Optimization: A Journal of Mathematical Programming and Operations Research

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/gopt20

An algorithm for minimizing clustering functions

Adil M. Bagirov & Julien Ugon

[a] Centre for Informatics and Applied Optimization , School of Information Technology and Mathematical Sciences, University of Ballarat , Victoria, 3353, Australia

Published online: 08 Aug 2006.

To cite this article: Adil M. Bagirov & Julien Ugon (2005) An algorithm for minimizing clustering functions , Optimization: A Journal of Mathematical Programming and Operations Research, 54:4-5, 351-368, DOI: 10.1080/02331930500096155

To link to this article: http://dx.doi.org/10.1080/02331930500096155

Optimization,
Vol. 54, Nos. 4–5, August–October 2005, 351–368

An algorithm for minimizing clustering functions[†]

ADIL M. BAGIROV* and JULIEN UGON

Centre for Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Victoria, 3353, Australia

(*Received in final form 15 December 2004*)

The problem of cluster analysis is formulated as a problem of nonsmooth, nonconvex optimization. An algorithm for solving the latter optimization problem is developed which allows one to significantly reduce the computational efforts. This algorithm is based on the so-called discrete gradient method. Results of numerical experiments are presented which demonstrate the effectiveness of the proposed algorithm.

Keywords: Clustering functions; Cluster analysis; Nonsmooth, nonconvex optimization; Discrete gradient method

2000 Mathematics Subject Classifications: 49M25; 49M37; 49J52; 65K05; 90C26

1. Introduction

This article develops an algorithm for minimizing the so-called clustering functions.

Clustering is an important application area in data mining. Clustering is also known as the *unsupervised* classification of patterns. The cluster analysis deals with the problems of organization of a collection of patterns into clusters based on similarity. It has found many applications, including information retrieval, medicine, etc.

In cluster analysis we assume that we have been given a finite set A of points in the n-dimensional space $\mathbb{R}^n$, that is

$$A = \{a^1, \dots, a^m\}, \quad \text{where } a^i \in \mathbb{R}^n,\ i = 1, \dots, m.$$

There are different types of clustering such as partition, packing, covering and hierarchical clustering [1]. In this article we consider partition clustering, that is,

*Corresponding author. Email: a.bagirov@ballarat.edu.au
[†]Dedicated to V.F. Demyanov on the occasion of his 65th birthday.

the distribution of the points of the set A into a given number q of disjoint subsets A^i, $i = 1, \dots, q$ with respect to predefined criteria such that:

(1) $A^i \neq \emptyset$, $i = 1, \dots, q$;
(2) $A^i \cap A^j = \emptyset$, $i, j = 1, \dots, q$, $i \neq j$;
(3) $A = \cup_{i=1}^{q} A^i$.

The sets A^i, $i = 1, \dots, q$ are called clusters. The strict application of these rules is called *hard clustering*, unlike *fuzzy clustering*, where the clusters are allowed to overlap. In this article we consider the hard unconstrained clustering problem, that is, no constraints are imposed on the clusters A^i, $i = 1, \dots, q$.

An up-to-date survey of existing approaches to clustering is provided in [2] and a comprehensive list of literature on clustering algorithms is available in this article.

We assume that each cluster A^i, $i = 1, \dots, q$ can be identified by its center (or centroid). Then the clustering problem can be reduced to the following optimization problem (see [3–5]):

$$\text{minimize } \varphi(C, x) = \frac{1}{m} \sum_{i=1}^{q} \sum_{a \in A^i} \|x^i - a\|_p^{\gamma} \tag{1}$$
$$\text{subject to } C \in \bar{C}, \quad x = (x^1, \dots, x^q) \in \mathbb{R}^{n \times q}$$

where $C = \{A^1, \dots, A^q\}$ is a set of clusters, $\bar{C}$ is the set of all possible q-partitions of the set A and x^i is the center of the cluster A^i, $i = 1, \dots, q$. Here $\gamma \geq 1$ and $\| \cdot \| = \| \cdot \|_p$, $p \geq 1$. Recall that

$$\|x\|_p = \left(\sum_{l=1}^{n} |x_1|^p \right)^{1/p}, \quad 1 \leq p < +\infty, \qquad \|x\|_\infty = \max_{l=1,\dots,n} |x_l|$$

where x_l is the lth coordinate of a vector $x \in \mathbb{R}^n$.

If $\gamma = 2$ and $p = 2$ then the problem (1) is also known as the sum-of-squares clustering problem.

It should be noted that numerical methods of optimization cannot be directly applied to solve problem (1).

To solve this problem many different heuristic approaches have been developed, mainly for $\gamma = 2$ and $p = 2$. We can mention here agglomerative and divisive hierarchical clustering algorithms. The k-means algorithms and their variations (h-means, j-means etc.) are known to be the most effective among them to solve large-scale clustering problems. However, the effectiveness of these algorithms highly depends on an initial point. Descriptions of many of these algorithms can be found, for example, in [5–8].

In order to make problem (1) suitable for the application of optimization methods we need to reformulate it. This problem is equivalent to the following problem:

$$\text{minimize } \psi(x, w) = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{q} w_{ij} \|x^j - a^i\|_p^{\gamma} \tag{2}$$
$$\text{subject to } x = (x^1, \dots, x^q) \in \mathbb{R}^{n \times q},$$
$$\sum_{j=1}^{q} w_{ij} = 1, \quad i = 1, \dots, m,$$

and

$$w_{ij} = 0 \text{ or } 1, \quad i = 1, \ldots, m, \; j = 1, \ldots, q$$

where w_{ij} is the association weight of pattern a^i with cluster j, given by

$$w_{ij} = \begin{cases} 1 & \text{if pattern } i \text{ is allocated to cluster } j \; \forall i = 1, \ldots, m, \; j = 1, \ldots, q, \\ 0 & \text{otherwise} \end{cases}$$

and

$$x^j = \frac{\sum_{i=1}^{m} w_{ij} a^i}{\sum_{i=1}^{m} w_{ij}}, \quad j = 1, \ldots, q.$$

Different methods of mathematical programming can be applied to solve problem (2) including dynamic programming, branch and bound, cutting planes. A review of these algorithms can be found in [1]. Dynamic programming approach can be effectively applied to the clustering problem when the number of instances is not large (see [9]). Branch and bound algorithms are effective when the database contain only hundreds of records and the number of clusters is not large (less than 5) (see [1,10–12]). For these methods the solution of large clustering problems is out of reach.

