TOP/DOMDEC - a Software Tool for Mesh Partitioning and Parallel Processing

Charbel Farhat and Stephane Lanteri
Department of Aerospace Engineering Sciences
and Center for Space Structures and Controls
University of Colorado at Boulder
Boulder, CO 80309-0429, U. S. A.
e-mail:charbel@boulder.colorado.edu — Ph:(303)-492-3992

Horst D. Simon
Computer Sciences Corporation
Applied Research Branch
Mail Stop T045-1
NASA Ames Research Center
Moffett Field, CA 94035-1000
e-mail:simon@nas.nasa.gov— Ph:(415)-604-4322

TOP/DOMDEC is an interactive software package for mesh partitioning and parallel processing. It offers several state-of-the-art graph decomposition algorithms in a user friendly environment. Generated mesh partitions can be smoothed and optimized for minimum interface and maximum load balance using one of several non-deterministic optimization algorithms. TOP/DOMDEC also provides real-time means for assessing a priori the quality of a mesh partition and discriminating between different partitioning algorithms. The user interface includes high speed three-dimensional graphics, an interprocessor communication simulator for today's massively parallel systems, and an output function with parallel I/O data structures. In this paper, we describe the basic features of TOP/DOMDEC and highlight their application to the parallel solution of computational fluid and solid mechanics problems.

1. The partitioning problem and its significance

Unstructured meshes are used in several large-scale scientific and engineering problems, including finite-volume methods for computational fluid dynamics and finite element methods for structural analysis. Because of their large size and computational requirements these problems are increasingly solved on highly parallel machines and clusters of high-end workstations. If unstructured problems

such as these are to be solved on distributed-memory parallel computers, their data structures must be partitioned and distributed across processors; if they are to be solved *efficiently*, the partitioning process must maximize load balance and minimize interprocessor communication. Recent investigations have also shown that even when computing on a parallel machine that offers a virtual shared memory environment, mesh partitioning is still desirable because it explicitly enforces data locality and therefore ensures high levels of performance [1].

Given a mesh M and a number of processors N_p, one generally would like to automatically partition M into N_p submeshes $\{M^s\}_{s=1}^{s=N_p}$ such that: (a) the subproblems associated with $\{M^s\}_{s=1}^{s=N_p}$ have about the same complexity — they are load balanced —, and (b) the amount of communication between the processors assigned to $\{M^s\}_{s=1}^{s=N_p}$ is minimized. Some parallel solution algorithms may also impose other requirements on the mesh partition such as nearly perfect subdomain aspect ratios or large/small subdomain interconnectivity bandwidth. Such additional requirements are discussed in [8] but are beyond the scope of this paper.

The various forms of mesh partitioning algorithms that try to achieve load balancing and minimum communication are as different as the number of researchers working on the problem. However, recently published mesh decomposition algorithms can be grouped into three categories: engineering, optimization, and graph theory based heuristics [2-9]. Deciding which specific algorithm or category of algorithms is best is as difficult as defining what exactly constitutes an "efficient" partition. The answers to both questions are mesh, problem, and machine dependent. The main idea behind designing TOP/DOMDEC is to let each analyst decide himself which partitioning algorithm is best for his problem. Therefore, we have integrated in TOP/DOMDEC several popular and state-of-the-art mesh partitioning algorithms and the simulation tools that are necessary to evaluate them for different problems and different computing platforms.

2. TOP/DOMDEC

TOP/DOMDEC is a Totally Object-oriented Program written in C++ and GL for automatic DOMain DEComposition. It is both a software tool and a software environment for mesh partitioning and parallel processing. It is a software tool because it contains several algorithms for automatic mesh decomposition and a set of relevant decision making tools for selecting the best mesh partition for a given problem and a given multiprocessor. It is also a software environment

because it allows advanced users to "plug in" their own mesh partitioning algorithm and benefit from all the interactive features of TOP/DOMDEC that include the evaluation of load balancing, network traffic and communication costs, the generation of parallel data structures, and the use of state-of-the-art high-speed graphics.

