A Robust Iterative Refinement Clustering Algorithm with Smoothing Search Space

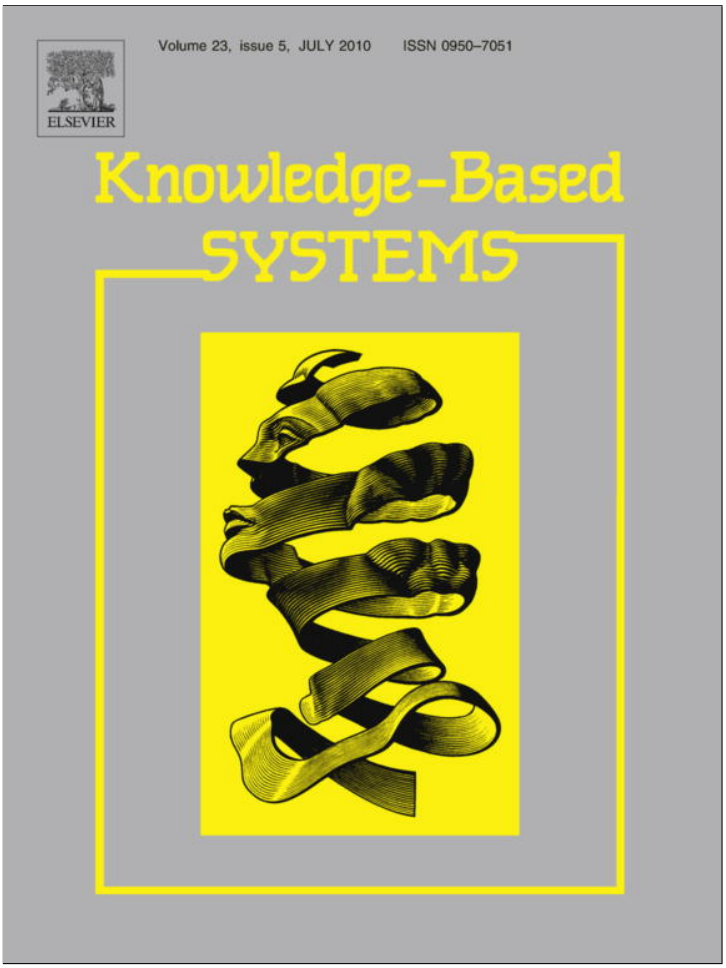
Zong, Yu; Xu, Guandong; Zhang, Yanchun; Jiang, He; Li, Mingchu

Published in:
Knowledge-Based Systems

DOI (link to publication from Publisher):
10.1016/j.knosys.2010.01.012

Publication date:
2010

Document Version
Early version, also known as pre-print



Knowledge-Based Systems 23 (2010) 389–396

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A robust iterative refinement clustering algorithm with smoothing search space

Yu Zong [a,b,*], Guandong Xu [b], Yanchun Zhang [b], He Jiang [a], Mingchu Li [a]

[a] School of Software, Dalian University of Technology, Dalian 116621, China
[b] School of Science and Engineering, Center for Applied Informatics, Victoria University, Melbourne VIC8001, Australia

ARTICLE INFO

Article history:
Received 26 March 2009
Received in revised form 12 January 2010
Accepted 27 January 2010
Available online 4 February 2010

Keywords:
Clustering
Smoothing search space
Kernel smoothing
Heuristic algorithm

ABSTRACT

Iterative refinement clustering algorithms are widely used in data mining area, but they are sensitive to the initialization. In the past decades, many modified initialization methods have been proposed to reduce the influence of initialization sensitivity problem. The essence of iterative refinement clustering algorithms is the local search method. The big numbers of the local minimum points which are embedded in the search space make the local search problem hard and sensitive to the initialization. The smaller number of local minimum points, the more robust of initialization for a local search algorithm is. In this paper, we propose a Top–Down Clustering algorithm with Smoothing Search Space (TDCS3) to reduce the influence of initialization. The main steps of TDCS3 are to: (1) dynamically reconstruct a series of smoothed search spaces into a hierarchical structure by 'filling' the local minimum points; (2) at the top level of the hierarchical structure, an existing iterative refinement clustering algorithm is run with random initialization to generate the clustering result; (3) eventually from the second level to the bottom level of the hierarchical structure, the same clustering algorithm is run with the initialization derived from the previous clustering result. Experiment results on 3 synthetic and 10 real world data sets have shown that TDCS3 has significant effects on finding better, robust clustering result and reducing the impact of initialization.

1. Introduction

Clustering is a useful approach in data mining processes for identifying patterns and revealing underlying knowledge from large data collections. The application areas of clustering include image segmentation, information retrieval, and document classification, associate rule mining, web usage tracking and transaction analysis. Generally, clustering is defined as the process of partitioning unlabelled data set into meaningful groups (clusters) so that intra-group similarities are maximized and inter-group similarities are minimized at the same time.

In essence, clustering involves the following unsupervised learning process, which can be written as:

Define an 'encoder' function $c(x)$ to map each data object x_i into a particular group $G_k(c(x) = k \Rightarrow x \in G_k\ k = 1,\ldots,K)$, so that a cluster criterion $Q(c) = \sum_{k=1}^{K}\sum_{c(x_i)=k,c(x_j)=k} dist(x_i, x_j)$ is minimized.

As we know, this is a classical combinatorial optimization problem and solving it is exactly NP-hard, even with just two clusters [6]. According to computation complexity theory [22], no complete algorithm can get the overall optimal solutions in a polynomial time, unless $P = \mathrm{NP}$. Iterative refinement method, a popular approximate algorithm, is widely adopted by various unsupervised learning algorithms. A general iterative refinement clustering process can be summarized as Algorithm 1 [19].

Algorithm 1. General iterative refinement clustering

Initialization: Initialize the parameters of the current cluster model.
Refinement: Repeat until the cluster model converges.
(1) Generate the cluster membership assignments for all data objects, based on the current model;
(2) Refine the model parameters based on the current cluster membership assignments.