Much better results have been obtained with metaheuristics, such as simulated annealing, tabu search and genetic algorithms [13]. The simulated annealing approaches to clustering have been studied, for example, in [14–16]. Application of tabu search methods for solving clustering is studied in [17]. Genetic algorithms for clustering have been described in [13]. The results of numerical experiments, presented in [18] show that even for small problems of cluster analysis when the number of entities $m \leq 100$ and the number of clusters $q \leq 5$ these algorithms take 500–700 (sometimes several thousands) times more CPU time than the k-means algorithms. For relatively large databases one can expect that this difference will increase. This makes metaheuristic algorithms of global optimization ineffective for solving many clustering problems.

In [19] an interior point method for minimum sum-of-squares clustering problem is developed. The paper [20] develops variable neighborhood search algorithm and paper [7] presents j-means algorithm which extends k-means by adding a jump move. The global k-means heuristic, which is an incremental approach to minimum sum-of-squares clustering problem, is developed in [21]. The incremental approach is also studied in the paper [22]. Results of numerical experiments presented show the high effectiveness of these algorithms for many clustering problems.

The paper [23] describes a global optimization approach to clustering and demonstrates how the supervised data classification problem can be solved via clustering.

The objective function ψ in problem (2) has many local minima. Many of these local minima do not provide good clusterings of the dataset. The formulation (2) induces that a good clustering may be provided by the global minimum of the objective function ψ. However, due to the large number of variables and the complexity of the objective function, general-purpose global optimization techniques, as a rule, fail to solve such problems.

In many clustering algorithms it is assumed that the number of clusters is known *a priori*. However, this usually is not the case.

In [24] an algorithm for clustering based on nonsmooth optimization techniques was developed. Here we present this algorithm, which calculates clusters step-by-step, gradually increasing the number of data clusters until termination conditions are met. In this approach the clustering problem is reduced to an unconstrained optimization problem with nonsmooth objective function. In the present article we develop an algorithm for solving this optimization problem. Results of numerical experiments using real-world datasets are presented and their comparison with the best known solutions from the literature are provided.

The article is organized as follows: the nonsmooth optimization approach to clustering is discussed in section 2. Section 3 describes an algorithm for solving clustering problems. An algorithm for solving optimization problems is described in section 4. Section 5 presents the results of the numerical experiments and section 6 concludes the article.

2. The nonsmooth optimization approach to clustering

In this section we present a reformulation of the clustering problem (2) in terms of nonsmooth, nonconvex optimization.

In [23,25] the cluster analysis problem is reduced to the following problem of mathematical programming

$$\text{minimize } f(x^1, \dots, x^q) \quad \text{subject to } (x^1, \dots, x^q) \in \mathbb{R}^{n\times q}, \tag{3}$$

where

$$f(x^1, \dots, x^q) = \frac{1}{m}\sum_{i=1}^{m} \min_{s=1,\dots,q} \|x^s - a^i\|_p^\gamma. \tag{4}$$

If $q > 1$, the objective function (4) in problem (3) is nonconvex and nonsmooth.

Note that the number of variables in the optimization problem (3) is $q \times n$. If the number q of clusters and the number n of attributes are large, we have a large-scale global optimization problem. Moreover, the form of the objective function in this problem is complex enough not to be amenable to the direct application of general purpose global optimization methods.

It is shown in [3] that problems (1), (2) and (3) are equivalent. The number of variables in problem (2) is $(m+n) \times q$ whereas in problem (3) this number is only $n \times q$ and the number of variables does not depend on the number of instances. It should be noted that in many real-world databases the number of instances m is substantially greater than the number of features n. On the other hand in the hard clustering problems the coefficient w_{ij} are integer, that is the problem (2) contains both integer and continuous variables. In the nonsmooth optimization formulation of the clustering problem we have only continuous variables. All these circumstances can be considered as advantages of the nonsmooth optimization formulation (3) of the clustering problem.

3. An optimization clustering algorithm

In this section we describe an algorithm for solving cluster analysis problems.

Algorithm 1 An algorithm for solving the clustering problem.

Step 1 (Initialization). Select a tolerance $\epsilon > 0$. Select a starting point $x^0 = (x_1^0, \dots, x_n^0) \in \mathbb{R}^n$ and solve the minimization problem (3) for $q = 1$. Let $x^{1*} \in \mathbb{R}^n$ be a solution to this problem and f^{1*} be the corresponding objective function value. Set $k = 1$.

Step 2 (Computation of the next cluster center). Select a starting point $y^0 \in \mathbb{R}^n$ and solve the following minimization problem:

$$\text{minimize}\, \bar{f}^{\,k}(y) \quad \text{subject to } y \in \mathbb{R}^n \tag{5}$$

where

$$\bar{f}^{\,k}(y) = \sum_{i=1}^{m} \min\left\{ \|x^{1*} - a^i\|_p^\gamma, \dots, \|x^{k*} - a^i\|_p^\gamma, \|y - a^i\|_p^\gamma \right\}. \tag{6}$$

Step 3 (Refinement of all cluster centers). Let $y^{k+1,*}$ be a solution to problem (5). Take $x^{k+1,0} = (x^{1*}, \dots, x^{k*}, y^{k+1,*})$ as a new starting point and solve the problem (3) for $q = k+1$. Let $x^{k+1,*}$ be a solution to problem (3) and $f^{\,k+1,*}$ be the corresponding objective function value.

Step 4 (Stopping criterion). If

$$\frac{f^{\,k*} - f^{\,k+1,*}}{f^{\,1*}} < \epsilon$$

then stop, otherwise set $k = k+1$ and go to Step 2.

Algorithm 1 contains some steps which deserve some explanations. In Step 1 the center of the entire set A is calculated with respect to a given norm. In this case the problem (3) is a convex programming problem. In Step 2 we calculate a center of the next $(k+1)$-th cluster, assuming the previous k cluster centers to be known and fixed. It should be noted that the number of variables in problem (5) is n which is substantially less than, if we calculate, all cluster centers simultaneously. In Step 3 the refinement of all $k+1$ cluster centers is carried out. One can expect that the starting point $x^{k+1,0}$ calculated in Step 2 is not far from the solution to problem (3). Therefore it takes only a moderate number of iterations to calculate it. Such an approach allows one to significantly reduce the computational time for solving problem (3).

It is clear that $f^{\,k*} \geq 0$ for all $k \geq 1$ and the sequence $\{f^{\,k*}\}$ is decreasing, that is,

$$f^{\,k+1,*} \leq f^{\,k,*} \quad \text{for all } k \geq 1.$$

The latter implies that after $\bar{k} > 0$ iterations the stopping criterion in Step 4 will be satisfied.

One of the important questions when one tries to apply Algorithm 1 is the choice of the tolerance $\epsilon > 0$. Large values of ϵ can result in the appearance of large clusters whereas small values can produce small and artificial clusters. Results presented in [24,25] show that appropriate values for ϵ are $\epsilon \in [10^{-1}, 10^{-2}]$.

