International Journal of New Computer Architectures and their Applications (IJNCAA) 5(2): 94-101
The Society of Digital Information and Wireless Communications, 2015 ISSN 2220-9085 (Online); ISSN 2412-3587 (Print)

A Parallel Load Balancing Based on Pseudo-Cliques

Liviu Octavian Mafteiu-Scai
West University of Timisoara,
Timisoara, Romania
lscai@info.uvt.ro

ABSTRACT

In solving systems of equations on parallel computers, one very important problem is load balancing, because this affects the process efficiency. One way of achieving this goal is a good partitioning of the equations system. In this work, a connection between partitioning and pseudo-cliques from graph theory is shown. A preconditioning that uses pseudo-cliques balancing is also proposed.

KEYWORDS

system equations, parallel, partitioning, preconditioning, pseudo-cliques, iterative methods

1 INTRODUCTION

A lot of real applications involves solving large sparse linear systems of equations and, because in the current real problems these systems have dimensions of millions [2], parallel computing is one way to increase performance.

In the case of small systems, direct methods like Gauss are good choices but these are prohibitive for medium and large systems. A good choice in last case are iterative methods like Newton, Conjugate Gradient, Broyden, etc. But, these methods are more dependent on the properties of the systems. Furthermore, the iterative methods require systems preconditioning to obtain a good approximation of the solution in a reasonable time. From a global perspective ie preconditioning and effective solving, we consider that a preconditioner is good only if it 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 numerous studies that show the importance of preconditioning for solving linear equations systems on parallel computers [1, 2].

One of the most used iterative methods is the Conjugate Gradient (CG) [3], being seen as a special case of Gaussian elimination. CG method is very effective when the associated matrix of equations system is symmetric and positive definite, but there are versions of CG for nonsymetric case. The CG method involves small errors and exact solutions are generally obtained after at most n steps in the case of a well conditioned system, where n is the size of the equations system. But, if there is a rapid convergence for well conditioned systems, this can be arbitrary if the matrix is ill conditioned. Moreover, 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 seems to be the CG, as shown in [4].

An important factor in parallel solving of equations systems is the load balancing of processors. In general, there is a good load balancing of the processors if each of them has roughly the same amount of work to perform. If for each partition in part coresponds an equations subsystem, the load balancing can be seen from three perspectives: the size of the subsystems, the number of nonzero elements in each subsystem in part and the difficulty of solving these subsystems (condition number of the associated

matrix). The first can be easily controlled and does not involve a considerable computational effort. The second can be controlled in most cases but sometimes a computational effort is required: preconditioning techniques like bandwidth or average bandwidth reduction [5]. The third -and most important in our opinion- unfortunately it can not be practical considered due to large computational effort required ie computing the condition number for each subsystem in part. It is well known that the condition number has a major influence in terms of convergence speed, therefore the execution time [4,5,6]: in the case of parallel solving, the condition number of each diagonal submatrix in part, influence the local convergence for each subprocess in part and implicitly the global convergence. In paper [5] other factors that influence the performances of parallel solving equations systems with iterative methods were experimentally identified: a distribution of non-zero elements along and around the main diagonal of the associated matrix, the ratio between the number of nonzero elements inside and outside of Jacobian sub-matrix for each partition in part and of course the number of partitions.

In CG method the parallelism is derived mainly from parallel matrix-vector product and this represents the main computational effort. Because this product is directly proportional to the number of nonzero values, an implementation of a load balanced distribution can be viewed as an uniformly non-zero elements distribution between processors. That is, the nonzero elements must be uniform distributed along and around the main diagonal: this should to be as uniform as to ensure a more balanced load of processors. At the same time, the syncronization between processors can be viewed as a result of a load balancing. For example, in parallel conjugate gradient the updating of the residual vector and the vector solution do not depend on each other and can be performed at the same time, but these operations cannot be performed before performing the matrix-vector product. But the matrix-vector product in a new iteration can't be performed until the residual vector is updated. Thus, there are two moments in which processors must synchronize before they can move on to the next iteration. It is very important that the work to be balanced between processors such synchronization moments so the processors do not have any periods of inactivity between these two moments (ideally) or these periods to be the 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 a suitable division of the processes to be performed, as well as finding the most appropriate mechanisms for synchronization and communication between different processes.