The intuitionistic denotation of iterative refinement clustering algorithm is shown in Fig. 1. The horizontal axis denotes feasible solutions of clustering problem and the vertical axis is the corresponding objective function values of feasible solutions. In this paper, the feasible solution is the results of 'encode' function (or the clustering results) and the objective function value is the values of cluster criterion $Q(c) = \sum_{k=1}^{K}\sum_{c(x_i)=k,c(x_j)=k} dist(x_i, x_j)$. Without loss of generality, we assume that point 3 is selected as the initialization of an iterative refinement clustering algorithm, and by repeating step (1) and (2), the algorithm will converge to point 4, one of the feasible solutions with sub-optimal objective function value.

* Corresponding author. Address: Center for Applied Informatics, Victoria University, Melbourne VIC8001, Australia. Tel.: +61 3 9919 9750; fax: +61 3 9919 5060.
E-mail address: Yu.Zong@vu.edu.au (Y. Zong).

0950-7051/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2010.01.012

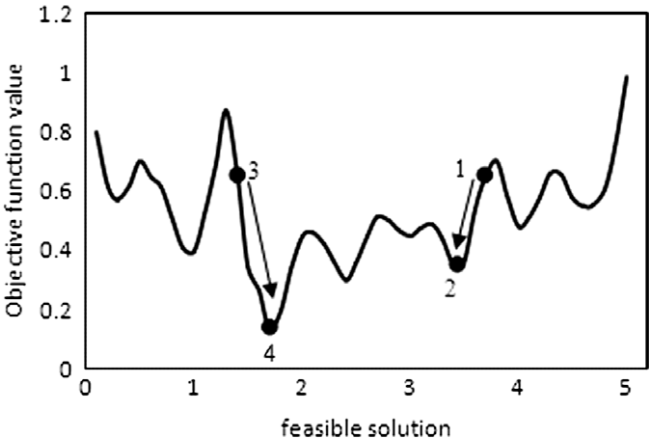


Fig. 1. An example of iterative refinement clustering algorithm.

If point 1 is chosen as the initialization of the same clustering algorithm, it will lead the algorithm converges to point 2, a worse solution with a higher cluster criterion value.

That the initialization model must be correct is an important underlying assumption for iterative refinement clustering algorithm. It can determine the clustering solution [19], that is, different initialization models will produce different clustering results (or different local minimum points as shown in Fig. 1). Since the problem of obtaining a globally optimal initial state has been shown to be NP-hard [9], the study on the initialization methods towards a sub-optimal clustering result hence is more practical, and is of great value for the clustering research. Recently, initialization methods have been categorized into three major families: random sampling methods, distance optimization methods and density estimations [11]. Forgy adopted uniformly random input objects as the seed clusters [8], and MacQueen gave an equivalent way with selecting the first K input objects as the seed clusters [17]. In the FASTCLUS, a K-means variance implemented in SAS [21], the simple cluster seeking (SCS) initialization method is adopted [23]. Katsavounidis et al. proposed a method that utilizes the sorted pairwise distances for initialization [15]. Kaufman and Rousseeuw introduced a method that estimates the density through pairwise distance comparison, and initializes the seed clusters using the input objects from areas with high local density [14]. In Ref. [7], a method which combines local density approximation and random initialization is proposed. Belal et al. find a set of medians extracted from a dimension with maximum and then use the medians as the initialization of K-means [3]. Niu et al. give a novel algorithm called PR (Pointer Ring), which initializes cluster centers based on pointer ring by partition traditional hyper-rectangular units further to hyper-triangle subspaces [18]. The initialization steps of K-means++ algorithm can be described as: choosing an initial center m_1 uniformly at random from data set; and then selecting the next center $m_i = x'$ from data set with probability $dist(x\prime, m)^2 / \sum_{x \in D} dist(x, m)$, where $dist(x, m)$ denote the shortest distance from a data object x to the closest center m; iterative until find K centers [1]. The main steps of initialization centers of K-means by kd-tree are: first, the density of a data at various locations are estimated by using kd-tree; and then use a modification of Katsavounidis' algorithm, which incorporates this density information, to choose K seeds for K-means algorithm [20]. And recently, Lu et al. treat the clustering problem as a weighted clustering problem so as to find a better initial cluster center based on the hierarchical approach [16].

The goal of these modified initialization methods, is to reduce the influence of sub-optimal solutions (the local minimum points) bestrewed in the whole search space, as shown in Fig. 1. Although iterative refinement clustering algorithms with these modified initialization methods have some merits in improving the quality of cluster results, they are also have high probability to be attracted by local minimum points. Local search method is the essence of iterative refinement clustering algorithms. Lots of the local minimum points make a local search problem hard and sensitive to the initialization. Those proposed modified initialization methods are only focused on how to select an initialization which can improve the quality of iterative refinement clustering algorithm, but the search space embedded lots of local minimum points is ignored.

Smoothing search space method reconstructs the search space by filling local minimum points, to reduce the influence of local minimum points. In this paper, we first design two smoothing operators to reconstruct the search space by filling the minimum 'traps' (points) based on the relationship between distance metric and cluster criterion. Each smoothing operator has a parameter, smoothing factor, to control the number of minimum 'traps'. And then, we give a top–down clustering algorithm with smoothing search space (TDCS3) to reduce the influence of initialization. The main steps of TDCS3 are to: (1) dynamically reconstruct a series of smoothed search space as a hierarchical structure: the most smoothed search space at the top, and the original search space at the bottom, other smoothed search spaces are distributed between them, by 'filling' the local minimum points; (2) at the top level of the hierarchical structure, an existing iterative refinement clustering algorithm is run with random initialization to generate the cluster result; (3) from the second level to the bottom level of the hierarchical structure, the same clustering algorithm is run with the initialization derived from the cluster result on the previous level. Experiment results on 3 synthetic data sets and 10 real world data sets have shown that TDCS3 has significant effects on finding better, robust cluster result and reducing the influence of initialization.