An algorithm for solving problems (3) and (5) is discussed in section 4. Since this algorithm is a local one, the choice of a good initial guess for problem (5) is very important. An algorithm for finding such an initial guess is described below. It should be noted that starting points in problem (3) are predetermined by Algorithm 1.

3.1. *An algorithm for finding the initial points in problem (5)*

The main idea behind Step 2 in Algorithm 1 is that a new cluster is added to the pre-existing ones.

In [7] an improvement for the *h*-means algorithm, *j*-means, is given, to avoid so-called *degenerated solutions*, where one cluster is empty: the furthest point from all cluster centers is taken as a new center. This leads to an interesting idea, however it needs to be further improved: often real-world datasets contain erroneous records which can be quite far from the rest of the dataset. Taking such an erroneous point as an initial guess may lead to a shallow local minimum. In figure 1, the point P_1 is the furthest point from the cluster centers. However, if this point is chosen as the center of a cluster, this cluster will consist of only P_1. The point P_2 is closer to the centers, however, it would induce a much better cluster.

The following algorithm allows one to avoid this difficulty.

Algorithm 2 An algorithm for finding the initial point in problem (5).

Step 1 (Initialization). Let $C_1, \dots, C_{q-1}$, $q \geq 2$ be the preexisting centers and $\rho > 0$ a tolerance. Let $A_1 = A$, and $i = 1$.
Step 2 Let C be the point in A_i the furthest from the centers $C_1, \dots, C_{q-1}$.
Step 3 Find the set

$$\mathcal{C} = \left\{ a \in A_i \colon \|C - a\| < \min_{j=1,\dots,q-1} \|C_j - a\| \right\}.$$

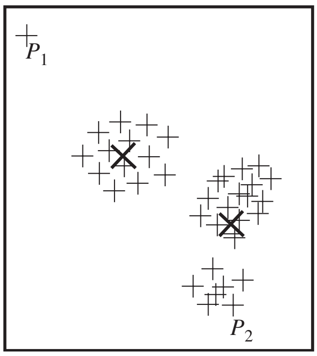

Figure 1. An example of a dataset containing an erroneous point.

If the cardinality of the set $\mathcal{C}$ card$\{\mathcal{C}\} > \rho$, then $C_q = C$ and the algorithm terminates. Otherwise go to Step 4.

Step 4 Set $A_{i+1} = A_i \backslash \{C\}$, $i = i + 1$, and go to Step 2.

Remark 1 Since A contains a finite number of points the initial guess will be found after a finite number of steps. If the tolerance ρ is too small then the initial guess may be an erroneous point, but if ρ is large then no initial point can be found. Results of numerical experiments show that the tolerance ρ should be chosen in $[0.1(m/q), 0.4(m/q)]$.

4. Solving optimization problems

Both problems (3) and (5) in the clustering algorithm are nonsmooth optimization problems. Since the objective functions in these problems are represented as a sum of minimum of norms they are Lipschitz continuous. In order to describe differential properties of the objective functions f and $\bar{f}$ we recall some definitions from nonsmooth analysis (see [26,27]).

4.1. *Differential properties of the objective functions*

We consider a locally Lipschitz function φ defined on $\mathbb{R}^n$. This function is differentiable almost everywhere and one can define for it a Clarke subdifferential (see [26]), by

$$\partial\varphi(x) = \operatorname{co}\left\{ v \in \mathbb{R}^n : \exists (x^k \in D(\varphi), x^k \to x, k \to +\infty) : v = \lim_{k \to +\infty} \nabla\varphi(x^k) \right\},$$

here $D(\varphi)$ denotes the set where φ is differentiable, co denotes the convex hull of a set.

The function φ is differentiable at the point $x \in \mathbb{R}^n$ with respect to the direction $g \in \mathbb{R}^n$ if the limit

$$\varphi'(x, g) = \lim_{\alpha \to +0} \frac{\varphi(x + \alpha g) - \varphi(x)}{\alpha}$$

exists. The number $\varphi'(x, g)$ is said to be the derivative of the function φ with respect to the direction $g \in \mathbb{R}^n$ at the point x.

The upper derivative $\varphi^0(x, g)$ of the function φ at the point x with respect to the direction $g \in \mathbb{R}^n$ is defined as follows:

$$\varphi^0(x, g) = \limsup_{\alpha \to +0,\, y \to x} \frac{\varphi(y + \alpha g) - \varphi(y)}{\alpha}.$$

The following is true (see [26])

$$\varphi^0(x, g) = \max\{\langle v, g\rangle : v \in \partial\varphi(x)\}$$

where $\langle \cdot, \cdot \rangle$ stands for the inner product in $\mathbb{R}^n$. It should be noted that the upper derivative always exists for locally Lipschitz functions. The function φ is said to be regular at the point $x \in \mathbb{R}^n$ if

$$\varphi'(x, g) = \varphi^0(x, g)$$

for all $g \in \mathbb{R}^n$. For regular functions there exists a calculus (see [26,27]). However, in general for non-regular functions such a calculus does not exist.

The function φ is called semismooth at $x \in \mathbb{R}^n$, if it is locally Lipschitz continuous at x and for every $g \in \mathbb{R}^n$, the limit

$$\lim_{v \in \partial\varphi(x+tg'),\, g' \to g,\, t \to +0} \langle v, g \rangle$$

exists (see [28]).

Since both functions f from (4) and $\bar{f}$ from (6) are locally Lipschitz continuous they are subdifferentiable.

Proposition 1 *The functions f and $\bar{f}$ are semismooth.*

Proof The sum and the minimum of semismooth functions are semismooth (see [28]). A norm as a convex function is semismooth. Then the function f, which is the sum of functions represented as the minimum of norms, is semismooth. Similarly the function $\bar{f}$, which is the sum of functions represented as the minimum of constants and a norm, is semismooth as well. ■

Functions represented as a minimum of norms in general are not regular. Since the sum of non-regular functions in general is not regular, the functions f and $\bar{f}$ in general are not regular. Therefore, subgradients of these functions cannot be calculated using subgradients of their terms. We can conclude that the calculation of the subgradients of the functions f and $\bar{f}$ is a very difficult task and therefore the application of methods of nonsmooth optimization requiring a subgradient evaluation at each iteration, including bundle method and its variations ([29–31]), cannot be effective to solve problems (3) and (5).

The problems (3) and (5) are global optimization problems. However, they have to be solved many times in Algorithm 1, and thus require fast algorithms. Moreover, the number of variables in the problem (3) is large and the global optimization methods cannot be directly applied to solve it. Therefore, we discuss algorithms for finding local minima of the functions f and $\bar{f}$.

