ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2008, Vol. 37, No. 1

PARALLEL MULTIDIMENSIONAL SCALING USING GRID COMPUTING: ASSESSMENT OF PERFORMANCE

Audrius Varoneckas[1], Antanas Žilinskas[2], Julius Žilinskas[2,3]

[1]*Vytautas Magnus University, Vileikos 8, LT-44404 Kaunas, Lithuania*
[2]*Institute of Mathematics and Informatics, Akademijos 4, LT-08663 Vilnius, Lithuania*
[3]*Institute of Psychophysiology and Rehabilitation, Vydūno 4, Palanga LT-00135, Lithuania*

Abstract. Multidimensional scaling is a technique for visualization and exploratory analysis of multidimensional data aiming to discover a structure of sets of objects using information on similarities/dissimilarities between those objects. A difficult global optimization problem should be solved to minimize the error of visualization. A hybrid optimization algorithm has been constructed combining evolutionary global search with efficient local descent. A parallel version of the proposed optimization algorithm is implemented to enable solution of large scale problems in acceptable time. In the present paper we investigated the efficiency of the parallel version of the algorithm on PC clusters and computational grids.

Keywords: grid computing, multidimensional scaling, evolutionary algorithm.

1. Introduction

Multidimensional scaling (MDS) is an exploratory technique for data analysis [1, 3], widely usable in different applications, e.g. psychometrics, market analysis, data mining, visualization of general multidimensional data, visualization of observation points in interactive global optimization.

The points $\mathbf{x}_i=(\mathbf{x}_{i1},...,\mathbf{x}_{im})$, $i=1,...,n$ representing n objects in m-dimensional embedding space should be found fitting pairwise distances of points to given pairwise dissimilarities of the objects (δ_{ij}, $i,j=1,...,n$). It is supposed that dissimilarities are symmetric ($\delta_{ij}=\delta_{ji}$) and ($\delta_{ii}=0$).

The implementation of a MDS method is reduced to minimization of a fitness criterion, e.g. the so called *STRESS* function:

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

where $\mathbf{x}=(\mathbf{x}_1,...,\mathbf{x}_n)$ is a vector aggregating coordinates of points $\mathbf{x}_i=(\mathbf{x}_{i1},...,\mathbf{x}_{im})$, $d(\mathbf{x}_i,\mathbf{x}_j)$ denotes the distance between the points $\mathbf{x}_i$ and $\mathbf{x}_j$.

The distances may be estimated using different norms in $\mathbb{R}^m$. Most often a Minkowski distance is used:

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

The formula (2) defines Euclidean distances when $r=2$, and city-block distances when $r=1$. The points $\mathbf{x}_i$ defined by means of minimization of (1), but using different distances in the embedding space, can be interpreted as different nonlinear projections of the objects from the original space to the embedding space. When objects of problem are defined by multidimensional points, dissimilarities can be found estimating pairwise distances of points in the original multidimensional space.

MDS is a difficult global optimization problem. Although *STRESS* is defined by an analytical formula, which seems rather simple, its minimization is difficult. The function normally has many local minima. When city-block distances are used, *STRESS* can be non differentiable even at the minimum point [10]. The minimization problem is high dimensional (number of variables is $N = n \times m$) global optimization problem.

When computing power of usual computers is not sufficient to solve a problem, the high performance parallel computers, clusters of computers and computational grids may be helpful. An algorithm is more applicable in case its parallel implementation is available, because larger practical problems may be solved by means of parallel computation. Because of this, implementation of parallel algorithms and investigation of their efficiency on PC clusters and computational grids are considered.

2. Evolutionary Algorithm for Multidimensional Scaling

As it was shown in [6, 7, 8], the hybrid algorithm combining evolutionary global search with efficient local descent is the most reliable though the most time consuming method for MDS with Euclidean distances. Therefore a similar hybrid algorithm has been constructed.