Concepts from graph theory are often used in theoretical formalization of parallelization issues [7,8,9,10,11,12]. In our approach the basic concept used from graph theory is *clique.* So, there are known definitions for a *graph* G and a *subgraph* S:

$$G(X,U) = \{X = \{x_1, x_2, \ldots . x_n\}, U = \{(x_i, x_j), x_{i,j} \in X\}\}$$

$$S(X', U') = \{ X' \subset X, U' \subset U\}$$

where x_i are ***vertices***, and the pairs (x_i,x_j) are *edges* that connect two vertices. If $(x_i,x_j)=(x_j,x_i)$ we have an undirected graph.

Clique is an old term [13] and is one of the basic concepts of graph theory. A clique is a *complete subgraph* of an undirected graph, ie every two distinct vertices in the clique are adjacent:

$$C(X', U') = \{X' = \{x_1, x_2, \ldots . x_k\}, X' \subset X, U' = \{\forall i, j \; \exists (x_i, x_j), x_{i,j} \in X'\}, U' \subset U\}$$

The finding cliques problem is NP-complete [14]. Because cliques are used in a great number of theoretical and real problems, many algorithms have been developed. In particular, some issues present a high interest:

- *maximal clique:* a clique that cannot be extended by including one more adjacent vertex;
- *maximum clique:* a clique, such that there is no others cliques with more vertices;
- *clique number:* represents the number of vertices in a maximum clique;
- *intersection number:* the smallest number of cliques that together cover all the graph's edges.

Cliques are considered as a part of dense graphs and can be considered that they do not have much practical effect in the case of sparse structures. For these cases, the *pseudo-cliques* or *quasi-cliques* concept was proposed. Mainly, there are two models to explain pseudo-cliques [15]. First, the pseudo-clique is a subgraph that is obtained from a clique by removing a constant number of clique's edges. From a second point of view a pseudo-clique is a subgraph that has at least a constant ratio of edges compared to a clique of the same size. Note that this second perspective is considered in our paper. Also, the pseudo-clique problem is NP-complete.

2 THE PROPOSED APPROACH

The partitioning in terms of graph theory is an old problem in parallel computing, a relevant paper that shows partitioning models being [16]. In this paper we propose a partioning model of parallel solving equations system based on pseudo-cliques concept.

This paper will show the connection between cliques/pseudo-cliques and partitioning in parallel solving of equations systems. In our approach it is considered the case of parallel solving of an equations system using Kyrlov iterative methods that using the Jacobian, a synchronous model of parallelization and equal (or about equal) row-blocks partitioning. Also we take into consideration the undirected and unweighted graph corresponding for associated matrix of equations system.

From matriceal point of view, in an ideal case all nonzero elements should be distributed inside of diagonal blocks /submatrices. In the case of equal row-blocks partitioning, partitions with same number of nonzeros represent independent and equal subsystems of equations. In this case we have a minimum of communication processes ie a better load balancing is achieved. From the graph point of view, this situation corresponds to a graph that has cliques with same sizes and all edges of associated graph are covered by these. Of course, another ideal case in terms of communication processes is that in which the graph can be divided completely in cliques, but not necessarily of equal sizes.

Because such ideal situations are difficult or impossible to obtain in practice, we consider that a reasonable solution is when 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 to ensure a smaller number of communication processes. In addition, it is desirable that blocks $Ji(x)$ to contain close values of the number of nonzeros to ensure a good processors balancing. It is obvious that such a situation corresponds to a division of the associated graph in pseudo-cliques with about equal sizes. From associated matrix point of view, a relabeling in graph according to pseudo-cliques can conduct to an uniform distribution of nonzeros along the main diagonal. That is, that situation corresponds to close values of nonzeros in diagonal blocks of Jacobian, a situation that is reasonable and can be obtained in practice.