Since the evaluation of subgradients of the function f is difficult, direct search methods of optimization seem to be the best option for solving problems (3) and (5). Among such methods we mention here two widely used methods: Powell's method (see [32]) which is based on a quadratic interpolation of the objective function and Nelder–Mead's simplex method [33]. Both methods perform well for smooth functions. Moreover, they are only effective when the number of variables is less than 20. However, in both problems (3) and (5) the objective functions are quite complicated nonsmooth functions. In the problem (3) the number of variables is $n_v = n \times q$ where n is the dimension of the dataset A (ranging from 2 to thousands in real-world datasets), and q is the number of clusters. In many cases the number n_v is greater than 20.

In this article we use the discrete gradient method to solve the problems (3) and (5). The description of this method can be found in [34] (see, also, [35,36]). The discrete gradient method can be considered as a version of the bundle method ([29–31]), where subgradients of the objective function are replaced by its discrete gradients.

The discrete gradient method uses only values of the objective function. It should be noted that the calculation of the objective functions in the problems (3) and (5) can be expensive as the number of instances in the dataset grows. We show that the use of the discrete gradient method allows one to significantly reduce the number of objective function evaluations.

4.2. *Discrete gradient method*

In this subsection we briefly describe the discrete gradient method. We start with the definition of the discrete gradient.

4.2.1. Definition of the discrete gradient. Let φ be a locally Lipschitz continuous function defined on $\mathbb{R}^n$. Let

$$S_1 = \{g \in \mathbb{R}^n \colon \|g\| = 1\}, \qquad G = \{e \in \mathbb{R}^n \colon e = (e_1, \dots, e_n),\ |e_j| = 1, j = 1, \dots, n\},$$
$$P = \{z(\lambda) \colon z(\lambda) \in \mathbb{R}^1,\ z(\lambda) > 0, \lambda > 0, \lambda^{-1} z(\lambda) \to 0, \lambda \to 0\},$$
$$I(g, \alpha) = \{i \in \{1, \dots, n\} \colon |g_i| \geq \alpha\},$$

where $\alpha \in (0, n^{-1/2}]$ is a fixed number.

Here S_1 is the unit sphere, G is the set of vertices of the unit hypercube in $\mathbb{R}^n$ and P is the set of univariate positive infinitesimal functions.

We define operators $H_i^j \colon \mathbb{R}^n \to \mathbb{R}^n$ for $i = 1, \dots, n, j = 0, \dots, n$ by the formula

$$H_i^j g = \begin{cases} (g_1, \dots, g_j, 0, \dots, 0) & \text{if } j < i, \\ (g_1, \dots, g_{i-1}, 0, g_{i+1}, \dots, g_j, 0, \dots, 0) & \text{if } j \geq i. \end{cases} \tag{7}$$

We can see that

$$H_i^j g - H_i^{j-1} g = \begin{cases} (0, \dots, 0, g_j, 0, \dots, 0) & \text{if } j = 1, \dots, n,\ \ j \neq i, \\ 0 & \text{if } j = i. \end{cases} \tag{8}$$

Let $e(\beta) = (\beta e_1, \beta^2 e_2, \dots, \beta^n e_n)$, where $\beta \in (0, 1]$. For $x \in \mathbb{R}^n$ we consider vectors

$$x_i^j \equiv x_i^j(g, e, z, \lambda, \beta) = x + \lambda g - z(\lambda) H_i^j e(\beta), \tag{9}$$

where $g \in S_1$, $e \in G$, $i \in I(g, \alpha)$, $z \in P$, $\lambda > 0, j = 0, \dots, n, j \neq i$.

It follows from (8) that

$$x_i^{j-1} - x_i^j = \begin{cases} (0, \dots, 0, z(\lambda) e_j(\beta), 0, \dots, 0) & \text{if } j = 1, \dots, n,\ \ j \neq i, \\ 0 & \text{if } j = i. \end{cases} \tag{10}$$

It is clear that $H_i^0 g = 0$ and $x_i^0(g, e, z, \lambda, \beta) = x + \lambda g$ for all $i \in I(g, \alpha)$.

Definition 1 (see [36,37]) The discrete gradient of the function φ at the point $x \in \mathbb{R}^n$ is the vector $\Gamma^i(x, g, e, z, \lambda, \beta) = (\Gamma^i_1, \ldots, \Gamma^i_n) \in \mathbb{R}^n$, $g \in S_1$, $i \in I(g, \alpha)$, with the following coordinates:

$$\Gamma^i_j = [z(\lambda)e_j(\beta)]^{-1}\Big[\varphi(x^{j-1}_i(g, e, z, \lambda, \beta)) - \varphi(x^j_i(g, e, z, \lambda, \beta))\Big], \quad j = 1, \ldots, n,\ j \neq i,$$

$$\Gamma^i_i = (\lambda g_i)^{-1}\left[\varphi(x^n_i(g, e, z, \lambda, \beta)) - \varphi(x)) - \sum_{j=1, j\neq i}^{n} \Gamma^i_j(\lambda g_j - z(\lambda)e_j(\beta))\right].$$

A more detailed description of the discrete gradient and examples can be found in [34,36].

Remark 2 It follows from Definition 1 that for the calculation of the discrete gradient $\Gamma^i(x, g, e, z, \lambda, \beta)$, $i \in I(g, \alpha)$ we define a sequence of points

$$x^0_i, \ldots, x^{i-1}_i, x^{i+1}_i, \ldots, x^n_i.$$

For the calculation of the discrete gradient it is sufficient to evaluate the function φ at each point of this sequence.

Remark 3 The discrete gradient is defined with respect to a given direction $g \in S_1$. We can see that for the calculation of one discrete gradient we have to calculate $(n+1)$ values of the function φ: at the point x and at the points $x^0_i, \ldots, x^{i-1}_i$, $x^{i+1}_i, \ldots, x^n_i$, $i \in I(g, \alpha)$. For the calculation of another discrete gradient at the same point with respect to another direction $g^1 \in S_1$ we have to calculate this function n times, because we have already calculated φ at the point x.

4.2.2. Calculation of the discrete gradients of the objective function (4). Now let us return to the objective function f of problem (3). This function depends on $n \times q$ variables where q is the number of clusters. This function is represented as a sum of min-type functions

$$f(x) = \sum_{i=1}^{m} \psi_i(x)$$

where

$$\psi_i(x) = \min_{j=1,\ldots,q} [\eta_{ij}(x)]^\theta, \quad i = 1, \ldots, m$$

with $\theta = \gamma/p$, and

$$\eta_{ij}(x) = \sum_{u=1}^{n} |x^j_u - a^i_u|^p.$$

It should be noted that the functions η_{ij} are separable. We can see that for every $i = 1, \ldots, m$, each variable x^j appears in only one function η_{ij}.