The idea is to maintain a population of best (with respect to *STRESS* value) solutions whose crossover can generate better solutions. The size of population p is a parameter of the algorithm. An initial population is generated by performing local searches from p starting points that are the best (with respect to *STRESS* value) from a sample of N_{init} randomly generated points. The population evolves generating offspring of two randomly selected parents. Two point crossover is used. Adaptation of the offspring to environment is modelled by local search. The fitness of the offspring is defined by the locally optimal value found by local descent. Elitist survival is performed: if the offspring is better fitted than the worst individual of the current population then the later is replaced by the offspring. Minimization terminates after predetermined computing time t_c.

The structure of the hybrid algorithm with parameters (p, N_{init}, t_c):

```
Generate the initial population:
   Generate N_init random points.
   Perform search for local minima
      starting from the best p points.
   Form the initial population from the
      found local minimizers.
while not time limit t_c exceeded
   Select two uniformly random parents
      from the current population.
   Produce an offspring by means of 2-
      point crossover and local
      minimization.
   if it is better than the worst
      individual of the current
      population,
   then replace the offspring with the
      latter.
```

The upper level genetic algorithm ensures globality of search while at the lower level local descent ensures efficient search for local minima. Well known direction set algorithm by Powell has been used for local search for MDS problems with Euclidean distances. In the case of city-block distances *STRESS* is a piecewise quadratic (over simply defined polyhedra) function of $\mathbf{x}$ [10]; a special local search method has been proposed taking into account the latter property. It has been shown experimentally that these local search strategies perform best in MDS with Euclidean and city-block metrics.

Parallel version of genetic algorithm with multiple populations [2] has been developed. Communications between processors have been kept to minimum to enable implementation of the algorithm on clusters of personal computers and computational grids. 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. The results of different processors are collected when search is finished after predefined time. To make parallel implementation as much portable as possible the general message-passing paradigm of parallel programming has been chosen. A standardized message-passing communication protocol MPI is used for communication between parallel processors.

3. Experimental investigation in grid environment

Grid computing [4] is emerging as new paradigm for distributed problem solving for a wide range of application domains. Supercomputers, high performance computing clusters, clusters of personal computers are distributed and independent but, on the other hand, form a large scale dynamic grid environment.

When assigning tasks onto the grid computing resources, users can perform manual look-up and select resources or let the central scheduler select available resources. Since grid environment is often dynamic, the same computational conditions will never occur.

3.1. Test bed

We have used standard C++ and MPI library to implement the parallel algorithm for multidimensional scaling. The algorithm is available under BalticGrid Special Interest Groups portal (*http://sig.balticgrid.org*).

Test bed, used in our experiments, is a heterogeneous cluster environment. It contains eight high-performance clusters. Their configuration is shown in Table 1. gLite middleware [5] is installed on each cluster.

3.2. Experimental investigation

To exclude the impact of number of objects and of used metric, a relative error

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

is used for comparison. Performance of the global optimization algorithm for multidimensional scaling 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}.

Table 1. Summary of the grid environment considered for experimental investigation of the algorithm for multidimensional scaling

Cluster	Number of CPU(s)	CPU Type	Memory
KTU-BG-GLITE	41	Intel PIII 700MHz	5.24Gb
KTU-ELEN-LCG2	10	Intel P4 3GHz	3Gb
SU-GRID	15	Intel P4 1.7GHz	7.68Gb
VU-MIF-LCG2 (grid6)	112	AMD Opteron 2.4GHz	224Gb
CYFRONET-LCG2	264	Intel Xeon 2.8GHz	264Gb
VDU-IF-LCG2	22	Intel P4 3 GHz	12Gb
VU-MIF-LCG2 (grid5)	25	Intel PIII 1GHz	25Gb
RTUETF	20	AMD Opteron 2GHz	20Gb

3.3. Data sets

Several sets of multidimensional points corresponding to well understand geometric objects were used for the experimental investigation. We want to choose difficult test problems, i.e. difficult to visualize geometric objects. The data with desired properties correspond to the multidimensional objects equally extending in all dimensions of the original space, e.g. sets of vertices of multidimensional cubes and simplices. Dissimilarity between vertices is measured by the distance in the original vector space defined by its metric. Global optimization problems of different difficulty can be constructed by defining dimensionality of the original spaces. Below we use shorthand 'cube' and 'simplex' for sets of their vertices.

