Parallel Genetic Algorithm: Assessment of Performance in Multidimensional Scaling

Antanas Žilinskas
Institute of Mathematics and Informatics
Akademijos 4
Vilnius LT-08663, Lithuania
antanasz@ktl.mii.lt

Julius Žilinskas
Institute of Mathematics and Informatics
Akademijos 4
Vilnius LT-08663, Lithuania
julius.zilinskas@mii.lt

ABSTRACT

Visualization of multidimensional data by means of Multidimensional Scaling (MDS) is a popular technique of exploratory data analysis widely usable, e.g. in analysis of bio-medical data, behavioral science, marketing research, etc. Implementations of MDS methods include a subroutine for an auxiliary global optimization problem. The latter is difficult because of high dimensionality, absence of overall smoothness, and a large number of local minima. In such a situation application of a genetic algorithm (GA) seems reasonable. A favorable assessment of application of GAs in MDS in previous publications is based on heuristic arguments without estimating quantitatively the precision of GA while applied to the solution of corresponding global optimization problems. Indeed, the estimation of precision is difficult because of complexity to find the actual global minimum not only in routine use but also in unique research experiments. Quantitatively the precision of GA was estimated, at least in the experimental problems of modest dimensionality, using global minima found by means of the developed parallel version of explicit enumeration algorithm. To cope with high complexity of the minimization problem a parallel version of GA is developed, and its efficiency for problem of higher dimensionality is investigated.

Track: Genetic Algorithms.

Categories and Subject Descriptors

G.1.6 [**Numerical analysis**]: Optimization—*Global optimization, integer programming, quadratic programming methods*; G.4 [**Mathematics software**]: Parallel and vector implementations

General Terms

Algorithms, Performance

Keywords

genetic algorithms, multidimensional scaling

1. INTRODUCTION

To grasp the structure of a set of multi-parametric objects a two-dimensional image of the corresponding multidimensional points frequently can be helpful. Multidimensional Scaling (MDS) is a widely usable technique for mapping from a multidimensional space of originals to an embedding space of lower dimensionality, e.g. to two-dimensional plane of images [2]. For the detailed description of MDS methods and examples of application we refer to [2] and to the publications cited there.

The implementation of MDS methods is reduced to a difficult problem of global optimization [8]. Genetic algorithms (GA), who are among the most popular global optimization algorithms [6, 13, 14], can be applied for the solution of MDS problems. Because of robustness, the application of GA in MDS allows various modifications of the underlying optimization problem implying non differentiability of corresponding objective functions [5]. For MDS problems with the standard *STRESS* function (see next section) a combination of GA with an efficient local method seems promising [12]. Implementations of such a method (GA-MDS) are more reliable than the other MDS algorithms, as shown by the results of experiments in [9, 11]. A heuristic method avoiding solution of an auxiliary optimization problem is described, e.g. in [15]; but such an approach seems reasonable mainly to non metric scaling. For a general discussion on various methods of MDS we refer to [1].

Our recent results [17] support the conclusion of the authors of the cited above papers that application of GA in MDS is highly promising. However, these conclusions are mainly heuristic without quantitative estimation of precision in solving the corresponding global optimization problems. Indeed, an objective assessment is difficult because of complexity in finding global minima of the considered problems.

In the present paper an attempt is made to assess the efficiency of GA-MDS quantitatively. To estimate errors of GA-MDS true global minima of corresponding optimization problems should be known. We consider a version of MDS problems where global minimum can be found with guarantee, at least for problems of modest dimensionality; i.e. we considered problems where distances in the embedding space are measured according to the city block metric. The auxiliary optimization problem in this case is piecewise quadratic, and global minimum can be found by a combination of enumeration with quadratic programming. A parallel algorithm is developed enabling to find global optima in the problems of moderate dimensionality with guarantee.

The efficiency of the developed GA-MDS was assessed quantitatively for the problems where global minima have been found by the described above enumeration algorithm. The presented below quantitative results can be summarized as follows: GA-MDS solved all these problems very fast with reliability depending on dimensionality of the problem. A parallel version of the proposed GA-MDS is implemented and tested for large scale problems.

2. GLOBAL OPTIMIZATION IN MDS

We are interested to grasp the structure of a set of n multi-parametric objects whose pairwise dissimilarities (δ_{ij}) are known. For example, the objects can be represented by a set of points in a metric space where (δ_{ij}) are defined as distances between i-th and j-th points. We are interested to visualize the set of objects by their images in two-dimensional plane where heuristic human ability to investigate the structure of a set of points is much stronger than the ability to investigate such structures in abstract/multidimensional spaces of objects. By means of MDS the points $\mathbf{x}_i = (x_{i1}, x_{i2})$, $i = 1, \ldots, n$, representing the considered original objects in the two-dimensional embedding space should be found fitting pairwise distances between the image points to the given pairwise dissimilarities of the objects (δ_{ij}).

