Structural Engineering and Mechanics, Vol. 14, No. 1 (2002) 57-70
DOI: http://dx.doi.org/10.12989/sem.2002.14.1.057

Iterative mesh partitioning strategy for improving the efficiency of parallel substructure finite element computations

Shang-Hsien Hsieh†

Computer-Aided Engineering, Department of Civil Engineering, National Taiwan University, Taipei, Taiwan

Yuan-Sen Yang‡

National Center for Research on Earthquake Engineering, Taipei, Taiwan

Po-Liang Tsai‡†

Military Police Headquarter, Taipei, Taiwan

(Received March 28, 2001, Accepted April 24, 2002)

Abstract: This work presents an iterative mesh partitioning approach to improve the efficiency of parallel substructure finite element computations. The proposed approach employs an iterative strategy with a set of empirical rules derived from the results of numerical experiments on a number of different finite element meshes. The proposed approach also utilizes state-of-the-art partitioning techniques in its iterative partitioning kernel, a cost function to estimate the computational cost of each submesh, and a mechanism that adjusts element weights to redistribute elements among submeshes during iterative partitioning to partition a mesh into submeshes (or substructures) with balanced computational workloads. In addition, actual parallel finite element structural analyses on several test examples are presented to demonstrate the effectiveness of the approach proposed herein. The results show that the proposed approach can effectively improve the efficiency of parallel substructure finite element computations.

Key words: mesh partitioning; graph partitioning; parallel finite element computations; parallel substructure method.

1. Introduction

The parallel substructure method (Nour-Omid *et al.* 1987) is extensively used in parallel finite element computations. This method first partitions the structure into several substructures and assigns each substructure to a separate processor, as shown in Fig. 1, from (a) to (b). Each processor then forms its substructure equilibrium equations:

† Associate Professor
‡ Associate Research Fellow
‡† Captain

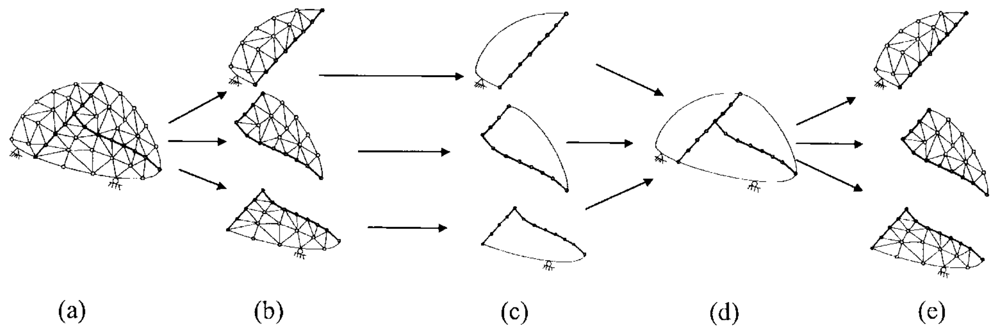

Fig. 1 Sketch of procedures of parallel substructure method

$$[K]_k\{u\}_k = \begin{bmatrix} K_{II} & K_{IE} \\ K_{EI} & K_{EE} \end{bmatrix}_k \begin{Bmatrix} u_I \\ u_E \end{Bmatrix}_k = \begin{Bmatrix} f_I \\ f_E \end{Bmatrix}_k \tag{1}$$

in which the subscript index k denotes the k^{th} substructure, while the subscript indices I and E denote the internal and interface degrees of freedom, respectively. Then, static condensations of substructure interior degrees-of-freedom are performed independently and concurrently within each substructure without inter-process communication:

$$[\bar{K}]_k = [K_{EE}]_k - [K_{EI}]_k[K_{II}]_k^{-1}[K_{IE}]_k \tag{2}$$

$$\{\bar{f}_E\}_k = \{f_E\}_k - [K_{EI}]_k[K_{II}]_k^{-1}\{f_I\}_k \tag{3}$$

The condensed system of each substructure is then gathered and assembled, and then usually solved by a parallel solver:

$$[\bar{K}_{EE}]_g = \sum_{k=1}^{N_P}([P]_k[\bar{K}_{EE}]_k[P]_k^T) \tag{4}$$

$$\{\bar{f}_e\}_g = \sum_{k=1}^{N_p}([P]_k\{\bar{f}_E\}_k) \tag{5}$$

$$\{u_E\}_g = [\bar{K}_{EE}]_g^{-1}\{\bar{f}_E\}_g \tag{6}$$

in which the subscript g denotes the global system. Finally, the internal degrees of freedom of each substructure are then solved concurrently within each processor:

$$\{u_E\}_k = [P]_k^T\{u_E\}_g \tag{7}$$

$$\{u_I\}_k = [K_{II}]_k^{-1}(\{f_I\}_k - [K_{IE}]_k\{u_E\}_k) \tag{8}$$

Fig. 1 shows the procedures of the parallel substructure method. Notably either the modified condensation method (Han and Abel 1984) or the multifrontal method (Geng *et al.* 1997) is usually used to perform the static condensation procedures rather than directly computing the matrix inverse and multiplications. Parallel finite element computations using the substructure method reduce the cost of inter-process communication and are, therefore, usually more efficient than the parallel solver in solving the entire set of uncondensed equations.

The finite element mesh of the structure is partitioned so that, among processors (or substructures), computational workloads are well balanced and inter-process communication is minimized, to make the parallel substructure method highly efficient. Although many mesh partitioning algorithms have been proposed (Hsieh *et al.* 1997), most of them use only simple and general assumptions in their load balancing optimization process without taking into account the characteristics of parallel solution algorithms. Therefore, the mesh partitions produced by these algorithms usually do not result in optimal parallel efficiency for a specific solution algorithm (Vanderstraeten *et al.* 1996, Yang and Hsieh 1997).

This work presents an iterative mesh partitioning approach to obtain better mesh partitions for parallel substructure finite element computations. The approach employs an iterative strategy with a set of empirical rules derived from results of numerical experiments on a number of different finite element meshes. The approach utilizes an existing partitioning algorithm as the kernel mesh partitioning algorithm and employs a set of empirical rules in an iterative process to improve the mesh partitioning results. Actual parallel finite element structural analyses using several test examples are also presented to demonstrate the effectiveness of the approach.

2. Optimization criteria in mesh partitioning

Most mesh partitioning algorithms (e.g., Farhat 1988, Simon 1991) use the following two criteria for optimization in their heuristic approaches. (1) Balancing the size of the substructures (for example, balancing the number of elements or nodes) and (2) Minimizing the overall size of the interface among substructures (for example, the number of interface nodes). As well known, the more the workload is balanced among processors, the less waiting time is for those processors with lighter workloads. Accordingly, the first criterion is mainly to balance computational loads (in this case, static condensation) among substructures, assuming the workload is proportional to the number of elements or nodes. The second criterion is to reduce both the overall size of the message exchanged in the inter-process communications and the size of the linear system associated with the interface degrees-of-freedom. Minimizing the size of the linear system can help to improve the overall parallel efficiency because it is usually difficult for a parallel equation solver to achieve high efficiency.

In the parallel substructure method, however, a balanced number of elements or nodes among substructures does not imply balancing of workloads among processors. Mesh partitioning based on the above criteria often does not produce partitions that give a good balance of computational loads among processors. Accordingly, the overall efficiency of parallel analysis is not optimized. A better optimization criterion is needed to yield a better balance of workloads among substructures. However, the workload related to the matrix condensation of each substructure depends on several factors, such as nodal numbering, nodal adjacencies, and matrix sparsity, and can not easily be predicted using a simple formula.

However, the authors found that a substructure with more interface nodes should have fewer elements to achieve a good balance of workloads among substructures. Therefore, this work presents an iterative approach with a set of empirical rules to take into account the number of internal and interface nodes within each substructure to balance substructure workloads. Section 4 discusses the approach in detail.

3. Computing environments

The proposed mesh partitioning approach is implemented in an object-oriented mesh partitioning environment, called MPE++ (Hsieh *et al*. 1998), in which several conventional algorithms have already been implemented for mesh partitioning. MPE++ can translate a finite element mesh into a communication graph (Hsieh *et al*. 1995), so that a graph partitioning algorithm or library can be implemented (or embedded) in MPE++. Each vertex in a communication graph denotes a single element, while each edge between two vertices denotes one or more nodes shared by the corresponding elements. This work embeds, in MPE++ a graph partitioning package, called JOSTLE (version 1.1.8) (Walshaw 1999). JOSTLE can not only produce graph partitioning results with balancing overall vertex weights in each submesh and a feasible minimum of edges crossing through different submeshes (so-called ‘edge cuts’), but it can also tune partitioning results as the vertex/edge weights are changed. These functions are useful for partitioning an adaptive mesh and dynamic load balancing.