Further, we will introduce some new terms which are useful in pseudo-cliques evaluation in our approach.

Definition 1. An edge is internal if their adjacent vertices are in same pseudo-clique.

Definition 2. An edge is external if their corresponding vertices are in different pseudo-cliques.

Something similar to external edges are the c*ritical edges* or *bridges*, edges that interconnect two partitions into a graph.

Definition 3. It is said that a set of pseudo-cliques cover the graph if all the vertices of the graph are contained in these pseudo-cliques.

It is obvious that a larger ratio between internal and external edges for all pseudo-cliques of an covered graph involves a smaller number of communication processes -relative to equations system's size- and that is better for parallelization. More, the values for internal and external edges for each pseudo-clique in part must be close, a fact that ensures a good balancing between processors. So, we can appreciate that the internal edges represent the equations subsystem processing and external edges represents the communication processes.

Definition 4. Degree of filling (DF) is the ratio between number of internal edges of a pseudo-clique and number of edges of a clique containing the same vertices.

The computing relation is:

$$DF = \frac{k}{\frac{v(v-1)}{2}} 100 \qquad (1)$$

for a pseudo-clique with v vertices and k internal edges. This value is expressed as a percentage and a value of this equal with 100% shows a clique. It is obvious that this particular case is an ideal situation, but, in practice we are satisfied with values above 60-80%.

Further, a simplified example for a better understanding of our approach is shown. Consider G(X,U) the associated graph of a 10x10 equations system where X={1,2,3,4,5,6,7,8,9,10} and U={(1,2),(1,3),(1,4),(2,5),(5,6),(5,7),(6,7),(8,9),(9,10),(8,10)}. In Figure 1 are represented three ways of partitioning.

As it can be seen in Figure 1, it is obvious that a good partitioning implies that the total number of external edges of the associated graph to be as small. More, it implies that the pseudo-cliques obtained by partitioning the associated graph to be approximately equal in terms of the number of vertices (in fact unknowns of the equations system). In conclusion, a bad balancing in terms of the size of equations subsystems assigned to each processor (pseudo-cliques) and a large number of communication processes (external edges) may lead to large downtime for some of the processors involved in computing ie a low efficiency of the entire parallel process.

The example from Figure 1 is relevant for the proposed approach. Thus, in a) we have a large number of processors involved in the computation (4) but there are three external edges. In b) we have the ideal case in terms of the number of external edges (0) but we have a disequilibrium in terms of equations subsystems sizes. The case from c) represents a compromise between case a) and b), with an acceptable balancing and a small number of communications.

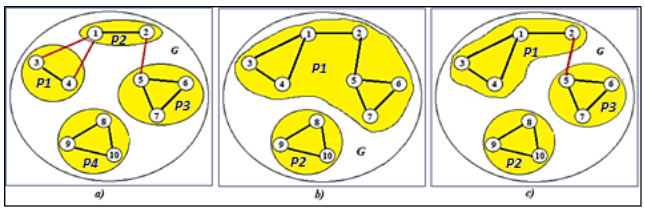

Figure 1 Examples of possible partitioning

Notes:

a) it should be noted that in our approach were not taking into account other factors that influence the computational effort per processor such as the condition number of each equations subsystem in part;

b) the partitioning used was a standard approach [16], ie each external edge causes one external communication process. In fact, the processor *p1* that has allocated partition *P1*, does not communicate twice to the processor *p2* (with the partition allocated *P2*) the value from vertex 1. So we have a single process of communication, not two. Consequently, from the communication processes point of view, we have a single external edge between *P1* and *P2*, thus the new graph obtained is a hipergraph, a much more realistic representation as shown in paper [17]. Such an approach will be the subject of our future research;

c) the partitioning must be treated differently, depending on the parallelization model: synchronous or asynchronous;

So, how can we practically use the pseudo-clique concept to optimize the parallelization

process? Before showing this, we must mention that there are two perspectives in terms of partitioning. First, for a given number of processors/partitions the equations system is ”arranged” so as to ensure an enhanced efficiency of parallelization process. The second approach consists of determining the best partitioning or one as close/satisfactory for a given equations system.