Mathematically the problem of MDS is formulated as the problem of minimization of a fitness criterion, e.g. the so called *STRESS* function:

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

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

where $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$; $d(\mathbf{x}_i, \mathbf{x}_j)$ denotes the distance between the points $\mathbf{x}_i$ and $\mathbf{x}_j$, $w_{ij} \geq 0$ denote weights, $i, j = 1, ..., n$. The distances can be defined using different norms in $\mathbf{R}^2$, but normally a Minkowski distance (2) is used. Most frequently the special cases are considered: the Euclidean distances ($r = 2$) and the city-block distances ($r = 1$) .

Although *STRESS* function is defined by an analytical formula, which seems rather simple, its minimization is a difficult global optimization problem [8]. *STRESS* function normally has many local minima. The minimization problem is high dimensional: the number of variables is $N = 2n$. Non-differentiability of *STRESS* normally cannot be ignored, although in case of $r > 1$ gradient based optimization algorithms can be applied in vicinities of local minimizers where *STRESS* is differentiable [7]. In the case of city-block metric ($r = 1$) differentiability of *STRESS* cannot be guaranteed even at a local minimum point.

Further analysis is concentrated on MDS with city-block metric; for specific details of this version of MDS we refer to [3, 10]. *STRESS* (1) with city block distances $d_1(\mathbf{x}_i, \mathbf{x}_j)$ can be redefined as

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

Let $A(\mathbf{P})$ be a set of $\mathbf{R}^N$ such that

$$A(\mathbf{P}) = \{\mathbf{X} | x_{ik} \leq x_{jk} \text{ for } p_{ki} < p_{kj},\ i, j = 1, \ldots, n; k = 1, 2\},$$

where $\mathbf{P} = (\mathbf{p}_1, \mathbf{p}_2)$, $\mathbf{p}_k = (p_{k1}, p_{k2}, \ldots, p_{kn})$ is a permutation of $1, \ldots, n$; $k = 1, 2$.

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

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

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 following quadratic programming problem

$$\min \Bigg(-\sum_{k=1}^{2} \sum_{i=1}^{n} x_{ik} \sum_{j=1}^{n} w_{ij} \delta_{ij} z_{kij} +$$

$$+ \frac{1}{2} \Bigg(\sum_{k=1}^{2} \sum_{l=1}^{2} \sum_{i=1}^{n} x_{ik} x_{il} \sum_{t=1, t \neq i}^{n} w_{it} z_{kit} z_{lit} -$$

$$- \sum_{k=1}^{2} \sum_{l=1}^{2} \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} x_{ik} x_{jl} w_{ij} z_{kij} z_{lij} \Bigg) \Bigg)$$

$$\begin{aligned} \text{s.t.} \sum_{i=1}^{n} x_{ik} &= 0,\ k = 1, 2, \\ x_{\{j|p_{kj}=i+1\},k} - x_{\{j|p_{kj}=i\},k} &\geq 0,\ i = 1, \ldots, n-1. \end{aligned}$$

Polyhedron $A(\mathbf{P})$ is defined by linear inequality constrains, where equality constrains ensure centering excluding translated solutions.

The structure of the minimization problem (4) is favorable to apply a two level minimization [18]:

$$\min_{P} S(P),$$

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

where the upper level problem is a combinatorial problem defined over the set P of two permutations (one permutation per each coordinate of embedding space) of $1, ..., n$, and the lower level problem is a quadratic programming problem with a positively defined quadratic objective function and linear constraints. The problem at the lower level is solved using a standard quadratic programming algorithm. Globality of search is ensured by the upper level algorithms.

The number of feasible solutions of the upper level combinatorial problem is $(n!)^2$. The time required for explicit enumeration depends on the number of feasible solutions and grows very fast with increasing n. Solution of MDS with city-block metric is invariant with respect to mirroring around coordinate axes and exchanging of coordinates. The required number of explicitly enumerated solutions can be reduced taking into account such symmetries. The number is reduced to $(n!/2)^2$ refusing mirrored solutions around each coordinate axis. The number is further reduced refusing mirrored solutions with exchanged coordinates to $(n!)^2/8+n!/4$. However, it is still a large number.

For the problems of exponential complexity random heuristics normally are applied. Indeed, applicability of GA in MDS problems is claimed very promising in [5, 9, 11, 12, 15, 17]. However, this claim is not supported by quantitative results concerning accuracy in estimating the corresponding global minima. Similar problem can be noted in assessment of other metaheuristic algorithms (e.g. simulated annealing) with respect to their applicability to MDS [3, 10]. To get an objective assessment of a metaheuristic algorithm, global minima of the mentioned above combinatorial problems should be known. Generally speaking, theoretically guaranteed estimates of global minima can be obtained by enumeration but limitations on the dimensionality of practically solvable problems can be very restrictive; we attempt to lift these limitations by means of parallelization of the explicit enumeration algorithm.

