DOI: 10.2478/awutm-2014-0009

DE GRUYTER OPEN

Analele Universităţii de Vest,
Timişoara
Seria Matematică – Informatică
LII, 1, (2014), 141– 156

Experiments and Recommendations for Partitioning Systems of Equations

Liviu Octavian Mafteiu-Scai

Abstract. Partitioning the systems of equations is a very important process when solving it on a parallel computer. This paper presents some criteria which leads to more efficient parallelization, that must be taken into consideration. New criteria added to preconditioning process by reducing average bandwidth are proposed in this paper. These new criteria lead to a combination between preconditioning and partitioning of systems equations, so no need two distinct algorithms/processes. In our proposed methods - where the preconditioning is done by reducing the average bandwidth- two directions were followed in terms of partitioning: for a given preconditioned system determining the best partitioning (or one as close) and the second consist in achieving an adequate preconditioning, depending on a given/desired partitioning. A mixed method it is also proposed. Experimental results, conclusions and recommendations, obtained after parallel implementation of conjugate gradient on IBM BlueGene /P supercomputer- based on a synchronous model of parallelization- are also presented in this paper.

AMS Subject Classification (2000). 65F50, 65F10, 68T, 08, 68R10, 65F15
Keywords. average bandwidth, system of equations, parallel, preconditioning, partitioning, load-balancing, conjugate gradient, recommendation system

1 Introduction

A lot of engineering applications involves solving large sparse linear systems of equations. Parallel computing has been imposed because in the current real problems, the systems of equations have dimensions of millions equations [5] which require the use of high performance computing (HPC) systems, fact that enables to speedup the rate of performance.
Solving linear system of equation by direct methods is a good choice in the case of small systems, but not for large systems due to their relatively high memory and computational requirements. Iterative methods are good for large systems of equations but on the other hand they are more dependent on the properties of the systems. Furthermore, the iterative methods require systems preconditioning if we want to obtain a good approximations of the solution in a short time. The preconditioning consists in transforming the associated system matrix into one that is more favorable to solving process. We consider that a preconditioner is good if improves the convergence of the iterative method, sufficiently to overcome the extra cost of applying it. The importance of preconditioning becomes even more important in the case of parallel solving and there are many studies that show this [3, 4, 33, 37].
One of the very important elements in parallel computing is the "load-balancing". This was first introduced by Shivaratry in paper [38], where are proposed and compared some load-balancing strategies. Load-balancing increase the performance of parallel computer applications by reduction the processor idle time and interprocessor communication time. Therefore, the ideal case is that in which all processors contain the same amount of computational work, and does not exist data dependencies between processors. Because the ideal cases are rare, the goal remains to minimize these two factors for a good load-balancing of processors.
From another point of view, because many problems are often expressed as graphs, the load-balancing problem can be seen as a graph partitioning: vertices represent the data and edges represent relationships between data/vertices. Thus, in the case of weighted graph, the main goals of partitioning are to assign equal total vertex weight to partitions and to minimize the weight of cut edges. There are many algorithms that fiind good partitionings based on graph theory [10, 19–21, 34–36]. But the consideration in which each vertex represent an equal amount of work is a limitation of graph teory application in load-balancing [18]. Another limitation lies in the type of equations systems that can be represented, ie only systems with square and symmetric matrices [22]. To solve these shortcomings, hypergraphs are used as models for partitioning in parallel computing [6, 7, 17, 23].

New methods of partitioning in parallel computing, inspired from artificial intelligence began to be proposed lately, methods that in many cases outperforms the conventional heuristics methods : [11, 12, 25, 27, 40].
There are many software package used for partitioning and load-balancing in parallel computing some of these being:
-JOSTLE: a library for multilevel graph partitioning and load balancing developed at the University of Greenwich, London, UK (http://staffweb.cms.gre.ac.uk/ c.walshaw/jostle/), used for mesh-based parallel scientific computing applications and load-balancing, on distributed memory parallel computers;
-PSPIKE (Parallel General Sparse Linear System Solver): a software package solver for parallel solution of large sparse linear systems (http://www.pspike-project.org/), that use matrix reordering and partitioning routines;
-METIS: a software for unstructured graph partitioning and sparse matrix ordering system developed at University of Minnesota (http://www.cs.umn.edu/karypis), that include some partitioning method like: spectral bisection, multilevel spectral bisection, multilevel partitioning, geometric partitioning, geometric partitioning-KL etc.
So, in parallel solving systems of equations the load-balancing of processors is also a very important factor. This isn't an easy problem, with two main issues: how to partition the data between processors and how to parallelize the iterations in case of iterative methods. There are three ways of matrices partitioning: by rows, by columns or by blocks: the first two are suitable for distributed memory architecture while the last is used in vector processors.
We consider that an important cause of bad load-balancing in parallel solving of system equations is the dynamism of the process over time, namely in computational and communication costs, such as, for example the condition numbers of equations subsystems. Therefore, along the time, numerous methods for dynamic load-balancing have been proposed [8, 9, 15, 16, 42], a part of the benefits and limitations of dynamic partitioning across a wide range of parallel system environments being described in the paper [39].
One of the most used iterative methods for solving systems of linear equations is the conjugate gradient (CG) that is very effective when the associated matrix of system is symmetric and positive definite. The method was proposed by M.R. Hestenes, and E. Stiefel in [6], being seen as a special case of Gaussian elimination. The method involves small errors, exact solutions are generally obtained after at most n steps, where n is the size of a well conditioned system of equations. We have rapid convergence in case of well conditioned systems, but can be arbitrary if the matrix is ill conditioned. In Krylov type iterative methods, the convergence decreases or is lost after parallelization of these methods -compared with serial variants of these-, the least affected seem to be the CG, as was shown in [33].