The contributions of this paper are: (1) we discuss the question why iterative refinement clustering algorithm are sensitive to initialization; (2) we deal with the initialization sensitivity problem by smoothing the search space of iterative refinement clustering algorithms; (3) two smoothing operators are designed based on distance metric; (4) based on the smoothed search spaces, a top–down clustering algorithm is proposed to reduce the influence of initialization. More importantly the existing iterative refinement clustering algorithm can be run in TDCS3 to improve the quality of cluster results.

This paper is organized as follows: in Section 2, we first discuss the local search method and the definition of smoothing search space, and then two smoothing operators are designed based on distance metrics. In Section 3 the top–down clustering algorithm with smoothing search space is proposed, and the strength of the proposed algorithm is also discussed. Then, the experiments on 3 synthetic and 10 real world data sets are conducted and results are presented in Section 4. Finally, we conclude this paper in Section 5.

2. Smoothing search space and smoothing operator

2.1. Local search and smoothing search space

Local search method is the essence of iterative refinement clustering algorithms. During the mid-sixties, local search method was first proposed to cope with the overwhelming computational intractability of NP-hard combinatorial optimization problems. Give a minimization (or maximization) problem with objective function f and feasible region F, a typical local search algorithm requires that, with each solution $x_i \in R^d$, there is associated a

predefined neighbourhood $N(x_i) \subset R^d$. Given a current solution point $x_i \in R^d$, the set $N(x_i)$ is searched for a point x_{i+1} with $f(x_{i+1}) < f(x_i)$ (or $f(x_{i+1}) > f(x_i)$). If such a point exists, it becomes the new current solution point ($x_i \leftarrow x_{i+1}$), and then the process is iterated. Otherwise, x_i is retained as a local optimum with respect to $N(x_i)$. Then a set of feasible solution points is generated, and each of them is 'locally' improved within its neighborhood. Local search methods only check the neighbourhood of current feasible solution x_i, so the search range has been dramatically reduced and the convergence speed has been accelerated. A major shortcoming of local search is that the algorithm has a tendency to get stuck at a locally optimum configuration, i.e., a local minima point, as the point 2 or 4 shown in Fig. 1.

Different neighbourhood structures result in difference terrain surface structures of the search space and produce different numbers of local minimum points. The effectiveness of a local search algorithm relies on the number of local minimum points in the search space [10], that is, local minimum points make a search problem hard. The smaller the number of local minimum points, the more effective a local search algorithm is. In order to reduce the influence of local minimum to local search algorithm, some local minimum 'traps' must be filled. Gu and Huang [10] has called the method of filling minimum 'trap' as the smoothing search space, and it is able to dynamically reconstruct the problem structure and smooth the rugged terrain surface of the search space. The smoothed search space could 'hide' some local minimum points, therefore, improving the performance of the traditional local search algorithm. Fig. 2 is the illustration of smoothing search space.

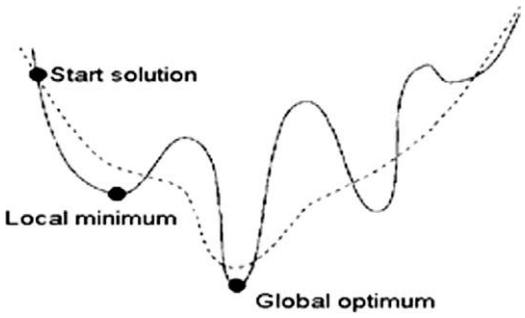


Fig. 2. The illustration of smoothing search space. Many local minimum 'traps' are filled after running a smoothing operator. The real line curve shows the original search space which has lots of minimum 'trap', and dashed shows the smoothed search space with fewer minimum 'trap'.

At the former discussing, we can find that lots of the local minimum points which are embedded in the search space make a local search problem hard and sensitive to the initialization. The essence of iterative refinement clustering algorithms is the local search method, thus they have the same real reason for initialization sensitivity problem.

The main idea of smoothing search space is always common, but different application areas have different ways to smoothing the search space. In clustering area, clustering is defined as the process of partitioning unlabelled data objects into meaningful groups (clusters) so that the value of cluster criterion $Q(c)$ is minimized. Minimizing $Q(c)$ value means that the intra-similarities of all clusters are maximized or the distances of each data object to its cluster center is minimized. So the cluster criterion $Q(c)$ has a close relationship with the similarity or distance between data objects.

2.2. Smoothing operator

In this section, we designed two smoothing operators based on the relationship between $Q(c)$ and distance measure, to fill the minimum 'traps' embedded in the rugged surface of search space.

Let $D = \{x_1, x_2, \ldots, x_N\}, x_i \in R^d$ be a set of data objects that needs to be clustered. And note *dist*: $R^d \times R^d \mapsto R_+$ be a given distance function between any two data objects inR^d. *Dist* is a distance matrix which contains the distances between all data objects of D, and $Dist(i,j)$ denotes the distance between data object x_i and x_j, $dist(x_i, x_j)$.

2.2.1. Displacement smoothing operator

Based on average distance of distance matrix *Dist*, we design the displacement smoothing operator as below.

Definition 1. Given a data set $D = \{x_1, x_2, \ldots, x_N\}$, and its distance matrix *Dist*, the average distance of *Dist* is defined as:

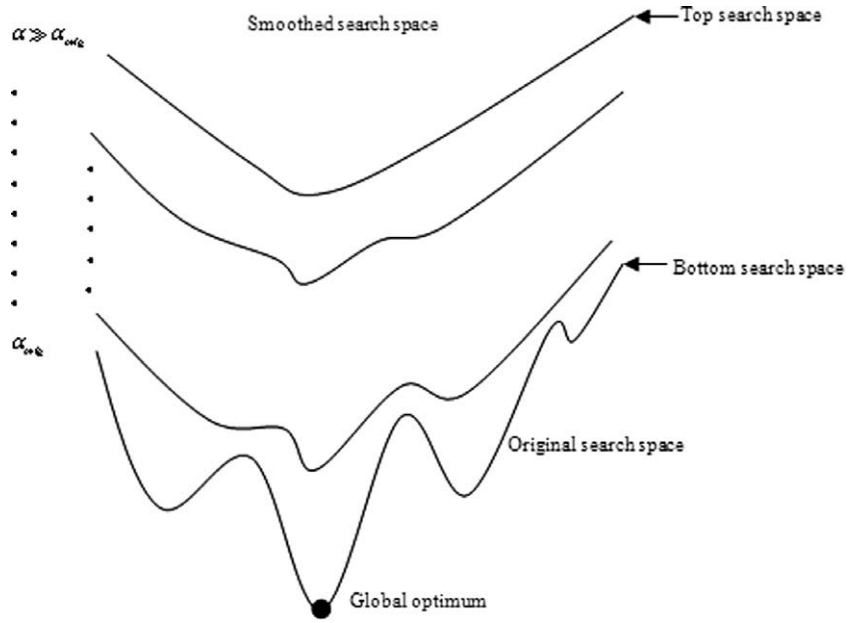


Fig. 3. The illustration of a series smoothed search spaces with different terrain surface, which are generate by displacement methods with different smoothing factor α.

$$\overline{Dist} = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=1}^{N} Dist(i,j) \quad (1)$$

Definition 2. Given a smoothing factor $\alpha \geqslant \alpha_{org}$, the displacement smoothing operator reconstructs the smoothed search space according to:

$$Dist_\alpha(i,j) = \begin{cases} \overline{Dist} + (Dist(i,j) - \overline{Dist})^\alpha & if \quad Dist(i,j) \geqslant \overline{Dist} \\ \overline{Dist} - (\overline{Dist} - Dist(i,j))^\alpha & if \quad Dist(i,j) < \overline{Dist} \end{cases} \quad (2)$$

According to Definitions 1 and 2, a series of smoothed search spaces with different numbers of minimum 'traps' will be reconstructed during $\alpha \to \alpha_{org}$. A smoothed search space generated from a large α exhibits a smoother terrain surface, and a search space generated from a smaller α exhibits a more rugged terrain surface. The search space will return to the original search space when $\alpha = \alpha_{org}$. Let's note the smoothed search space according to the largest α as the top search space and the original search space as the bottom search space, as shown in Fig. 3.

Algorithm 2 describes the details of the reconstruction process for smoothing the search spaces. In the first step, we calculate the average distance of *Dist*, and during the second step, a distance transformation is run to change each distance $Dist(i,j) \in Dist$ with average distance $\overline{Dist}$ and the difference between $dist(x_i,x_j)$ and $\overline{Dist}$. The main time cost of displacement smoothing operator is the process of the distance transformation. For a distance $Dist(i,j) \in Dist$, the time cost of distance transformation is $O(1)$. For all the distances belong to *Dist*, the total time consumed is $O(N^2)$.

Algorithm 2. Displacement smoothing operator

Input: distance matrix *Dist*, smoothing factor α
Output: smoothed search space S^α
(1) Calculate the average distance $\overline{Dist}$ of *Dist*;
(2) For any $Dist(i,j) \in Dist$
If $Dist(i,j) < \overline{Dist}$ then
$Dist_\alpha(i,j) = \overline{Dist} - (\overline{Dist} - Dist(i,j))^\alpha$;
Else
$Dist_\alpha(i,j) = \overline{Dist} + (Dist(i,j) - \overline{Dist})^\alpha$;
End if;
End for;
(3) $S^\alpha \leftarrow Dist_\alpha$ and return.

In this paper, we set $\alpha_{org} = 1$, then there are two extreme cases of the series of the clustering instances, which are based on the distance, are: (1) if $\alpha \gg \alpha_{org}$, then $Dist_\alpha(i,j) \to \overline{Dist}$, this is the trivial case; (2) if $\alpha = \alpha_{org}$, then $Dist_\alpha(i,j) = Dist(i,j)$, which is the original problem.

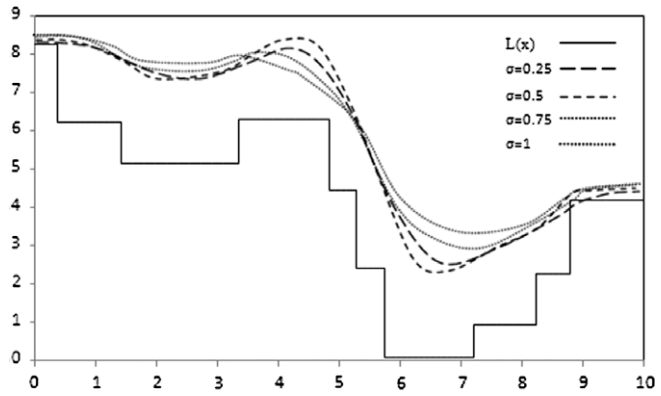


Fig. 4. The illustration of smoothing space after running kernel smoothing operator on piece-wise function. The real line curve illustrates the original search space, and all the dashed are the smoothed search space, by run a kernel smoothing operator with different smoothing factor σ on the original search space.

2.2.2. Kernel smoothing operator

The main idea of the displacement smoothing operator is the linear transformation of distance based on $\overline{Dist}$ and the exponential of the difference between $dist(x_i,x_j)$ and $\overline{Dist}$. This smoothing operator fits linear problems well, but is weak to solving non-linear problems. So another smoothing operator which could be extended to non-linear situation is designed to deal with non-linear clustering problem. This smoothing operator, named kernel smoothing operator, is based on the smoothing kernel.

Definition 3 [2]. Given a real value function $f:R^n \to R$ and a smoothing kernel $g:R \to R$, which is a continuous, bounded, non-negative, and symmetric function whose integral is one, the g-transform of f is defines as

$$\langle f \rangle_g(x) = \int_{R^n} f(y) g(\|y - x\|) dy. \quad (3)$$

The Gaussian kernel function $g(z) = \exp(-z^2/(2\sigma^2))$ is the most widely used kernel function. Fig. 4 give an example of applying a smoothing transformation to the piece-wise constant function L and we estimate the transformed function.

$$\langle L \rangle_g(x) = \int_{R^n} L(y) g(\|y - x\|) dy. \quad (4)$$

From Fig. 4, we can see the 'traps' of minimum point has been smoothed by the Gaussian kernel function with different σ, the smoothing factor.

