INFORMATICA, 2012, Vol. 23, No. 4, 601–620

Multimodal Evolutionary Algorithm for Multidimensional Scaling with City-Block Distances

Juana López REDONDO[1], Pilar Martínez ORTIGOSA[2], Julius ŽILINSKAS[3]

[1]*Department of Computer Architecture and Technology, University of Granada*
Periodista Daniel Saucedo Aranda, s/n. 18071 Granada, Spain
[2]*Department of Informatics, University of Almería*
Campus de Excelencia Internacional Agroalimentario (ceiA3), Spain
[3]*Vilnius University Institute of Mathematics and Informatics*
Akademijos 4, LT-08663 Vilnius, Lithuania
e-mail: jlredondo@atc.ugr.es, ortigosa@ual.es, julius.zilinskas@mii.vu.lt

Received: September 2012; accepted: December 2012

Abstract. Multidimensional scaling with city-block distances is considered in this paper. The technique requires optimization of an objective function which has many local minima and can be non-differentiable at minimum points. This study is aimed at developing a fast and effective global optimization algorithm spanning the whole search domain and providing good solutions. A multimodal evolutionary algorithm is used for global optimization to prevent stagnation at bad local optima. Piecewise quadratic structure of the least squares objective function with city-block distances has been exploited for local improvement. The proposed algorithm has been compared with other algorithms described in literature. Through a comprehensive computational study, it is shown that the proposed algorithm provides the best results. The algorithm with fine-tuned parameters finds the global minimum with a high probability.

Keywords: multidimensional scaling, city-block distances, evolutionary algorithms, multimodal algorithms.

1. Introduction

Multidimensional scaling (MDS) is a technique employed for exploratory analysis of multidimensional data (Borg and Groenen, 2005; Dzemyda *et al.*, 2013). It can be used to represent multidimensional data as a set of points in a low-dimensional embedding space and expose it as an image for heuristic analysis by human experts. Application areas of MDS vary from psychometrics (Takane, 2006) and market analysis (DeSarbo *et al.*, 1998; Nelson and Rabianski, 1988) to mobile communications (Groenen *et al.*, 2000) and pharmacology (Žilinskas, 2006).

MDS aims at finding the points in a low dimensional space whose inter-point distances fit the given dissimilarities of multidimensional objects. This can be achieved by

optimization of a criterion of fit (Žilinskas and Žilinskas, 2009b). Unfortunately the optimization problem is not easy to solve since there are normally many local minima. Moreover, the objective function is invariant with respect to translation and rotation or mirroring. The number of decision variables depends on the number of multidimensional objects and the dimensionality of the embedding space, so, the number can be large.

The Euclidean distances are commonly used in the definition of the objective function for MDS. Efficient algorithms have been proposed for such problems (Everett, 2001; Groenen *et al.*, 2000; Mathar, 1996; Mathar and Žilinskas, 1993). However, MDS with other Minkowski distances can be more informative than MDS with the Euclidean distances (Arabie, 1991). The images of geometrical data obtained using MDS with the city-block distances highlight the structural properties of the original data better than the images obtained using MDS with the Euclidean distances (Žilinskas and Žilinskas, 2006, 2007, 2009b). Agonist and antagonist ligands can be separated by a line in a two-dimensional image obtained using MDS with the city-block distances (Žilinskas, 2006), but this is not possible in a corresponding image obtained using MDS with the Euclidean distances.

In this study, MDS with the city-block distances is considered. In the case of the city-block distances, the least squares objective function can be non-differentiable at a minimum point (Žilinskas and Žilinskas, 2007). Therefore special attention should be given to this point. However, the least squares objective function with city-block distances is piecewise quadratic. Such a structure is exploited to develop a local optimization algorithm.

A successful algorithm for global optimization has to provide a balance between the *exploration* of the search space to identify regions with good solutions and the *exploitation* of the accumulated search experience. In this work, a multimodal evolutionary algorithm (MEAL) for MDS with the city-block distances has been designed. MEAL forms and maintains a population of individuals. Individual represents the attraction region of a local minimum point. MEAL could also be identified as a memetic algorithm (Speer *et al.*, 2004), in the sense that it uses local optimization in the evolution process.

Performance of MEAL has been compared with two algorithms applicable to MDS with the city-block distances: GENSSCAL (Vera *et al.*, 2007) and HA (Žilinskas and Žilinskas, 2008). GENSSCAL applies a multivariate randomly alternating simulated annealing procedure with permutation and translation phases. HA is a hybrid algorithm of evolutionary combinatorial search and convex continuous quadratic programming. Proposers of GENSSCAL and HA have shown that their algorithms perform better than a heuristic algorithm based on simulated annealing for two-dimensional city-block scaling (Brusco, 2001) and a distance smoothing approach (Groenen *et al.*, 1998). Through a comprehensive computational study we will show that using the same computational resources and CPU time, the solutions obtained by MEAL are better than those obtained by GENSSCAL and HA. Furthermore, with a suitable parameter setting MEAL is able to obtain the global optimum with a high probability.

The paper is organized as follows: MDS with the city-block distances based on quadratic programming is presented in Section 2. The metaheuristic algorithm for multidimensional scaling is described in Section 3. A computational study to analyze the

performance of the different algorithms is presented in Section 4. The paper ends with some conclusions and directions for future research in Section 5.

2. Multidimensional Scaling with City-Block Distances Based on Quadratic Programming

Multidimensional scaling aims at visualization of multidimensional data which cannot be visualized directly. A set of n objects whose pairwise dissimilarities are represented by an $(n \times n)$ matrix (δ_{ij}), $i, j = 1, \ldots, n$, is considered. It is supposed that dissimilarities are non-negative: $\delta_{ij} \geqslant 0$, symmetric: $\delta_{ij} = \delta_{ij}$, and $\delta_{ii} = 0$. A set of points $\mathbf{x}_i \in \mathbb{R}^m$, $i = 1, \ldots, n$ in an m-dimensional embedding metric space is considered as an image of the set of objects. Ideally, we want to find the points whose inter-point distances are equal to the given dissimilarities. However such points usually do not exist. Therefore, we minimize a criterion which measures the differences between the distances and the given dissimilarities. The least squares *Stress* function is used most frequently:

$$S(\mathbf{x}) = \sum_{i<j}^{n} w_{ij} \big(d(\mathbf{x}_i, \mathbf{x}_j) - \delta_{ij}\big)^2, \tag{1}$$

where $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ represents the set of points, $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})$; it is supposed that the weights are positive: $w_{ij} > 0$, $i, j = 1, \ldots, n$; $d(\mathbf{x}_i, \mathbf{x}_j)$ denotes the distance between the points $\mathbf{x}_i$ and $\mathbf{x}_j$. In this work, we consider the city-block distances

$$d_1(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{m} |x_{ik} - x_{jk}|.$$

Therefore, *Stress* function (1) with the city-block distances can be redefined as

$$S(\mathbf{x}) = \sum_{i<j}^{n} w_{ij} \left(\sum_{k=1}^{m} |x_{ik} - x_{jk}| - \delta_{ij} \right)^2. \tag{2}$$

Stress function with city-block distances can be non-differentiable at a minimum point when $m \geqslant 2$ (Žilinskas and Žilinskas, 2007). This does not happen with other Minkowski distances, where positiveness of the distances $d(\mathbf{x}_i^*, \mathbf{x}_j^*)$ at a minimum point $\mathbf{x}^*$ (when $w_{ij}\delta_{ij} > 0$ for $i, j = 1, \ldots, n$, $i \neq j$) implies differentiability of *Stress* (de Leeuw, 1984; Groenen *et al.*, 1995). *Stress* with Minkowski distances is not differentiable where at least one distance is zero. However, this can be taken into account when choosing starting points for local optimization, but it can be ignored later during local optimization. Non-differentiability of the objective function at a minimum point cannot be ignored.