For conjugate gradient algorithm, the main effort consists in computing the matrix-vector product and because it is directly proportional to the number of nonzero values in the matrix, an implementation of a load balanced distribution corresponds to an uniform non-zero elements distribution between processors. The parallelism in the conjugate gradient algorithm is derived from parallel matrix-vector product and other inner products. The rest of the operations involved are trivial in relation to them. The operations can be performed in parallel as long as no dependency between them. For example, the updating of the residual vector and the vector solution does not depend on each other and can be performed at the same time. But these operations can not be performed before performing the matrix-vector product. And matrix-vector product in a new iteration can not be performed until the residual vector is updated. So, there are two points on which processors must synchronize before they can move on to the next iteration. It is very important that the work be balanced between processors such synchronization points so that the processors does not have any periods of inactivity beetwen two moments (ideally) or these periods to be as smallest possible. Thus, minimizing this waiting/idle time is an important goal in parallel solving systems of equations.
In conclusion, the problems to be solved in parallel solving systems of equations are related to finding an suitable division of the processes to be performed, as well as finding the most appropriate mechanisms for synchronization and communication between different processes. The system equations partitioning can be done staticaly or dinamicaly: in first case the way to partitioning is fixed and is independent of variable system status.

2 Theoretical considerations and experiments

Partitioning the systems of equations is a very important process when solving it on a parallel computer and more criteria must be taken into consideration in this case. In our study, the main criterion considered was the indicator average bandwidth. As shown in the papers [28, 29, 31], the optimization of this indicator leads to a more uniform distribution of non-zero elements *around* and *along* the main diagonal of the associated matrix. This distribution, somewhat similar to a band, ensures a better load balancing and the use of a larger number of processors, which leads to more efficient parallelization, as mentioned in [1, 2].
As will be shown further, additional criteria added to the average bandwidth optimization process lead to better results in terms of execution time, in case

of iterative methods that are based on the Jacobian, like conjugate gradient. We consider the case of parallel solving a system of equations using Kyrlov type iterative methods (Newton, conjugate gradient or preconditioned conjugate gradient), a synchronous model of parallelization and a row-blocks matrix partitioning with equals partitions. In such a case it is desirable that inside of diagonal blocks $Ji(x), i = 1, ..., p$ (p is the number of partitions) of the Jacobian $J(x)$ to be as many nonzero values (nonzero coefficients of equations system unknowns) for a greater efficiency of computing process (a smaller number of communication processes). At the same time it is desirable that blocks $Ji(x)$ to contain close values of the number of nonzero, to have a better balance of processors. In the ideal case -synchronous model, equal row-blocks partitioning and iterative methods that using the Jacobian- all nonzero elements should be distributed inside of diagonal submatrices with size k situated along the main diagonal, where $k = n/p$, n being the size of the system and p the number of partitions, like in Figure 1. It is obvious that in such a situation when must be solved p independent subsystems of equations, the efficiency parallelization process is high. How such an ideal situation is difficult/impossible to obtain in practice, we consider that a reasonable solution is to bring as many nonzero elements inside of diagonal submatrices, that can lead to improved performance.

