J Glob Optim (2009) 43:357–372
DOI 10.1007/s10898-008-9306-x

Branch and bound algorithm for multidimensional scaling with city-block metric

Antanas Žilinskas · Julius Žilinskas

Received: 3 April 2007 / Accepted: 28 March 2008 / Published online: 17 April 2008

Abstract A two level global optimization algorithm for multidimensional scaling (MDS) with city-block metric is proposed. The piecewise quadratic structure of the objective function is employed. At the upper level a combinatorial global optimization problem is solved by means of branch and bound method, where an objective function is defined as the minimum of a quadratic programming problem. The later is solved at the lower level by a standard quadratic programming algorithm. The proposed algorithm has been applied for auxiliary and practical problems whose global optimization counterpart was of dimensionality up to 24.

Keywords Multidimensional scaling · City-block metric · Branch and bound

1 Introduction

Multidimensional scaling (MDS) is a technique for exploratory analysis of multidimensional data widely usable in different applications [2,6]. Pairwise dissimilarities among n objects are given by the matrix (δ_{ij}), $i, j = 1, \ldots, n$. A set of points in an embedding metric space is considered as an image of the set of objects. Normally, an m-dimensional vector space is used, and $\mathbf{x}_i \in \mathbf{R}^m$, $i = 1, \ldots, n$, should be found whose inter-point distances fit the given dissimilarities. Images of the considered objects can be found minimizing a fit criterion, e.g. the most frequently used least squares *STRESS* function:

A. Žilinskas
Department of Mathematics and Informatics, Vilnius University, Naugarduko 24, 03225 Vilnius, Lithuania
e-mail: antanasz@ktl.mii.lt

J. Žilinskas (✉)
Vilnius Gediminas Technical University, Saulėtekio 11, 10223 Vilnius, Lithuania
e-mail: zilinskasjulius@gmail.com

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

where $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})$; $d(\mathbf{x}_i, \mathbf{x}_j)$ denotes the distance between the points $\mathbf{x}_i$ and $\mathbf{x}_j$; it is supposed that the weights are positive: $w_{ij} > 0, i, j = 1, \ldots, n$. Usually $\delta_{ij} = \delta_{ji}$, $\delta_{ii} = 0$, $w_{ij} = w_{ji}$, and $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$, therefore it is possible to sum only either $j > i$ or $i > j$ terms and multiply by two, but (1) is more convenient for the derivation of the lower level quadratic programming problems in the next section and definition of them in matrix form. Usually Minkowski distances are used:

$$d_r(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{k=1}^{m} \left| x_{ik} - x_{jk} \right|^r \right)^{1/r}. \tag{2}$$

Equation (2) defines Euclidean distances when $r = 2$, and city-block distances when $r = 1$. The most frequently used distances are Euclidean. However, MDS with other Minkowski distances in the embedding space can be even more informative than MDS with Euclidean distances [1].

In the present paper the MDS problem with the *STRESS* criterion and city-block distances in the embedding space are considered. *STRESS* normally has many local minima. It is invariant with respect to translation and mirroring. It can be non differentiable even at a minimum point [20]; the case of city-block metric is different from the other cases of Minkowski metric where positiveness of distances at a local minimum point imply differentiability of *STRESS* [8,10]. MDS with city-block distances is a difficult high dimensional ($\mathbf{x} \in \mathbf{R}^N$, $N = n \times m$) global optimization problem. However *STRESS* with city-block distances is piecewise quadratic, and such a structure can be exploited for tailoring of an ad hoc algorithm.

A survey of city-block MDS was presented in [1]. A combinatorial approach for city-block MDS was proposed in [13], where combinatorial local search is used to construct good object orders along dimensions, and least-squares are used to estimate the coordinates of image points for the objects based on the object orders. A smoothing approach for city-block MDS was proposed in [11], where smoothing excludes some local minima of *STRESS*. The technique was extended to any Minkowski distance in [12]. A heuristic algorithm based on simulated annealing for two-dimensional city-block scaling was proposed in [3]. Each coordinate axis is partitioned by uniformly spaced points, and a simulated annealing algorithm is used to search the lattice defined by these points aiming to minimize one of two types of *STRESS* either the sum of squares (1), or the sum of corresponding absolute values. The solution found is locally improved by quadratic programming.

A two level minimization method for the two-dimensional embedding space was proposed in [20] where a problem of combinatorial optimization is tackled by evolutionary search at the upper level, and a problem of quadratic programming is tackled at the lower level. The parallel version of the algorithm is proposed and investigated in [19]. The generalized method for arbitrary dimensionality of the embedding space is developed and experimentally compared with distance smoothing approach and simulated annealing in [21].

Most papers on MDS consider local optimization or metaheuristic search which provide rather good values of objective function but cannot guarantee that the global minimum is found. In the present paper a version of two level algorithm is proposed where the upper level combinatorial problem is tackled by branch and bound. The proposed algorithm is suitable to solve middle size MDS problems exactly. The lower bound for the objective function is found solving lower level problems of smaller size taking into account only some of n objects.

2 Multidimensional scaling with city-block distances

STRESS (1) with city-block distances $d_1(\mathbf{x}_i, \mathbf{x}_j)$ can be redefined as

$$S(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \left(\sum_{k=1}^{m} |x_{ik} - x_{jk}| - \delta_{ij} \right)^2. \tag{3}$$

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

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

where $\mathbf{P} = (\mathbf{p}_1, \ldots, \mathbf{p}_m)$, $\mathbf{p}_k = (p_{k1}, p_{k2}, \ldots, p_{kn})$ is a permutation of $1, \ldots, n$; $k = 1, \ldots, m$. $A(\mathbf{P})$ is not empty since it contains for example the points with equal coordinates $x_{ik} = c,\ i = 1, \ldots, n,\ k = 1, \ldots, m$, where c is an arbitrary constant.