The main idea in the first approach consists of determining a number of cliques (ideal case) or pseudo-cliques (frequent case) equal to the number of partitions desired and rearranging the equations system using a relabelling of associated graph’s vertices. A load balancing of processors and a small number of communication between them will constitute the main goals of algorithms proposed for this issue.

In the second approach the proposed algorithm searches the best division of the associated graph in pseudo-cliques that ensure an efficient parallelization. This means pseudo-cliques with *sizes degree of filling* and *number of external edges* close as values. More, it will be chosen the partitioning where the pseudo-cliques have a larger value for *degree of filling* and a smaller value of *external edges.*

3 EXPERIMENTS

In the following, the experiments performed that validate the proposed approach are presented.

Finding all cliques of an undirected graph is a hard task in terms of memory and runtime because the number of cliques can grow exponentially with every node added. There are many algorithms proposed for this problem but that proposed by Bron & Kerbosch [18] seems to be one of the fastest. An adapted form of this was used in our experiments, ie an algorithm that can find a given number of cliques with appropriate sizes, without common vertices between them. More, these cliques must cover the graph in sense of Definition 3. A small example is presented in Figure 2. In Figure 2a) are represented all cliques found by an algorithm inspired from that proposed by Bron & Kerbosch. In Figure 2b) are represented new cliques found after making a balancing between those initially determined. It can be seen in Figure 2b) that the intersection of these new cliques in terms of vertices is an empty set.

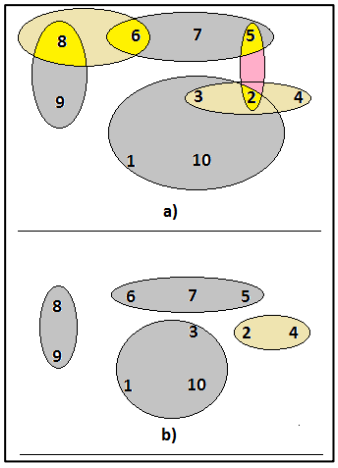

Figure 2 Balancing example

After experiments with cliques we observe that in the case of big and sparse graph the sizes of cliques after balancing are very small in relation to the size of the graph, most of them consisting only of two vertices. This situation in terms of solving large and sparse equations systems in parallel, would require a relatively large number of processors, that is not an option in most situations.

In such a situation -more and small cliques- the research was oriented mainly towards the pseudo-cliques problem. An efficient algorithm for enumerating all pseudo-cliques of a graph was proposed in paper [19]. This was a good starting point for an incipient algorithm that could find pseudo-cliques closer in size, independent in relation to the contained vertices and with a closer number of external vertices. The situation is similar to that one illustrated in Figure 2, but instead of cliques here are pseudo-cliques.

The case of a 20x20 linear system of equations exemplifies the proposed approach. The number of possible partitions/pseudo-cliques analyzed was 4, 5 and 10. The associated graph with 5 balanced pseudo-cliques has the lowest

number of external edges -equal to 32- as can be seen in Table 1.

After relabelling the associated graph, we obtain an equivalent equations system that has a better convergence in the case of the conjugate gradient method.

