] T.A. Runkler, Ant colony optimization of clustering models, International Journal of Intelligent Systems 20 (12) (2005) 1233–1251.
[44] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Information Processing and Management: An International Journal 24 (5) (1988) 513–523.
[45] G. Salton, Automatic Text Processing, Addison-Wesley, 1989.
[46] S. Saatchi, C.C. Hung, Hybridization of the ant colony optimization with the *K*-means algorithm for clustering, in: Lecture Notes in Computer Science, Image Analysis, vol. 3540, Springer, Berlin, 2005.
[47] S.Z. Selim, M.A. Ismail, *K*-means-type algorithms: a generalized convergence theorem and characterization of local optimality, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6 (1984) 81–87.
[48] W. Song, C. Hua Li, S. Cheol Park, Genetic algorithm for text clustering using ontology and evaluating the validity of various semantic similarity measures, Expert Systems with Applications 36 (2009) 9095–9104.
[49] M. Steinbach, G. Karypis, V. Kumar, A Comparison of Document Clustering Techniques, KDD2000, Technical report of University of Minnesota, 2000.
[50] S. Vaithyanathan, B. Dom, Model selection in unsupervised learning with applications to document clustering, in: Proceedings International Conference on Machine Learning, 1999.
[51] S. Xu, J. Zhang, A parallel hybrid web document clustering algorithm and its performance study, The Journal of Supercomputing 30 (2004) 117–131.
[52] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large databases, in: Proc. of ACM-SIGMOD Int. Conf. Management of Data (SIG-MOD96), 1996, pp. 103–114.
[53] L. Zhang, Q. Cao, A novel ant-based clustering algorithm using the kernel method, Information Sciences (in press) http://dx.doi.org/10.1016/j.ins.2010.11.005 (corrected proof).
[54] Y. Zhao, G. Karypis, Criterion Functions for Document Clustering Experiments and Analysis, Technical Report #01-40, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, 2001.
[55] Y. Zhao, G. Karypis, Empirical and theoretical comparisons of selected criterion functions for document clustering, Machine Learning 55 (3) (2004) 311–331.

Please cite this article in press as: R. Forsati et al., Efficient stochastic algorithms for document clustering, Inform. Sci. (2012), http://dx.doi.org/10.1016/j.ins.2012.07.025