3. PARALLEL IMPLEMENTATION OF GA AND EXPLICIT ENUMERATION ALGORITHM

Because of complexity, guaranteed solutions of the considered problem can be found only for the problems of modest dimensionality. For larger dimensionalities genetic algorithms seem prospective. In this case the guarantee to find the exact solution is lost, but good solutions may be found in acceptable time.

The general idea of GA is to maintain a population of best (with respect to *STRESS* value) solutions whose crossover can generate better solutions. The permutations in $\mathbf{P}$ are considered as a chromosome representing an individual. The initial population of individuals is generated randomly and improved by local search. The population evolves generating offspring from two randomly chosen individuals of the current population with the chromosomes $\mathbf{Q}$ and $\mathbf{U}$, where the first corresponds to the better fitted parent. 2-point crossover is used. The chromosome of the offspring is defined by the following formula

$$\mathbf{p}_k = (q_{k1}, ..., q_{k\xi_1}, v_{k1}, ..., v_{k(\xi_2-\xi_1)}, q_{k\xi_2}, ..., q_{kn}),$$

where $k = 1, 2$; ξ_1, ξ_2 are two integer random numbers with uniform distribution over $1, ..., n$; and v_{ki} constitute the subset of $1, ..., n$ complimentary to $q_{k1}, ..., q_{k\xi_1}, q_{k\xi_2}, ..., q_{kn}$; the numbers v_{ki} are ordered in the same way as they are ordered in $\mathbf{u}_k$. The offspring is improved by local search and its fitness is defined by the optimal value of the corresponding lower level problem. An elitist selection is applied: if the offspring is better fitted than the worst individual of the current population, then the offspring replaces the latter. Minimization continues generating new offsprings and terminates after the predetermined computing time t_c.

A parallel version of the genetic algorithm with multiple populations, similar to [4, 17] has been developed. Communications between processors have been kept to minimum to enable implementation of the algorithm on clusters of personal computers. Each processor runs the same genetic algorithm with different sequences of random numbers. This is ensured by initializing different seeds for random number generators in each processor. 60 individuals are used in every population, this value is tuned using experimental investigation in [16]. Mutation and migration were not used in this version of the algorithm, however in the future we are going to research their potential to improve effectiveness and robustness of global search. The results of different processors are collected when search is finished after the predefined time t_c. To make parallel implementation as much portable as possible the general message-passing paradigm of parallel programming has been chosen. The standardized message-passing communication protocol MPI is used for communication between parallel processors.

Solutions found by means of GA are assessed heuristically in the papers cited above as very promising. However, quantitative assessment is of great interest because of extending applications of MDS to practical problems. This can be done using exact minimum values of the considered global optimization problems found, e.g. by means of enumeration. Although the dimensionality of MDS problems solvable by means of enumeration can not be large because of exponentially growing number of potential solutions, it is important to implement and apply such an algorithm for the problems of highest possible dimensionality. Parallel computation may help us to cope with a large number of feasible solutions to be explicitly enumerated.

In the developed parallel implementation for explicit enumeration each processor runs the same algorithm generating feasible solutions which should be enumerated explicitly, but only each p-th is explicitly enumerated on a processor where p is the number of processors. The first processor explicitly enumerates the first, $(p+1)$-th and so on generated solutions. The second processor explicitly enumerates the second, $(p+2)$-th and so on generated solutions. The p-th processor explicitly enumerates the p-th, $2p$-th and so on generated solutions. It is assumed that generation of the solutions to be explicitly enumerated requires much less computational time than the explicit enumeration which requires solution of the lower level quadratic programming problem. The results of different processors are collected when the generation of solutions and explicit enumeration are finished. The standardized message-passing communication protocol MPI is used for communication between parallel processors.

4. EXPERIMENTAL INVESTIGATION

Sets of vertices of multidimensional cubes and simplices have been used as data sets of experiments. The number of vertices of d-dimensional cube is $n = 2^d$, and the dimensionality of the global minimization problem is $N = 2^{d+1}$. The coordinates of i-th vertex of a d-dimensional cube are equal either to 0 or to 1, and they are defined by the binary code of $i = 0, ..., n-1$. The number of vertices of d-dimensional simplex is $n = d+1$, and the dimensionality of the global minimization problem is $N = 2(d+1)$. The distances between any two vertices of the standard simplex are equal: $\delta_{ij} = 1$, $i \neq j$. Vertices of the unit simplex can be defined by

$$v_{ij} = \left\{ \begin{array}{ll} 1, & \text{if } i = j+1, \\ 0, & \text{otherwise}, \end{array} \right. \Bigg| \, i = 1, \ldots, \text{dim}+1, j = 1, \ldots, \text{dim}.$$

Below we use shorthand ‘cube’ and ‘simplex’ for sets of their vertices.