For $\mathbf{x} \in A(\mathbf{P})$, (3) can be rewritten in the following form

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

where

$$z_{kij} = \begin{cases} 1, & p_{ki} > p_{kj}, \\ -1, & p_{ki} < p_{kj}. \end{cases}$$

Since the function $S(\mathbf{x})$ is quadratic over polyhedron $\mathbf{x} \in A(\mathbf{P})$, the minimization problem

$$\min_{\mathbf{x} \in A(\mathbf{P})} S(\mathbf{x}) \tag{4}$$

can be reduced to the quadratic programming problem

$$\min \left(- \sum_{k=1}^{m} \sum_{i=1}^{n} x_{ik} \sum_{j=1}^{n} w_{ij} \delta_{ij} z_{kij} + \right.$$

$$\left. \frac{1}{2} \left(\sum_{k=1}^{m} \sum_{l=1}^{m} \sum_{i=1}^{n} x_{ik} x_{il} \sum_{t=1, t \ne i}^{n} w_{it} z_{kit} z_{lit} - \sum_{k=1}^{m} \sum_{l=1}^{m} \sum_{i=1}^{n} \sum_{j=1, j \ne i}^{n} x_{ik} x_{jl} w_{ij} z_{kij} z_{lij} \right) \right)$$

$$\text{s.t.} \sum_{i=1}^{n} x_{ik} = 0,\ k = 1, \ldots, m, \tag{5}$$

$$x_{\{j | p_{kj} = i+1\}, k} - x_{\{j | p_{kj} = i\}, k} \ge 0,\ k = 1, \ldots, m,\ i = 1, \ldots, n-1. \tag{6}$$

Polyhedron $A(\mathbf{P})$ is defined by the linear inequality constraints (6), and the equality constraints (5) ensure centering to avoid translated solutions. The feasible set defined by (5) and (6) is not empty since at least the point $x_{ik} = 0,\ i = 1, \ldots, n,\ k = 1, \ldots, m$ is feasible. A standard quadratic programming method can be applied to solve this problem.

The structure of the minimization problem (4) is favorable to apply a two level minimization:

$$\min_{\mathbf{P}} S(\mathbf{P}), \tag{7}$$

$$\text{s.t. } S(\mathbf{P}) = \min_{\mathbf{x} \in A(\mathbf{P})} S(\mathbf{x}), \tag{8}$$

Fig. 1 Mirrored solutions with respect to changing direction of coordinate axes in the case of $m = 2$ and corresponding permutations defining the sequence of coordinate values of image points representing objects a and b

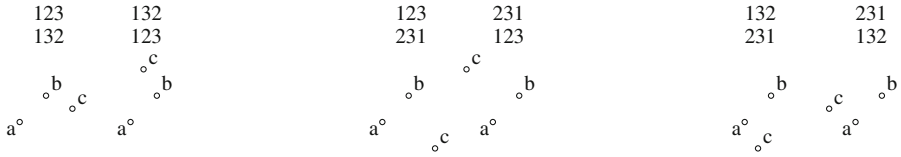


Fig. 2 Pairs of mirrored solutions with respect to exchanging of coordinates in the case of $m = 2$ and corresponding permutations defining the sequence of coordinate values of image points representing objects a, b and c

where (7) is a problem of combinatorial optimization, and (8) is a problem of quadratic programming.

The upper level (7) objective function is defined over the set of m-tuple of permutations of $1, \ldots, n$ representing sequences of coordinate values of image points. The number of feasible solutions of the upper level combinatorial problem is $(n!)^m$. A solution of MDS with city-block distances is invariant with respect to mirroring when changing direction of coordinate axes or exchanging of coordinates. Figures 1 and 2 illustrate mirrored solutions in the case of $m = 2$. As it can be seen from Fig. 1 there are 2^m equivalent solutions which can be represented by one of them. Similarly it can be seen from Fig. 2 that there are groups of $m!$ equivalent solutions which can be represented by one of them. The feasible set can be reduced taking into account such symmetries. The number of feasible solutions can be reduced to $(n!/2)^m$ refusing mirrored solutions changing direction of each coordinate axis. It can be further reduced to approximately $(n!/2)^m/m!$ refusing mirrored solutions with exchanged coordinates. It is not exactly $(n!/2)^m/m!$ as not all possible solutions belong to the groups of $m!$ equivalent solutions. Denoting $u = n!/2$, the number of feasible solutions is u in the case $m = 1$, $(u^2 + u)/2$ in the case $m = 2$, $(u^3 + 3u^2 + 2u)/6$ in the case $m = 3$, and $(u^4 + 6u^3 + 11u^2 + 6u)/24$ in the case $m = 4$.

The upper level combinatorial problem can be solved using different algorithms, e.g. small problems can be solved by explicit enumeration. In this paper a branch and bound algorithm for optimization of the upper level problem is proposed enabling guaranteed global minimum of problems of dimensionality up to $n = 12$, $m = 2$ and $n = 8$, $m = 3$ using a PC with Pentium IV 3 GHz processor.

3 Branch and bound for multidimensional scaling

The main concept of branch and bound is to search for the optimum, constructing a search tree so that only some of the feasible solutions should be explicitly evaluated detecting subsets of feasible solutions, represented by branches of the search tree, which cannot contain optimal solutions. The bound for the objective function over a subset of feasible solutions should be evaluated and compared with the best objective function value found. If the evaluated bound

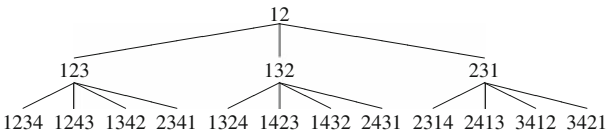


Fig. 3 A search tree for $m = 1$

is worse than the known function value, the subset cannot contain optimal solutions and the branch describing the subset can be pruned. The fundamental aspect of branch and bound is that the earlier the branch is pruned, the smaller the number of complete solutions that will need to be explicitly evaluated.

Evaluation of the bounds for the objective function is the most important part of the branch and bound technique. If the bounds are not tight, the search may lead to complete enumeration of all feasible solutions. This is not acceptable practically for all but the smallest problems, because the number of feasible solutions grows exponentially with the size of problem. Construction of the bound depends on the objective function and the type of subsets of feasible solutions over which the bound is evaluated.