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

Vertices of multidimensional simplex can be defined by

$$v_{ij} = \begin{cases} 1, & if\ i = j+1, \\ 0, & otherwise, \end{cases}$$

$$i = 1,...,\dim+1,\ j = 1,...,\dim.$$

Dimensionality of this global minimization problem is $N = 2 \times (\dim + 1)$.

3.4. Results

For experiments a parallel version of the hybrid algorithm was used with parameters chosen according to the results of [9]. The experimental data obtained in the grid environment are presented in Table 2, where proc denotes the number of processors. Performance improvement is significant, especially comparing single processor results with ten processors results. Dimensionality of the original spaces essentially influences the complexity of the corresponding global optimization problem.

The experimental results show variable performance depending on the cluster used. However one may expect more variable performance: the performance of the algorithm on a cluster of PCs with Intel P4 3GHz CPUs (KTU-ELEN-LCG2, VDU-IF-LCG2) is just a bit better than that on a cluster of PCs with Intel PIII 700MHz-1GHz CPUs (VU-MIF-LCG2 (grid5), KTU-BG-GLITE).

Although improvement of reliability using parallel computation is significant, it is difficult to judge about the efficiency of parallelization. The efficiency of parallelization of the algorithm is investigated by comparing the performance when total computing time is the same for all cluster configurations: t_c=10s/(number of processors).

Such results on supercomputer of grid environment are presented in Table 3. The percentage of runs finding the best known solution (perc) when total computing time of the algorithm is t_c = 10s/(number of processors) decreases a bit or remains approximately the same when the number of processors is increased.

4. Conclusions

The considered algorithm combining evolutionary global search with properly chosen local minimization is well scalable with respect to parallelization.

Performance improvement using parallel version of the algorithm is significant for all considered problems comparing with the performance on single processor.

Performance of the parallel algorithm depends on the computer cluster used. Therefore it is not always good to rely on the central scheduler of grid resources. When assigning tasks onto the grid computing resources users can perform manual look-up and selection.

Acknowledgements

The research is supported by Lithuanian State Science and Studies Foundation, LitGRID programme and the NATO Reintegration grant CBP.EAP.RIG. 981300.

Table 2. Performance of the parallel algorithm for multidimensional scaling on grid clusters. City-block distances, data sets of multidimensional cubes are used. Algorithm parameters: $p = 60$, $N_{init} = 6000$, $t_c = 10$, number of algorithm execution = 100

	dim	*f**	1proc	2proc	3proc	4proc	5proc	6proc	7proc	8proc	9proc	10proc
KTU-BG-GLITE	3	0.224472	100	100	100	100	100	100	100	100	100	100
	4	0.296531	100	100	100	100	100	100	100	100	100	100
	5	0.331341	30	48	60	78	85	87	92	93	94	97
	6	0.351327	7	7	11	16	16	23	27	29	33	38
KTU-ELEN-LCG2	3	0.224472	100	100	100	100	100	100	100	100	100	100
	4	0.296531	100	100	100	100	100	100	100	100	100	100
	5	0.331337	74	47	56	70	91	99	100	100	91	95
	6	0.351388	16	7	7	10	51	53	63	68	27	34
SU-GRID	3	0.224472	100	100	100	100	100	100	100	100	100	100
	4	0.296531	100	100	100	100	100	100	100	100	100	100
	5	0.331334	62	73	85	86	93	98	99	99	100	100
	6	0.351401	14	13	21	32	35	49	52	59	59	62
VU-MIF-LCG2 (grid6)	3	0.224472	100	100	100	100	100	100	100	100	100	100
	4	0.296531	100	100	100	100	100	100	100	100	100	100
	5	0.331335	82	95	100	100	100	100	100	100	100	100
	6	0.351366	18	21	31	42	52	63	68	75	84	85
CYFRONET-LCG2	3	0.224472	100	100	100	100	100	100	100	100	100	100
	4	0.296531	100	100	100	100	100	100	100	100	100	100
	5	0.331337	60	88	92	98	100	100	100	100	100	100
	6	0.351375	13	22	32	44	52	60	62	69	75	77
VDU-IF-LCG2	3	0.224472	100	100	100	100	100	100	100	100	100	100
	4	0.296531	100	100	100	100	100	100	100	100	100	100
	5	0.331334	59	77	76	92	90	97	94	97	99	90
	6	0.351401	14	12	21	33	23	47	24	56	65	31
VU-MIF-LCG2 (grid5)	3	0.224472	100	100	100	100	100	100	100	100	100	100
	4	0.296531	100	100	100	100	100	100	100	100	100	100
	5	0.33134	32	61	61	72	83	97	83	98	96	99
	6	0.351449	8	10	7	9	14	23	25	38	36	37
RTUETF	3	0.224472	100	100	100	100	100	100	100	100	100	100
	4	0.296531	100	100	100	100	100	100	100	100	100	100
	5	0.331337	61	86	95	100	98	99	100	100	100	61
	6	0.351381	13	22	27	41	44	55	62	66	73	13