For a given $k = 1, \dots, nq$ we set

$$r_k = \left\lfloor \frac{k-1}{n} \right\rfloor + 1, \qquad d_k = k - (r_k - 1)n$$

where $\lfloor c \rfloor$ stands for the floor of a number c. We define by X the vector of all variables x^j, $j = 1, \dots, q$:

$$X = (X_1, X_2, \dots, X_{nq})$$

where

$$X_k = x^{r_k}_{d_k}.$$

We use the vector of variables X to define a sequence

$$X^0_t, \dots, X^{t-1}_t, X^{t+1}_t, \dots, X^{nq}_t, \quad t \in I(g, \alpha), \quad g \in \mathbb{R}^{nq}$$

as in Remark 2. It follows from (10) that the points X^{k-1}_t and X^k_t differ by one coordinate only ($k = 0, \dots, nq, k \neq t$). For every $i = 1, \dots, m$ this coordinate appears in only one function η_{ir_k}. It follows from the definition of the operator H that $X^t_t = X^{t-1}_t$ and thus this observation is also true for X^{t+1}_t. Then we get

$$\eta_{ij}(X^k_t) = \eta_{ij}(X^{k-1}_t) \quad \forall j \neq r_k$$

which means that when we change the kth coordinate of the point X, only one function (namely the function η_{ir_k}) changes its value.

Moreover, the function η_{ir_k} can be calculated at the point X^k_t using the value of this function at the point X^{k-1}_t, $k \geq 1$:

$$\eta_{ir_k}(X^k_t) = \eta_{ir_k}(X^{k-1}_t) + \left|x^{r_k}_{d_k} + z(\lambda)e_k(\beta) - a^i_{d_k}\right|^p - \left|x^{r_k}_{d_k} - a^i_{d_k}\right|^p. \tag{11}$$

Because η is separable, only one term in the sum changes. Thus we need to add the new term, and subtract the old one. For $p = 2$ (11) can be simplified as follows:

$$\eta_{ir_k}(X^k_t) = \eta_{ir_k}(X^{k-1}_t) + z(\lambda)e_k(\beta)\left(2x^{r_k}_{d_k} + z(\lambda)e_k(\beta) - 2a^i_{d_k}\right).$$

In order to calculate the function f at the point X^k_t, $k \geq 1$ first we have to calculate the values of the functions η_{ir_k} for all $a^i \in A$, $i = 1, \dots, m$ using (11). Then we update f using these values and the values of all other functions η_{ij}, $j \neq r_k$ at the point X^{k-1}_t according to (4). Thus we have to apply a full calculation of the function f using the formula (4) only at the point $X^0_t = X + \lambda g$. Hence for the calculation of each discrete gradient, we have to apply a full calculation of the objective function f only at the point $X^0_t = X + \lambda g$ and this function can be updated at the points X^k_t, $k \geq 1$ using a simplified scheme.

We can conclude that for the calculation of the discrete gradient at a point X with respect to the direction $g^0 \in S_1$ we calculate the function f at two points: X and $X_t^0 = X + \lambda g^0$. For the calculation of another discrete gradient at the same point X with respect to another direction $g^1 \in S_1$ we calculate the function f only at the point: $X + \lambda g^1$.

The function $\bar{f}(x) \equiv \bar{f}^{\,u}(x)$ defined by (6) can be rewritten as follows:

$$\bar{f}(x) = \sum_{i=1}^{m} \min\left\{c_i, [\bar{\eta}_i(x)]^{\theta}\right\}$$

where

$$c_i = \min\left\{\|x^{1*} - a^i\|_p^{\gamma}, \ldots, \|x^{u*} - a^i\|_p^{\gamma}\right\}$$

and

$$\bar{\eta}_i(x) = \sum_{l=1}^{n} |x_l - a_l^i|^p, \quad x \in \mathbb{R}^n.$$

Here $\theta > 0$ is defined as above. The functions $\bar{\eta}_i$ are separable and have a similar structure as the function η_{ij}. Therefore we can apply a simplified scheme for their calculation as well. Hence we can accelerate the calculation of each discrete gradient of the function $\bar{f}$ in a similar way as we do for the function f (see above).

Since the number of variables nq in the problem (3) can be large, this algorithm allows one to significantly reduce the number of objective function evaluations during the calculation of a discrete gradient. Although the number of variables n in the problem (5) is significantly less than in the problem (3), this algorithm still allows one to accelerate the calculation of the discrete gradients of the objective function.

4.2.3. The method. We consider the following unconstrained minimization problem:

$$\text{minimize } \varphi(x) \quad \text{subject to } x \in \mathbb{R}^n \tag{12}$$

where the function φ is assumed to be semismooth. An important step in the discrete gradient method is the calculation of a descent direction of the objective function φ. Therefore, we first describe an algorithm for the computation of this direction.

Let $z \in P$, $\lambda > 0$, $\beta \in (0, 1]$, the number $c \in (0, 1)$ and a tolerance $\delta > 0$ be given.

Algorithm 3 An algorithm for the computation of the descent direction.

Step 1 Choose any $g^1 \in S_1$, $e \in G$, $i \in I(g^1, \alpha)$ and compute a discrete gradient $v^1 = \Gamma^i(x, g^1, e, z, \lambda, \beta)$. Set $\bar{D}_1(x) = \{v^1\}$ and $k = 1$.

Step 2 Calculate the vector $\|w^k\| = \min\{\|w\| : w \in \bar{D}_k(x)\}$. If

$$\|w^k\| \leq \delta, \tag{13}$$

then stop. Otherwise go to Step 3.

Step 3 Calculate the search direction by $g^{k+1} = -\|w^k\|^{-1} w^k$.

Step 4 If

$$\varphi(x+\lambda g^{k+1})-\varphi(x)\leq -c\lambda\|w^k\|, \tag{14}$$

then stop. Otherwise go to Step 5.

Step 5 Calculate a discrete gradient

$$v^{k+1}=\Gamma^i(x,g^{k+1},e,z,\lambda,\beta),\quad i\in I(g^{k+1},\alpha),$$

construct the set $\bar{D}_{k+1}(x)=\mathrm{co}\{\bar{D}_k(x)\cup\{v^{k+1}\}\}$, set $k=k+1$ and go to Step 2.

We give some explanations to Algorithm 3. In Step 1 we calculate the first discrete gradient with respect to an initial direction $g^1\in\mathbb{R}^n$. The distance between the convex hull $\bar{D}_k$ of all calculated discrete gradients and the origin is calculated in Step 2. This problem can be solved using Wolfe's algorithm [38]. If this distance is less than the tolerance $\delta>0$ then we accept the point x as an approximate stationary point (Step 2), otherwise we calculate another search direction in Step 3. In Step 4 we check whether this direction is a descent direction. If it is, we stop and the descent direction has been calculated, otherwise we calculate another discrete gradient with respect to this direction in Step 5 and update the set $\bar{D}_k$. At each iteration k we improve the approximation $\bar{D}_k$ of the subdifferential of the function φ.