In the proposed algorithm, a subset of feasible solutions is represented by a partial solution where only $\overline{n}$ of n objects are considered. Such a partial solution is defined by m-tuple of permutations $\overline{P}$ of $1, \ldots, \overline{n}$. The lower bound for *STRESS* is a value of partial *STRESS* at the minimum point of the lower level quadratic programming problem for $\overline{n}$ objects over a polyhedron $A(\overline{\mathbf{P}})$:

$$\min_{\overline{\mathbf{X}} \in A(\overline{\mathbf{P}})} \overline{S}(\overline{\mathbf{X}}) = \min_{\overline{\mathbf{X}} \in A(\overline{\mathbf{P}})} \sum_{i=1}^{\overline{n}} \sum_{j=1}^{\overline{n}} w_{ij} \left(\sum_{k=1}^{m} \left| x_{ik} - x_{jk} \right| - \delta_{ij} \right)^2, \tag{9}$$

where $\overline{\mathbf{X}} = (\mathbf{x}_1, \ldots, \mathbf{x}_{\overline{n}})$. Assignment of other objects later in the search must not change the sequence of coordinate values of image points of earlier assigned $\overline{n}$ objects and consideration of other objects cannot decrease value of *STRESS* function (it can only be increased).

A search tree for $m = 1$ is shown in Fig. 3. Every numeral represents a value of p_{1i}, $i = 1, \ldots, \overline{n}$. To refuse mirrored solutions the search tree starts with a root node representing partial solution "12" and therefore the image point representing the second object will never be to the left from the image point representing the first object. The lower bound for the objective function over this partial solution is not evaluated because it represents the feasible set of solutions. The image point representing the third object can be added to the right of the first two, between them or to the left of them. Therefore assignment of the third object is represented by three branches in the search tree ending with the nodes "123", "132" and "231". Although the sequence numbers of the first two objects may be changed (13 and 23) after assignment of the third object, their sequence is not changed ($1 < 2$, $1 < 3$ and $2 < 3$). The image point representing the fourth object has four possible positions and therefore assignment of the fourth object is represented by four branches. Again although the sequence numbers of the first three objects may be changed after assignment of the fourth object, their sequence is not changed. Assignment of the fifth object would be represented by five branches and so on. If the value of partial *STRESS* (9) at the minimum point of the lower level quadratic programming problem for a partial solution is greater than the value of *STRESS* at the minimum point of an already evaluated complete solution, the subset of feasible solutions represented by this partial solution cannot contain an optimal solution. Therefore the branch representing such a partial solution can be pruned, which means that the further search in this subset is not performed. For $m = 1$ the number of feasible solutions is $n!/2$.

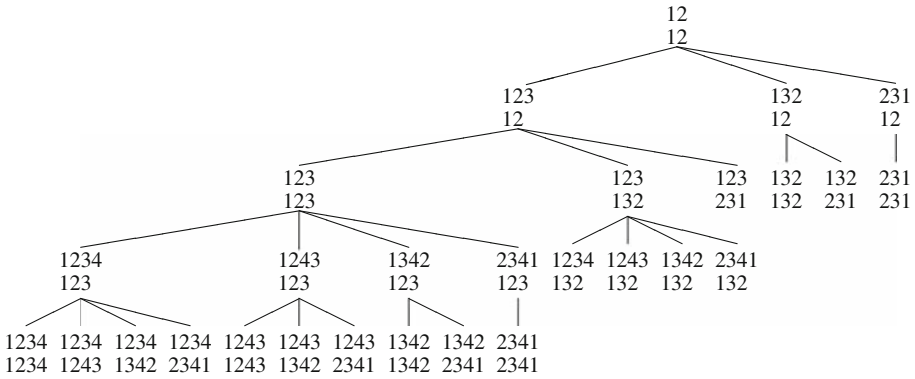


Fig. 4 A search tree for $m = 2$

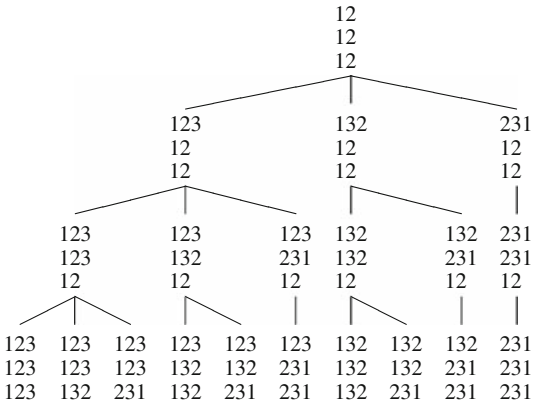
A search tree for $m = 2$ is shown in Fig. 4. Every numeral represents a value of p_{ki}, $k = 1, \ldots, m$, $i = 1, \ldots, \overline{n}$. Lower rows represent greater k. To refuse mirrored solutions the search tree starts with a root node representing partial solution "12/12" and therefore the image point representing the second object will never be to the left or up from the image point representing the first object (see Fig. 1). The lower bound over this partial solution is not evaluated because it represents the feasible set of solutions. The image point representing the third object horizontally can be to the right of the first two, between them or to the left of them. Therefore assignment of the third object is represented by three branches in the search tree ending with the nodes "123/12", "132/12" and "231/12". Vertically it can be below the first two, between them or above of them. Therefore the node "123/12" has three branches ending with the nodes "123/123", "123/132" and "123/231". To refuse mirrored solutions with exchanged coordinates some restrictions on permutations of equal sizes are set. Let us define the order of permutations like one in Fig. 3: for permutations of $1, \ldots, 3$ it is "123" $\prec$ "132" $\prec$ "231" and for permutations of $1, \ldots, 4$ it is "1234" $\prec$ "1243" $\prec$ "1342" $\prec$ "2341" $\prec$ "1324" $\prec$ "1423" $\prec$ "1432" $\prec$ "2431" $\prec$ "2314" $\prec$ "2413" $\prec$ "3412" $\prec$ "3421". A permutation $\mathbf{p}_k$ cannot precede $\mathbf{p}_l$ for $k > l$ ($l < k \Rightarrow \mathbf{p}_l \preceq \mathbf{p}_k$). Therefore partial solutions "132/123", "231/123" and "231/132" are not allowed, as they represent solutions symmetric to "123/132", "123/231" and "132/231" respectively (see Fig. 2). For $m = 2$ the number of feasible solutions is $n!^2/8 + n!/4$. When $m > 1$ some subsets of feasible solutions are represented with partial solutions defined by permutations of different sizes, for example "123/12". However the value of partial *STRESS* (9) is evaluated only over the partial solutions defined by permutations of equal sizes.

A search tree for $m = 3$ is shown in Fig. 5. Similar restrictions hold as for the case $m = 2$. For $m = 3$ the number of feasible solutions is $n!^3/48 + n!^2/8 + n!/6$. Similarly a search tree can be built for larger values of m.