In this paper, we use the kernel smoothing method to smooth the distance function $dist(x_i,x_j)$ and reduce the influence of lots of minimum value embedded in the search space. We assume that there is no missing value in data set D, that is, distance function *dist* is a continuous function. Let $L(x) = dist(x_i,y_j) = \sum_{l=1}^{d} \|x_{il} - y_{jl}\|^2$, the smoothing method for clustering is defined as:

$$dist(x_i,y_j)_g = dist(x_i,x_j) * \exp(-dist(x_i,x_j)^2/(2\sigma^2)) \quad (5)$$

Algorithm 3. Kernel smoothing operator

Input: data set D, smoothing factor σ
Output: smoothed search space S^α
(1) $S^\alpha = zeros(N, N)$;
(2) for any pairwise data objects $x_i,x_j \in D$
$S^\alpha(i,j) = dist(x_i,x_j)^* \exp(-dist(x_i,x_j)^2/2\sigma^2)$
(3) return S^α

The main steps of kernel smoothing operator are shown in Algorithm 3. For any pairwise data objects belonging to the data set D, a Gaussian kernel influence is added to distance function $dist(x_i,x_j)$ to smooth the surface of search space. Once transformation on a pairwise data object x_i,x_j needs $O(1)$ time, so for the transformation of N^2 pairwise data objects, at least need $O(N^2)$ time.

3. Clustering algorithm based on smoothing search space

Based on the smoothing operator and the smoothing factor, a series of smoothed search spaces with different number of minimum 'traps', are reconstructed as a hierarchical structure. Any iterative refinement clustering algorithm can be run on each smoothed search space from the top search space down to the bottom search space. A clustering algorithm is proposed to realize this process in this section. For simple description, we use the symbol α to denote the smoothing factor in the rest of this paper.

3.1. Top–down clustering algorithm based on smoothing search space

We give a top–down clustering algorithm with the smoothing search space (TDCS3) in this section. The main steps of TDCS3 are as follows: (1) dynamically reconstruct a series of smoothed search space by running a smoothing operator; (2) an existing iterative refinement clustering algorithm is run on the current smoothed search space and the clustering result is generated; (3) based on the clustering results, a new initialization is generated and service it as the initialization to the next smoothed search space; (4) repeat step (1)–(3) until returning to the original search space of the hierarchical structure (see Fig. 3). Algorithm 4 is the main description of TDCS3.

Algorithm 4. TDCS3

Input: Data set D, Cluster number K and Smoothing factor α
Output: cluster results C

(1) Generate the top smoothed search space S^{α} with α;
(2) Run any iterative refinement clusteringΛ on $S^{\alpha}$with random initialization, and generate the cluster resultsC^{α};
(3) while $\alpha \geqslant \alpha_{org}$
  (3.1) Generate the initialization$Init^{\alpha}$ from C^{α};
  (3.2) $\alpha' \leftarrow \alpha - \lambda$ and generate new smoothed search space$S^{\alpha'}$ with α';
  (3.3) Run Λ on $S^{\alpha'}$with$Init^{\alpha}$,and generate the cluster results$C^{\alpha'}$;
  (3.4) $\alpha \leftarrow \alpha', C^{\alpha} \leftarrow C^{\alpha'}$;
(4) Return$C \leftarrow C^{\alpha}$.

From the top search space to the bottom search space of the hierarchical structure, any existing iterative refinement clustering algorithm Λ, such as K-means [13], MeanShift [24] and so on, could be run on those search spaces. At the top smoothed search space, algorithm Λ is run with random initialization, and the correlative clustering result is generated. The initialization from the clustering result on the former smoothed search space of hierarchical structure will be regarded as the initialization of Λ on the current smoothed search space and lead the search of algorithm Λ to converge to a better sub-optimal result (a local minimum point with lower value of cluster criterion). Iteratively run these steps until returning to the bottom search space.

3.2. Strengths of TDCS3

TDCS3 focuses on reducing the influence of lots of minimum 'traps' which are embedded in the search space of iterative refinement clustering algorithms. Compared to traditional iterative refinement clustering algorithms, TDCS3 has the following benefits:

(1) Intelligent characteristicIn TDCS3, a series of smoothed search spaces with different numbers of minimum points are reconstructed. The smoothed search spaces are the different level topological structures of the original search space. The quality of cluster results on more smoothed search space is high, for the number of minimum 'traps' are less. So the initialization from the clustering result on more smoothed search space can capture good structure of clusters. The initialization from the clustering results on the former search space can lead the search on current search space to a better minimum point.
(2) Flexible characteristicIn TDCS3, the smoothing operator and the iterative clustering algorithm Λ are not solely fixed. According to different applications and demands, the smoothing operator could be redesigned and the iterative refinement algorithm would be reselected.
(3) Adaptive characteristicTDCS3 inherits the merits of iterative refinement clustering algorithms and reconstructs their search space to reduce the probability of getting stuck into a worse sub-optimal minimum point. So it is able to improve the quality of cluster result of any existing iterative refinement clustering algorithm.

4. Experiments

4.1. Benchmark

In TDCS3, any existing clustering algorithm based on iterative refinement algorithm could be integrated to run with two smoothing operators: the displacement smoothing operator (DS) and the kernel smoothing operator (KS). We denote an iterative refinement clustering algorithm Λ run in TDCS3 with displacement smoothing operator DS as TDCS3_Λ_DS. In similar way, the meaning of TDCS3 _Λ_KS means that an iterative refinement algorithm Λ is run in TDCS3 with kernel smoothing operator. Λ is a common symbol for any existing iterative refinement clustering algorithm, and can be replaced by an existing clustering algorithm's name. For example, if we run the classical iterative refinement clustering algorithm, K-means, in TDCS3 with displacement smoothing operator, then we could denote it as TDCS3 _K-means_DS.