It is proved that Algorithm 3 is terminating (see [34]).

Now we can describe the discrete gradient method. Let sequences $\delta_k>0$, $z_k\in P$, $\lambda_k>0$, $\beta_k\in(0,1]$, $\delta_k\to+0$, $z_k\to+0$, $\lambda_k\to+0$, $\beta_k\to+0$, $k\to+\infty$ and numbers $c_1\in(0,1)$, $c_2\in(0,c_1]$ be given.

Algorithm 4 Discrete gradient method

Step 1 Choose any starting point $x^0\in\mathbb{R}^n$ and set $k=0$.

Step 2 Set $s=0$ and $x^k_s=x^k$.

Step 3 Apply Algorithm 3 for the calculation of the descent direction at $x=x^k_s$, $\delta=\delta_k$, $z=z_k$, $\lambda=\lambda_k$, $\beta=\beta_k$, $c=c_1$. This algorithm terminates after a finite number of iterations $l>0$. As a result we get the set $\bar{D}_l(x^k_s)$ and an element v^k_s such that

$$\|v^k_s\|=\min\{\|v\|:v\in\bar{D}_l(x^k_s)\}.$$

Furthermore, either $\|v^k_s\|\leq\delta_k$ or for the search direction $g^k_s=-\|v^k_s\|^{-1}v^k_s$

$$\varphi(x^k_s+\lambda_k g^k_s)-\varphi(x^k_s)\leq -c_1\lambda_k\|v^k_s\|. \tag{15}$$

Step 4 If

$$\|v^k_s\|\leq\delta_k \tag{16}$$

the set $x^{k+1}=x^k_s$, $k=k+1$ and go to Step 2. Otherwise go to Step 5.

Step 5 Construct the following iteration $x_{s+1}^k = x_s^k + \sigma_s g_s^k$, where σ_s is defined as follows

$$\sigma_s = \arg\max\{\sigma \geq 0: \varphi(x_s^k + \sigma g_s^k) - \varphi(x_s^k) \leq -c_2\sigma\|v_s^k\|\}.$$

Step 6 Set $s = s+1$ and go to Step 3.

For the point $x^0 \in \mathbb{R}^n$ we consider the set $M(x^0) = \{x \in \mathbb{R}^n: \varphi(x) \leq \varphi(x^0)\}$.

THEOREM 1 *Assume that the set $M(x^0)$ is bounded for starting points $x^0 \in \mathbb{R}^n$. Then every accumulation point of $\{x^k\}$ belongs to the set $X^0 = \{x \in \mathbb{R}^n: 0 \in \partial\varphi(x)\}$.*

Since the objective functions in problems (3) and (5) are semismooth the discrete gradient method can be applied to solve them. Discrete gradients in Step 5 of Algorithm 3 can be calculated using the simplified schemes described earlier.

5. Results of numerical experiments

To verify the effectiveness of the proposed approach, a number of numerical experiments with real-world data sets have been carried out on a Pentium-4, 1.7 GHz, PC. The following datasets have been used in numerical experiments:

(1) Fisher's iris dataset ($m = 150$, $n = 4$) [39];
(2) Image segmentation dataset ($m = 2310$, $n = 19$) [40];
(3) The traveling salesman problem – TSPLIB1060 ($m = 1060$, $n = 2$) [41];
(4) The traveling salesman problem – TSPLIB3038 ($m = 3038$, $n = 2$) [41].

In order to implement Algorithm 4 we have to select the sequences $\{\delta_k\}$, $\delta_k > 0$, $\{z_k\}$, $z_k \in P$, $\{\lambda_k\}$, $\lambda_k > 0$, $\{\beta_k\}$, $\beta_k \in (0, 1]$. In the numerical experiments we choose these sequences as follows: $\delta_k = 10^{-9}$, $\beta_k = 1$ for all k, $z_k(\lambda) = \lambda^\alpha$, $\alpha \in [1.5, 4]$, $\lambda_k = d^k\lambda_0$, $d = 0.5$, $\lambda_0 = 0.9$. We take $\lambda_{\min} \in (0, \lambda_0)$. If $\lambda_k < \lambda_{\min}$ then Algorithm 4 terminates. In order to get a solution with high accuracy one has to take $\lambda_{\min}$ very small, for example $\lambda_{\min} \leq 10^{-5}$. Larger values of $\lambda_{\min}$ may lead to more inaccurate solutions, however, Algorithm 4 calculates such solutions very quickly. In our numerical experiments, unless specified otherwise, we set $\lambda_{\min}$ large.

In the numerical experiments we also take $c_1 = 0.2$ and $c_2 = 0.001$. The starting point for solving problem (5) is generated by Algorithm 2 and the starting point for solving problem (3) is generated in Algorithm 1. In Algorithm 2 we fix the parameter $\rho = (rm)/q$ where $r \in [0, 0.5]$.

For image segmentation dataset we take $\lambda_{\min} = 10^{-5}$. For all other datasets this parameter is: $\lambda_{\min} = 0.01$.

In the numerical experiments we consider the squared Euclidean norm, that is $\gamma = 2$ and $p = 2$.

In order to provide comparison with the best known solutions from the literature, we calculated 10 clusters for iris dataset and 50 clusters for all other datasets.

Algorithm 1 is a deterministic one. However, it requires the selection of the parameter ρ, which determines the starting points. To see how this parameter influences the results, numerical experiments have been carried out for $r_i = 0.05i$, $i = 0, \ldots, 10$.

We present the results of the numerical experiments in tables 1–4. In these tables k represents the number of clusters. We give the best known function value f_{opt} from

Table 1. Results for Fisher's iris dataset.

k	f_{opt}	E_{best}	E_{mean}	N_s	N_g/N_s	t	σ_t
2	152.348	0.00	0.00	1.43×10^5	3.63	0.10	0.02
3	78.851	0.00	0.00	3.22×10^5	5.00	0.20	0.03
4	57.226	0.00	0.00	5.83×10^5	5.95	0.34	0.05
5	46.446	0.00	0.68	8.73×10^5	7.33	0.49	0.05
6	39.040	0.00	0.00	1.21×10^6	8.23	0.65	0.04
7	34.298	0.00	0.86	1.57×10^6	9.31	0.82	0.05
8	29.989	0.00	0.11	2.03×10^6	10.62	1.04	0.07
9	27.786	0.00	2.11	2.46×10^6	11.86	1.23	0.09
10	25.834	0.52	1.78	300×10^6	13.04	1.47	0.07

Table 2. Image segmentation dataset.