We use a special local optimization algorithm for MDS with city-block distances. The algorithm exploits that *Stress* function with city-block distances is piecewise quadratic.

Let $A(\mathbf{P})$ be a set such that

$$A(\mathbf{P}) = \big\{\mathbf{x} | x_{ik} \leqslant x_{jk} \quad \text{for } p_{ki} < p_{kj},\ i,j = 1,\ldots,n,\ k = 1,\ldots,m\big\},$$

where $\mathbf{P} = (\mathbf{p}_1,\ldots,\mathbf{p}_m)$, $\mathbf{p}_k = (p_{k1}, p_{k2},\ldots,p_{kn})$ is a permutation of $1,\ldots,n$; $k = 1,\ldots,m$. For $\mathbf{x} \in A(\mathbf{P})$, (2) can be rewritten in the following form

$$S(\mathbf{x}) = \sum_{i<j}^{n} w_{ij}\left(\sum_{k=1}^{m}(x_{ik} - x_{jk})z_{kij} - \delta_{ij}\right)^2,$$

where $z_{kij} = 1$ if $p_{ki} > p_{kj}$ and $z_{kij} = -1$ if $p_{ki} < p_{kj}$, and therefore z_{kij} is constant for $\mathbf{x} \in A(\mathbf{P})$. The function can be rewritten as

$$\begin{aligned}
S(\mathbf{x}) &= \sum_{i<j}^{n} w_{ij}\left(\sum_{k=1}^{m}(x_{ik} - x_{jk})z_{kij} - \delta_{ij}\right)^2 \\
&= \sum_{i<j}^{n} w_{ij}\left(\sum_{k=1}^{m}(x_{ik} - x_{jk})z_{kij}\right)^2 \\
&\quad - 2\sum_{i<j}^{n} w_{ij}\delta_{ij}\sum_{k=1}^{m}(x_{ik} - x_{jk})z_{kij} + \sum_{i<j}^{n} w_{ij}\delta_{ij}^2 \\
&= \sum_{k=1}^{m}\sum_{l=1}^{m}\sum_{i=1}^{n} x_{ik}x_{il}\sum_{t=1,t\neq i}^{n} w_{it}z_{kit}z_{lit} \\
&\quad - \sum_{k=1}^{m}\sum_{l=1}^{m}\sum_{i=1}^{n}\sum_{j=1,\ j\neq i}^{n} x_{ik}x_{jl}w_{ij}z_{kij}z_{lij} \\
&\quad - 2\sum_{i=1}^{n}\sum_{k=1}^{m} x_{ik}\sum_{j=1,\ j\neq i}^{n} w_{ij}\delta_{ij}z_{kij} + \sum_{i<j}^{n} w_{ij}\delta_{ij}^2,
\end{aligned}$$

where we can see quadratic, linear and constant parts of the function. Since the function $S(\mathbf{x})$ is quadratic over polyhedron $\mathbf{x} \in A(\mathbf{P})$, the optimization problem

$$\min_{\mathbf{x}\in A(\mathbf{P})} S(\mathbf{x}) \tag{3}$$

can be reduced to the quadratic programming problem (Žilinskas and Žilinskas, 2008)

$$\min\left(-\mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{D}\mathbf{x}\right) \tag{4}$$

$$\text{s.t. } \mathbf{A}^0\mathbf{x} = 0, \tag{5}$$

$$\mathbf{A}^k\mathbf{x} \geqslant 0, \quad k = 1,\ldots,m, \tag{6}$$

where $\mathbf{d}$, $\mathbf{D}$, $\mathbf{A}$ and $\mathbf{A}^k$ are computed from $\delta_{ij},\ w_{ij},\ i,j = 1,\ldots,n$ and $\mathbf{P}$. *Stress* function (2) is invariant with respect to translation and mirroring. The equality constraints (5)

Algorithm 1. Local optimization algorithm for MDS with city-block distances based on quadratic programming

Input: n; m; $\delta_{ij},\ w_{ij},\ i,j=1,\ldots,n$; $\mathbf{x}_{\text{init}}$
Output: S^*, $\mathbf{x}^*$

$S^* \leftarrow \infty$
Compute $\mathbf{P}$ representing $\mathbf{x}_{\text{init}}$
Compute $\mathbf{d}$, $\mathbf{D}$, $\mathbf{A}^0$ and $\mathbf{A}^k$ from n; m; $\delta_{ij},\ w_{ij},\ i,j=1,\ldots,n$ and $\mathbf{P}$
$\mathbf{x}^* = \arg\min(-\mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{D}\mathbf{x}),$ s.t. $\mathbf{A}^0\mathbf{x} = 0,\ \mathbf{A}^k\mathbf{x} \geqslant 0,\ k=1,\ldots,m.$
while $S(\mathbf{x}^*) < S^*$ **and** $\exists k,l\ \mathbf{A}_l^k\mathbf{x}^* = 0$ **do**
  $S^* \leftarrow S(\mathbf{x}^*)$
  for $k=1,\ldots,m$ **do**
    for all blocks of consequent active constraints $A_l^k\mathbf{x}^* = 0,\ i \leqslant l \leqslant j$ **do**
      for $t=1,\ldots,n$ **do**
        if $i \leqslant p_{kt} \leqslant j+1$ **then**
          $p_{kt} \leftarrow i+j+1-p_{kt}$
        end if
      end for
    end for
  end for
  Compute $\mathbf{d}$, $\mathbf{D}$, $\mathbf{A}^0$ and $\mathbf{A}^k$ from n; m; $\delta_{ij},\ w_{ij},\ i,j=1,\ldots,n$ and $\mathbf{P}$
  $\mathbf{x}^* = \arg\min(-\mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{D}\mathbf{x}),$ s.t. $\mathbf{A}^0\mathbf{x} = 0,\ \mathbf{A}^k\mathbf{x} \geqslant 0,\ k=1,\ldots,m.$
end while
for $k=1,\ldots,m$ **do**
  if $x_{1k} > x_{2k}$ **then**
    for $i=1,\ldots,n$ **do**
      $x_{ik} \leftarrow -x_{ik}$
    end for
  end if
end for
$S^* \leftarrow S(\mathbf{x}^*)$

ensure centering of the image to avoid translated solutions. Polyhedron $A(\mathbf{P})$ is defined by the linear inequality constraints (6). Any convex quadratic programming method can be applied to solve the problems (4)–(6).

A local optimization algorithm for MDS with city-block distances based on quadratic programming is shown in Algorithm 1. A minimum point of a quadratic programming problem is not necessarily a local minimum point of the initial problem of minimization of *Stress* function (2). This is because *Stress* function is minimized with respect to $\mathbf{P}$ as well. If a minimum point of a quadratic programming problem is on the border of a polyhedron $A(\mathbf{P})$, a local minimum point of *Stress* function is possibly located in a neighboring polyhedron. Therefore, minimization is continued by solving a quadratic programming problem over the polyhedron on the opposite side of the active inequal-

ity constraints. The permutations in $\mathbf{P}$ are updated to define the neighboring polyhedron (lines 7–15 in Algorithm 1). If $i, \ldots, j$ inequality constraints $A^k\mathbf{x} \geqslant 0$ are active, $p_{kt} : \ i \leqslant p_{kt} \leqslant j+1$ are updated to $i+j+1-p_{kt}$. Quadratic programming is repeated while better values are found, and some inequality constraints are active (lines 5–18 in Algorithm 1).

In evolutionary optimization it is important to avoid invariant minimum points since they may be treated as different individuals although they may be exactly the same image just translated or mirrored in the embedding image space. To avoid different minimum points, invariant with respect to mirroring in the embedding space, we change the resulting local minimum point if $x_{1k} > x_{2k}$ so that $x_{1k} \leqslant x_{2k}$ (lines 19–25 in Algorithm 1).