3. Partitioning algorithms

The development of efficient heuristics for solving the NP hard problem of graph partitioning has been a very active research area in the last few years. In TOP/DOMDEC, we have implemented a number of recent and fast algorithms for graph and mesh partitioning that have been demonstrated to be useful in practical large-scale computational science and engineering problems. These algorithms are:

1. The Greedy algorithm (GR). This partitioning algorithm was first proposed in [4]. It is referred to as the Greedy algorithm because it essentially "bites" into the mesh in order to construct the subdomains. It exploits only the mesh connectivity information, which makes it is the fastest algorithm implemented in TOP/DOMDEC. In general, the GR algorithm tends to generate mesh partitions that are characterized by reasonable subdomain aspect ratios and a reduced number of interface points. Here, we point out that one statement is unfortunately missing in the Greedy Fortran code given in [4]. It is the statement that forces every subdomain to start with an element attached to the previously computed interface.

2. The RCM based algorithm (RCM). The use of a bandwidth reduction scheme such as the Reverse Cuthill-McKee [10] algorithm for mesh partitioning was first reported in [5] for parallel structural computations on the iPSC/2. The basic idea is an algebraic one and stems from the fact that if the $NEQ \times NEQ$ sparse matrix associated with the given problem is setup in banded form, and if an equal number of consecutive columns are assigned to each of the N_p available processors, then the maximum number of processors which need to communicate with any other processor is given by $N_p \times MAXBAND/NEQ$, where $MAXBAND$ denotes the maximum bandwidth of the system. Therefore if $MAXBAND$ is minimized , the interconnectivity bandwidth of the subdomains is also minimized, and the startup costs associated with the messages issued by any processor are minimized. However, the RCM algorithm tends to produce elongated subdomains with large interfaces, and hence may not be suitable for parallel processors with relatively low startup but high transmission costs.

3. The Recursive RCM algorithm (RRCM). This is a recursive version of the RCM algorithm that strikes a compromise between reducing the bandwidth of the mesh partition and its interface size. The partitions generated with the RRCM algorithm have always better subdomain aspect rations than those produced by the RCM scheme.

4. The Principal Intertia algorithms (PI). Slicing is probably the most intuitive approach for partitioning any object. In the case of discrete meshes, it can be implemented as follows [8]. First, a direction u is specified and the mesh points are projected onto it. Next, the projected points are sorted along u then gathered into N_p subdomains. A reasonable choice for u is given by any of the three principal inertia directions of the given mesh. These are the eigenvectors I_1, I_2, I_3 of the 3×3 inertia matrix:

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

where

$$I_{xx} = \sum_{i=1}^{i=N_n} (y_i - y_{cg})^2 + (z_i - z_{cg})^2 \qquad I_{yy} = \sum_{i=1}^{i=N_n} (x_i - x_{cg})^2 + (z_i - z_{cg})^2$$

$$I_{zz} = \sum_{i=1}^{i=N_n} (x_i - x_{cg})^2 + (y_i - y_{cg})^2 \qquad I_{xy} = I_{yx} = -\sum_{i=1}^{i=N_n} (x_i - x_{cg})(y_i - y_{cg})$$

$$I_{xz} = I_{zx} = -\sum_{i=1}^{i=N_n} (x_i - x_{cg})(z_i - z_{cg}) \qquad I_{yz} = I_{zy} = -\sum_{i=1}^{i=N_n} (y_i - y_{cg})(z_i - z_{cg})$$

and N_n, x_i, y_i, z_i, and x_{cg}, y_{cg}, z_{cg} denote respectively the number of nodal points in the mesh, their coordinates, and the coordinates of the mesh center of gravity. It should be noted that the PI algorithms include the Coordinate Bisection algorithms as particular cases [7].

5. The Recursive Principal Intertia algorithms (RPI). These algorithms are recursive variations of the PI partitioning schemes. They require solving at each recursive step a 3×3 eigenvalue problem in order to find the principal inertia directions of the current mesh partition.