Table 3. Performance of the parallel algorithm for multidimensional scaling on VU-MIF-LCG2 (grid6) cluster. Algorithm parameters: $p = 60$, $N_{init} = 6000$, $t_c = 10/$(number of processors), number of algorithm execution = 100

dim	f^*	1proc	2proc	4proc	5proc	7proc	8proc	9proc
	Euclidean distances, multidimensional cubes							
3	0.243852	100	100	100	100	100	100	100
4	0.300323	100	100	100	100	100	100	100
5	0.332046	100	100	100	100	100	100	100
6	0.350552	86	86	94	89	94	94	97
7	0.362889	48	69	91	93	97	97	97
8	0.382309	47	69	1	1	2	2	2
	city-block distances, multidimensional cubes							
3	0.224472	100	100	100	100	100	100	100
4	0.296531	100	100	100	100	100	100	100
5	0.331364	82	78	81	82	74	71	74
6	0.351512	16	12	19	16	23	21	16
	Euclidean distances, multidimensional simplices							
16	0.359307	100	100	100	100	100	100	100
17	0.362790	100	100	100	100	100	100	100
18	0.365953	100	100	100	100	100	100	100
19	0.368722	100	100	100	100	100	100	100
20	0.371271	100	100	100	100	100	100	100
31	0.388863	100	100	100	100	100	100	100
	city-block distances, multidimensional simplices							
16	0.348424	75	95	100	94	68	69	68
17	0.352688	58	82	47	38	28	28	26
18	0.356216	59	82	26	18	11	17	10
19	0.359641	38	28	4	2	4	4	3
20	0.362490	29	18	3	2	5	4	2
31	0.382309	2	8	10	19	8	8	7

References

[1] **I. Borg, P. Groenen.** Modern Multidimensional Scaling. *Springer, New York,* 2005.

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

[3] **T. Cox, M. Cox.** Multidimensional Scaling. *Chapman and Hall/CRC, Boca Raton,* 2001.

[4] **I. Foster, C. Kesselman** (Eds.). The Grid: Blueprint for New Computing Infrastructure. *Morgan Kaufman Publishers,* 1999.

[5] **gLite:** *http://glite.web.cern.ch/glite/*

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

[7] **R. Mathar.** A hybrid global optimization algorithm for multidimensional scaling. In *R. Klar, O. Opitz (Eds.), Classification and Knowledge Organization. Springer, Berlin,* 1996, 63–71.

[8] **R. Mathar, A. Žilinskas.** On global optimization in two dimensional scaling. *Acta Applicandae Mathematicae,* 1993, 33, 109–118.

[9] **A. Varoneckas, A. Žilinskas, J. Žilinskas.** Multidimensional scaling using parallel genetic algorithm. *In: I.D.L. Bogle, J. Žilinskas (Eds.) Computer Aided Methods in Optimal Design and Operations. Vol.*7 *of Series on Computers and Operations Research, World Scientific,* 2006, 129–138.

[10] **A. Žilinskas, J. Žilinskas.** Two level minimization in multidimensional scaling. *Journal of Global Optimization,* 2007, 38, 581–596.

Received December 2007.