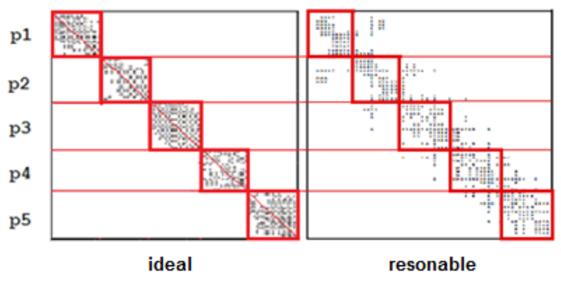


Figure 1: Ideal and reasonable nonzero elements distribution

2.1 A few factors that influence the efficiency of parallelization

In papers [28, 29] was proposed a new indicator called average bandwidth (mbw), used in preconditioning systems of linear equations. Its relevance in parallel solving systems of linear equations was shown in paper [31]. It has also been shown in [31] that in some cases such preconditioning by reducing the average bandwidth leads to a drastic decrease of the efficiency of parallel computing process in terms of convergence and global runtime. To note that this unwanted effect was also observed in the case of preconditioning

Iterations needed for convergence	before *mbw* reduction	after *mbw* reduction
Partitions (processors)		
2	500	527
4	442	**434**
5	441	697
10	577	**429**
20	455	456
25	474	**417**
50	482	519

Table 1: Iterations needed for convergence

by bandwidth reducing. In the following example is shown such a case for a 100x100 linear system of equations, symmetric matrix with 1182 nonzero values, solved with parallel conjugate gradient method for an accuracy equal with e-30 and more partitioning on IBM Blue Gene /P supercomputer. Experimental results obtained for different partitioning in terms of convergence rate, before and after average bandwidth reduction can be seen in Figure 2 and Table 1. As can be seen in Table 1, the partitioning by 2, 5, 20 and 50 processors are not favorable. Experiments and further study have resulted

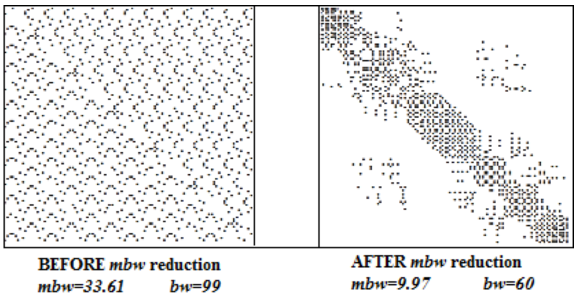


Figure 2: Matrix configuration before and after average bandwidth reduction

in the identification of several factors that can led in some conditions to these undesired effects. The main factors, experimental identified by us are:
a) distribution of non-zero elements along and around the main diagonal of the associated matrix of the equations system;
b) diagonal-blocks configurations ie the ratio between the number of nonzero elements inside and outside of Jacobian sub-matrix for each partition in part;
c) condition numbers of diagonal-blocks submatrix;
d) number of partitions.

2.2 Proposed solutions for improving the partitioning

In the following will be considered a synchronous model of parallelization, row-blocks matrix partitioning with equals partitions and Krylov type iterative methods that using the Jacobian, ie conjugate gradient. Below are decribed our proposals for a more efficient parallelization, in terms of the factors mentioned before:

a) distribution of non-zero elements in associated matrix of system:

The non-zero elements distribution plays an important role in efficient solving of equations systems in parallel. From our point of view is of interest the distributions of nonzero elements along and around the main diagonal of the associated matrix of the equations system:
- around the main diagonal: it is desirable that as many elements to be as close to the main diagonal [1, 2, 37], which is in fact a smaller value for average bandwidth [28, 29]. This will ensure the efficiency of the parallel computing mainly by the possibility of using a larger number of processors.
- along the main diagonal: this should to be as uniform as to ensure a more load balanced of processors. These improvements can be achieved by reducing the average bandwidth, using for example algorithms described in [29]. Figure 3 shows an example of the average bandwidth reduction compared with bandwidth reduction [14], using the same initial matrix.

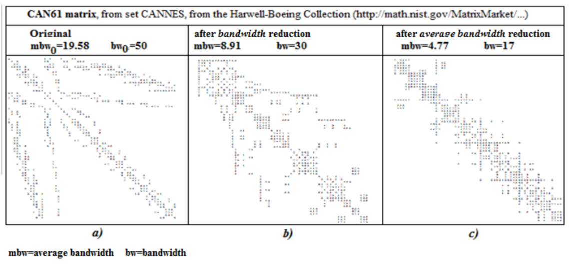

CAN61 matrix, from set CANNES, from the Harwell-Boeing Collection (http://math.nist.gov/MatrixMarket/...)
Original
mbw_0=19.58 bw_0=50
after *bandwidth* reduction
mbw=8.91 bw=30
after *average bandwidth* reduction
mbw=4.77 bw=17
a)
b)
c)
mbw=average bandwidth bw=bandwidth

Figure 3: The average bandwidth relevance

b) diagonal-blocks configurations:

In our approach were followed two directions:
- a good ratio of nonzero elements inside and outside of diagonal blocks, for each partition in part ie. inside as much, outside as few, for a smaller number of communication processes;
- diagonal blocks to contain close values of the number of nonzeros, to have a better load balancing of processors.