6. The Recursive Graph Bisection algorithm (RGB). Conceptually, the RGB algorithm is similar to the RPI schemes. However, whereas the RPI algorithms use a Euclidean metric for sorting the vertices, the RGB scheme uses a graph metric for that purpose. Therefore, the RGB partitioning algorithm exploits the mesh connectivity information but the RPI schemes do not.

7. The 1D Topology Frontal algorithm (1DTF). This is the only partitioning algorithm in TOP/DOMDEC that guarantees that every generated subdomain will have at most two neighbors. It is based on the work presented in [11] and can be useful for subdomain-based multifrontal solution schemes.

8. The Recursive Spectral Bisection algorithm (RSB). This algorithm is at the same time the least intuitive partitioning scheme and the one that has most attracted the attention of the parallel computing community. It is derived from a graph bisection strategy based on the computation of the Fiedler vector — that is, the second eigenvector of the Laplacian matrix of the graph associated with the given problem [6]. Thanks to the multigrid strategy and the special version of the Lanczos algorithm described in [12] for extracting the Fiedler vector, the computational requirements of this partitioning scheme are no longer overwhelming, even on a simple workstation.

Some of the above algorithms were not designed to minimize the subdomain interfaces, and as a matter of fact, will always generate a larger number of interface points than other partitioning algorithms. However, this does not disqualify them from being of practical use. For example, the slicing algorithms (PI) are the only partitioning algorithms that can enforce a strip-wise decomposition parallel to a specific direction, which is often required in the solution of anisotropic elliptic problems via a domain decomposition based iterative solver.

Once a partitioning scheme is specified in TOP/DOMDEC, the user can impose either a "gathered" or a "scattered" decomposition mode. In the gathered mode, the generated partition is represented by a single object and any further action on that object applies to all of the subdomains. In the scattered mode, each generated subdomain is represented by a separate object, has its own data structures, and is a potential target to an independent action. For example, the scattered mode allows the user to recursively construct a hybrid mesh partition where a different decomposition algorithm is applied at a different level of the recursion.

4. Optimization and smoothing

The algorithms mentioned above are recommended for "initial" decompositions. After an initial mesh partition is generated, TOP/DOMDEC offers three optimization algorithms for improving it. The list of subdomain related items that can be optimized includes:

1. interface size.

2. subdomain frontwidth (for frontal or skyline direct solvers).
3. the product of the above two items.
4. node-wise load balance.
5. element-wise load balance.
6. edge-wise load balance.
7. subdomain aspect ratio.
8. a weighted function of any of the above items.

Item 8 is particularly useful for heterogeneous problems that combine node-wise, and/or element-wise, and/or edge-wise computations.

Three optimization algorithms based on the work described in [11] have been implemented in TOP/DOMDEC: (1) Tabu Search, (2) Simulated Annealing, and (3) Stochastic Evolution. The optimization process is quite fast because it is limited to the component of the mesh that neighbors the initial interface. Having three optimization algorithms available allows the user to switch between them when one of them gets entrapped in a local extremum. Optimized mesh partitions have usually smooth interfaces and therefore are suitable for variational domain decomposition methods that are popular in both computational fluid dynamics [13] and solid mechanics [14] problems.

5. Real-time evaluators

As we have said earlier, defining what exactly constitutes an efficient partition is both problem and machine dependent. Therefore, only the analyst interested in running a specific application using a well-defined computational algorithm on a particular computing platform should decide which partitioning algorithm is best suited for his/her problem. For this reason, we have packaged in TOP/DOMDEC all of the mesh partitioning algorithms described above, together with simulation tools that allow their evaluation for a specific problem and a particular parallel architecture. The list of partitioning attributes that can be evaluated includes:

1. Interface size. The total number of nodes or edges on the subdomain interfaces is today the most popular criterion for assessing a mesh partition. Because of interprocessor communication cost considerations, it is often believed that the mesh partition with the smallest interface size is the best one, even though this is not necessarily always true [8]. For each mesh partition, TOP/DOMDEC outputs the interface size of every subdomain as well as the maximum subdomain interface size and the total interface size (without duplication).