Thus, for the following linear equations system:

$7x_4 + 10x_7 + 13x_{10} + 16x_{13} + x_{18} + 3x_{19} = 511$
$8x_4 + 11x_7 + 14x_{10} + 17x_{13} + x_{16} + 4x_{19} = 562$
$10x_5 + 13x_8 + 16x_{11} + 3x_{17} + 6x_{20} = 501$
$7x_1 + 8x_2 + 13x_7 + 16x_{10} + 3x_{16} + 6x_{19} = 436$
$10x_3 + 14x_7 + 17x_{10} + x_{13} + 4x_{16} + 7x_{19} = 508$
$16x_8 + 3x_{14} + 6x_{17} + 9x_{20} = 452$
$10x_1 + 11x_2 + 13x_4 + 14x_5 + 3x_{13} + 6x_{16} + 9x_{19} = 460$
$13x_3 + 16x_6 + x_{10} + 4x_{13} + 7x_{16} + 10x_{19} = 499$
$3x_{11} + 6x_{14} + 9x_{17} + 12x_{20} = 510$
$13x_1 + 14x_2 + 16x_4 + 17x_5 + x_8 + 6x_{13} + 9x_{16} + 12x_{19} = 648$
$16x_3 + 3x_9 + 7x_{13} + 10x_{16} + 13x_{19} = 573$
$9x_{14} + 12x_{17} + 15x_{20} = 630$
$16x_1 + 17x_2 + x_5 + 3x_7 + 4x_8 + 6x_{10} + 7x_{11} + 12x_{16} + 15x_{19} = 722$
$3x_6 + 6x_9 + 9x_{12} + 13x_{16} + 16x_{19} = 692$
$15x_{17} + 18x_{20} = 615$
$x_2 + 3x_4 + 4x_5 + 6x_7 + 7x_8 + 9x_{10} + 10x_{11} + 12x_{13} + 13x_{14} + 18x_{19} = 1012$
$3x_3 + 6x_6 + 9x_9 + 12x_{12} + 15x_{15} = 495$
$x_1 + 2x_{20} = 41$
$3x_1 + 4x_2 + 6x_4 + 7x_5 + 9x_7 + 10x_8 + 12x_{10} + 13x_{11} + 15x_{13}\ 16x_{14} + 18x_{16} = 1183$
$6x_3 + 9x_6 + 12x_9 + 15x_{12} + 18x_{15} + 2x_{18} = 666$

the equivalent system after pseudo-cliques balancing and associated graph relabelling is:

$8x_2 + 11x_3 + 14x_5 + 17x_6 + 4x_7 + x_8 = 562$
$8x_1 + 13x_3 + 7x_4 + 16x_5 + 6x_7 + 3x_8 = 436$
$11x_1 + 13x_2 + 10x_4 + 3x_6 + 9x_7 + 6x_8 + 14x_9 = 460$
$7x_2 + 10x_3 + 13x_5 + 16x_6 + 3x_7 + x_{20} = 511$
$14x_1 + 16x_2 + 13x_4 + 6x_6 + 12x_7 + 9x_8 + 17x_9 + x_{10} = 648$
$17x_1 + 3x_3 + 16x_4 + 6x_5 + 15x_7 + 12x_8 + x_9 + 4x_{10} + 7x_{11} = 722$
$4x_1 + 6x_2 + 9x_3 + 3x_4 + 12x_5 + 15x_6 + 18x_8 + 7x_9 + 10x_{10} + 13x_{11} + 16x_{13} = 1183$
$x_1 + 3x_2 + 6x_3 + 9x_5 + 12x_6 + 18x_7 + 4x_9 + 7x_{10} + 10x_{11} + 13x_{13} = 1012$
$14x_3 + 17x_5 + x_6 + 7x_7 + 4x_8 + 10x_{12} = 508$
$x_5 + 4x_6 + 10x_7 + 7x_8 + 13x_{12} + 16x_{15} = 499$
$7x_6 + 13x_7 + 10x_8 + 16x_{12} + 3x_{14} = 573$
$10x_9 + 13x_{10} + 16x_{11} + 3x_{16} + 6x_{18} = 501$
$16x_7 + 13x_8 + 6x_{14} + 3x_{15} + 9x_{17} = 692$
$3x_{11} + 6x_{13} + 9x_{16} + 12x_{18} = 510$
$16x_{10} + 3x_{13} + 6x_{16} + 9x_{18} = 452$
$3x_{12} + 9x_{14} + 6x_{15} + 12x_{17} + 15x_{19} = 495$
$9x_{13} + 12x_{16} + 15x_{18} = 630$
$6x_{12} + 12x_{14} + 9x_{15} + 15x_{17} + 18x_{19} + 2x_{20} = 666$
$15x_{16} + 18x_{18} = 615$
$x_4 + 2x_{18} = 41$