3. Multimodal Evolutionary Algorithm for Multidimensional Scaling

Multimodal evolutionary algorithm (MEAL) uses terms from genetics and evolutionary theory. A candidate solution (with some region around) is called an *individual*. An individual can be thought of as a hyper-spherical sub-region of the search domain. It is defined by the center (c_i) and the radius (R_i) (see Fig. 1). The center represents a local minimum point and the radius is a positive number which defines the sub-space of the search domain. Such a definition of an individual has been borrowed from UEGO algorithm (Jelásity *et al.*, 2001) which has been successfully used in a wide range of applications (González-Linares *et al.*, 2000; Redondo *et al.*, 2004, 2009, 2012). A particular individual is not fixed, it can move and shrink as optimization proceeds. The entire set of individuals is called a *population* (*pop*). Basically, MEAL is a method of managing such a population ensuring the exploration of the search space. Local optimization helps quick identification of "good" areas in the search space.

In MEAL, every individual is intended to occupy a local minimum point of the objective function. Although the total number of local minima points in the search domain is not known in advance, the user should specify a maximum number of individuals (*maxPop*) allowed.

Initially, a population of a single, randomly generated individual is created. Later, an iterative process is carried out. The number of iterations is given by another input parameter *maxIter*.

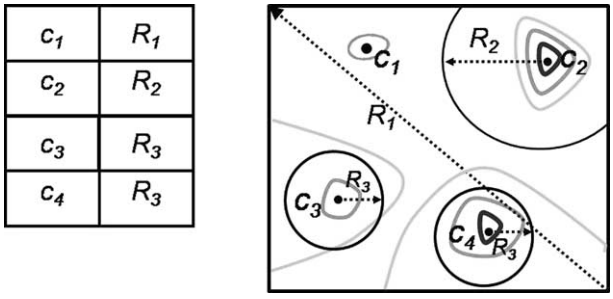

Fig. 1. Concept of both individual and population.

At each iteration, a new offspring is generated through evolutionary operators. A single individual can create a new sub-population without participation of the remaining ones. The reproduction operators are applied inside the sub-region defined by its radius. This means that exploration is carried out in the area defined by the radius.

Every time a new individual is created, it is associated with a radius whose value depends on the current iteration. Individuals created at later iterations have a smaller radius, which means that they examine a relatively smaller area of the search domain. Therefore, they are able to differentiate between local minima points that are relatively close to each other. As the algorithm progresses, the optimization process is directed towards smaller regions by creating new sets of individuals defining smaller sub-domains. It is important to mention that individuals with different radii can coexist in the population.

Additionally, an improvement method is deployed. Every created individual is improved by a local optimization algorithm. If a new point with a better objective function value is found during local optimization, then the new point becomes the center of the individual. As a consequence, the individuals move towards the locations of better local minima points.

The individuals compete among themselves to belong to the population at the next iteration. The fitness (objective function value) is used to determine the relative merit of each individual.

3.1. *Input Parameters*

MEAL has three user given parameters:

- *maxIter*: the maximum number of iterations of the algorithm.
- *maxPop*: the maximum allowed population size.
- R_{mI}: the radius associated with the last iteration of the algorithm.

The radius of an individual created at iteration i, is given by the following exponential function:

$$R_i = R_1 \left(\frac{R_{mI}}{R_1} \right)^{\frac{i-1}{maxIter-1}}, \quad i = 2, \ldots, \mathit{maxIter},$$

where R_1 is the diameter of the search domain (the largest radius) and R_{mI} is the smallest radius given by the input parameter. The maximum number of function evaluations allowed when creating new individuals in each iteration is $6 \cdot \mathit{maxPop}$. The budget of function evaluations allowed per current individual is $bc_i = 6 \cdot \mathit{maxPop}/\mathit{length}(pop_i)$.

3.2. *Stages of the Algorithm*

The structure of MEAL is given in Algorithm 2. The key stages of the algorithm are as follows:

- *Init_population:* In the beginning a population list containing one individual is created. Local optimization is applied from a random point in the search domain.

Algorithm 2. Algorithm MEAL

```
1: Init_population
2: for i = 2, ..., maxIter do
3:    Generate_offspring(R_i, bc_i)
4:    Select_new_population(R_i, maxPop)
5: end for
```

The found local minimum point defines the center of the initial individual. The diameter of the search domain defines the radius of the initial individual.

- *Generate_offspring(R_i, bc_i)*: In terms of evolutionary computation, the procedure can be interpreted as creation of offspring. An individual generates new individuals that are within its sub-region, though later on new individuals can move away.
  For every individual in the population, random trial points are generated in the area defined by its center and radius. For every pair of trial points, the middle of the *segment* connecting the pair is computed. Local optimization algorithm is then applied to all trial points. As a consequence, points can move towards local minima points. If a better fitness value is found than that of the current individual, the center of the individual is set to the better point. The same radius of the current individual is kept.
  The minimized members of the pair are inserted into the population list if the fitness value at the minimized midpoint is worse than that at the corresponding minimized members (see Fig. 2, left). On the contrary, if the minimized midpoint is better than the extreme points, then it will be included into the population (see Fig. 2, right). Every new inserted individual is assigned the current radius value (R_i). As a result of this procedure the population list eventually contains several individuals with different radii.
  Every individual in the population list has a fixed number of evaluations for the creation of new points bc_i. Notice that each individual is allowed to generate more trial points when the number of individuals in the population list $length(pop_i)$ is small. The number of trial points is smaller when the population list contains more individuals.

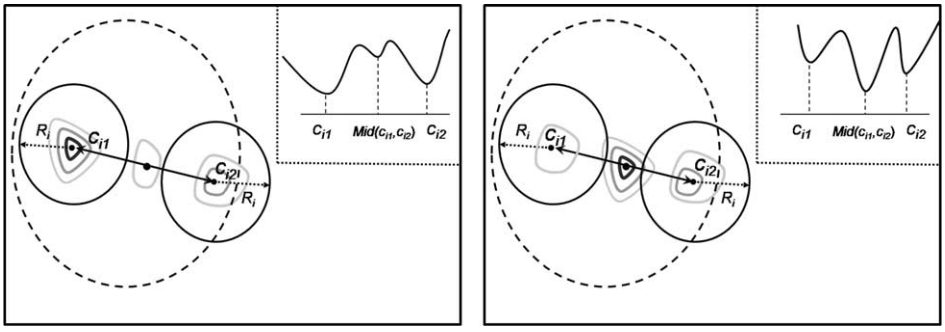

Fig. 2. Individual generation procedure.

Algorithm 3. Algorithm Select_new_population(R_i, *maxPop*)

1: Merge_individuals(R_i)
2: *radius* $= R_i$
3: **while** *length(pop)* $>$ *maxPop* **do**
4: *radius* $:=$ *radius* $* 2$
5: Merge_individuals(*radius*)
6: **end while**

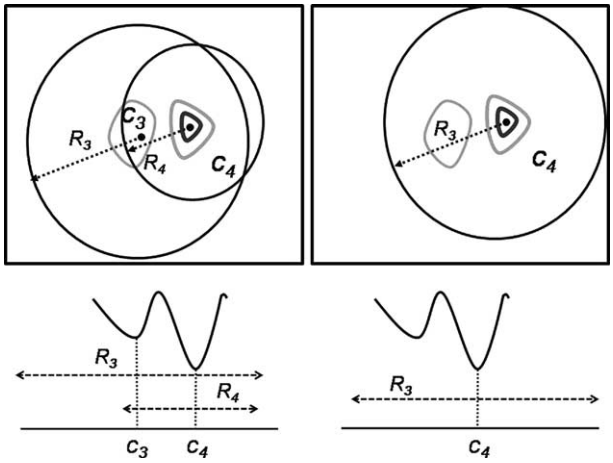

Fig. 3. Merging procedure.