2. Memory requirements. On a distributed memory parallel processor, memory limitations can be imposed either by the maximum memory requirements of a given subdomain, or by the total memory requirements of all subdomains. Once a mesh partition has been created, TOP/DOMDEC allows the user to specify the number of unknowns at a grid point and the mode of computations, and to inquire about both the local and global memory requirements of the partitioned problem. For simplicity, TOP/DOMDEC views the world of computations as either locally explicit or locally implicit. By locally implicit, we refer to a computational algorithm that does require the factorization of some subdomain matrix, and by locally explicit, we refer to an algorithm that does not. In locally implicit mode of computations, a profile/skyline [15] storage is assumed and an RCM renumbering option is available; both Dirichlet and Neumann interface boundary conditions can be specified in that case. The reader should note that, even when the subdomains are element-wise or node-wise balanced, their skylines are not necessarily balanced, and therefore the memory requirements for locally implicit computations can vary from one subdomain to another.

3. Load balance. TOP/DOMDEC outputs load balance factors for both locally explicit and locally implicit modes of subdomain computations. For locally explicit computations, load balance factors can be requested for element-wise, node-wise, edge-wise, subdomain-by-subdomain and/or interface-by-interface computations. For locally implicit computations, the load balance factors can be based either on the subdomain matrix envelope sizes, or on the subdomain matrix maximum bandwidths.

4. Communication pattern and network traffic. For a given problem, a given computational algorithm, and a given parallel processor, the communication pattern of a generated mesh partition can be an important factor for accepting it or rejecting it. For example, if the target parallel processor is sensitive to wire conflicts or a good processor mapping is essential for reducing interprocessor communication costs, one should select a mesh partition where every subdomain has a reduced number of neighbors. TOP/DOMDEC allows the user to analyze graphically the communication pattern of a mesh partition and assess its impact on network traffic.

5. Communication costs. TOP/DOMDEC has built-in communication cost models for the iPSC-860, the Paragon XP/S, the CM-5, the KSR-1, as well as a generic message-passing based parallel processor. The analyst can use these models to compare the relative communication costs of different mesh partitions and their load balancing.

In summary, the analyst can generate several mesh partitions using different

decomposition algorithms, then use the evaluators described above to select the most appropriate partition for his/her problem.

5. User interface, import functions and structured output

TOP/DOMDEC's user-friendly interface is built around browsers, pop-up menus, and click-on buttons. It features high performance three-dimensional graphics, a visual dynamic data base, and a powerful interactive postprocessor for visualizing scientific and engineering global or domain decomposed numerical results. FIG. 1-7 highlight some of the components of this interface.

FIG.1. A versatile file reader

FIG.2. Decomposition and optimization algorithms

FIG.3. Interface inspection

FIG.4. Memory requirements

FIG.5. Load balance factors

FIG.6. Communication pattern

FIG.7. Communication costs

TOP/DOMDEC also features import and output functions. The import function is useful for reading industrial meshes written in the format of some major finite difference and finite element commercial software. Also, the user can import to TOP/DOMDEC mesh partitions that were generated by a different program and take advantage of the graphics features and evaluation/simulation tools available in TOP/DOMDEC. The output command dumps a specified mesh partition into an ASCII or binary file, including the parallel data structures needed for local computations and for message passing. This command can modify the subdomain definitions to allow a specified amount of overlapping. The output data is blocked in a subdomain-by-subdomain fashion and preprocessed for parallel read instructions.

7. Applications

Here we illustrate the use of TOP/DOMDEC for partitioning realistic unstructured meshes, and highlight the impact of mesh partitions on the parallel performance of a computational fluid dynamics solver.

First, we consider three meshes corresponding to the finite element discretization of a High Speed Civil Transport (HSCT) aircraft (FIG. 8), a Stiffened Wing Panel (SWP) (FIG. 9), and a Turbine Blade (TB) (FIG. 10).

FIG.8. The HSCT mesh

FIG.9. The SWP mesh

FIG.10. The TB mesh

For simplicity, we assume that for all three problems, the user is interested in the mesh partition with the smallest total interface size, and/or the smallest subdomain interface size.