As has been mentioned, the *ideal solution* lies in bringing all nonzero values inside of diagonal blocks and a *reasonable solution* is to bring as many non-zero values inside of diagonal blocks, as can be seen in Figure 1.

Our solution in this respect consists in improving the average bandwidth reduction algorithms by adding additional decision criteria for

lines/columns interchange operations. Several criteria for determining the interchange opportunity of lines/columns in the preconditioning process by average bandwidth reduction in sparse matrices have been described in the paper [30]. A new one, especially designed for preconditioning by reducing average bandwidth, correlated with increasing the number of non-zero elements inside the diagonal blocks, in the case of systems parallel solving is further proposed. Figure 4 shows a small example for an easy understanding of our proposed approach. There is represented an associated matrix of a 12x12 system of equations, where the matrix elements are identified by numbers between 1 and 144. It is desired a preconditioning process by reducing the average bandwidth so as to be maximized the number of nonzero elements inside jacobians. In the Figure 4 is analyzed the interchange opportunity of lines/columns (1,3). The matrix elements whose position is affected by the interchange are colored. Different colors were used for highlighted the new positions of each group in part. As can be seen we have eight distinct groups and each group can have null and nonzero elements, depending on the initial configuration of the matrix. The four groups are with positive contribution to the number of non-zero elements inside submatrix blocks $(C1, C2, C3, C4)$ and the other four $(C5, C6, C7, C8)$ are with contributions to the number of non-zero elements outside the submatrix blocks. The first four represent optimization and the last four represent non-optimization.
The condition to improving the initial state is
$C1 + C2 + C3 + C4 > C5 + C6 + C7 + C8$ (1)
which means that the total nonzero values on the left side of inequality must be greater than the number of non-zero values on the right side. In the case of symmetric matrices, equation (1) becomes:
$C1 + C3 > C5 + C7$ (2)
Should be noted that the proposed method allows to determine how will be affected the number of nonzero values from inside of diagonal blocks by an interchange lines/columns (i, j), without to perform an effective interchange, just using the relation (3):
$C = C1 + C2 + C3 + C4 - C5 - C6 - C7 - C8$ (3)
ie only a positive value of C represent an interchange opportunity.
Note: method is simple and efficient because an interchange (i, j) affects only partitions which includes lines i and j of a sparse matrix, as can be seen in Figure 4.
In our experiments, in order to select the most favorable interchanges, was used a greedy selection of lines, selection that depend on the num-

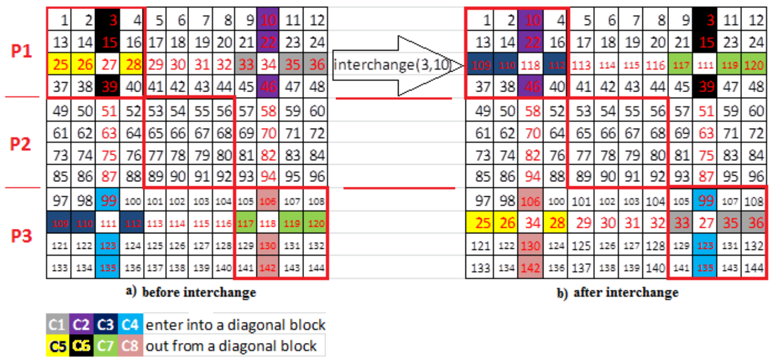

P1
P2
P3
interchange(3,10)
a) before interchange
b) after interchange
C1 C2 C3 C4 enter into a diagonal block
C5 C6 C7 C8 out from a diagonal block

Figure 4: Example for interchange opportunity in parallel case

bers of nonzero values inside and outside of partition for each line in part.

c) condition numbers of diagonal blocks:

It is well known that the condition number has a major influence in terms of convergence speed, therefore the execution time, both in the case of serial and parallel [33, 37, 41]. It is evident that in the case of parallel solving, the diagonal submatrix condition numbers influence the local convergence for each subprocess in part and implicitly the global convergence. We consider that this local influences are responsible for the overall decrease convergence in some partitioning cases even if the system of equations was preconditioned, as was highlighted in [31]. Such an unfavorable change of local condition numbers after a preconditioning process is exemplified in Figure 5. It was experimentally observed that the condition numbers of diago-