An iteration of a classical branch and bound algorithm processes a node in the search tree representing a not yet explored subspace of the solution space. Iteration has three main components: selection of the node to process, branching, and bound calculation. There are three main strategies of selection:

- Best first – select the node with minimal lower bound.
- Depth first – select the node which is farthest from the root node.
- Breadth first – select the node which is closest to the root node.

Fig. 5 A search tree for $m = 3$



The proposed branch and bound algorithm is built using depth first selection and its structure is similar to the structure of algorithms in [4]. The advantage of the depth first strategy is that the search tree may be constructed sequentially to avoid storing of unbranched nodes. This is very important when the search tree is big. The breadth first search requires storing of unbranched nodes in a first-in-first-out structure. The best first search requires storing of unbranched nodes in a priority queue structure, which not only requires memory resources, but insertion and removal of nodes takes at least logarithmic time to the number of nodes in the queue in the worst case.

The proposed branch and bound algorithm for MDS with city-block distances is shown in Algorithm 1. Evaluation of partial solution corresponds to the bound calculation. The search tree is constructed sequentially implementing depth first search. Order of permutations is described earlier in this section. Verification of order of permutations in a tuple ($\mathbf{p}_k \prec \mathbf{p}_{k-1}$) enables refusal of mirrored solutions with exchanged coordinates. The main cycle continues while $\overline{n} > 2$ refusing mirrored solutions with changed direction of coordinate axis.

Future work may be devoted for improvement of the proposed algorithm using heuristic rearrangement of objects before optimization or different selection strategies in branch and bound, improvement of bound analyzing minimal impact of unassigned objects to the value of objective function, and development of parallel versions of the algorithm.

4 Experimental investigation

The efficiency of the proposed algorithm has been evaluated experimentally. The proposed algorithm has been implemented in C++, and g++ 3.2.3 has been used for compilation. External quadratic programming package quadprog 1.4–7 available at http://cran.r-project.org is used. Experiments have been performed on a PC with 3 GHz Pentium IV processor and Scientific Linux 3.0.5 operating system.

The accuracy of fit while evaluated via minimum of $S(\mathbf{x})$ depends on n and $\delta_{ij}, i, j = 1, \ldots, n$. Therefore such a criterion is not very convenient to compare accuracies of scaling for different sets of objects. To reduce such an undesirable impact, a relative error

$$f(\mathbf{x}) = \sqrt{S(\mathbf{x}) \Big/ \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \delta_{ij}^2}$$

Algorithm 1 Branch and bound algorithm for multidimensional scaling

Input: $n; m; \delta_{ij},\ w_{i,j},\ i, j = 1, \ldots, n$
Output: S^*
1: $p_{ki} \leftarrow i,\ i = 1, ..., n,\ k = 1, ..., m$
2: $\overline{n} \leftarrow n + 1; k \leftarrow m + 1; S^* \leftarrow \infty$
3: **while** $\overline{n} > 2$ **do**
4: **if** $k > m$ **then**
5: **if** $\overline{n} > 2$ **and** $\overline{n} < n$ **then**
6: **if** $\min_{\overline{\mathbf{X}} \in A(\overline{\mathbf{P}})} \overline{S}(\overline{\mathbf{X}}) \geq S^*$ **then** // Evaluate partial solution
7: $k \leftarrow k - 1$
8: **else**
9: $\overline{n} \leftarrow \overline{n} + 1; k \leftarrow 1$
10: **end if**
11: **else**
12: $\overline{n} \leftarrow \overline{n} + 1; k \leftarrow 1$
13: **end if**
14: **end if**
15: **if** $\overline{n} > n$ **then**
16: $S^* \leftarrow \min\left(S^*, \min_{\mathbf{x} \in A(\mathbf{P})} S(\mathbf{x})\right)$ // Evaluate complete solution
17: $\overline{n} \leftarrow n; k \leftarrow m$
18: **end if**
19: **if** $\overline{n} > 2$ **then**
20: // Form next tuple of permutations
21: **if** $p_{k\overline{n}} = 0$ **then**
22: $p_{k\overline{n}} \leftarrow \overline{n}$
23: **if** $k > 1$ **and** $\mathbf{p}_k \prec \mathbf{p}_{k-1}$ **then** // Detect refusable symmetries
24: $p_{ki} \leftarrow p_{k-1i},\ i = 1, ..., \overline{n}$
25: **end if**
26: $k \leftarrow k + 1$
27: **else**
28: $p_{k\overline{n}} \leftarrow p_{k\overline{n}} - 1$
29: **if** $p_{k\overline{n}} = 0$ **then**
30: $p_{ki} \leftarrow p_{ki} - 1,\ i = 1, ..., \overline{n} - 1$
31: $k \leftarrow k - 1$
32: **if** $k < 1$ **then**
33: $\overline{n} \leftarrow \overline{n} - 1; k \leftarrow m$
34: **end if**
35: **else**
36: find i: $p_{ki} = p_{k\overline{n}},\ i = 1, ..., \overline{n} - 1$
37: $p_{ki} \leftarrow p_{ki} + 1; k \leftarrow k + 1$
38: **end if**
39: **end if**
40: **end if**
41: **end while**

is used in the presentation of the results. Performance of deterministic global optimization algorithms is measured using the optimization time t and the smallest function value found. We also present the number of the lower level quadratic programming problems solved (NQPP).

Several sets of multidimensional points corresponding to well understood geometric objects have been used for experimental investigation: the sets of vertices of multidimensional simplices and cubes. Dissimilarity between vertices is measured by city-block distance in the original vector space. Global optimization problems of increasing complexity correspond to increasing dimensionality of the original space d. Below we use shorthand 'simplex' and 'cube' for the sets of their vertices.

The number of vertices of multidimensional simplex is $n = d + 1$, and the dimensionality of global minimization problem is $N = m \times (d + 1)$. Two types of multidimensional simplices are used. The distances between any two vertices of standard simplex are equal: $\delta_{ij} = 1, i \neq j$. Vertices of unit simplex can be defined by

$$v_{ij} = \left\{ \begin{array}{l} 1, \text{ if } i = j + 1, \\ 0, \text{ otherwise,} \end{array} \right| \; i = 1, \ldots, d + 1, \;\; j = 1, \ldots, d.$$

The number of vertices of multidimensional cube is $n = 2^d$, and the dimensionality of global minimization problem is $N = m \times 2^d$. The coordinates of i-th vertex of a d-dimensional cube are equal either to 0 or to 1, and they are defined by a binary code of $i = 0, \ldots, n - 1$.