In Figure 3 it can be seen a partitioning with 5 partitions/pseudo-cliques, before and after preconditioning the given equations system by pseudo-cliques balancing. The experimental results obtained using a parallelized implementation of conjugate gradient on IBM Blue Gene/P supercomputer are shown in Table 1.

A set of experiments was done using systems of equations with sizes between 10 and 300. The goal was to improve the convergence of parallelized conjugate gradient method after preconditioning by pseudo-cliques balancing. The average results of these experiments can be seen in Figure 4.

It is to be mentioned that there isn't a general rule to increase the speed of convergence with system size, as one might believe of Figure 4. In some cases, we obtained better results and in others worse. The graphic represents the average of results obtained for about a hundred systems for each size in part, ie 10, 50, 100, 200 and 300.

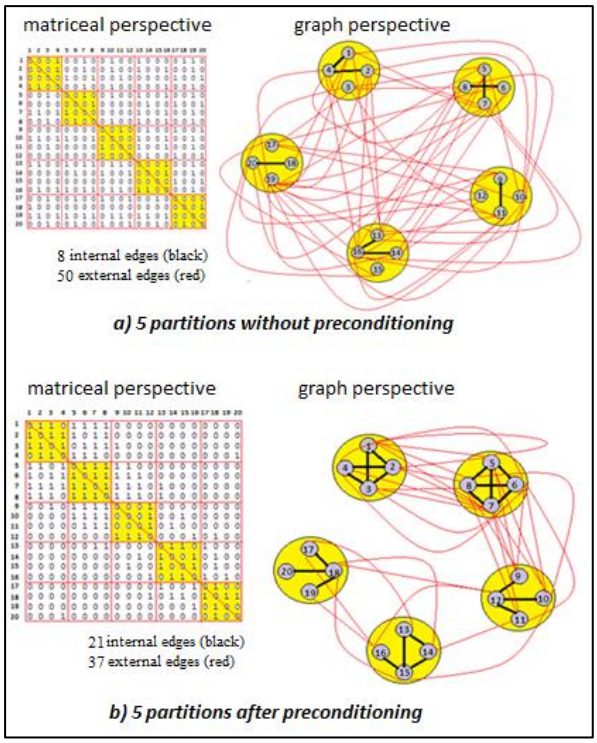

Figure 3 An example of partitioning

Table 1. Experimental results

Partitions (pseudo-cliques)	Iterations necessary for convergence		Internal/external edges	
	Initial	After balancing	Initial	After balancing
4	47	47	9/49	9/49
5	59	47	8/50	21/37
10	55	55	8/50	8/50

After experiments, it has been observed:

- only pseudo-cliques balancing is not sufficient for improving the parallel computing process, but in most cases the pseudo-cliques balancing assure a minimal number of external edges for some partitioning, that can lead to a better convergence.

- the partitioning scheme is very important, sometimes even determinant;
- other factors, such as condition numbers of equations subsystems, are also very important in convergence;
- a lower number of external edges in pseudo-cliques lead to efficiencies of up to 30% in terms of convergence (or more in some cases).