The performance results of various mesh partitioning algorithms are summarized in Tables 1-3 where NS, TIN, SIN and PCPU denote respectively the number of subdomains, the total number of interface nodes, the maximum number of interface nodes per subdomain, and the CPU time in seconds for generating the mesh partition on an IRIS/Crimson Silicon Graphics workstation.

Table 1
HSCT mesh - 12644 nodes - 31544 elements

NS	16	32	64	128
ALG.	TIN/SIN ; PCPU	TIN/SIN ; PCPU	TIN/SIN ; PCPU	TIN/SIN ; PCPU
GR	1329/291 ; 0.34 s	2088/179 ; 0.42 s	2818/136 ; 0.53 s	3871/105 ; 0.72
RGB	1420/342 ; 2.88 s	2121/241; 4.03 s	3021/164; 5.74 s	4117/118 ; 8.58 s
RPI	2168/478 ; 2.28 s	3608/364; 2.91 s	5346/307; 3.74 s	7726/248 ; 4.96 s
RRCM	1359/273 ; 2.04 s	2020/235 ; 2.65 s	2938/176 ; 3.42 s	3982/165; 4.80 s
RSB	1335/259 ; 15.86 s	1928/171; 20.66 s	2836/148; 30.34 s	4025/104; 38.69 s

Table 2
SWP mesh - 9486 nodes - 9136 elements

NS	16	32	64	128
ALG.	TIN/SIN ; PCPU	TIN/SIN ; PCPU	TIN/SIN ; PCPU	TIN/SIN ; PCPU
GR	1075/173 ; 0.11 s	1432/147 ; 0.16 s	1947/100 ; 0.26 s	2579/63 ; 0.42 s
RGB	1079/194 ; 0.51 s	1602/158 ; 0.77 s	2262/118 ; 1.09 s	3126/83 ; 1.84 s
RPI	1224/226; 1.45 s	1979/166; 1.83 s	2819/164; 2.35 s	4232/118; 3.23 s
RRCM	1121/231 ; 1.14 s	1568/160 ; 1.57 s	2135/140 ; 2.20 s	2927/85 ; 3.24 s
RSB	754/117 ; 6.32 s	1076/113 ; 8.49 s	1640/90 ; 11.98 s	2448/64 ; 14.05 s

Table 3
TB mesh - 11096 nodes - 7552 elements

NS	16	32	64	128
ALG.	TIN/SIN ; PCPU	TIN/SIN ; PCPU	TIN/SIN ; PCPU	TIN/SIN ; PCPU
GR	2382/396 ; 0.14 s	3012/296 0.17 s	3943/191 0.28 s	5361/149 ; 0.58 s
RGB	2642/573; 0.67 s	3666/400; 0.88 s	4820/267; 1.32 s	5969/184; 1.93 s
RPI	2504/459; 1.87 s	3680/393; 2.34 s	4976/270; 3.05 s	6465/190; 4.12 s
RRCM	2562/563 ; 2.68 s	3483/441; 3.62 s	4695/274 ; 4.64 s	5905/163; 6.49 s
RSB	2196/460 ; 7.60 s	3085/294; 9.78 s	4037/199; 12.14 s	5256/143; 14.05 s

The above results show that for all three problems, the GR and RSB algorithms outperform all of the other partitioning algorithms in reducing the interface size. For the HSCT mesh, the GR compares favorably with the RSB. However for the SWP mesh, the RSB outperforms the GR. For the TB mesh, the results oscillate with the number of requested subdomains. Note that in general, the GR algorithm is the cheapest one and yet delivers good results.

Next, we focus on the winning mesh partitions for the HSCT and SWP meshes and optimize them with the Simulated Annealing (SA) algorithm.