A frequently used test problem for MDS algorithms is based on experimental testing of several soft drinks [9]. 38 students have tested ten different brands of soft drinks. Each pair was judged on its dissimilarity on a 9 point scale (1 – very similar, 9 – completely different). The accumulated dissimilarities have been used as a practical data set in our experiments. This problem is referred as 'cola' problem in the results below, $n = 10$ in this problem.

Problems of analysis of pharmacological binding affinity data [17] have been used as other practical data sets. 'ruusk' represents binding affinity data of Ruuskanen et al. [15] analyzed as properties of three human and five zebrafish α_2-adrenoceptor proteins, $n = 8$; 'uhlen' represents binding affinity data of Uhlèn et al. [16] analyzed as properties of human, rat, guinea pig and pig α_2-adrenoceptor proteins, $n = 12$; 'hwa12' represents binding affinity data of Hwa et al. [14] analyzed as properties of ligands, $n = 9$; 'hwa21' represents binding affinity data of Hwa et al. [14] analyzed as properties of wild type and mutant proteins, $n = 12$. The binding affinity data is represented through a matrix, one dimension formed by the different ligands tested in a series of experiments while the other dimension represents the different proteins. Dissimilarities of proteins are computed as city-block distances between vectors of the $\log_{10}$-transformed binding affinities representing properties of the proteins. Dissimilarities of ligands are computed as city-block distances between vectors of the $\log_{10}$-transformed binding affinities representing ligands.

Problems with $m = 1$ can be solved efficiently maximizing Defays [7] criterion using branch and bound [5]. However problems with $m = 1$ have been included in the experimental investigation of the proposed algorithm to have a larger set of problems and to investigate the worst case scenario. Exchange of vertices of standard simplex does not impact the problem as all vertices are equally distant from all others. Therefore all lower level quadratic programming problems (8) for the problems of standard simplices with $m = 1$ have the same minimum and no partial solution in the search tree can be pruned. In such a worst case the branch and bound algorithm is more costly than explicit enumeration of all feasible solutions because of additional evaluation of bounds over partial solutions. It is interesting to measure the decrease of efficiency in the worst case. Similarly exchange of non zero vertices of unit simplex does not impact the problem. Cube also has some symmetries. It would be possible to take into account symmetries of the data sets restricting possible permutations and increase the efficiency [18], however symmetries in practical data sets are rare known and therefore this is not considered in the present paper.

Performance of two level algorithm for MDS with explicit enumeration of feasible solutions of the upper level combinatorial problem and standard quadratic programming algorithm for the lower level problem is shown in Table 1. The numbers of objects in the problems are shown in the first column. The results of experimental investigation are shown using two columns for each dimensionality of the embedding space ($m = 1$, $m = 2$ and $m = 3$).

Table 1 Performance of MDS algorithm based on explicit enumeration of the upper level problem

n	$m=1$		$m=2$		$m=3$	
	t, s (NQPP)	f^*	t, s (NQPP)	f^*	t, s (NQPP)	f^*
			unit simplices			
3	0.00 (3)	0.00	0.00 (6)	0.00	0.00 (10)	0.00
4	0.00 (12)	0.3651	0.00 (78)	0.00	0.02 (364)	0.00
5	0.00 (60)	0.4140	0.05 (1830)	0.00	2.02 (37820)	0.00
6	0.00 (360)	0.4554	2.02 (64980)	0.1869	653.91 (7840920)	0.00
7	0.04 (2520)	0.4745	133.28 (3176460)	0.2247	334788 (2670344040)	0.00
8	0.24 (20160)	0.4917	11631 (203222880)	0.2569		
9	2.48 (181440)	0.5018				
10	33.16 (1814400)	0.5113				
11	372.59 (19958400)	0.5176				
12	5208.0 (239500800)	0.5236				
13	78579 (3113510400)	0.5279				
			standard simplices			
3	0.00 (3)	0.3333	0.00 (6)	0.00	0.00 (10)	0.00
4	0.00 (12)	0.4082	0.01 (78)	0.00	0.02 (364)	0.00
5	0.00 (60)	0.4472	0.05 (1830)	0.1907	1.79 (37820)	0.00
6	0.01 (360)	0.4714	1.73 (64980)	0.2309	580.87 (7840920)	0.00
7	0.02 (2520)	0.4879	113.77 (3176460)	0.2621	301860 (2670344040)	0.0945
8	0.21 (20160)	0.5000	10183 (203222880)	0.2825		
9	2.22 (181440)	0.5092				
10	27.39 (1814400)	0.5164				
11	334.30 (19958400)	0.5222				
12	4687.0 (239500800)	0.5270				
13	68762 (3113510400)	0.5311				
			cubes			
4	0.00 (12)	0.4082	0.00 (78)	0.00	0.02 (364)	0.00
8	0.23 (20160)	0.4787	12518 (203222880)	0.2245		
			ruusk			
8	0.25 (20160)	0.2975	12183 (203222880)	0.1096		
			hwa12			
9	3.06 (181440)	0.0107				
			cola			
10	27.07 (1814400)	0.3642				
			uhlen			
12	6413.0 (239500800)	0.2112				
			hwa21			
12	6648.0 (239500800)	0.1790				

Time of optimization in seconds (t, s) and the number of the lower level quadratic problems solved (NQPP) are given in the left columns. The value of relative error (f^*) corresponding to the minimum of *STRESS* is given in the right columns. The number of quadratic programming problems solved grows very fast with the number of objects n and they are exactly as predicted in Sect. 3.

Performance of the proposed two level algorithm for MDS with branch and bound for the upper level combinatorial problem and standard quadratic programming algorithm for the lower level problem is shown in Table 2. The numbers of quadratic programming problems solved include the lower level problems corresponding to evaluated complete and partial solutions of the upper level problems, however quadratic programming problems for partial solutions are smaller than ones for complete solutions as only some of objects are considered there.