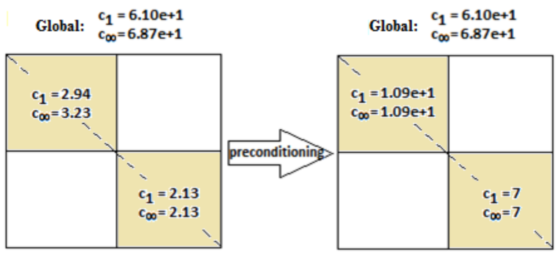

Global: c_1 = 6.10e+1 c_∞ = 6.87e+1
c_1 = 2.94 c_∞ = 3.23
c_1 = 2.13 c_∞ = 2.13
preconditioning
Global: c_1 = 6.10e+1 c_∞ = 6.87e+1
c_1 = 1.09e+1 c_∞ = 1.09e+1
c_1 = 7 c_∞ = 7

Figure 5: Preconditioning influence on the local condition numbers

nal blocks are influenced by the:
- preconditioning by average bandwidth reducing process (in some cases the preconditioning process increases the condition numbers of diagonal blocks involved in interchange, as seen in Figure 5);

- spectrum of nonzero values in each diagonal blocks;
- number of partitions.

Our proposed solution consists in improving the average bandwidth reduction algorithms by adding additional decision criteria for lines/columns interchange operation, ie. the interchange will be made only if the condition numbers of diagonal-blocks involved in interchange do not increase by more than an order of magnitude, because it was experimentally observed that increasing less than an order of magnitude does not significantly influence the convergence.

2.3 The proposed partitioning methods

In our approach, where the preconditioning is done by reducing the average bandwidth, two directions were followed in terms of partitioning:
- for a given preconditioned system of equations, the goal consist in determining the best partitioning or one as close/satisfactory;
- achieving an adequate preconditioning, depending on a given/desired partitioning.

Finaly, a mixed solution for partitioning in the case of conjugate gradient is proposed.

2.3.1 Determining partitioning for a given preconditioned system

For a given preconditioned system of equations, it is aimed determining the best partitioning or one as close/satisfactory. This goal can be achieved using:

a) a formula:

Experimentally was observed that around 20 percent of cases after preconditioning by reducing average bandwidth (mbw), -a better (5 percent) or a closer (15 percent)-, the number of partitions p is given by the formula:
$p = n/mbw$ (4)
where mbw is the average bandwidth and n is the size of equations system.

<u>Note:</u> We consider that the small successful percentage is due to:
- the value with decimals obtained for p (p value must be integer);
- working with equal partitions;
- condition numbers of the diagonal blocks;
- the ratio nonzero/zero inside/outside of the diagonal blocks.

b) a technique from artificial intelligence:

The prototype of a recommendation system for partitioning in parallel conjugate gradient, based on feature vectors dissimilarity using the indicator Reference Distance Weighted [32] was tested and presented in Figure 6. A

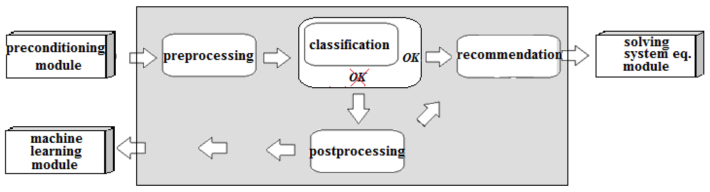


Figure 6: Recommendation system prototype

summary of the main processing is done next. *Preprocessing* represents the feature extraction from the associated matrix, such as dimension, average bandwidth, bandwidth, profile, patterns of nonzero etc. *Classification* is the process by which the system of equations is included in one of the classes existing in knowledge base of the recommendation system. *Postprocessing* is called if the classification process has failed. *Postprocessing* represents the process of "adjustment" of some features in order to approximate the problem to be solved by one of the existing classes in the knowledge base. Such adjustable characteristics could be: bandwidth, average bandwidth, profile etc. After this, the *machine learning module* is called, to extend an existing class or create a new one. Finally, a *recommendation for partitioning* is done and then the *parallel solving module* is called.
Experimentally was observed that around 5 percent, we can find a good partitioning using the proposed prototype. We consider that the small successful percentage and the prohibitive runtime (even for small dimension of systems) is due to: too small knowledge base, too few features used in the features vectors, too little knownledge about the relevance of each feature in part, the large number of parameters which must be calculated and the condition numbers of the diagonal blocks which can not be known apriori.

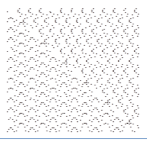
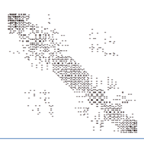
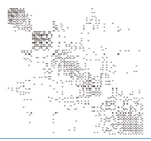
2.3.2 Preconditioning according to a given partitioning

In this approach, for a given/desired partitioning it is aimed achieving the best possible preconditioning.
Experimentally was observed that around 50 percent, we can find a good pattern of matrix, based on average bandwidth reduction conducted by:
- row-blocks reconfiguration (to increase the number of nonzero inside and decrease them outside)
- condition numbers of diagonal blocks involved in lines/columns interchange;

About last criteria we have a dilemma: computing OR estimation the condition numbers for the two blocks affected by interchange? It is well known that computing the condition number its a hard problem from runtime point of view. At the same time, there are many methods for estimating the condition number, that determine in a reasonable time an approximate value of the condition number [13, 24, 26, 41]. But the approximative value obtained by these methods (the accuracy is not as precise as the general direct computation methods) it is not satisfactory in our case because for example, we need to compare two exactly values of condition numbers for the same diagonal submatrix, if the preconditioning by reducing average bandwidth is guided by decreasing the condition numbers.