Table 4
HSCT mesh - 12644 nodes - 31544 elements

NS	16	32	64	128
ALG.	TIN/SIN ; PCPU	TIN/SIN ; PCPU	TIN/SIN ; PCPU	TIN/SIN ; PCPU
GR	1329/291 ; 0.34 s	2088/179 ; 0.42 s	2818/136 ; 0.53 s	3871/105 ; 0.72
GR+SA	1272/282; 12.76 s	1951/159; 21.20 s	2674/131; 33.58 s	3549/87; 55.52 s

Table 5
SWP mesh - 9486 nodes - 9136 elements

NS	16	32	64	128
ALG.	TIN/SIN ; PCPU	TIN/SIN ; PCPU	TIN/SIN ; PCPU	TIN/SIN ; PCPU
RSB	754/117 ; 6.32 s	1076/113 ; 8.49 s	1640/90 ; 11.98 s	2448/64 ; 14.05 s
RSB+SA	690/115; 10.66 s	983/103; 16.44 s	1474/80; 24.50 s	2116/63; 31.22 s

The results reported in Tables 4-5 show that optimization did not help a lot the mesh partitions of the HSCT problem, but improved by over 15% the mesh partitions of the SWP mesh, at a reasonable additional cost. The interface smoothing effect of the optimization phase is highlighted in FIG. 11-12.

FIG.11. SWP mesh - RSB: 16 subdomains before optimization

FIG.12. SWP mesh - RSB: 16 subdomains after optimization

Finally, we consider the partitioning in 32 subdomains of a large-scale two-dimensional NACA0012 mesh, for transonic viscous flow computations on the iPSC-860/32 and the KSR-1/32 systems. The mesh has 262717 vertices, 523914 elements, and 786631 edges. The parallel CFD algorithm blends finite volume and finite element spatial discretizations and employs an explicit Runge-Kutta scheme for time integration [1]. It involves both edge-wise and element-wise computations, for the evaluation of the convective and diffusive fluxes, respectively. Therefore ideally, all mesh partitions should be balanced both edge-wise and element-wise.

FIG.13. NACA0012- partial view - GR: 32 subdomains

FIG.14. NACA0012- partial view - RGB: 32 subdomains

FIG.15. NACA0012- partial view - RSB: 32 subdomains

Table 6 summarizes the characteristics of the mesh partitions obtained with the GR, RGB, and RSB algorithms without optimization (FIG. 13-15), and Tables 7-8 report the performance results on the iPSC-860/32 and KSR-1/32 parallel processors. In Table 6, EL_LBF, ED_LBF, and P_COM_TIME denote respectively the element-wise load balance factor, the edge-wise load balance factor, and the communication time predicted by TOP/DOMDEC.

Table 6
NACA0012 mesh - 262717 vertices - 523914 elements - 786631 edges
Partitioning in 32 subdomains

	GR	RGB	RSB
TIN	6526	7309	6592
SIN	813	734	626
EL_LBF	98.1%	97.0 %	99.8%
ED_LBF	98.2%	97.1 %	99.9%
PCPU	8.19 s	62.91 s	325.66 s
P_COM_TIME (iPSC-860)	46.8 s	61.4 s	54.1 s
P_COM_TIME (KSR-1)	19.5 s	19.9 s	20.7 s

Table 7
NACA0012 mesh - 262717 vertices - 523914 elements - 786631 edges
iPSC-860 - 32 processors

	GR	RGB	RSB
Communication Time	47.4 s	62.7 s	53.8 s
Solution Time	550.0 s	550.3 s	538.0 s

Table 8
NACA0012 mesh - 262717 vertices - 523914 elements - 786631 edges
KSR-1 - 32 processors

	GR	RGB	RSB
Communication Time	18.9 s	18.5 s	14.2 s
Solution Time	343.7 s	340.1 s	322.5 s

Clearly, the communication costs predicted by TOP/DOMDEC for all three mesh partitions and for the iPSC-860 are in excellent agreement with the measured communication costs. As expected (from the interface sizes) the RSB partition communicates longer on the iPSC-860 than the GR partition; however, it

produces faster results. The reason is that the edge-wise load balance factor is slightly better for the RSB partition than for the GR one, which highlights the dominating effect of load balance over communication. On the KSR-1 system, the measured communication costs differ from the predicted ones. We believe that this is due to cache effects that are not yet modeled in TOP/DOMDEC.