k	f_{opt}	E_{best}	E_{mean}	N_s	N_g/N_s	t	σ_t
2	3.5606×10^7	−0.01	2.08	8.14×10^6	20.69	19.43	3.12
3	2.7416×10^7	−0.02	4.90	1.86×10^7	27.31	40.05	6.93
4	1.9456×10^7	−0.03	15.82	3.60×10^7	33.70	69.37	12.92
5	1.7143×10^7	−0.03	13.16	5.45×10^7	40.41	98.53	22.23
6	1.5209×10^7	−0.03	16.36	7.73×10^7	47.34	131.88	32.67
7	1.3404×10^7	0.33	10.50	1.04×10^8	54.77	169.15	47.53
8	1.2030×10^7	2.28	14.29	1.30×10^8	62.96	205.52	49.40
9	1.0784×10^7	1.36	9.28	1.61×10^8	70.12	248.53	70.28
10	9.7952×10^6	1.51	8.72	1.90×10^8	77.12	289.24	74.52
20	5.1283×10^6	−0.01	8.76	2.48×10^8	111.15	897.46	198.27
30	3.5076×10^6	5.62	10.21	3.33×10^8	184.89	1815.14	255.73
40	2.7398×10^6	9.96	13.52	3.50×10^8	207.80	2396.03	247.34
50	2.2248×10^6	15.26	19.30	3.65×10^8	231.72	3011.10	241.75

Table 3. TSPLIB3038 dataset.

k	f_{opt}	E_{best}	E_{mean}	N_s	N_g/N_s	t	σ_t
2	3.1688×10^9	0.00	0.00	1.78×10^6	1.69	0.74	0.17
3	2.1763×10^9	0.00	1.51	4.31×10^6	2.24	1.65	0.19
4	1.4790×10^9	0.00	0.03	6.62×10^6	2.82	2.48	0.14
5	1.1982×10^9	0.00	0.21	9.56×10^6	3.35	3.50	0.16
6	9.6918×10^8	0.00	0.05	1.26×10^7	3.90	4.56	0.16
7	8.3966×10^8	1.73	1.91	1.52×10^7	4.54	5.53	0.11
8	7.3475×10^8	0.00	0.75	1.94×10^7	5.20	6.99	0.32
9	6.4477×10^8	0.00	0.44	2.34×10^7	5.94	8.41	0.40
10	5.6025×10^8	0.00	0.68	2.78×10^7	6.73	9.98	0.37
20	2.6681×10^8	0.00	0.90	3.40×10^7	10.07	29.35	1.04
30	1.7557×10^8	0.27	1.55	4.32×10^7	16.56	61.04	1.35
40	1.2548×10^8	−0.08	1.48	5.45×10^7	24.70	104.74	1.72
50	9.8400×10^7	0.62	1.63	6.82×10^7	33.88	158.03	3.83

Table 4. TSPLIB1060 dataset.

k	f_{opt}	E_{best}	E_{mean}	N_s	N_g/N_s	t	σ_t
10	1.75484×10^9	0.00	0.18	1.39×10^7	6.00	4.61	0.42
20	4.91794×10^9	0.32	3.01	1.77×10^7	8.74	13.62	1.08
30	4.81251×10^9	1.29	3.76	2.46×10^7	14.03	29.02	1.48
50	2.55509×10^9	1.36	2.46	4.48×10^7	28.12	81.90	3.72

the literature corresponding to k clusters ([7,22]) and the % error E_{best} and E_{mean} of respectively the best value and the average value obtained by the proposed algorithm. The error E is calculated as

$$E = \frac{(\bar{f} - f_{\text{opt}})}{f_{\text{opt}}} \times 100$$

where $\bar{f}$ is the function value obtained by the algorithm. A negative error means that the proposed algorithm improved the best known solution. We also give the average number of calculations of norms for reaching the solutions. N_s and N_g represent the average number of norm evaluations for finding k clusters using the simplified and general schemes, respectively. In the tables we give the values of N_s and the ratio N_g/N_s. Finally, in these tables we present the average CPU time (t) and its standard deviation (σ_t).

Results presented in tables 1–4 show that for Fisher's iris, TSPLIB3038 and TSPLIB1060 datasets, the proposed algorithm allows one to calculate either the best known solution or a solution which is very close to the best one. For image segmentation dataset the results for large numbers of clusters are not so good which can be explained by the existence of erroneous points. These results demonstrate that if the algorithm gets stuck in a shallow minimum then it may affect the next iterations. However, results for this dataset show that the algorithm in most cases reaches a solution which is close to the best one. It should be noted that the problem of finding 50 clusters in the image segmentation dataset has 950 variables which is challenging for many global optimization techniques.

The results for the error of average values presented in these tables show that for Fisher's iris, TSPLIB3038 and TSPLIB1060 datasets the results obtained by the proposed algorithm do not strongly depend on the initial point (that is values of ρ) and they are always close to the best solutions. The results for the image segmentation dataset are more dependent on the initial point. This is an indicator of the existence of erroneous points in this dataset. This shows that if a dataset does not have a good cluster structure the proposed algorithm may lead to different solutions starting from different initial points.

The results for the number N_s show that one can calculate a large number of clusters using a reasonable number of norm evaluations. The ratio N_g/N_s demonstrate that the simplified scheme allows one to significantly reduce the computational effort. This complexity reduction becomes larger as the number of clusters or the number of features increase.

The results presented in the tables show that the clustering problem can be solved by the algorithm within reasonable CPU time. CPU time depends on the parameter $\lambda_{\min}$. Small values of $\lambda_{\min}$ lead to a better solution, however, much more CPU time is required as in the case of the image segmentation dataset. The results for σ_t show that the CPU time does almost not depend on starting points.

6. Conclusions

In this article we have formulated the clustering problem as a nonsmooth nonconvex optimization problem and proposed a clustering algorithm based on nonsmooth optimization techniques. The discrete gradient method has been used for solving the nonsmooth optimization problems. We have adapted this method to this problem which allows one to considerably accelerate its convergence. The clustering algorithm calculates clusters incrementally. This algorithm calculates as many clusters as a dataset contains with respect to some tolerance. The proposed algorithm has been tested using real-world datasets. The results of these experiments demonstrate that the algorithm can calculate a satisfactory solution within a reasonable CPU time. These results explicitly show that the algorithm allows one to significantly reduce the computational effort which is extremely important for clustering in large scale data sets.

Acknowledgments

This research was supported by the Australian Research Council.

References

[1] Hansen, P. and Jaumard, B., 1997, Cluster analysis and mathematical programming. *Mathematical Programming*, **79**(1–3), 191–215.

[2] Jain, A.K., Murty, M.N. and Flynn, P.J., 1999, Data clustering: a review. *ACM Computing Surveys*, **31**(3), 263–323.

[3] Bock, H.H., 1974, *Automatische Klassifikation* (Gottingen: Vandenhoeck & Ruprecht).