- *Select_new_population(R_i, maxPop):* During each successive generation, a proportion of the current population is selected to breed a new generation. Individuals are selected through a fitness-based process, where fitter individuals (as measured by a fitness function) are selected. However, although an individual may be weak in terms of its fitness function value, it may include some components, i.e., the radius, which could be useful later in the optimization process. In our particular selection procedure, information about weaker solutions can also survive and be considered in the next stage of the algorithm.
  Initially, a merging procedure is carried out: if the centers of any pair of individuals from the population list are closer to each other than R_i, the two individuals are merged. The center with the better function value is chosen as the center of the new individual. The larger radius of the original individuals is retained (see Fig. 3).
  Nevertheless, in spite of the execution of the merging procedure, the length of the population can be larger than the maximum allowed, i.e., *maxPop*. In such a case, the number of individuals has to be reduced. To this aim, the merging procedure is repeated with a progressively increasing radius (see Algorithm 3). Note that the points with the best objective function values are maintained as the centers and larger radii are preserved. There always exists an individual with the radius equal to the diameter of the search domain. This helps us to keep a wide population

diversity and prevent premature convergence to poor solutions.

4. Computational Experiments

MEAL and HA (Žilinskas and Žilinskas, 2008) have been implemented in C++, and g++−4.3 has been used for compilation. The experiments using these two algorithms have been performed on a LINUX computer with AMD Athlon(tm) 64 X2 Dual Core Processor 4200+, 2200 MHz, 2 GB RAM. Only Windows executable of the algorithm GENSSCAL (Vera *et al.*, 2007) was available to us. Therefore, the experiments with this algorithm have been performed on a Windows Vista computer with AMD Athlon(tm) 64 X2 Dual Core Processor 5000+, 2600 MHz, 4 GB RAM. Both computers are quite similar, although the latter is around 15% faster. This is acceptable since the proposed algorithm runs on a slower computer and is in a less favorable situation in comparison.

4.1. *Test Problems*

In order to have an overall view of the performance of the algorithms, several data-sets of multidimensional data have been used in the experiments.

Some data-sets can be obtained from well understood multidimensional geometric objects: vertices of simplices and hyper-cubes. Data of various dimensionality can be generated. In the case of simplices dimensionality of multidimensional data is by one smaller than the number of vertices. One class of data-sets corresponds to the vertices of the standard simplex. The distances between any two vertices of the standard simplex are equal in any norm. Such data-sets are referred to as 'r8', 'r12', 'r16' and 'r20', where the number indicates the number of vertices. Another class of data-sets corresponds to the city-block distances between the vertices of the unit simplices. One vertex of the unit simplex is at the origin and the others are at the unit distance from the origin in each coordinate direction. Such data-sets are referred to as 'u8', 'u12', 'u16' and 'u20', where the number represents the number of multidimensional objects – vertices of the unit simplex. Finally, the city-block distances between the vertices of the multidimensional unit cubes are called 'c8' and 'c16', where the number represents the number of vertices. Therefore, 'c8' is the usual three-dimensional cube and 'c16' is the four-dimensional one.

Another class of data-sets are error-perturbed distance data proposed by Groenen *et al.* (1999). The data is generated using uniformly distributed random points in m-dimensional space, whose pairwise dissimilarities are computed by

$$\delta_{ij} = \sum_{k=1}^{m} \left| x_{ik}^{(e)} - x_{jk}^{(e)} \right|,$$

where $x_{ik}^{(e)} = x_{ik} + N(0, e(x_{ik}))$, and $N(0, e)$ denotes the normally distributed random variable with zero mean and standard deviation e. Eight problems defined by all combinations of the parameters ($n = 10, 20$; $m = 2, 3$; $e(x) = 0.15x, 0.3x$) have been

generated and are referred to as 'gABC' in the results below. A represents m, B represents n ($B = 1$ means $n = 10$ and $B = 2$ means $n = 20$), and C represents e ($C = 1$ means $e(x) = 0.15x$ and $C = 3$ means $e(x) = 0.3x$).

Another class of empirical data-sets is obtained from pharmacological binding affinity data (Žilinskas, 2006). The binding affinity data is represented through a matrix, one dimension formed by different ligands tested in a series of experiments while the other dimension represents different proteins. Heuristic analysis can be performed visualizing data as properties of proteins or ligands. Dissimilarities of proteins are computed as the city-block distances between vectors of the $\log_{10}$-transformed binding affinities representing proteins. Dissimilarities of ligands are computed as the city-block distances between vectors of the $\log_{10}$-transformed binding affinities representing ligands. We refer to dissimilarities of three human and five zebrafish α_2-adrenoceptor proteins obtained from binding affinity data of (Ruuskanen *et al.*, 2005) as 'ru1' ($n = 8$) and dissimilarities of $n = 20$ ligands obtained from the same binding affinity data as 'ru2'. Dissimilarities of human, rat, guinea pig and pig α_2-adrenoceptor proteins obtained from binding affinity data of (Uhlén *et al.*, 1998) are referred to as 'uh1' ($n = 12$). Dissimilarities of ligands obtained from binding affinity data of (Hwa *et al.*, 1995) are referred to as 'hw12' ($n = 9$) while dissimilarities of wild type and mutant proteins obtained from the data of (Hwa *et al.*, 1995) are referred to as 'hw21' ($n = 12$).

A frequently used test problem for MDS algorithms is based on experimental testing of $n = 10$ soft drinks (Green *et al.*, 1989), where dissimilarities are measured by means of psychological experiments. This problem is referred to as 'cola'. The thirteen ($n = 13$) ethnic sub-groups data (Funk *et al.*, 1974) is referred to as 'funk'.

To check the reliability of different algorithms, it is interesting to have test problems with the known global minima. Algorithms are usually compared on the best objective function value they have found. However the minimum of $S(\mathbf{x})$ depends on the problem (n and $\delta_{ij},\ i, j = 1, \ldots, n$) and is not very convenient when comparing accuracies of scaling for different sets of objects. To reduce this undesirable impact, a relative error (normalized *Stress* function)

$$f(\mathbf{x}) = \sqrt{S(\mathbf{x}) \Big/ \sum_{i<j}^{n} w_{ij}\delta_{ij}^2}$$

is used when presenting the results. In Table 1, the best known values of relative error for the considered test problems are presented. Bold font numbers indicate the exact global minima found by explicit enumeration with avoidance of symmetries (Žilinskas, 2007) for vertices of simplices ('u8', 'u12', 'r8', 'r12') and by a branch and bound algorithm (Žilinskas and Žilinskas, 2009a) for vertices of cubes ('c8') and empirical data-sets ('g211', 'g213', 'ru1', 'hw12', 'hw21', 'cola'). Of course, the exact solution requires enormous computational resources and it is not possible for larger problems. A parallel branch and bound algorithm solves 'hw21' with $m = 2$ in 10 hours and 'hw12' with $m = 3$ in more than 60 hours on 16 processors (Žilinskas, 2012).

Table 1
The best known values of relative error for the considered test problems

Data	n	$\min f$, $m = 2$	$\min f$, $m = 3$
u8	8	**0.2569**	**0.0992**
u12	12	**0.3167**	0.1874
u16	16	0.3439	0.2243
u20	20	0.3595	0.2455
r8	8	**0.2825**	**0.1250**
r12	12	**0.3300**	0.2013
r16	16	0.3525	0.2321
r20	20	0.3657	0.2505
c8	8	**0.2245**	**0.0000**
c16	16	0.2965	0.1590
gm11	10	**0.1293**	0.0906
gm13	10	**0.2711**	0.1298
gm21	20	0.1868	0.1610
gm23	20	0.2966	0.2284
ru1	8	**0.1096**	**0.0188**
hw12	9	**0.0000**	**0.0000**
uh1	12	**0.0825**	0.0356
hw21	12	**0.0497**	0.0183
ru2	20	0.0518	0.0243
cola	10	**0.1675**	0.0676
funk	13	0.2275	0.1074

4.2. *Comparing the Reliability and Performance of the Algorithms for Short Runs*

In this section, an exhaustive computational study has been carried out to compare the robustness and efficiency of GENSSCAL, HA and MEAL algorithms.