8. Software availability

TOP/DOMDEC runs on all Silicon Graphics (IRIS) workstations and on the IBM RS 6000 machine equipped with the GL graphics library. The size of the executable program is less than 1.5 Mbytes.

Acknowledgments

The authors would like to thank S. Barnard, M. Lesoinne, S. Maurich, R. Partch, B. Stoner, and D. Vanderstraeten, for their contributions to TOP/DOMDEC. The first author acknowledges partial support by the National Science Foundation under Grant ASC-9217394 and partial support by RNR NAS at NASA Ames Research Center under Grant NAG 2-827. The third author is an employee of Computer Sciences Corporation. His work was supported through NASA Contract NAS 2-12961.

References

[1] S. Lanteri and C. Farhat, Viscous flow computations on MPP systems: implementational issues and performance results for unstructured grids, in: R. F. Sincovec *et. al.*, ed., Parallel Processing for Scientific Computing, SIAM, (1993) 65-70.

[2] J. Flower, S. Otto and M. Salama, Optimal mapping of irregular finite element domains to parallel processors, in A. K. Noor, ed., Parallel Computations and Their Impact on Mechanics, The American Society of Mechanical Engineers, AMD-Vol. 86, (1987) 239-252.

[3] B. Nour-Omid, A. Raefsky and G. Lyzenga, Solving finite element equations on concurrent computers, in A. K. Noor, ed., Parallel Computations and Their Impact on Mechanics, The American Society of Mechanical Engineers, AMD-Vol. 86, (1987) 209-228.

[4] C. Farhat, A simple and efficient automatic FEM domain decomposer, Comp. & Struct., Vol. 28, No. 5, (1988) 579-602.

[5] J. G. Malone, Automated mesh decomposition and concurrent finite element analysis for hypercube multiprocessors computers, Comp. Meth. Appl. Mech. Eng., Vol. 70, No. 1, (1988) 27-58.

[6] A. Pothen, H. Simon and K. P. Liou, Partitioning sparse matrices with eigenvectors of graphs," SIAM J. Mat. Anal. Appl., Vol. 11, No. 3, (1990) 430-452. (1990)

[7] H. D. Simon, Partitioning of unstructured problems for parallel processing, Comput. Sys. Engrg., Vol. 2, No. 3, (1991) 135-148.

[8] C. Farhat and M. Lesoinne, Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics, Internat. J. Numer. Meth. Engrg., Vol. 36, No. 5, (1993) 745-764.

[9] S. Hsieh and J. F. Abel, Use of networked workstations for parallel nonlinear structural dynamic simulations of rotating bladed-disk assemblies, Comp. Syst. Engrg. (to appear).

[10] W. Chan and A. George, A linear time implementation of the Reverse Cuthill Mc Kee Algorithm, BIT, Vol. 20, (1980) 8-14.

[11] D. Vanderstraeten, O. Zone, R. Keunings, L. Wolsey, Non-deterministic heuristics for automatic domain decomposition in direct parallel finite element calculations, in: R. F. Sincovec *et. al.*, ed., Parallel Processing for Scientific Computing, SIAM, 929-932 (1993).

[12] S. T. Barnard and H. D. Simon, A fast implementation of recursive spectral bisection for partitioning unstructured problems, in: R. F. Sincovec *et. al.*, ed., Parallel Processing for Scientific Computing, SIAM, (1993) 711-718.

[13] Q. V. Dihn, R. Glowinski and J. Periaux, Solving elliptic problems by domain decomposition methods with applications, in: A. Schoenstadt, ed., Elliptic Problem Solvers II, Academic Press, (1984).

[14] C. Farhat, A saddle-point principle domain decomposition method for the solution of solid mechanics problems, in: D. E. Keyes, T. F. Chan, G. A. Meurant, J. S. Scroggs and R. G. Voigt, ed., Proc. Fifth SIAM Conference on Domain Decomposition Methods for Partial Differential Equations, SIAM (1991) 271-292.

[15] A. George and J. W. Liu, Computer solution of large sparse positive definite systems, Prentice-Hall, Inc., Englewood Cliffs, N. J. (1981).