Table 2 Performance of MDS algorithm based on branch and bound for the upper level problem

n	$m = 1$		$m = 2$		$m = 3$	
	t, s (NQPP)	f^*	t, s (NQPP)	f^*	t, s (NQPP)	f^*
			unit simplices			
3	0.00 (3)	0.00	0.00 (6)	0.00	0.00 (10)	0.00
4	0.00 (14)	0.3651	0.00 (73)	0.00	0.02 (313)	0.00
5	0.00 (73)	0.4140	0.03 (662)	0.00	0.49 (9837)	0.00
6	0.01 (432)	0.4554	0.51 (16076)	0.1869	46.67 (578691)	0.00
7	0.03 (2951)	0.4745	17.65 (422940)	0.2247	2652.0 (20674563)	0.00
8	0.32 (23110)	0.4917	1675.0 (29943080)	0.2569		
9	2.77 (204549)	0.5018	134281 (1905072549)	0.2759		
10	31.67 (2018948)	0.5113				
11	404.64 (21977347)	0.5176				
12	5545.0 (261478146)	0.5236				
13	86436 (3374988545)	0.5279				
			standard simplices			
3	0.00 (3)	0.3333	0.00 (6)	0.00	0.00 (10)	0.00
4	0.00 (14)	0.4082	0.00 (63)	0.00	0.01 (133)	0.00
5	0.00 (73)	0.4472	0.03 (1322)	0.1907	1.12 (23017)	0.00
6	0.01 (432)	0.4714	0.85 (27255)	0.2309	25.49 (335771)	0.00
7	0.05 (2951)	0.4879	59.61 (1655631)	0.2621	11111 (92710201)	0.0945
8	0.24 (23110)	0.5000	5107.0 (102073658)	0.2825		
9	2.47 (204549)	0.5092	502844 (3574743410)	0.2991		
10	28.33 (2018948)	0.5164				
11	361.60 (21977347)	0.5222				
12	4970.0 (261478146)	0.5270				
13	73714 (3374988545)	0.5311				
			cubes			
4	0.00 (14)	0.4082	0.00 (73)	0.00	0.02 (353)	0.00
8	0.12 (11260)	0.4787	124.68 (2157090)	0.2245	6189 (35216122)	0.00
			ruusk			
8	0.02 (665)	0.2975	3.85 (82617)	0.1096	838.68 (6381457)	0.0188
			hwa12			
9	0.02 (2217)	0.0107	203.25 (2344833)	0.00		
			cola			
10	0.78 (60077)	0.3642	15594 (204022569)	0.1675		
			uhlen			
12	0.62 (36559)	0.2112	35951 (312924750)	0.0825		
			hwa21			
12	1.49 (71748)	0.1790				

For better comparison of the performance of algorithms the results are plotted in Fig. 6. Time of optimization is given in logarithmic scale. Dots represent time of optimization for considered n-dimensional data sets in m-dimensional embedding space. Results for various dimensional simplices and cubes are joined by lines. Results of the algorithm based on explicit enumeration depend mostly on dimensionality of data and embedding space as all dots are near the lines, the influence of type of data set is modest and therefore the data sets are not specified in this plot.

The proposed algorithm behaves in the worst case scenario when highly symmetric data sets of simplices are used with $m = 1$. In the case of standard simplices all lower level problems for complete solutions are equivalent as they are symmetric [18]. Because of the symmetric data no partial solution can be pruned. In the worst case scenario almost all partial solutions should be evaluated as well as all complete solutions. The number of quadratic

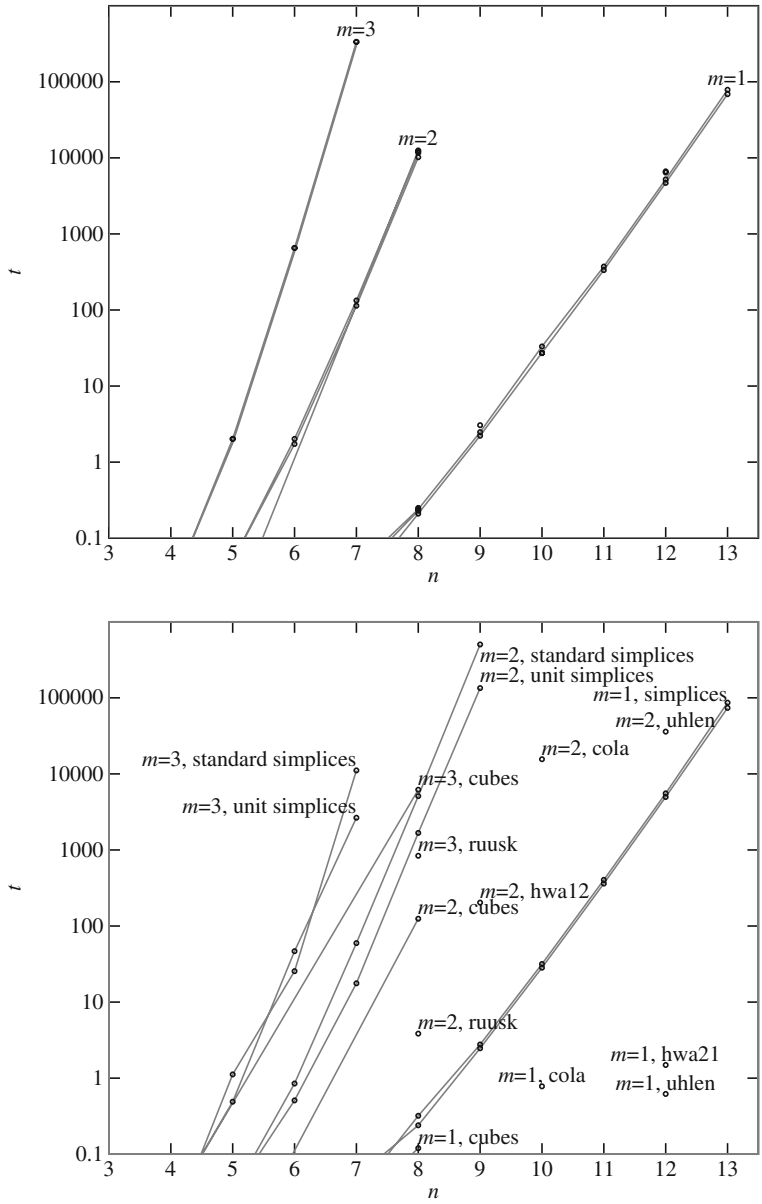


Fig. 6 Comparison of performance of algorithms based on explicit enumeration (upper plot) and branch and bound (lower plot) for the upper level problem

programming problems solved is $\sum_{i=3}^{n} \frac{i!}{2} - n + 3$ comparing to $n!/2$ of complete solutions evaluated by the algorithm of explicit enumeration. Optimization time and the number of quadratic programming problems of the proposed algorithm are up to 13% larger than these of the algorithm based on explicit enumeration for problems which optimization is longer than 1 s. Plots of results for $m = 1$ simplices are very similar in both plots of Fig. 6. However for

$m > 1$ even for simplices the proposed algorithm performs better than explicit enumeration. Plots of these results move to the right as optimization becomes faster. Standard simplices are harder problems than unit simplices in the case $m > 1$ for the proposed algorithm as they are more symmetric. The plots for unit simplices are lower than ones for standard simplices.