The set of problems described in Section 4.1 have been solved with all the heuristics, i.e., GENSSCAL, HA and MEAL. The default parameters provided by the authors have been used for the executions of GENSSCAL algorithm. The average computational time of GENSSCAL has been used as a stopping criterion for HA for each problem. Notice that we want to determine which is the best algorithm for this set of problems. Hence, in order to have a fair comparison, both the reliability and performance of the algorithms have to be compared when they are allowed to run a similar (averaged across all instances) amount of time. Since each run of a randomized algorithm may provide a different solution, each algorithm was run 40 times for each problem. The input parameters of MEAL were chosen so that the CPU times were, on average (when considering the whole set of problems), similar to the CPU times used by both GENSSCAL and HA. In particular, the considered set of parameters was: *maxPop* $= 5$, *maxIter* $= 10$, $R_{mI} = 0.02$.

For each execution, we obtain the minimal relative error f^* provided by the algo-

rithm, the point at which this value is attained and the CPU time (in seconds). With this information, for a given algorithm and a given problem, we compute the average computing time $Av(t)$ (in seconds) and the minimum relative error found in 40 runs ($\min f^*$). The number of times (in percentage: $At\%$) the algorithm has found the $\min f^*$ value has also been computed. Notice that a success is considered when the difference between f^* and $\min f^*$ is less than 10^{-6}. Table 2 summarizes the results obtained by the three algorithms.

The available executable program of GENSSCAL provides the best point achieved over all runs, but the best point of each run is not printed. To be able to compare the solutions provided by the GENSSCAL program with the results obtained by both HA and MEAL algorithms, the relative error of the best point provided by the GENSSCAL program is evaluated and presented as $\min f^*$. For all runs, the program GENSSCAL provides normalized minimal function values with 8 decimal digits in each run. To find the corresponding value of the relative error f^*, the square root of normalized minimal function value is computed. These values are used to compute the percentage of runs ($At\%$) where this number f^* differs from the best f^* of 40 runs by less than 10^{-6}.

The $At\%$ values obtained by HA suggest this algorithm has a good behavior in terms of effectiveness. Nevertheless, for 20 out of 42 test problems, its $\min f^*$ values were larger than the ones provided by both GENSSCAL and MEAL. See, for example, problems 'g213' and 'cola' with $m = 2$. For these two problems, the number of times HA has found its $\min f^*$ value is relatively high, but the obtained minimum relative errors are worse than the ones provided by GENSSCAL and MEAL. However, for problems where the three algorithms have obtained the same $\min f^*$, the percentages of success reached by the HA algorithm are larger than the ones obtained by GENSSCAL, in most cases. See for instance, problems 'u8', 'u12', 'c8' with $m = 2, 3$ to name a few. On the contrary, the $At\%$ values achieved by HA are, in general, worse than those obtained by MEAL. Only for problem 'u12', 'r12', 'ru1' and 'uh1' with $m = 2$, HA is able to improve MEAL results.

Notice that GENSSCAL and HA invest the same computing times (although the experiments with GENSSCAL have been performed on around 15% faster computer), while computing times for MEAL are a bit different (see columns $Av(t)$ in Table 2). This is because MEAL adapts itself to the difficulty of the problem at hand, whereas HA does not take the difficulty of the problem into account. HA executes an iterative process until a CPU time is obtained (in this study, the average execution time of GENSSCAL has been considered as a stopping criterion for each problem). Nevertheless, computing times for MEAL are very close to that of other algorithms although for half of the problems they are a bit shorter while for others they are a bit longer.

MEAL is a clear leader in terms of reliability. It finds the best relative error in many cases: for 12 of 42 test problems it is able to obtain 100% success, for 22 of 42 $At\% \geqslant 50$. Only for 3 test problems ('hw21', 'ru2' and 'funk' with $m = 3$) $At\% = 2.5$, i.e., the algorithm finds the $\min f^*$ value only once in 40 runs. However, GENSSCAL provides results differing by more than 10^{-6} in most of the runs. In fact, for 28 of 42 problems it

Table 2
Comparison of algorithms: short runs

Data	MEAL			GENSSCAL			HA		
	$Av(t)$	$\min f^*$	$At\%$	$Av(t)$	$\min f^*$	$At\%$	$Av(t)$	$\min f^*$	$At\%$
	$m = 2$								
u8	0.07	0.2569	100.0	0.19	0.2569	5.0	0.19	0.2569	100.0
u12	0.23	0.3167	82.5	0.34	0.3167	7.5	0.34	0.3167	97.5
u16	0.47	0.3439	65.0	0.52	0.3439	65.0	0.52	0.3445	10.0
u20	0.94	0.3595	50.0	0.73	0.3595	60.0	0.73	0.3611	2.5
r8	0.06	0.2825	100.0	0.17	0.2825	92.5	0.17	0.2825	100.0
r12	0.19	0.3300	90.0	0.31	0.3300	60.0	0.31	0.3300	92.5
r16	0.54	0.3525	42.5	0.47	0.3525	40.0	0.47	0.3533	2.5
r20	0.76	0.3657	27.5	0.67	0.3657	12.5	0.67	0.3670	2.5
c8	0.10	0.2245	100.0	0.22	0.2245	2.5	0.22	0.2245	100.0
c16	0.92	0.2965	27.5	0.60	0.2965	2.5	0.60	0.3001	2.5
g211	0.29	0.1293	100.0	0.27	0.1293	5.0	0.27	0.1293	100.0
g213	0.22	0.2711	100.0	0.28	0.2711	2.5	0.28	0.2713	85.0
g221	1.77	0.1868	100.0	0.78	0.1868	2.5	0.78	0.1872	7.5
g223	1.38	0.2966	65.0	0.78	0.2966	2.5	0.78	0.3090	2.5
ru1	0.08	0.1096	32.5	0.27	0.1096	2.5	0.27	0.1096	85.0
hw12	0.14	0.0000	100.0	0.26	0.0001	2.5	0.26	0.0000	100.0
uh1	0.39	0.0825	55.0	0.43	0.0825	2.5	0.43	0.0825	60.0
hw21	0.41	0.0497	82.5	0.41	0.0497	2.5	0.41	0.0497	55.0
ru2	1.58	0.0518	15.0	1.09	0.0518	2.5	1.09	0.0523	2.5
cola	0.21	0.1675	22.5	0.25	0.1675	5.0	0.25	0.1681	22.5
funk	0.40	0.2275	12.5	0.36	0.2275	2.5	0.36	0.2467	2.5
	$m = 3$								
u8	0.24	0.0992	100.0	0.31	0.0992	2.5	0.31	0.0992	100.0
u12	0.76	0.1874	97.5	0.52	0.1874	2.5	0.52	0.1874	47.5
u16	0.95	0.2243	25.0	0.81	0.2243	12.5	0.81	0.2267	2.5
u20	2.65	0.2458	10.0	1.15	0.2455	2.5	1.15	0.2474	2.5
r8	0.29	0.1250	100.0	0.27	0.1250	37.5	0.27	0.1250	100.0
r12	0.45	0.2013	47.5	0.47	0.2013	30.0	0.47	0.2013	22.5
r16	1.06	0.2321	30.0	0.73	0.2321	25.0	0.73	0.2329	2.5
r20	1.30	0.2505	5.0	1.03	0.2505	2.5	1.03	0.2531	2.5
c8	0.21	0.0000	100.0	0.32	0.0001	2.5	0.32	0.0000	100.0
c16	0.60	0.1590	80.0	0.93	0.1590	2.5	0.93	0.1590	52.5
g311	0.61	0.0906	100.0	0.43	0.0906	2.5	0.43	0.0906	97.5
g313	0.64	0.1298	95.0	0.42	0.1298	2.5	0.42	0.1298	90.0
g321	2.05	0.1610	22.5	1.23	0.1610	2.5	1.23	0.1653	2.5
g323	1.95	0.2284	12.5	1.26	0.2284	2.5	1.26	0.2457	2.5
ru1	0.29	0.0188	5.0	0.41	0.0188	2.5	0.41	0.0188	15.0
hw12	0.46	0.0000	100.0	0.40	0.0001	2.5	0.40	0.0000	100.0
uh1	0.74	0.0356	27.5	0.66	0.0356	2.5	0.66	0.0356	2.5
hw21	0.52	0.0183	2.5	0.44	0.0191	2.5	0.44	0.0186	2.5
ru2	2.30	0.0255	2.5	1.67	0.0244	2.5	1.67	0.0252	2.5
cola	0.35	0.0676	7.5	0.38	0.0676	2.5	0.38	0.0771	2.5
funk	0.69	0.1074	2.5	0.55	0.1075	2.5	0.55	0.1328	2.5