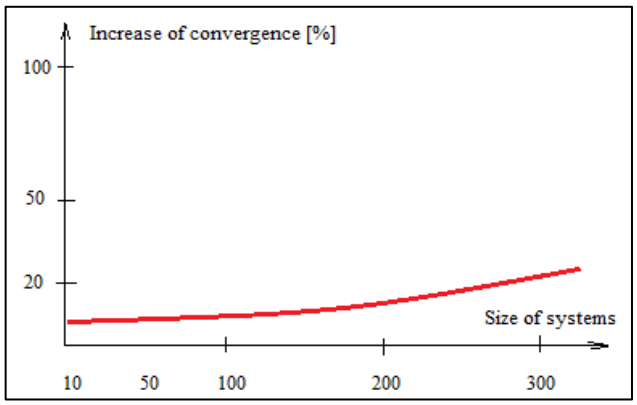

Figure 4. Improving the CG convergence

4 CONCLUSIONS

The proposed analogy *graph-equations* in this paper represents a new approach in terms of partitioning equations systems to solve these on a parallel computer and opens new research opportunities. This approach was validated through experiments.

An efficient algorithm for determining an optimal division of the associated graph in pseudo-cliques will be one of our research concerns.

At the same time, a study in terms of convergence depending by degree of filling must be taken into consideration.

REFERENCES

[1] O. Axelsson, ”A survey of preconditioned iterative methods for linear systems of algebraic equations”, Springer, BIT Numerical Mathematics 1985, 25(1), 165-187, 1985

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

[3] M.R. Hestenes, E. Stiefel, ”Methods of Conjugate Gradients for Solving Liniar Systems”, Journal of Research of the National Bureau of Standards, 49(6), 1952

[4] 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, 2013

[5] L.O. Mafteiu-Scai, ”Experiments and Recommendations for Partitioning Systems of Equations”, in West University of Timisoara Annals, 52(1), ISSN 1841-3307, DOI: 10.2478/awutm-2014-0009, 141-156, 2014

[6] Y. Saad, ”Iterative methods for sparse liniar systems”, Chapter 6: Krylov Subspace Methods, Part I”, SIAM, ISBN 978-0-89871-534-7, 2003

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

[8] U.V. Catalyurek C. Aykanat, ”A Hypergraph-Partitioning Approach for CoarseGrain Decomposition”, Supercomputing, ACM/IEEE 2001 Conference, ISBN:1-58113- 293-X, IEEE, 2001

[9] 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, IEEE, ISBN:1- 4244-0910-1, 2007

[10] A. Cevahir, A. Nukada, S. Matsuoka, ”High performance conjugate gradient solver on multi-GPU clusters using hypergraph partitioning”, Computer Science - Research and Development, 25/1-2, 83-91, 2010

[11] 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

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

[13] R.D. Luce, A.D. Perry, ”A method of matrix analysis of group structure", Psychometrika 14 (2): 95–116, doi:10.1007/BF02289146, PMID 18152948, 1949

[14] R.M. Karp, ”Reducibility among combinatorial problems”, in Miller, R. E.; Thatcher, J. W., Complexity of Computer Computations, New York: Plenum, 85–103, 1972

[15] T. Uno, ”An Efficient Algorithm for Solving Pseudo Clique Enumeration”, Problem, Algorithmica, 56 (1), 3-16, Springer, 2010

[16] B. Hendrickson, T. Kolda, ”Graph Partitioning Models for Parallel Computing”, Elsevier, Journal Parallel Computing - Special issue on graph partioning and parallel computing”, Volume 26 Issue 12, Nov. 2000, 1519 – 1534, 2000

[17] B. Hendrickson, ”Graph partitioning and parallel solvers: Has the emperor no clothes?”, în A. Ferreira et al. editors, Solving Irregularly Structured Problems în Parallel: 5th Intl. Symposium, Berkeley, California,USA, August 9-11, 1998, LNCS 1457, 218-225. Springer, 1998.

[18] C. Bron, J. Kerbosch, ”Algorithm 457: Finding all cliques of an undirected graph”, Communication of ACM 16 (9), 575–577, 1973

[19] T. Uno, ”An Efficient Algorithm for Enumerating Pseudo Cliques”, Algorithms and Computation, Lecture Notes in Computer Science, Springer, vol. 4835, 2007, 402-414, 2007