2.3.3 A mixed solution for partitioning in the case of conjugate gradient

Further is described a mixed solution obtained from the two previous models, an approach that has the main steps:
*Step*1 : average bandwidth reduction without regard to partitioning;
*Step*2 : computing the number of processors using formula $p = n/mbw$;
*Step*3 : improvement previous p partitioning through reconfiguration matrix pattern ie. average bandwidth reduction guided by row-blocks reconfiguration and reduction the condition numbers of these.
An example of running of such a model can be viewed in Figure 7. The effectiveness of the proposed model is proved by decreasing the number of iterations required for convergence. Also, is observed the relevance of the two additional criteria added to preconditioning process by reducing the average bandwidth: condition numbers and diagonal blocks patterns optimization. So, in Figure 7 can be observed that in first stage, only average bandwidth reduction (Step 1) conduct to a value of partitions number equal with 10. After this, a new average bandwidth reduction conducted by row-blocks configuration and condition numbers, lead to a new configuration of the matrix, which provides a faster convergence in the case of 10 partitions, even if the new value of average bandwidth (mbw) has increased. Can be also observed the decrease of local condition numbers, which explains the increase of convergence speed.

Example: 100x100 1182 nonzero uniform distrib. PCG e-30			

State	Initial
	mbw=33.61 bw=99
		Only mbw optimization
		mbw=9.97 bw=60
		p=100/9.97≈10
			mbw (diag.blocks + cond.numbers)
			optimization for 10 partitions
			mbw=13.19 bw=73

Nonzero in diag. blocks(10 part.)	10 12 14 6 12 10 8 14 10 12 = 108	74 32 44 30 44 46 40 34 42 72 = 458	72 28 72 24 40 32 40 30 44 48 = 430
Cond. Numb.: *Example for p1*	Fail(singular matrix) 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 14 0 0 0 0 0 0 0 0 14 0 7 0 0 0 0 0 0 0 15 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 14 15 0 0 0 0 0 0 0 14 0 0 17 0 0 0 0 0	$c_1=c_\infty$ =6.89e+4 0 3 4 16 1 7 10 0 12 0 3 0 13 6 10 16 0 0 2 0 4 13 0 7 11 17 1 10 3 10 16 6 7 0 4 10 13 3 15 0 1 10 11 4 0 14 17 7 0 7 7 16 17 10 14 0 4 13 6 13 10 0 1 13 17 4 0 16 9 16 0 0 10 3 7 13 16 0 18 0 12 2 3 15 0 6 9 18 0 18 0 0 10 0 7 13 16 0 18 0	$c_1=c_\infty$ =3.12e+2 0 14 17 8 1 4 16 13 0 10 14 0 1 11 4 7 0 16 3 13 17 1 0 14 7 10 3 0 6 16 8 11 14 0 17 1 13 10 0 7 1 4 7 17 0 13 6 3 9 0 4 7 10 1 13 0 9 0 12 0 16 0 3 13 6 9 0 18 0 15 13 16 0 10 3 0 18 0 2 0 0 3 6 0 9 12 0 2 0 18 10 13 16 7 0 0 15 0 18 0
Iterations for convergence	487	434	409

Figure 7: An example running of mixed model

3 Conclusions and future work

Average bandwidth reducing is a good choice for preconditioning in parallel solving of equations systems. In this work the average bandwidth reducing process additionally depending on the diagonal blocks configuration and their condition numbers. Adding these new additional criteria leads to a substantial improvement of convergence. The major advantage of the proposed methods is that combine *partitioning* with *preconditioning*, so no need for two distinct algorithms.
The relation $p = n/mbw$ determine with a good approximation the optimal number of equal partitions and this value represent a good start for an efficient preconditioning guided by diagonal blocks configuration.
One of the concerns in the future will be the performance improvement of proposed recommendation system, in particular by: increase and diversify the knowledge base, increasing the number of features to be analyzed, determining the weight/relevance of each feature in part in computing the degree of dissimilarity and the selection a simple but efficient method for estimating the condition number to increase the preconditioning efficiency.
Another concern for the future will be to work with unequal partitions respectively overlapped partitions in the case of proposed approach.