obtains the $\min f^*$ value only once in 40 runs ($At\% = 2.5$). Regarding HA, $At\% = 2.5$ is obtained for 18 of 42 problems.

4.3. *Comparing the Performance of the Algorithms for Reliable Runs*

In the previous section, MEAL was tuned to obtain similar running times to the ones of HA and GENSSCAL, so as to be able to compare all three algorithms. Of course, if the set of parameters is not reliable enough, the search cannot be exhaustive and the algorithm may become trapped in attraction regions of bad local minima. This section is aimed at designing a suitable parameter setting that allows MEAL to explore the search domain deeper and to provide better quality using reasonable computing times. An analysis of the effects of the different parameters of MEAL was carried out in order to obtain a robust parameter setting. From such experiments, it was determined that, for the problem at hand, a more reliable set of the parameters for MEAL is: *maxPop* $= 50$, *maxIter* $= 10$, $R_{mI} = 0.02$.

Furthermore, the behavior of the HA algorithm is also studied when its running times are similar to those of MEAL for each problem. Notice that the comparison with GENSSCAL is not suitable in this case, since only the default parameters of the program are available.

Again, each algorithm has been executed 40 times on every test problem. Each problem has been solved first by MEAL, using the reliable set of parameters. For each problem, the average computing time $Av(t)$ (in seconds) has been computed and used as a stopping criterion for HA. For each execution, we obtain the minimal relative error f^* provided by the algorithm. For a given algorithm and a given problem, we compute the minimum relative error found in 40 runs ($\min f^*$), the average value ($\overline{f^*}$) and the standard deviation (s.d. f^*). As a criterion of reliability, we also present the percentage of runs ($At\%$) where the minimum relative error $\min f^*$ has been found. Again, success is considered when the difference between f^* and $\min f^*$ is less than 10^{-6}. The results are shown in Table 3.

It is important to mention that $\min f^*$ values obtained by both HA and MEAL coincide with the exact global minima found by exact methods (see Tables 1 and 3). As can be observed in Table 3, MEAL finds the $\min f^*$ value with 100% success in most cases (32 out of 42 test problems). In cases where 100% success is not obtained, the difference between $\min f^*$ and $\overline{f^*}$ values is very small and the standard deviation is very small too. The worst percentage of success is $At\% = 20$, which is obtained for the problem 'u20' with $m = 3$. HA obtains 100% success only for 20 of 42 test problems. Moreover, there exist two problems, 'funk' with $m = 2$ and 'r20' with $m = 3$, where the $At\%$ values are only 2.5. For problems 'g323' and 'hw21' (with $m = 3$) the percentage of success obtained by HA is slightly higher than the one achieved by MEAL. However, the average relative error ($\overline{f^*}$) and the s.d. are smaller for MEAL for these problems. On average, MEAL obtains 88% success at finding the minimum relative error, while HA achieves 77% success. The computing time is the same for both algorithms. Therefore, it is possible to conclude that MEAL is more reliable than HA.

Table 3
Comparison of algorithms: reliable runs

Data	$Av(t)$	MEAL				HA			
		min f^*	$\overline{f^*}$	s.d. f^*	$At\%$	min f^*	$\overline{f^*}$	s.d. f^*	$At\%$
	$m = 2$								
u8	3.98	0.2569	0.2569	0.0000	100.0	0.2569	0.2569	0.0000	100.0
u12	10.23	0.3167	0.3167	0.0000	100.0	0.3167	0.3167	0.0000	100.0
u16	21.28	0.3439	0.3439	0.0000	100.0	0.3439	0.3440	0.0001	92.5
u20	40.80	0.3595	0.3595	0.0000	100.0	0.3595	0.3596	0.0002	55.0
r8	4.20	0.2825	0.2825	0.0000	100.0	0.2825	0.2825	0.0000	100.0
r12	11.28	0.3300	0.3300	0.0000	100.0	0.3300	0.3300	0.0001	97.5
r16	23.38	0.3525	0.3525	0.0000	100.0	0.3525	0.3527	0.0002	60.0
r20	43.25	0.3657	0.3657	0.0000	100.0	0.3657	0.3661	0.0003	20.0
c8	3.35	0.2245	0.2245	0.0000	100.0	0.2245	0.2245	0.0000	100.0
c16	22.70	0.2965	0.2965	0.0000	100.0	0.2965	0.2966	0.0002	95.0
g211	11.45	0.1293	0.1293	0.0000	100.0	0.1293	0.1293	0.0000	100.0
g213	8.18	0.2711	0.2711	0.0000	100.0	0.2711	0.2711	0.0000	100.0
g221	59.43	0.1868	0.1868	0.0000	100.0	0.1868	0.1868	0.0000	100.0
g223	67.65	0.2966	0.2966	0.0000	100.0	0.2966	0.2966	0.0000	92.5
ru1	4.03	0.1096	0.1096	0.0000	100.0	0.1096	0.1096	0.0000	100.0
hw12	4.45	0.0000	0.0000	0.0000	100.0	0.0000	0.0000	0.0000	100.0
uh1	8.68	0.0825	0.0825	0.0000	100.0	0.0825	0.0825	0.0000	97.5
hw21	10.10	0.0497	0.0497	0.0000	100.0	0.0497	0.0497	0.0000	100.0
ru2	31.93	0.0518	0.0518	0.0000	42.5	0.0518	0.0518	0.0000	35.0
cola	6.85	0.1675	0.1676	0.0002	67.5	0.1675	0.1676	0.0002	35.0
funk	15.93	0.2275	0.2282	0.0013	75.0	0.2282	0.2327	0.0022	2.5
	$m = 3$								
u8	7.35	0.0992	0.0992	0.0000	100.0	0.0992	0.0992	0.0000	100.0
u12	27.53	0.1874	0.1874	0.0000	100.0	0.1874	0.1874	0.0000	100.0
u16	70.68	0.2243	0.2243	0.0000	100.0	0.2243	0.2244	0.0004	87.5
u20	134.03	0.2455	0.2457	0.0002	20.0	0.2455	0.2460	0.0004	15.0
r8	11.90	0.1250	0.1250	0.0000	100.0	0.1250	0.1250	0.0000	100.0
r12	40.75	0.2013	0.2013	0.0000	100.0	0.2013	0.2013	0.0000	100.0
r16	85.93	0.2321	0.2321	0.0000	100.0	0.2321	0.2322	0.0003	72.5
r20	177.00	0.2505	0.2506	0.0001	30.0	0.2505	0.2511	0.0005	2.5
c8	8.25	0.0000	0.0000	0.0000	100.0	0.0000	0.0000	0.0000	100.0
c16	68.55	0.1590	0.1590	0.0000	100.0	0.1590	0.1590	0.0000	100.0
g311	16.03	0.0906	0.0906	0.0000	100.0	0.0906	0.0906	0.0000	100.0
g313	27.15	0.1298	0.1298	0.0000	100.0	0.1298	0.1298	0.0000	100.0
g321	239.08	0.1610	0.1610	0.0000	100.0	0.1610	0.1613	0.0005	80.0
g323	268.13	0.2284	0.2285	0.0003	45.0	0.2284	0.2293	0.0023	60.0
ru1	12.35	0.0188	0.0188	0.0000	100.0	0.0188	0.0188	0.0000	100.0
hw12	15.65	0.0000	0.0000	0.0000	100.0	0.0000	0.0000	0.0000	100.0
uh1	26.85	0.0356	0.0356	0.0000	100.0	0.0356	0.0359	0.0007	82.5
hw21	41.50	0.0183	0.0183	0.0001	77.5	0.0183	0.0184	0.0002	92.5
ru2	271.80	0.0243	0.0244	0.0001	70.0	0.0243	0.0245	0.0002	55.0
cola	21.10	0.0676	0.0684	0.0010	37.5	0.0676	0.0688	0.0017	17.5
funk	57.35	0.1074	0.1085	0.0009	27.5	0.1074	0.1099	0.0021	10.0