To compare minima found for different problems the normalized value of *STRESS* is used:

$$f(\mathbf{X}) = \sqrt{S(\mathbf{X}) \Bigg/ \sum_{i<j} w_{ij}\delta_{ij}^2}\ .$$

Performance of deterministic global optimization algorithms is measured using the optimization time t and the smallest function value found. Efficiency of the parallelization can be evaluated using standard criteria taken into account the optimization time and the number of processors. A commonly used criterion of parallel algorithms is the speedup:

$$s_p = \frac{t_1}{t_p},$$

where t_p is the time used by the algorithm implemented on p processors. The speedup divided by the number of processors is called the efficiency:

$$e_p = \frac{s_p}{p}.$$

Performance of the parallel algorithm composed of explicit enumeration of combinatorial problem and quadratic programming on SUN Fire E15k high performance computer for test problems 'simplex' with $n = 7$ is shown in Table 1. Dimensionality of the global optimization problems is $N = 14$. On a single processor optimization takes up to 20 minutes. Different numbers of processors from 1 to 24 have been used. On 24 processors optimization takes less than one minute. The speedup is almost linear and equal to the number of processors, the efficiency of parallel algorithm is close to one.

Performance of the same parallel algorithm on a cluster of personal computers is shown in Table 2 for the problems of 'simplex' with $n = 7$, $n = 8$ and 'cubes' with $n = 8$. Dimensionality of the global optimization problems is $N = 14$ and $N = 16$. The cluster is composed of 3 personal computers with 3GHz Pentium 4 processors and hyper-threading technology allowing simultaneous multithreading. When the number of processors is up to 3, the speedup is almost linear and equal to the number of processors, the efficiency is close to 1. In this case one processor per each personal computer is used. If the number of processors is larger than 3, the efficiency of the parallel algorithm is around 0.6 as at least one processor should be emulated using multithreading. Because of static distribution of workload, the efficiency is determined by the slowest element of the system. Therefore the efficiency is similar for the number of processors 4-8. The speedup of approximately 3.6 has been reached. Parallel computation yielded the speedup of approximately 3, hyper-threading yielded approximately 20% improvement. With the help of parallel computation problems with $n \leq 9$ have been solved.

In the case of stochastic global optimization algorithms, the reliability is the most important criterion. In our experiments with large scale problems, performance of GA-MDS with quadratic programming at the lower level is measured using the best estimate of the global minimum f^* in 100 runs, and the reliability is measured as percentage of runs (*perc*) when the estimate of the global minimum differs from f^* by less than 10^{-4}.

The parallel genetic algorithm has been used to solve problems with different multidimensional data. When 10s

Table 1: Performance of explicit enumeration on SUN Fire E15k parallel computer for test problems 'simplex' with $n = 7$

	standard simplex			unit simplex		
p	t, s	s_p	e_p	t, s	s_p	e_p
1	1037	1.00	1.00	1299	1.00	1.00
2	518	2.00	1.00	650	2.00	1.00
3	349	2.97	0.99	438	2.97	0.99
4	261	3.97	0.99	327	3.97	0.99
5	210	4.95	0.99	262	4.95	0.99
6	175	5.91	0.98	219	5.93	0.99
7	151	6.88	0.98	188	6.89	0.98
8	134	7.73	0.97	168	7.75	0.97
9	118	8.80	0.98	147	8.85	0.98
10	107	9.71	0.97	134	9.66	0.97
11	97	10.72	0.97	120	10.78	0.98
12	90	11.58	0.96	111	11.69	0.97
13	82	12.62	0.97	102	12.68	0.98
14	77	13.54	0.97	95	13.62	0.97
15	72	14.39	0.96	89	14.55	0.97
16	67	15.44	0.97	84	15.54	0.97
17	64	16.12	0.95	79	16.47	0.97
18	60	17.23	0.96	76	17.18	0.95
19	58	17.90	0.94	72	18.03	0.95
20	55	18.95	0.95	68	19.17	0.96
21	52	19.96	0.95	65	19.96	0.95
22	49	20.95	0.95	62	20.85	0.95
23	48	21.81	0.95	59	22.06	0.96
24	46	22.50	0.94	57	22.60	0.94

Table 2: Performance of explicit enumeration on a cluster of 3 personal computers for test problems 'simplex' and 'cube'

	$n = 7$			$n = 8$		
p	t, s	s_p	e_p	t, s	s_p	e_p
			standard simplices			
1	120	1.00	1.00	10401	1.00	1.00
2	60	1.99	1.00	5304	1.96	0.98
3	41	2.93	0.98	3554	2.93	0.98
4	49	2.43	0.61	4292	2.42	0.61
5	40	3.00	0.60	3468	3.00	0.60
6	34	3.57	0.60	2926	3.55	0.59
			unit simplices			
1	141	1.00	1.00	11821	1.00	1.00
2	71	1.99	0.99	6019	1.96	0.98
3	48	2.93	0.98	4021	2.94	0.98
4	57	2.46	0.62	4884	2.42	0.61
5	47	3.02	0.60	3915	3.02	0.60
6	39	3.64	0.61	3269	3.62	0.60
			cube			
1				12729	1.00	1.00
2				6472	1.97	0.98
3				4341	2.93	0.98
4				5199	2.45	0.61
5				4193	3.04	0.61
6				3531	3.61	0.60