The proposed algorithm performs much better than the algorithm based on explicit enumeration for cubes and practical data sets even when $m = 1$. The number of lower level quadratic programming problems solved while applying the branch and bound algorithm is up to several thousand times less than the number of such problems solved while applying explicit enumeration; correspondingly the solution time in the first case is up to ten thousand times shorter than in the second case. Dots representing these results move down in the lower plot of Fig. 6. However, it is even more important that the proposed algorithm can solve larger problems in acceptable time. Dots representing these results appear in the lower plot of Fig. 6. Problems of up to $n = 12$ have been solved for the two-dimensional embedding space, and up to $n = 8$ for the three-dimensional embedding space. The largest solved problems involve global optimization in N-dimensional ($N = 24$) space. Performance of the proposed algorithm is better for practical data sets than for geometric data sets for the same m as it can be seen in Fig. 6 where dots corresponding to the results of practical data sets are to the right from corresponding plots of geometric data sets.

Minima of *STRESS* found for the two-dimensional scaling problems of binding affinity data coincide with those in [17] proving that the evolutionary algorithm used in [17] has found the true global minima.

To get some impression about the performance, the proposed algorithm was compared with the two algorithms referenced below; it should be taken into account however, that the proposed algorithm includes guaranteed finding of global minimum of *STRESS* while the competing algorithms are based on some approximations of global minimum. The first algorithm to compare is similar to the proposed one, and it differs from the proposed algorithm only in tackling of the upper level problems; the competing algorithm is a two level hybrid algorithm with evolutionary search at the upper level and standard quadratic programming algorithm at the lower level [19,20]. The second algorithm to compare is the well known distance smoothing algorithm [12]. The results are shown in Table 3. For the hybrid algorithm the following default parameters were used: the size of population of the evolutionary algorithm and the number of random initial individuals were equal to 60; 100 runs of 10 s length have been performed. Smooth 4 (implementation of the smoothing algorithm developed by their authors and available at http://people.few.eur.nl/groenen) has been used with the parameters recommended by its authors and given as defaults in his 'input' file; 100 random starts have been performed. For two embedding spaces (of the dimensionality $m = 2$ and $m = 3$) the best (min f^*) and the worst (max f^*) relative error corresponding to found minima of *STRESS* in 100 runs are given (in two columns if different, and in one column if equal).

Both competing algorithms terminate after shorter time than the proposed algorithm based on branch and bound. Such a result was expected as the proposed algorithm includes guaranteed finding of global minimum of *STRESS* while the competing algorithms terminate with some appropriate approximations. The hybrid algorithm finds better values of f more frequently (in 100 runs) than the distance smoothing algorithm, and the best value is found by the hybrid algorithm in larger number of runs. However, the distance smoothing algorithm requires shorter time.

Let us to complete the discussion on experimental results with a remark on the qualitative influence of properties of the used data to the experimental results. The geometric data sets imply properties of minimization problem (1) which are most disadvantageous for the branch and bound algorithm with respect to explicit enumeration; because of symmetries in

Table 3 Performance of other MDS algorithms

n	Hybrid algorithm with evolutionary search				Distance smoothing			
	$m = 2$		$m = 3$		$m = 2$		$m = 3$	
	Min f^*	Max f^*	Min f^*	Max f^*	Min f^*	Max f^*	Min f^*	Max f^*
				unit simplices				
3	0.00			0.00				
4	0.00			0.00	0.00		0.00	
5	0.00			0.00	0.00		0.00	
6	0.1869			0.00	0.1869		0.00	
7	0.2247			0.00	0.2247		0.00	
8	0.2569			0.0992	0.2569		0.0992	0.0993
9	0.2759			0.1272	0.2759		0.1272	0.1439
10	0.2936			0.1542	0.2936		0.1543	0.1548
11	0.3058			0.1679	0.3058		0.1679	0.1829
12	0.3167			0.1874	0.3167		0.1874	0.1969
13	0.3249	0.3259		0.2008	0.3249	0.3265	0.2008	0.2087
				standard simplices				
3	0.00			0.00	0.00		0.00	
4	0.00			0.00	0.00		0.00	
5	0.1907			0.00	0.1907		0.00	
6	0.2309			0.00	0.2309		0.00	
7	0.2621			0.0945	0.2621		0.0945	
8	0.2825			0.1250	0.2825		0.1250	0.1320
9	0.2991			0.1543	0.2991		0.1544	0.1548
10	0.3115			0.1682	0.3115		0.1682	0.1833
11	0.3217			0.1879	0.3217	0.3233	0.1879	0.1980
12	0.3300	0.3309		0.2013	0.3300	0.3307	0.2013	0.2096
13	0.3371	0.3377	0.2109	0.2118	0.3371	0.3379	0.2109	0.2143
				cubes				
4	0.00			0.00	0.00		0.00	
8	0.2245			0.00	0.2245	0.2478	0.00	
				ruusk				
8	0.1096		0.0188	0.0210	0.1096	0.1232	0.0189	0.0479
				hwa12				
9	0.00			0.00	0.0108	0.0114	0.0108	0.0127
				cola				
10	0.1675	0.1686	0.0676	0.0766	0.1694	0.2004	0.0676	0.0841
				uhlen				
12	0.0825	0.0826	0.0356	0.0381	0.0875	0.1188	0.0358	0.0481
				hwa21				
12	0.0497		0.0183	0.0191	0.0497	0.0502	0.0248	0.0482

data the objective function $S(\mathbf{x})$ has many global minimum points, and all of them should be explored by the algorithm guaranteeing finding the global minimum. In this case branching subsequently generates large number of subproblems with minimum lower bounds requesting further branching of these subproblems. Therefore performance of the branch and bound algorithm in this case is similar to the performance of explicit enumeration.