In this paper, we intend to run K-means and MeanShift (MS) algorithms in TDCS3 with the displacement and the kernel smoothing operator. We thus got four algorithms: TDCS3_K-means _DS, TDCS3_K-means_KS, TDCS3_ MS_DS, and TDCS3_MS_KS respectively. We then compare these four algorithms with three kinds of pure iterative clustering algorithms, i.e. K-means, MS and K-means++ on 3 synthetic data sets and 10 real world data sets. K-means++ was downloaded from http://www.jihe.net/research/ijcnn04, but it was rewritten in Matlab. The popular classical K-means and its general version- MeanShift clustering algorithms, are downloaded from the MathWork website. TDCS3 algorithm is implemented in Matlab 7.0 too. The settings of parameters for K-means, K-means++, and MeanShift are similar to those in Refs. [13,1,24], respectively. For TDCS3, only the smoothing factor needs to be set. The chosen smoothing factor for DS and KS are tabulated in Table 1.

In general, the quality of our clustering algorithm heavily relies on the number of smoothed search spaces, which is calculated by $(\alpha - \alpha_{org})/\lambda$. The bigger number of smoothed space will lead to the better quality of clusters but the improvement of the clustering quality becoming less and less when the number of smoothed space increases. In experiments, we intend to choose a small value of λ and a large value of α. On the other hand, however, the bigger number of smoothed space will substantially increase the time cost of the proposed iterative refinement. Thus there is a contradiction between clustering quality and time cost, and in order to make a trade-off between them we need to choose the appropriate parameters that indicated in Table 1.

In TDCS3, a hierarchy structure of smoothed search spaces is reconstructed by running a smoothing operator with different value of smoothing factor. The change of smoothing factor value

Table 1
The smoothing factor parameters setting.

Smoothing operator	α	α_{org}	λ
DS	10	1	1
KS	1	0.25	0.25

decides the layout of hierarchy structure. For any smoothing factor, it changes $(\alpha - \alpha_{org})/\lambda$ times and running a smoothing operator with one smoothing factor value lead to a smoothed search space. As discussed in Section 2.2.1, reconstructing a smoothed search space needs $O(N^2)$ time cost, so the time cost of reconstructing $(\alpha - \alpha_{org})/\lambda$ smoothed search spaces is $((\alpha - \alpha_{org})/\lambda)^* O(N^2)$. On each smoothed search space, an existing clustering algorithm is additionally run to get the clustering result. So the total time cost of TDCS3 is $((\alpha - \alpha_{org})/\lambda)^*(O(N^2) + O(\cdot))$, where $O(\cdot)$ the time cost of any existing iterative refinement clustering algorithms.

Since the objective of clustering is to find out groups embedded in data set so that intra-group similarities are maximized and inter-group similarities are minimized at the same time. So we adopted two measures i.e. cluster compactness (Cmp) and the cluster separation (Sep) indices [11,12] to assess the goodness of the clustering results.

Definition 4. Given a data set $D = \{x_1, x_2, \ldots, x_N\} \subset R^d$, the deviation of D is given by

$$dev(D) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|x_i - \overline{D}\|} \tag{6}$$

Where N is the number of members in D, and $\overline{D} = 1/N\sum_{i=1}^{N} x_i$ is the mean of data set D.

The cluster compactness measure is based on the generalized definition of deviation, which is shown as Definition 4. The cluster compactness for the output clusters $C = \{C_1, C_2, \ldots, C_K\}$ is generated by Definition 5.

Definition 5. Given the output clustering result $C = \{C_1, C_2, \ldots, C_K\}$, the cluster compactness is described as:

$$\text{Cmp} = \frac{1}{K}\sum_{k=1}^{K}\frac{dev(C_k)}{dev(D)} \tag{7}$$

Where K is the number of clusters and $dev(C_k)$ is the deviation of clusterC_k.

Definition 6 gives the definition of cluster separation of the output cluster results.

Definition 6. Given the cluster centers $m = \{m_1, m_2, \ldots, m_K\}$ of the output clusters $C = \{C_1, C_2, \ldots, C_K\}$, cluster separation is described as:

$$\text{Sep} = \frac{1}{K(K-1)}\sum_{i=1}^{K}\sum_{j=1, j\neq i}^{K}\exp\left(-\frac{\|m_i - m_j\|}{2\sigma^2}\right) \tag{8}$$

Where σ is a Gaussian constant.

4.2. Synthetic data sets

In order to comprehensively compare the output clustering results with existing iterative refinement clustering algorithm, we use a random generator downloaded from http://www.jihe.net/research/ijcnn04, to produce three two dimensional synthetic data sets approximately in mixture of Gaussian distribution. Based on 15 predesigned cluster centers, 150,000 data points were generated for each data set using varying variances $v = 0.05{:}0.05{:}0.15$ and noise rates $r = 0{:}0.2{:}0.4$, where $v = 0.05{:}0.05{:}0.15$ describes the change of variance from 0.05 to 0.15 with step length of 0.05, and the same meanings of $r = 0{:}0.2{:}0.4$.

Table 2 gives the experiment results of seven algorithms on three synthetic data sets. In Table 2, *K*-means and MS are the original iterative refinement clustering algorithm without smoothing search space, while TDCS3_K-means_DS, TDCS3_K-means_KS, TDCS3_MS_DS, TDCS3_MS_KS are four enhanced clustering algorithms by combining *K*-means and MS algorithm with smoothing search space using DS and KS smoothing operator respectively.

Table 2
The experiment results of seven clustering algorithms on three synthetic data sets. Cluster compactness (Cmp) and cluster separation (Sep) with Gaussian constant $\sigma = 0.5$ are compared.