Table 3: **Performance of GA-MDS on SUN Fire E15k parallel computer; $t_c = 30\text{s}/p$**

	$p = 1$		$p = 4$		$p = 8$		$p = 12$		$p = 16$	
n	*perc*	f^*	*perc*	f^*	*perc*	f^*	*perc*	f^*	*perc*	f^*
					standard simplices					
4	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**
5	100	**0.1907**	100	**0.1907**	100	**0.1907**	100	**0.1907**	100	**0.1907**
6	100	**0.2309**	100	**0.2309**	100	**0.2309**	100	**0.2309**	100	**0.2309**
7	100	**0.2621**	100	**0.2621**	100	**0.2621**	100	**0.2621**	100	**0.2621**
8	100	**0.2825**	100	**0.2825**	100	**0.2825**	100	**0.2825**	100	**0.2825**
9	100	**0.2991**	100	**0.2991**	100	**0.2991**	100	**0.2991**	100	**0.2991**
10	99	0.3115	100	0.3115	100	0.3115	100	0.3115	100	0.3115
11	95	0.3217	100	0.3217	100	0.3217	100	0.3217	100	0.3217
12	79	0.3300	100	0.3300	100	0.3300	100	0.3300	98	0.3300
13	60	0.3371	95	0.3371	86	0.3371	52	0.3371	29	0.3371
14	45	0.3429	87	0.3429	34	0.3429	6	0.3429	1	0.3429
15	35	0.3481	20	0.3481	2	0.3481	1	0.3481	1	0.3484
16	26	0.3525	7	0.3525	1	0.3527	1	0.3527	1	0.3527
17	13	0.3565	5	0.3567	5	0.3572	1	0.3569	1	0.3572
18	4	0.3599	1	0.3600	2	0.3606	1	0.3607	1	0.3609
19	1	0.3630	1	0.3636	2	0.3640	2	0.3639	1	0.3643
20	2	0.3658	1	0.3665	2	0.3671	1	0.3670	1	0.3676
21	4	0.3685	1	0.3689	1	0.3696	1	0.3698	1	0.3702
					unit simplices					
4	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**
5	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**
6	100	**0.1869**	100	**0.1869**	100	**0.1869**	100	**0.1869**	100	**0.1869**
7	100	**0.2247**	100	**0.2247**	100	**0.2247**	100	**0.2247**	100	**0.2247**
8	100	**0.2569**	100	**0.2569**	100	**0.2569**	100	**0.2569**	100	**0.2569**
9	100	**0.2759**	100	**0.2759**	100	**0.2759**	100	**0.2759**	100	**0.2759**
10	100	0.2936	100	0.2936	100	0.2936	100	0.2936	100	0.2936
11	100	0.3058	100	0.3058	100	0.3058	100	0.3058	100	0.3058
12	100	0.3167	100	0.3167	100	0.3167	100	0.3167	100	0.3167
13	91	0.3249	100	0.3249	100	0.3249	77	0.3249	62	0.3249
14	92	0.3325	100	0.3325	49	0.3325	20	0.3325	13	0.3325
15	69	0.3384	61	0.3384	4	0.3384	3	0.3384	2	0.3384
16	64	0.3439	8	0.3439	1	0.3439	1	0.3443	3	0.3443
17	36	0.3484	3	0.3487	2	0.3490	3	0.3495	1	0.3494
18	19	0.3526	1	0.3527	1	0.3533	2	0.3534	2	0.3537
19	7	0.3562	1	0.3568	2	0.3576	1	0.3576	2	0.3585
20	1	0.3595	1	0.3601	2	0.3607	2	0.3611	3	0.3615
21	1	0.3623	2	0.3634	1	0.3637	1	0.3645	2	0.3652
					cubes					
8	100	**0.2245**	100	**0.2245**	100	**0.2245**	100	**0.2245**	100	**0.2245**
16	35	0.2965	1	0.2965	1	0.2965	1	0.2974	1	0.3009

is allocated for each number of processors improvement of the reliability is significant, especially while comparing the results of single processor with results of the maximum number of processors. Although improvement of the reliability is significant, it is difficult to judge about the efficiency of parallelization. We suggest to estimate the efficiency of parallelization running the algorithm for the fixed total computing (CPU) time (opposed to wall clock time) on the different number of processors. The experiment has been performed running the parallel genetic algorithm for $t_c = 30\text{s}/p$.