Contrary, for the hybrid algorithm and the distance smoothing algorithm these geometric data sets are favorable since the number of good local (including global) minimum points is relatively large implying not inconsiderable probability to find a good local minimum even in one local descent.

In data sets corresponding to practical problems symmetries are absent implying properties of (1) more favorable for the branch and bound algorithm and less favorable for the hybrid algorithm and the distance smoothing algorithm.

5 Conclusions and further research

Optimization problems occurring in MDS normally are tackled by heuristic algorithms because of multimodality and other unfavorable properties of objective functions. Although practically acceptable local minima frequently can be found in this way, it is interesting to research algorithms aiming to find global minimum with guarantee; besides of some practical applications such algorithms are of interest in further development of fast heuristic algorithms to evaluate their precision. In the present paper problems of MDS with city-block metric in the embedding space are considered. An algorithm is proposed to find global minimum with guarantee for problems of modest but practically important dimensionality. The proposed algorithm is of two level structure: a combinatorial problem at the upper level is solved by a branch and bound technique, and a quadratic programming algorithm is applied at the lower level. The proposed algorithm has been tested using data of several real world problems.

One of the main topics for the further research is the enhancement of efficiency of the proposed algorithm by means of choosing the most appropriate method for the lower level quadratic problems; among the candidates interior point methods can be mentioned. On the other hand, availability of the algorithm with guaranteed precision will be helpful in the further development of heuristic algorithms for upper level combinatorial problems where simulated annealing, swarm intelligence, and other algorithms will be compared with the presently used genetic algorithm.

Acknowledgements The research is supported by the Agency for International Science and Technology Development Programmes in Lithuania through COST programme, Lithuanian State Science and Studies Foundation within the project B-03/2007 “Global optimization of complex systems using high performance computing and GRID technologies” and the NATO Reintegration grant CBP.EAP.RIG.981300.

References

1. Arabie, P.: Was Euclid an unnecessarily sophisticated psychologist? Psychometrika **56**(4), 567–587 (1991)
2. Borg, I., Groenen, P.: Modern Multidimensional Scaling, 2nd edn. Springer, New York (2005)
3. Brusco, M.J.: A simulated annealing heuristics for unidimensional and multidimensional (city-block) scaling of symmetric proximity matrices. J. Classif. **18**(1), 3–33 (2001)
4. Brusco, M.J., Stahl, S.: Branch-and-Bound Applications in Combinatorial Data Analysis. Springer, New York (2005)
5. Brusco, M.J., Stahl, S.: Optimal least-squares unidimensional scaling: improved branch-and-bound procedures and comparison to dynamic programming. Psychometrika **70**(2), 253–270 (2005)
6. Cox, T., Cox, M.: Multidimensional Scaling. Chapman and Hall/CRC, Boca Raton (2001)
7. Defays, D.: A short note on a method of seriation. Br. J. Math. Stat. Psychol. **31**, 49–53 (1978)
8. de Leeuw, J.: Differentiability of Kruskal’s stress at a local minimum. Psychometrika **49**(1), 111–113 (1984)
9. Green, P., Carmone, F., Smith, S.: Multidimensional Scaling: Concepts and Applications. Allyn and Bacon, Boston (1989)
10. Groenen, P.J.F., Mathar, R., Heiser, W.J.: The majorization approach to multidimensional scaling for Minkowski distances. J. Classif. **12**(1), 3–19 (1995)

11. Groenen, P.J.F., Heiser, W.J., Meulman, J.J. : City-block scaling: smoothing strategies for avoiding local minima. In: Balderjahn, I., Mathar, R., Schader, M. (eds.) Classification, Data Analysis, and Data Highways, pp. 46–53. Springer, Heidelberg (1998)
12. Groenen, P.J.F., Heiser, W.J., Meulman, J.J.: Global optimization in least-squares multidimensional scaling by distance smoothing. J. Classif. **16**(2), 225–254 (1999)
13. Hubert, L., Arabie, P., Hesson-Mcinnis, M.: Multidimensional scaling in the city-block metric: a combinatorial approach. J. Classif. **9**(2), 211–236 (1992)
14. Hwa, J., Graham, R.M., Perez, D.M.: Identification of critical determinants of α_1-adrenergic receptor subtype selective agonist binding. J. Biol. Chem. **270**(39), 23189–23195 (1995)
15. Ruuskanen, J.O., Laurila, J., Xhaard, H., Rantanen, V.-V., Vuoriluoto, K., Wurster, S., Marjamaki, A., Vainio, M., Johnson, M.S., Scheinin, M.: Conserved structural, pharmacological and functional properties among the three human and five zebrafish α_2-adrenoceptors. Br. J. Pharmacol. **144**(2), 165–177 (2005)
16. Uhlén, S., Dambrova, M., Näsman, J., Schiöth, H.B., Gu, Y., Wikberg-Matsson, A., Wikberg, J.E.S.: [^{3}H]RS79948-197 binding to human, rat, guinea pig and pig α_{2A}-, α_{2B}- and α_{2C}-adrenoceptors, Comparison with MK. **912**, RX821002, rauwolscine and yohimbine. Eur. J. Pharmacol. **343**(1), 93–101 (1998)
17. Žilinskas, J.: Multidimensional scaling in protein and pharmacological sciences. In: Bogle, I.D.L., Žilinskas, J. (eds.) Computer Aided Methods in Optimal Design and Operations. Series on Computers and Operations Research, vol. 7, pp. 139–148. World Scientific, Singapore (2006)
18. Žilinskas, J.: Reducing of search space of multidimensional scaling problems with data exposing symmetries. Inform. Technol. Control **36**(4), 377–382 (2007)
19. Žilinskas, A., Žilinskas, J.: Parallel hybrid algorithm for global optimization of problems occurring in MDS-based visualization. Comput. Math. Appl. **52**(1–2), 211–224 (2006)
20. Žilinskas, A., Žilinskas, J.: Two level minimization in multidimensional scaling. J. Global Optim. **38**(4), 581–596 (2007)
21. Žilinskas, A., Žilinskas, J.: A hybrid method for multidimensional scaling using city-block distances. Math. Method. Oper. Res. accepted (2008)