[4] Bock, H.H., 1998, Clustering and neural networks. In: A. Rizzi, M. Vichi and H.H. Bock (Eds), *Advances in Data Science and Classification* (Berlin: Springer-Verlag), pp. 265–277.

[5] Spath, H., 1980, *Cluster Analysis Algorithms* (Chichester: Ellis Horwood Limited).

[6] Dubes, R. and Jain, A.K., 1976, Clustering techniques: the user's dilemma. *Pattern Recognition*, **8**, 247–260.

[7] Hansen, P. and Mladenovic, N., 2001, *J*-means: a new heuristic for minimum sum-of-squares clustering. *Pattern Recognition*, **4**, 405–413.

[8] Houkins, D.M., Muller, M.W. and Ten Krooden, J.A., 1982, Cluster analysis. In: *Topics in Applied Multivariate Analysis* (Cambridge: Cambridge University press).

[9] Jensen, R.E., 1969, A dynamic programming algorithm for cluster analysis. *Operations Research*, **17**, 1034–1057.

[10] Diehr, G., 1985, Evaluation of a branch and bound algorithm for clustering. *SIAM Journal on Scientific and Statistical Computing*, **6**, 268–284.

[11] Hanjoul, P. and Peeters, D., 1985, A comparison of two dual-based procedures for solving the *p*-median problem. *European Journal of Operational Research*, **20**, 387–396.

[12] Koontz, W.L.G., Narendra, P.M. and Fukunaga, K., 1975, A branch and bound clustering algorithm. *IEEE Transactions on Computers*, **24**, 908–915.

[13] Reeves, C.R. (Ed.), 1993, *Modern Heuristic Techniques for Combinatorial Problems* (London: Blackwell).

[14] Brown, D.E. and Entail, C.L., 1992, A practical application of simulated annealing to the clustering problem. *Pattern Recognition*, **25**, 401–412.

[15] Selim, S.Z. and Al-Sultan, K.S., 1991, A simulated annealing algorithm for the clustering. *Pattern Recognition*, **24**(10), 1003–1008.

[16] Sun, L.X., Xie, Y.L., Song, X.H., Wang, J.H. and Yu, R.Q., 1994, Cluster analysis by simulated annealing. *Computers and Chemistry*, **18**, 103–108.

[17] Al-Sultan, K.S., 1995, A tabu search approach to the clustering problem. *Pattern Recognition*, **28**(9), 1443–1451.

[18] Al-Sultan, K.S. and Khan, M.M., 1996, Computational experience on four algorithms for the hard clustering problem. *Pattern Recognition Letters*, **17**, 295–308.

[19] du Merle, O., Hansen, P., Jaumard, B. and Maldenovic, N., 2001, An interior point method for minimum sum-of-squares clustering. *SIAM Journal on Scientific Computing*, **21**, 1485–1505.

[20] Hansen, P. and Mladenovic, N., 2001, Variable neighborhood decomposition search. *Journal of Heuristic*, **7**, 335–350.

[21] Likas, A., Vlassis, M. and Verbeek, J., 2003, The global k-means clustering algorithm. *Pattern Recognition*, **36**, 451–461.
[22] Hansen, P., Ngai, E., Cheung, K. and Mladenovic, N., Analysis of global k-means, an incremental heuristic for minimum sum-of-squares clustering. *Journal of Classification*, to appear.
[23] Bagirov, A.M., Rubinov, A.M. and Yearwood, J., 2002, A global optimisation approach to classification. *Optimization and Engineering*, **3**(2), 129–155.
[24] Bagirov, A.M. and Yearwood, J., A new nonsmooth optimisation algorithm for minimum sum-of-squares clustering algorithms. *European Journal of Operational Research* (To appear).
[25] Bagirov, A.M., Rubinov, A.M., Soukhoroukova, N.V. and Yearwood, J., 2003, Supervised and unsupervised data classification via nonsmooth and global optimisation. *TOP: Spanish Operations Research Journal*, **11**(1), 1–93.
[26] Clarke, F.H., 1983, *Optimization and Nonsmooth Analysis* (New York: Wiley).
[27] Demyanov, V.F. and Rubinov, A.M., 1995, *Constructive Nonsmooth Analysis* (Frankfurt am Main: Peter Lang).
[28] Mifflin, R., 1977, Semismooth and semiconvex functions in constrained optimization. *SIAM Journal on Control and Optimization*, **15**(6), 959–972.
[29] Hiriart-Urruty, J.-P. and Lemarechal, C., 1993, *Convex Analysis and Minimization Algorithms*, Vol. 1 and 2 (Berlin, New York: Springer-Verlag).
[30] Kiwiel, K.C., 1985, *Methods of Descent for Nondifferentiable Optimization*. Lecture Notes in Mathematics, **1133** (Berlin: Springer-Verlag).
[31] Makela, M.M. and Neittaanmaki, P., 1992, *Nonsmooth Optimization* (Singapore: World Scientific).
[32] Powell, M.J.D., 2002, UOBYQA: unconstrained optimization by quadratic approximation, *Mathematical Programming*, *Series B*, **92**(3), 555–582.
[33] Nelder, J.A. and Mead, R., 1965, A simplex method for function minimization, *Computer Journal*, **7**, 308–313.
[34] Bagirov, A.M., 1999, Minimization methods for one class of nonsmooth functions and calculation of semi-equilibrium prices. In: A. Eberhard *et al.* (Eds), *Progress in Optimization: Contribution from Australasia* (Dordrecht: Kluwer Academic Publishers), pp. 147–175.
[35] Bagirov, A.M., 2002, A method for minimization of quasidifferentiable functions. *Optimization Methods and Software*, **17**(1), 31–60.
[36] Bagirov, A.M., 2003, Continuous subdifferential approximations and their applications. *Journal of Mathematical Sciences*, **115**(5), 2567–2609.
[37] Bagirov, A.M. and Gasanov, A.A., 1995, A method of approximating a quasidifferential. *Russian Journal of Computational Mathematics and Mathematical Physics*, **35**(4), 403–409.
[38] Wolfe, P.H., 1976, Finding the nearest point in a polytope. *Mathematical Programming*, **11**(2), 128–149.
[39] Fisher, R.A., 1936, The use of multiple measurements in taxonomic problems. *Ann Eugenics*, **VII** (part II), 179–188. Reprinted in R.A. Fisher, *Contributions to Mathematical Statistics*, Wiley, 1950.
[40] Murphy, P.M. and Aha, D.W., 1992, UCI repository of machine learning databases. Technical report, Department of Information and Computer Science. University of California, Irvine. www.ics.uci.edu/mlearn/MLRepository.html
[41] Reinelt, G., 1991, TSP-LIB-A Traveling Salesman Library. *ORSA J. Comput.*, **3**, 319–350.