5. Conclusions

In this paper, multidimensional scaling was considered. The essential part of this technique is optimization of an objective function with many adverse optimization properties. The objective function has many local minima and it is not differentiable everywhere. Usually, the Euclidean distances are used in the definition of the objective function. However, in the present study, the city-block distances are considered, since the obtained information could be more useful for experts. Multidimensional scaling with the city-block distances is more difficult since the objective function can be non-differentiable at a minimum point. Recently, two algorithms have been proposed to cope with this hard-to-solve optimization problem: GENSSCAL (Vera *et al.*, 2007) and HA (Žilinskas and Žilinskas, 2008). GENSSCAL is based on a multivariate randomly alternating simulated annealing procedure with permutation and translation phases. HA is a bi-level optimization algorithm which combines evolutionary global search and convex quadratic programming.

Any global optimization algorithm must be able to find the global optimum in the presence of many deceptive optima. Time should thus be spent on discovering new and promising regions rather than exploring the same region multiple times. In this study, a new algorithm with this aim is introduced: an evolutionary multimodal optimization algorithm called MEAL. A comprehensive computational study has been carried out to compare MEAL, GENSSCAL and HA. The computational studies show that, in similar CPU times, the results obtained by MEAL are better than those obtained by both GENSSCAL and HA. Furthermore, with a suitable parameter setting, it is able to obtain 100% success for the majority of the test problems (32 out of 42). When this is not the case, the standard deviation (s.d. f^*) is very small, which shows that the algorithm is robust.

In the future, we plan to design a parallel version of MEAL able to obtain solutions with higher quality using less CPU time.

Acknowledgments. This research was funded by a grant (No. MIP-063/2012) from the Research Council of Lithuania. This work has been funded by grants from the Spanish Ministry of Science and Innovation (TIN2008-01117), Program CEI from MICINN (PYR-2012-15 CEI BioTIC GENIL, CEB09-0010) and the Junta de Andalucía (P08-TIC-3518, P10-TIC-6002), in part financed by the European Regional Development Fund (ERDF). The authors thank COST Action "Open European Network for High Performance Computing on Complex Environments" IC0805 for enabling beginning of cooperation between the research groups at the University of Almería and Vilnius University Institute of Mathematics and Informatics. We thank J. Fernando Vera for providing the executable program of the GENSSCAL algorithm.

References

Arabie, P. (1991). Was Euclid an unnecessarily sophisticated psychologist? *Psychometrika*, 56(4), 567–587, doi:10.1007/BF02294491.

Borg, I., Groenen, P.J.F. (2005). *Modern Multidimensional Scaling: Theory and Applications*, 2nd edn., Springer, New York.

Brusco, M.J. (2001). A simulated annealing heuristic for unidimensional and multidimensional (city-block) scaling of symmetric proximity matrices. *Journal of Classification*, 18(1), 3–33, doi:10.1007/s00357-001-0003-4.

de Leeuw, J. (1984). Differentiability of Kruskal's stress at a local minimum. *Psychometrika*, 49(1), 111–113, doi:10.1007/BF02294209.

DeSarbo, W.S., Kim, Y., Wedel, M., Fong, D.K.H. (1998). A Bayesian approach to the spatial representation of market structure from consumer choice data. *European Journal of Operational Research*, 111(2), 285–305, doi:10.1016/S0377-2217(98)00150-7.

Dzemyda, G., Kurasova, O., Žilinskas, J. (2013). *Multidimensional Data Visualization: Methods and Applications*. Springer, New York, doi:10.1007/978-1-4419-0236-8.

Everett, J.E. (2001). Algorithms for multidimensional scaling. In: Chambers, L.D. (Ed.), *The Practical Handbook of Genetic Algorithms*, 2nd edn., Chapman & Hall/CRC, pp. 203–233.

Funk, S.G., Horowitz, A.D., Lipshitz, R., Young, F.W. (1974). The perceived structure of American ethnic groups: the use of multidimensional scaling in stereotype research. *Personality and Social Psychology Bulletin*, 1(1), 66–68, doi:10.1177/014616727400100122.

González-Linares, J., Guil, N., Zapata, E., Ortigosa, P., García, I. (2000). Deformable shapes detection by stochastic optimization. In: *2000 IEEE International Conference on Image Processing (ICIP'2000)*, Vancouver, Canada.

Green, P., Carmone, F., Smith, S. (1989). *Multidimensional Scaling: Concepts and Applications*. Allyn and Bacon, Boston.

Groenen, P.J.F., Mathar, R., Heiser, W.J. (1995). The majorization approach to multidimensional scaling for Minkowski distances. *Journal of Classification*, 12(1), 3–19, doi:10.1007/BF01202265.

Groenen, P.J.F., Heiser, W.J., Meulman, J.J. (1998). City-block scaling: smoothing strategies for avoiding local minima. In: Balderjahn, I., Mathar, R., Schader, M. (Eds.), *Classification, Data Analysis, and Data Highways*. Springer, Berlin, pp. 46–53.

Groenen, P.J.F., Heiser, W.J., Meulman, J.J. (1999). Global optimization in least-squares multidimensional scaling by distance smoothing. *Journal of Classification*, 16(2), 225–254, doi:10.1007/s003579900055.

Groenen, P., Mathar, R., Trejos, J. (2000). Global optimization methods for multidimensional scaling applied to mobile communication. In: Gaul, W., Opitz, O., Schander, M. (Eds.), *Data Analysis: Scientific Modeling and Practical Applications*. Springer, pp. 459–475.

Hwa, J., Graham, R.M., Perez, D.M. (1995). Identification of critical determinants of α_1-adrenergic receptor subtype selective agonist binding. *Journal of Biological Chemistry*, 270(39), 23189–23195, doi:10.1074/jbc.270.39.23189.

Jelásity, M., Ortigosa, P., García, I. (2001). UEGO, an abstract clustering technique for multimodal global optimization. *Journal of Heuristics*, 7(3), 215–233.

Mathar, R. (1996). A hybrid global optimization algorithm for multidimensional scaling. In: *Classification and Knowledge Optimization, Proceedings of the 20th Annual Conference of the Gesellschaft für Klassification e.V.* University of Freiburg, pp. 63–71.