The results of the experiment on SUN Fire E15k parallel computer are presented in Table 3. Global minima of some problems have been found by means of explicit enumeration; they are shown in bold font. These problems have been solved with 100% reliability by means of the genetic algorithm too. Larger problems cannot be solved in acceptable time by the algorithm with explicit enumeration, but the genetic algorithm still produces good solutions. The performance of the genetic algorithm does not depend significantly on the number of processors, when $t_c = 30\text{s}/p$.

Table 4: Performance of GA-MDS on a cluster of personal computers; $t_c = 30\mathrm{s}/p$

	$p=1$		$p=2$		$p=3$		$p=4$		$p=5$		$p=6$	
n	*perc*	f^*	*perc*	f^*	*perc*	f^*	*perc*	f^*	*perc*	f^*	*perc*	f^*
						standard simplices						
4	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**
5	100	**0.1907**	100	**0.1907**	100	**0.1907**	100	**0.1907**	100	**0.1907**	100	**0.1907**
6	100	**0.2309**	100	**0.2309**	100	**0.2309**	100	**0.2309**	100	**0.2309**	100	**0.2309**
7	100	**0.2621**	100	**0.2621**	100	**0.2621**	100	**0.2621**	100	**0.2621**	100	**0.2621**
8	100	**0.2825**	100	**0.2825**	100	**0.2825**	100	**0.2825**	100	**0.2825**	100	**0.2825**
9	100	**0.2991**	100	**0.2991**	100	**0.2991**	100	**0.2991**	100	**0.2991**	100	**0.2991**
10	100	0.3115	100	0.3115	100	0.3115	100	0.3115	100	0.3115	100	0.3115
11	100	0.3217	100	0.3217	100	0.3217	100	0.3217	100	0.3217	100	0.3217
12	86	0.3300	100	0.3300	100	0.3300	100	0.3300	100	0.3300	100	0.3300
13	68	0.3371	87	0.3371	98	0.3371	98	0.3371	100	0.3371	98	0.3371
14	57	0.3429	75	0.3429	91	0.3429	92	0.3429	95	0.3429	97	0.3429
15	38	0.3481	58	0.3481	70	0.3481	81	0.3481	86	0.3481	93	0.3481
16	20	0.3525	34	0.3525	49	0.3525	54	0.3525	76	0.3525	79	0.3525
17	14	0.3565	28	0.3565	36	0.3565	50	0.3565	52	0.3565	53	0.3565
18	9	0.3599	13	0.3599	18	0.3599	27	0.3599	30	0.3599	28	0.3599
19	8	0.3630	12	0.3630	14	0.3630	19	0.3630	16	0.3630	13	0.3630
20	4	0.3657	10	0.3657	12	0.3657	17	0.3657	17	0.3657	15	0.3657
21	4	0.3682	2	0.3682	9	0.3682	4	0.3682	6	0.3682	8	0.3684
						unit simplices						
4	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**
5	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**	100	**0.000**
6	100	**0.1869**	100	**0.1869**	100	**0.1869**	100	**0.1869**	100	**0.1869**	100	**0.1869**
7	100	**0.2247**	100	**0.2247**	100	**0.2247**	100	**0.2247**	100	**0.2247**	100	**0.2247**
8	100	**0.2569**	100	**0.2569**	100	**0.2569**	100	**0.2569**	100	**0.2569**	100	**0.2569**
9	100	**0.2759**	100	**0.2759**	100	**0.2759**	100	**0.2759**	100	**0.2759**	100	**0.2759**
10	100	0.2936	100	0.2936	100	0.2936	100	0.2936	100	0.2936	100	0.2936
11	100	0.3058	100	0.3058	100	0.3058	100	0.3058	100	0.3058	100	0.3058
12	100	0.3167	100	0.3167	100	0.3167	100	0.3167	100	0.3167	100	0.3167
13	93	0.3249	99	0.3249	100	0.3249	100	0.3249	100	0.3249	100	0.3249
14	89	0.3325	100	0.3325	100	0.3325	100	0.3325	100	0.3325	100	0.3325
15	75	0.3384	98	0.3384	99	0.3384	100	0.3384	100	0.3384	100	0.3384
16	72	0.3439	96	0.3439	97	0.3439	99	0.3439	100	0.3439	100	0.3439
17	54	0.3484	78	0.3484	84	0.3484	92	0.3484	93	0.3484	96	0.3484
18	40	0.3526	74	0.3526	87	0.3526	90	0.3526	93	0.3526	86	0.3526
19	36	0.3562	63	0.3562	71	0.3562	68	0.3562	62	0.3562	55	0.3562
20	35	0.3595	50	0.3595	70	0.3595	59	0.3595	40	0.3595	13	0.3595
21	15	0.3623	29	0.3623	44	0.3623	30	0.3623	19	0.3623	3	0.3623
						cubes						
8	100	**0.2245**	100	**0.2245**	100	**0.2245**	100	**0.2245**	100	**0.2245**	100	**0.2245**
16	89	0.2965	99	0.2965	100	0.2965	99	0.2965	100	0.2965	98	0.2965
32	8	0.3313	5	0.3314	1	0.3315	4	0.3319	1	0.3318	1	0.3348

For middle size problems the algorithm performs better on 4 processors than on a single processor. This indicates that for these problems randomness is too low when a single processor is used. For larger problems the algorithm performs better on a single processor. The performance on a larger number of processors may be improved by closer cooperation of processors, for example by interchange of the best solution found during the search.