References

[1] **P. Arbenz W. Gander**, A survey of direct parallel algorithms for banded linear systems, *Report Departement Informatik, ETH Zurich*, (1994)

[2] **P. Arbenz M. Hegland A. Cleary J. Dongarra**, *Paralel numerical liniar algebra, chapter: A comparison of paralel solvers for diagonally dominant and general narrowbanded liniar systems".*, Nova Science Publishers, Inc., Commack, NY, USA, 2001

[3] **O. Axelsson**, A survey of preconditioned iterative methods for linear systems of algebraic equations, *BIT Numerical Mathematics*, **25/1**, (1985), 165–187

[4] **M. Benzi**, Preconditioning Techniques for Large Linear Systems: A Survey, *Journal of Computational Physics*, **182**, (2002), 418–477

[5] **B. Carpentieri I.S. Duff L. Giraud G. Sylvand**, Combining Fast Multipole Techniques and an Approximate Inverse Preconditioner for Large Electromagnetism Calculations, *SIAM J. Sci. Comput.*, **27/3**, (2005), 774-792

[6] **U.V. Catalyurek C. Aykanat**, Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication, *Parallel and Distributed Systems, IEEE Transactions*, **10/7**, (1999)

[7] **U.V. Catalyurek C. Aykanat**, A Hypergraph-Partitioning Approach for Coarse-Grain Decomposition, *Supercomputing, ACM/IEEE 2001 Conference, ISBN:1-58113-293-X*, **IEEE**, (2001)

[8] **U.V. Catalyurek E.G. Boman K.D. Devine D. Bozdag**, Hypergraph-based Dynamic Load Balancing for Adaptive Scientific Computations, *Parallel MESH Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, ISBN:1-4244-0910-1*, (2007)

[9] **A. Cevahir A. Nukada S. Matsuoka**, High performance conjugate gradient solver on multi-GPU clusters using hypergraph partitioning, *Computer Science - Research and Development May 2010, Volume 25, Issue 1-2, pp 83-91*, (2010)

[10] **C.K. Cheng Y.C.A. Wei**, An improved two-way partitioning algorithm with stable performance, *IEEE Trans. Computer Aided Design*, **10/12**, (1991), 1502–1511

[11] **R. Cheng M. Gen**, Parallel machine scheduling problems using memetic algorithms, *Computers et. Industrial Engineering, Elsevier*, **33/3..4**, (1998), 761–764

[12] **C. Chevalier F. Pellegrini**, Improvement of the Efficiency of Genetic Algorithms for Scalable Parallel Graph Partitioning in a Multi-level Framework, *Euro-Par 2006 Parallel Processing, Lecture Notes in Computer Science,Springer*, **4128**, (2006), 243–252

[13] **A.K. Cline C.B. Moler G.W. Stewart J.H. Wilkinson**, An Estimate for the Condition Number of a Matrix, *SIAM J. Numer. Anal.*, **16/2**, (1979), 368375

[14] **E. Cuthill J. McKee**, Reducing the bandwidth of sparse symmetric matrices, *Proc. of ACM*, (1969), 157–172

[15] **G. Cybenko**, Dynamic load balancing for distributed memory multiprocessors, *Journal Parallel Distrib. Comput*, **7**, (1989), 279–301

[16] **K.D. Devine et. all**, New challenges in dynamic load balancing, *Applied Numerical Mathematics*, **52/(2-3)**, (2005), 133–152

[17] **K.D. Devine et. all**, Parallel hypergraph partitioning for scientific computing, *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International, ISBN: 1-4244-0054-6*, (2006)

[18] **G. Garcia R. Yahyapour A.Tchernykh**, Load Balancing for Parallel Computations with the Finite Element Method, *Computacin y Sistemas, ISSN 1405-5546*, **17/3**, (2013), 299–316

[19] **B.A. Hendrickson**, Fast spectral methods for ratio cut partitioning and clustering, *Proceedings of IEEE International Conference on Computer Aided Design*, (1991), 10–13

[20] **M.T. Heath P. Raghavan**, A Cartesian parallel nested dissection algorithm, *SIAM J. Matrix Anal. Appl.*, **16/1**, (1995), 235–253

[21] **B. Hendrickson R. Leland**, A Multilevel Algorithm for Partitioning Graphs, *Technical Report SAND93-1301, Sandia National Laboratories*, (1993)

[22] **B.A. Hendrickson**, Graph partitioning and parallel solvers: Has the emperor no clothes?, *Proceedings of the 5th Solving Irregularly Structured Problems in Parallel*, (1998), 218–225