Algorithm	Cmp	Sep
DB1: v = 0.05;r = 0.0;K = 15		
K-means	0.1574 ± 0.0155	0.6891 ± 0.0375
K-means++	0.1565 ± 0.0103	0.6328 ± 0.0131
MeanShift	0.1572 ± 0.0115	0.6294 ± 0.0295
TDCS3_MS_DS	0.1455 ± 0.0099	0.6094 ± 0.0121
TDCS3_MS_KS	0.1460 ± 0.0089	0.6062 ± 0.0110
TDCS3_K-means_DS	0.1495 ± 0.0097	0.6071 ± 0.0112
TDCS3_K-means_KS	0.1465 ± 0.0090	0.6001 ± 0.0096
DB2: v = 0.1;r = 0.2;K = 15		
K-means	0.2372 ± 0.0047	0.6490 ± 0.0275
K-means++	0.2375 ± 0.0057	0.6328 ± 0.0280
MeanShift	0.2388 ± 0.0039	0.6294 ± 0.0250
TDCS3_MS_DS	0.2054 ± 0.0015	0.6004 ± 0.0090
TDCS3_MS_KS	0.2102 ± 0.0012	0.6062 ± 0.0089
TDCS3_K-means_DS	0.2020 ± 0.0009	0.6060 ± 0.0070
TDCS3_K-means_KS	0.2089 ± 0.0015	0.6095 ± 0.0089
DB3: v = 0.15; r = 0.4; K = 15		
K-means	0.2591 ± 0.0029	0.6050 ± 0.0032
K-means++	0.2593 ± 0.0023	0.6019 ± 0.0034
MeanShift	0.2592 ± 0.0019	0.6045 ± 0.0034
TDCS3_MS_DS	0.2160 ± 0.0009	0.5730 ± 0.0027
TDCS3_MS_KS	0.2219 ± 0.0008	0.5608 ± 0.0030
TDCS3_K-means_DS	0.2111 ± 0.0012	0.5780 ± 0.0029
TDCS3_K-means_KS	0.2301 ± 0.0011	0.5602 ± 0.0024

From Table 2, we can find that the Cmp and Sep score of each algorithm on DB1 are smaller than those on DB2 and DB3. This phenomenon implies that seven clustering algorithms are sensitive to noise. The last four algorithms are modified based on smoothing search space method. By running the smoothing operator, a lot of minimum 'traps' embedded in the rugged search space of *K*-means and MeanShift are smoothed, so the influence of minimum 'traps' is reduced. The Cmp score and Sep score of these four algorithms are less than the scores of *K*-means and MeanShift on DB1. This phenomena show that the qualities of clustering results generated by our algorithms are better than other three algorithms. These four algorithms do not change the original essential of *K*-means and MeanShift, so they would not change their characteristic of sensitive to noise. The Cmp and Sep scores of the last four algorithms on each data set are lower than the scores of *K*-means, *K*-menas++, and MeanShift. The error range of Cmp and Sep scores are also both reduced significantly for the latter four algorithms. The reason for these phenomena is that the search space of *K*-means and MeanShift clustering algorithm has been reconstructed. Initialization from the clustering results of former smoothed search space can lead the iterative refinement clustering algorithm to converge to more better minimum point.

From the experiment results on three synthetic data sets, we can find that the existing iterative refinement clustering algorithms have the capability to become more robust to produce high quality cluster results via TDCS3.

4.3. Real world data sets

Ten data sets from the UCI machine learning repository [4] are used, all of which contains only numerical attributes except the class attributes. For the image segmentation, a constant attribute has been removed, just as done in Ref. [25]. The information about the data sets is tabulated in Table 3. Note that the class attributes of the data sets have not been used in the clustering algorithms.

Table 3
Data sets used in experiments.

Data set	Size	Attribute	Class
Iris	150	4	3
Liver disorder (LD)	345	6	2
Page blocks (PB)	5473	10	5
Wine	178	13	3
Vehicle	846	18	4
Image segmentation (IS)	2310	18	7
WaveForm21 (WF21)	5000	21	3
Wpbc	198	33	2
IOnoSphere (IOS)	351	34	2
WaveForm40 (WF40)	5000	40	3

Although the concept of cluster compactness and clustering separation is heterogeneous in essence, but the quantity property defined in Definitions 5 and 6 are homogenous, that is, they have the same changing trend. So it is reasonable to use the linear combination of Cmp and Sep metrics to define the overall index of clustering $quality(C) = Cmp + \beta^* Sep$ and to show how good the output clustering results are, where $\beta > 0$ is the combination parameter. According to the analysis in Ref. [5], we set parameter β equal to 1. Fig. 5 shows the experiment results of seven algorithms on 10 real world data sets.

Even if the dimension and the data size of Iris data set are smaller than liver disorder (LD) data set, the *quality*(*C*) values of Iris data set are higher than LD one. The reason for this phenomenon is that the three clusters embedded in Iris data set belong to different subspace of attributes, that is, the attributes of Iris data set have non-linear relationship. Classical iterative refinement clustering algorithm cannot separate them distinctly. Because the Kernel smoothing operator can deal with the non-linear clustering problem, the *quality*(*C*) values of TDCS3_K-means_ KS and TDCS3_MS_KS are lower than the values of other algorithms.

The dimension of page block (PB) data set is smaller than Wine data set, but the *quality*(*C*) value of PB is higher than that of Wine, for the data size of PB data set is higher than Wine data set. Vehicle and image segmentation (IS) data sets have the same dimension, but the data size of IS data set is higher than vehicle data set, so the *quality*(*C*) values of IS data set are higher than vehicle data set. The *quality*(*C*) values of the last four data sets have an increasing tendency with the increasing dimensions of them.

Experimental results on 10 data sets show that the iterative refinement clustering algorithms are sensitive to the data size and the data dimension. But from the Fig. 5, we can find the modified algorithm, TDCS3_MS_DS, TDCS3 _MS_KS,TDCS3_K-means_DS, and TDCS3_K-means_ KS have better *quality*(*C*) values

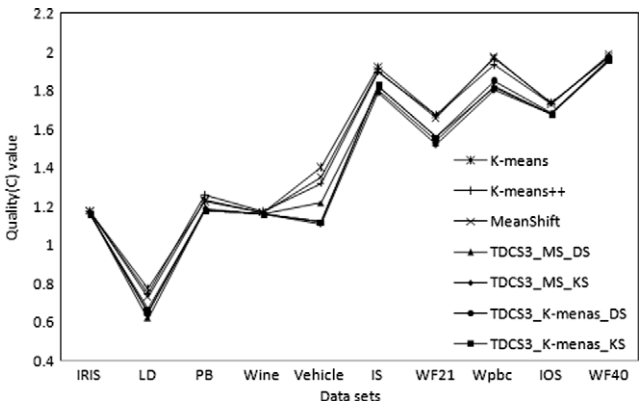


Fig. 5. Experiment results of seven algorithms on ten real world data sets.

than *K*-means, *K*-means++, and MeanShift on 10 data sets. The real reason for these phenomena is that the smoothing operator changes the search space of *K*-means and MeanShift, and reduces the probability of them to coverage to minimum points, leading to the qualities of cluster results increased.