The results of similar experiment on a cluster of personal computers are presented in Table 4. For most of the problems the algorithm performs better when more processors are used. This indicates that randomness is too low and the performance of the algorithm may be improved using mutations or using several populations even on a single processor. Only for the largest problems reliability decreases when more processors are used, indicating that for these

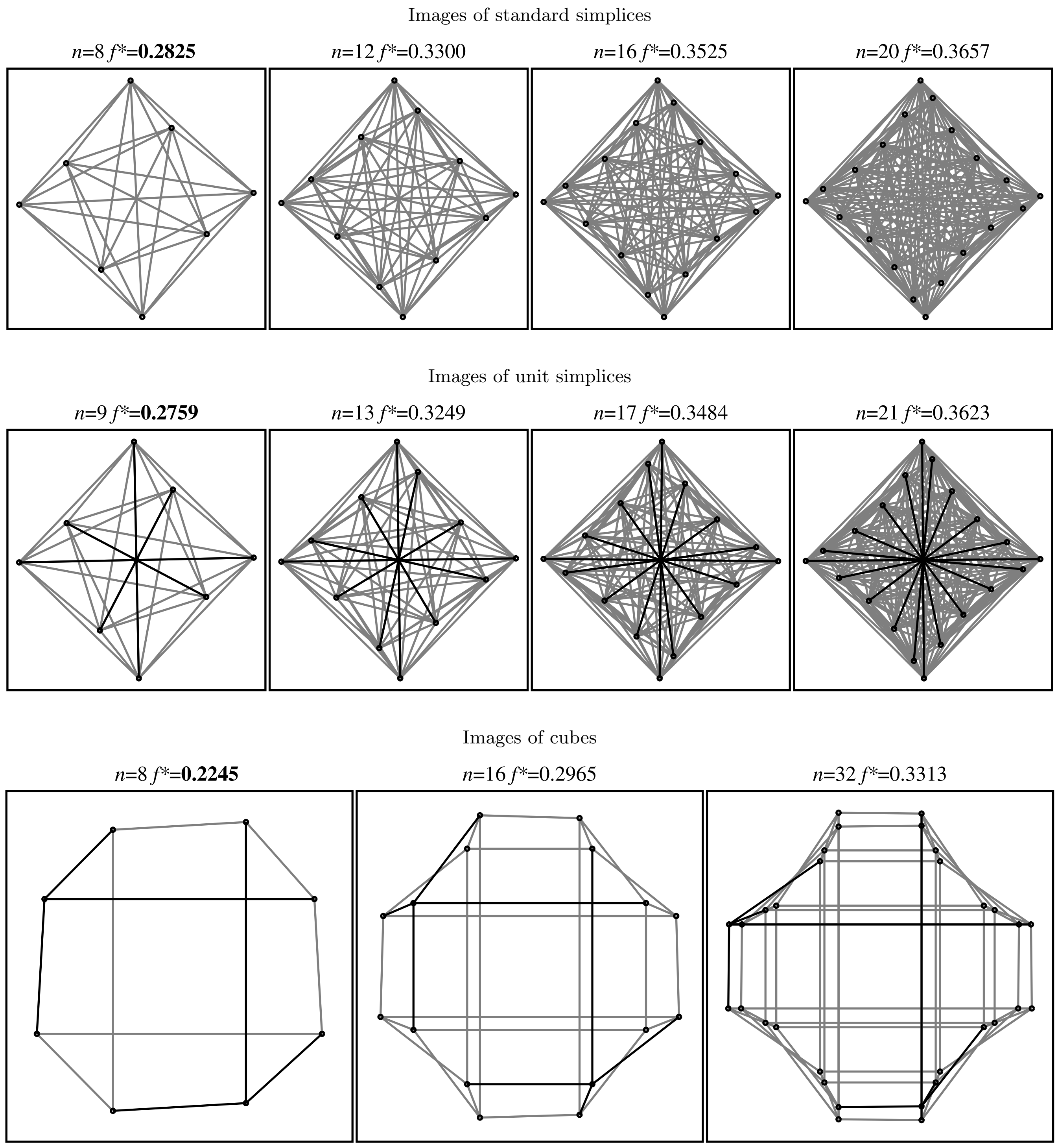

Figure 1: Images of multidimensional geometrical figures produced by GA-MDS.

problems the allowed computing time is too short when several processors are used.