[23] **G. Karypis R. Aggarwal V. Kumar S. Shashi**, Multilevel hypergraph partitioning: applications in VLSI domain, *Very Large Scale Integration (VLSI) Systems, IEEE Transactions*, **7/1**, (1999)

[24] **C.S. Kenney A. J. Laub M. S. Reese**, Statistical Condition Estimation for Linear Systems, *SIAM Journal on Scientific Computing*, **19/2**, (1998), 566–583

[25] **P. Korosec J. Silc B. Robic**, Mesh-Partitioning with the Multiple Ant-Colony Algorithm, *Ant Colony Optimization and Swarm Intelligence, Lecture Notes in Computer Science, ISBN:978-3-540-22672-7.*, **3172**, (2004)

[26] **A.J. Laub J. Xiai**, Statistical Condition Estimation for the Roots of Polynomials, *TSIAM Journal on Scientific Computing, Vol. 31, No. 1, pp. 624-643*, (2008)

[27] **G. Laszewski**, A Collection of Graph Partitioning Algorithms: Simulated Annealing, Simulated Tempering, Kernighan Lin, Two Optimal, Graph Reduction, Bisection, *Northeast Parallel Architectures Center at Syracuse University. Technical Report SCCS 477*, (1993)

[28] **L.O. Mafteiu-Scai**, Bandwidth reduction on sparse matrix, *West University of Timisoara Annals*, **XLVIII/3**, (2010)

[29] **L.O. Mafteiu-Scai V. Negru D. Zaharie O. Aritoni**, Average bandwidth reduction in sparse matrices using hybrid heuristics, *Studia Universitatis Babes-Bolyai University, Cluj Napoca*, **3/2011**, (2011)

[30] **L.O. Mafteiu-Scai**, Interchange opportunity in average bandwidth reduction in sparse matrix, *West Univ. of Timisoara Annals, Timisoara, Romania, ISSN:1841-3293*, (2012)

[31] **L.O. Mafteiu-Scai**, Average bandwidth relevance in parallel solving systems of linear equations, *Int. J. Eng. Res. Appl., ISSN 2248-9622*, **3/1**, (2013), 1898–1907

[32] **L.O. Mafteiu-Scai**, A new dissimilarity measure between feature-vectors, *Int. J. of Comp. Appl., ISSN: 0975 8887,ISBN: 973-93-80873-17-5*, **64/17**, (2013)

[33] **S. Maruster V. Negru L.O. Mafteiu-Scai**, Experimental study on parallel methods for solving systems of equations, *SYNACS Timisoara, 2012, IEEE Xplore CPS ISBN: 978-1-4673-5026-6, DOI: 10.1109/SYNASC.2012.7*, (Febr. 2013)

[34] **B. Nour-Omid A. Raefsky G. Lyzenga**, Solving finite element equations on concurrent computers, *American Society of Mechanical Engineering, A. K. Noor, Ed.*, (1986), 291–307

[35] **A. Pothen H.D. Simon K.P. Liou**, Partitioning sparse matrices with eigenvectors of graphs, *SIAM J. Matrix Anal. Appl.*, **11/3**, (1990), 430–452.

[36] **P. Raghavan**, Line and Plane Separators, *Technical Report UIUCDCS-R-93-1794, Department of Computer Science, University of Illinois, Urbana, IL 61801, Feb. 1993*, (1993)

[37] **Y. Saad**, *Iterative methods for sparse liniar systems (2nd ed.), Chapter 6: Krylov Subspace Methods, Part I".*, SIAM, ISBN 978-0-89871-534-7, 2003

[38] **N.G. Shivaratri P. Krueger M. Singhal**, Load distributing for locally distributed systems, *Computer*, **25/12**, (1992), 33–44

[39] **M.S. Squillante**, On the benefits and limitations of dynamic partitioning in parallel computer systems, *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, Springer*, **949**, (1995), 219–238

[40] **E.G.Talbi P. Bessiere**, A parallel genetic algorithm for the graph partitioning problem, *Proceeding ICS '91 Proceedings of the 5th international conference on Supercomputing, ACM New York, ISBN:0-89791-434-1*, **25/12**, (1991), 312–320

[41] **S. Xu J. Zhang**, A new data mining approach to predict matrix condition numbers, *Comunications in information and systems*, **4/4**, (2004), 325–340

[42] **M.H. Willebeek-LeMair A.P. Reeves**, Strategies for dynamic load balancing on highly parallel computers, *IEEE Transactions on Parallel and Distributed Systems*, **4/9**, (1993), 979–993

Liviu Octavian Mafteiu-Scai

Department of Computer Science
West University of Timisoara
V. Parvan nr. 4
Timisoara
Romania

E-mail: `lscai@info.uvt.ro`

Received: 27.11.2013

Accepted: 1.06.2014