5. Conclusion

Most of the initialization methods have some merits for reducing the influence initialization sensitivity of iterative refinement clustering algorithm. The real reason of an iterative refinement clustering algorithm always sensitive to the initialization is the rugged terrain surface. In this paper, two smoothing operators, which can deal with linear and non-linear data sets, is designed based on distance metric. A series of smoothed search spaces with different numbers of minimum 'traps' are reconstructed into a hierarchical structure. Based on the hierarchical structure of smoothed search space, a top–down clustering algorithm, TDCS3, is proposed. In TDCS3, any existing iterative refinement algorithm can be run to get better cluster results. Experiment results on synthetic and real world data set have shown that our algorithm could reduce the influence of initialization sensitivity and can get a better, robust cluster results.

Acknowledgments

This work was supported by the National 973 Plan of China under the Grant No. 2007CB714205, the Nature Science Foundation of China under Grant No. 60805024, the Ph.D. Programs Foundation of Ministry of Education of China under Grant No. 20070141020, the Australia Research Council Discovery Project DP0770479, and the Nature Science Foundation of Anhui Education Department under the Grant Nos. KJ2009A54 and KJ2007A072.

References

[1] D. Arthur, S. Vassilvitskii, K-means++: The Advantages of Careful Seeding, Technical Report, Standford, 2007, pp. 1027–1035.
[2] B. Addis, M. Locatelli, F. Schoen, Local optima smoothing for global optimization, Optimization Methods and Software (20) (2005) 417–437.
[3] M. Belal, A. Daoud, A new algorithm for cluster initialization, in: Proceeding of World Academy of Science, Engineering and Technology, vol. 04, 2005, pp. 74–76.
[4] C. Blake, E. Keogh, C.J. Merz, UCI Repository of Machine Learning Databases, Department of Information and Computer Science, University of California, Irvine, CA, 1998. Available from: %3chttp://www.ics.uci.edu/~mlearn/MLRepository.htm%3e.
[5] L.F. Chen, Q.S. Jiang, S.R. Wang, A hierarchical method for determining the number of clusters, Journal of Software 19 (1) (2008) 62–72.
[6] P. Drineas, R. Frieze, S. Vempala, et al., Clustering large graphs via singular value decomposition, Machine Learning 56 (1-3) (2004) 9–33.
[7] A.M. Daoud, S. Roberts, New Methods for Initialisation of Clusters, Technical Report 94.34, School of Computer Studies, University of Leeds, 1994.
[8] A. Forgy, N. Laird, D. Rubin, Maximum likelyhood from incomplete data via codes for analog input patterns, Applied Optics 26 (1987) 4919–4930.
[9] M.R. Garey, D.S. Johnson, H.S. Witsenhausen, The complexity of the generalized Lloyd-max problem, IEEE Transactions on Information Theory 28 (2) (1980) 255–256.
[10] J. Gu, X.F. Huang, Efficient local search with search space smoothing: a case study of the traveling salesman problem, IEEE Transactions on System, Man, and Cybernetics 24 (5) (1994) 728–735.
[11] J. He, L. Man, C.L. Tan et al., 2004. Initialization of cluster refinement algorithm: a review and comparative study, in: IEEE International Joint Conference on Neural Networks, pp. 297–302.
[12] J. He, A.H. Tan, C.L. Tan, et al., On quantitative evaluation of clustering systems, in: Weili Wu, Hui Xiong, Shashi Shekhar (Eds.), Information Retrieval and Clustering, Kluwer Academic Publishers., 2003, 2003.
[13] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data [M], Prentice Hall, Englewood Cliffs, New Jersey, 1998.
[14] Rousseeuw, Kaufman, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, New York, 1990.
[15] I. Katsavounidis, C. Kou, Z. Zhang, A new initialization technique for generalized Lloyd iteration, IEEE Signal Processing Letters 1 (10) (1994) 144–146.

[16] J.F. Lu, J.B. Tang, Z.M. Tang, et al., Hierarchical initialization approach for *K*-means clustering, Pattern Recognition Letters 29 (2008) 787–795.
[17] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the 5th Berkely Symposium in Mathematics and Probability, 1967, pp. 281–297.
[18] K. Niu, S.B. Zhang, J.L. Chen, 2006. An initializing cluster centers algorithm based on pointer ring, in: Proceedings of the 6th International Conference on Intelligent Systems Design and Applications (ISDA'06), pp. 655–660.
[19] S. Paul, U. Bradley, M. Fayyad, Refining Initial Points for *K*-means Clustering, in Proceeding of 15th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1998, pp. 91–99.
[20] S.J. Redmond, C. Heneghan, A method for initialising the *K*-means clustering algorithm using kd-tree, Pattern Recognition Letters 28 (2007) 965–973.
[21] SAS Institute Inc., SAS User's Guided: Statistics, Version 5 Ed., 1985.
[22] S. Sahni, T. Gonzalez, *P*-complete approximate problems [J], J ACM 23 (3) (1976) 555–565.
[23] J.T. Tou, R.C. Gonzalez, Pattern Recognition Principles, Addision Wesley, Massachusetts, 1974.
[24] K.L. Wu, M.S. Yang, Mean shift-based clustering, Pattern Recognition (40) (2007) 3035–3052.
[25] Z.H. Zhou, W. Tang, Cluster ensemble, Knowledge-Based System 19 (2006) 77–83.