In all cases GA-MDS finds the same global minima as found by explicit enumeration. Let us note that GA-MDS solves these problems in seconds, while the algorithm of explicit enumeration requires an hour on a cluster of three personal computers to solve problems with $n = 8$ and a day on a cluster of ten personal computers to solve problems with $n = 9$.

Images of multidimensional geometrical figures produced by GA-MDS are shown in Fig. 1. The images of vertices are shown by circles. To make representations more visual, adjacent vertices are joined by lines. The darker lines show joins adjacent to two opposite vertices in the case of cubes and adjacent to the 'zero' vertex in the case of unit simplices. Numbers of vertices and estimates of the global minima are given above the figures. The global minima found by means of explicit enumeration are shown in the bold font. Since the considered geometrical figures are highly symmetric, their images, corresponding to global minima of *STRESS*, are expected to be similarly symmetric. The images shown in Fig. 1 are very symmetric; only the image of 5-dimensional cube has some disturbances in the lower part of the image where oblique lines do not cross at the same point.

5. CONCLUSIONS

The problem of multidimensional scaling with city-block metric is reduced to a bilevel minimization problem with a combinatorial problem at the upper level, and a quadratic programming problem at the lower level. A parallel GA aimed for fast solution of the upper level problems is developed. The parallel GA scales well, and it finds the global minimum in the problems up to 32 variables with 100% reliability in 10 seconds on a cluster of three personal computers. To evaluate efficiency/reliability of GA for solution of MDS related global optimization problems guaranteed solutions were needed; a parallel version of explicit enumeration algorithm for the upper level combinatorial problem is developed to find a global minimum of interest with guarantee.

The developed parallel GA-MDS algorithm is sufficiently fast and reliable for many practical applications.

6. ACKNOWLEDGMENTS

The research is supported by Lithuanian State Science and Studies Foundation for the project by group of researchers T-07252 "Visualization of multidimensional data in interactive global optimization" and the NATO Reintegration grant CBP.EAP.RIG.981300.

7. REFERENCES

[1] W. Basalaj. *Proximity Visualization of Abstract Data.* http://www.pavis.org/essay, 2001.

[2] I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*, 2nd ed. Springer, New York, 2005.

[3] M.J. Brusco. A simulated annealing heuristics for unidimensional and multidimensional (city-block) scaling of symmetric proximity matrices. *Journal of Classification*, 18(1):3–33, January 2001.

[4] E. Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms.* Kluwer Academic Publishers, 2000.

[5] J.E. Everett. Algorithms for multidimensional scaling. In L.D. Chambers (ed.) *The Practical Handbook of Genetic Algorithms*, 2nd ed., pages 203–233. Chapman & Hall/CRC, 2001.

[6] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Adison-Wesley, Reading, 1989.

[7] P.J.F. Groenen, R. Mathar, and W.J. Heiser. The majorization approach to multidimensional scaling for Minkowski distances. *Journal of Classification*, 12(1):3–19, March 1995.

[8] P.J.F. Groenen. *The Majorization Approach to Multidimensional Scaling.* DSWO Press, Leiden, 1993.

[9] P. Groenen, R. Mathar, and J. Trejos. Global optimization methods for MDS applied to mobile communications. In W. Gaul, O. Opitz, and M. Schander (eds.) *Data Analysis: Scientific Models and Practical Applications*, pages 459–475. Springer, 2000.

[10] P.L. Leung and K. Lau. Estimating the city-block two-dimensional scaling model with simulated annealing. *European Journal of Operational Research*, 158(2):518–524, October 2004.

[11] R. Mathar. A hybrid global optimization algorithm for multidimensional scaling. In R. Klar and O. Opitz (eds.) *Classification and Knowledge Organization*, pages 63–71. Springer, 1997.

[12] R. Mathar and A. Žilinskas. On global optimization in two-dimensional scaling. *Acta Applicandae Mathematicae*, 33(1):109–118, October 1993.

[13] Z. Michalewich. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, Berlin, 1996.

[14] H.-P. Schwefel. *Evolution and Optimum Seeking.* John Wiley & Sons, NY, 1995.

[15] L. Tsogo, M.-H. Masson, and A. Bardot. Recovery of the metric structure of a pattern of points using minimal information. *IEEE Trans. Systems, Man and Cybernetics, Part A*, 31(1):30–42, January 2001.

[16] A. Varoneckas, A. Žilinskas, and J. Žilinskas. Multidimensional scaling using parallel genetic algorithm. In I.D.L. Bogle and J. Žilinskas (eds.) *Computer Aided Methods in Optimal Design and Operations*, pages 129–138. World Scientific, 2006.

[17] A. Žilinskas and J. Žilinskas. Parallel hybrid algorithm for global optimization of problems occurring in MDS based visualization. *Computers & Mathematics with Applications*, 52(1-2):211–224, July 2006.

[18] A. Žilinskas and J. Žilinskas. Two level minimization in multidimensional scaling. *Journal of Global Optimization*, ONLINE FIRST, October 2006.