Mathar, R., Žilinskas, A. (1993). On global optimization in two-dimensional scaling. *Acta Applicandae Mathematicae*, 33(1), 109–118, doi:10.1007/BF00995497.

Nelson, T.R., Rabianski, J. (1988). Consumer preferences in housing market analysis: an application of multidimensional scaling techniques. *Real Estate Economics*, 16(2), 138–159, doi:10.1111/1540-6229.00451.

Redondo, J.L., Ortigosa, P.M., García, I., Fernández, J.J. (2004). Image registration in electron microscopy. A stochastic optimization approach. In: *Proceedings of the International Conference on Image Analysis and Recognition (ICIAR 2004), Lecture Notes in Computer Science*, 3212(II), pp. 141–149.

Redondo, J.L., Fernández, J., García, I., Ortigosa, P.M. (2009). Solving the multiple competitive location and design problem on the plane. *Evolutionary Computation*, 17(1), 21–53.

Redondo, J.L., Fernández, J., Arrondo, A.G., García, I., Ortigosa, P.M. (2012). Fixed or variable demand? Does it matter when locating a facility? *OMEGA-International Journal of Management Science*, 40(1), 9–12, doi:10.1016/j.omega.2011.02.007.

Ruuskanen, J.O., Laurila, J., Xhaard, H., Rantanen, V.V., Vuoriluoto, K., Wurster, S., Marjamäki, A., Vainio, M., Johnson, M.S., Scheinin, M. (2005). Conserved structural, pharmacological and functional properties among the three human and five zebrafish α_2-adrenoceptors. *British Journal of Pharmacology*, 144(2), 165–177, doi:10.1038/sj.bjp.0706057.

Speer, N., Spieth, C., Zell, A. (2004). A memetic co-clustering algorithm for gene expression profiles and biological annotation. In: *Proceedings of the IEEE 2004 Congress on Evolutionary Computation (CEC 2004)*, Vol. 2. IEEE Press, pp. 1631–1638.

Takane, Y. (2006). Applications of multidimensional scaling in psychometrics. *Handbook of Statistics*, 26, 359–400, doi:10.1016/S0169-7161(06)26011-5.

Uhlén, S., Dambrova, M., Näsman, J., Schiöth, H.B., Gu, Y., Wikberg-Matsson, A., Wikberg, J.E.S. (1998). [^{3}H]RS79948-197 binding to human, rat, guinea pig and pig α_{2A}-, α_{2B}- and α_{2C}-adrenoceptors: comparison with MK912, RX821002, rauwolscine and yohimbine. *European Journal of Pharmacology*, 343(1), 93–101, doi:10.1016/S0014-2999(97)01521-5.

Vera, J.F., Heiser, W.J., Murillo, A. (2007). Global optimization in any Minkowski metric: a permutation-translation simulated annealing algorithm for multidimensional scaling. *Journal of Classification*, 24(2), 277–301, doi:10.1007/s00357-007-0020-1.

Žilinskas, A., Žilinskas, J. (2006). Parallel hybrid algorithm for global optimization of problems occurring in MDS based visualization. *Computers & Mathematics with Applications*, 52(1–2), 211–224, doi:10.1016/j.camwa.2006.08.016.

Žilinskas, A., Žilinskas, J. (2007). Two level minimization in multidimensional scaling. *Journal of Global Optimization*, 38(4), 581–596, doi:10.1007/s10898-006-9097-x.

Žilinskas, A., Žilinskas, J. (2008). A hybrid method for multidimensional scaling using city-block distances. *Mathematical Methods of Operations Research*, 68(3), 429–443, doi:10.1007/s00186-008-0238-5.

Žilinskas, A., Žilinskas, J. (2009a). Branch and bound algorithm for multidimensional scaling with city-block metric. *Journal of Global Optimization*, 43(2–3), 357–372, doi:10.1007/s10898-008-9306-x.

Žilinskas, A., Žilinskas, J. (2009b). Optimization-based visualization. In: Floudas, C.A., Pardalos, P.M. (Eds.), *Encyclopedia of Optimization*, 2nd edn., Springer, pp. 2785–2791, doi:10.1007/978-0-387-74759-0_478.

Žilinskas, J. (2006). Multidimensional scaling in protein and pharmacological sciences. In: Bogle, I.D.L., Žilinskas, J. (Eds.), *Computer Aided Methods in Optimal Design and Operations*. World Scientific, Singapore, pp. 139–148, doi:10.1142/9789812772954_0015.

Žilinskas, J. (2007). Reducing of search space of multidimensional scaling problems with data exposing symmetries. *Information Technology and Control*, 36(4), 377–382.

Žilinskas, J. (2012). Parallel branch and bound for multidimensional scaling with city-block distances. *Journal of Global Optimization*, 54(2), 261–274, doi:10.1007/s10898-008-9624-7.

J.L. Redondo is a fellow of the Spanish 'Juan de la Cierva' contract program. Her research interests include global optimization, high performance computing and location science.

P.M. Ortigosa received the diplomas (MSc) in physics and in electronic engineering from the University of Granada, Spain, in 1994 and 1996, respectively. She received the PhD degree in computer science from the University of Málaga, Spain, in 1999. She is currently a professor of computer architectures at the University of Almería, Spain. Since 2007, she began with the management at university level as director of Research Management Unit and as a member of the Research Commission of the University of Almería. In 2009 she was elected acting director of the Centre for Research on Information Technologies and Communication (CITIC) of the University of Almería. She is a principal researcher of the research group "Supercomputation. Algorithms" at the University of Almería. Her current research interests are in the fields of high performance computing and global optimization algorithms, focusing on evolutionary optimization strategies and competitive location problems. She is a reviewer of several scientific journals as *Journal of Global Optimization*, *IEEE Transactions on Evolutionary Computation*, *Evolutionary Computation*, *IEEE Transactions on Systems*, *Man and Cybernetics*, *Optimization Letters*, and *Journal of Heuristics*.

J. Žilinskas is a principal researcher and the head of Recognition Processes Department at Vilnius University Institute of Mathematics and Informatics, Lithuania. His research interests include global optimization, parallel computing, data analysis and visualization. He is a member of editorial boards of *Central European Journal of Computer Science*, *Central European Journal of Engineering*, *Informatica*, *Journal of Global Optimization*, *Mathematical Modelling and Analysis*, and *Optimization Letters*.

Daugiamodalus evoliucinis algoritmas daugiamatėms skalėms su miesto kvartalo atstumais

Juana López REDONDO, Pilar Martínez ORTIGOSA, Julius ŽILINSKAS

Šiame straipsnyje nagrinėjamos daugiamatės skalės su miesto kvartalo atstumais. Šiam metodui realizuoti reikalingas tikslo funkcijos su daug lokaliųjų minimumų taškų ir galimu nediferencijuojamumu minimumo taškuose optimizavimas. Tyrimo tikslas yra sukurti greitą ir efektyvų globaliojo optimizavimo algoritmą, peržvelgiantį visą paieškos sritį ir randantį gerus sprendinius. Siekiant išvengti stagnacijos bloguose lokaliuosiuose minimuose, daugiamodalinis evoliucinis algoritmas yra naudojamas globaliajam optimizavimui. Dalimis kvadratinė mažiausiųjų kvadratų tikslo funkcijos su miesto kvartalo atstumais struktūra yra išnaudojama lokaliajam pagerinimui. Pasiūlytas algoritmas yra palygintas su literatūroje aprašytais algoritmais. Nuodugniu skaičiuojamuoju tyrimu parodyta, kad pasiūlyto algoritmo rezultatai yra geriausi. Algoritmas su priderintomis parametrų reikšmėmis randa globalųjį minimumą su didele tikimybe.