An object-oriented parallel finite element program FE2000 is used to perform the parallel finite element analyses. The general sparse matrix technique is used using a package, SPARSPAK (George and Liu 1981). Much research has shown that the general sparse matrix approach usually outperforms the widely used skyline (or envelope) matrix approach in terms of numerical operation counts and storage requirements in solving a large-scale sparse linear system. In FE2000, SPARSPAK is slightly enhanced so that it can perform substructure matrix condensation using a modified decomposition method presented by Han and Abel (1984). The Multi-Stage Minimum Degree method (MSMD) (Ashcraft *et al.* 1999) is used to perform the substructure matrix ordering. This work performs all substructure computations in parallel. However, the present FE2000 performs sequential matrix factorization on a single processor to solve the condensed set of equations associated with the interface degrees-of-freedom because it is faster than the parallel one in the present implementation for all of the examples studied herein. A message-passing interface MPI (Message Passing Interface Forum 1994) is used by FE2000 to handle all message-passing tasks among processors.

A Pentium II-350 PC running LINUX RedHat operating system is used for mesh partitioning. A PC cluster consisting of four Pentium II-350 based PCs (each with 128 megabytes of memory and running the LINUX RedHat operating system) is used for parallel finite element analysis. The network system used in the PC cluster has a speed of 100 Mbps.

4. Iterative mesh partitioning approach

Iterative strategy has been widely used in optimization of graph partitioning. Most widely used methods, such as KL (Kernighan and Lin 1970) and SA (Kirkpatrick *et al*. 1983), involve iteratively exchanging some selected pairs of vertices among subgraphs to decrease the value of the specified cost function. These methods can also be applied to improve mesh partitioning results. For example,

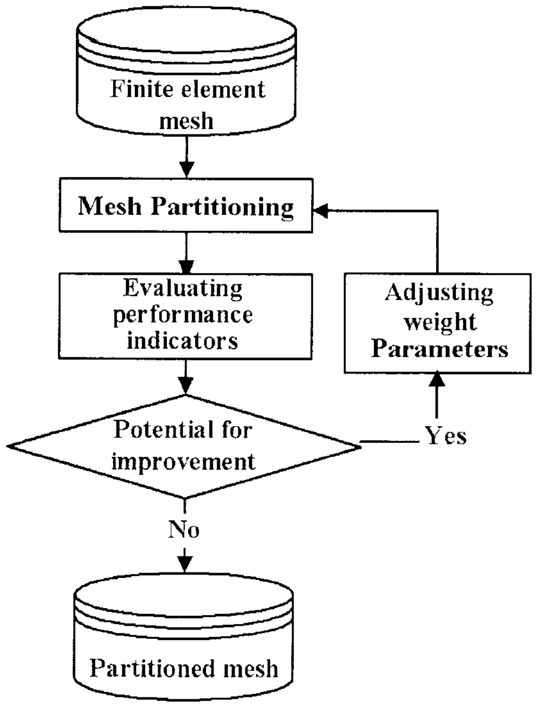
Finite element mesh
Mesh Partitioning
Evaluating performance indicators
Adjusting weight Parameters
Potential for improvement
Yes
No
Partitioned mesh

Fig. 2 Process of the proposed iterative mesh partitioning approach

Vanderstraeten and Keunings (1995) employed some optimization methods for mesh partitioning to reduce the total number of interface nodes among substructures.

An iterative approach with a set of empirical rules is presented to take into account the number of internal and interface nodes within each substructure, to balance the substructure workloads. After partitioning the finite element mesh using an existing mesh partitioning algorithm, this approach adjusts the weights of the elements in each substructure to promote redistribution of elements from substructures with more internal and interface nodes to substructures with fewer counterparts in the subsequent partitioning steps of the iterative process. The element weights of each substructure are adjusted mainly based on a cost function of the substructure workload calculated from the number of internal and interface nodes of each substructure. This rule for weight adjustment and other rules for terminating the iterative process are obtained empirically from results of several numerical experiments conducted by the authors. Further discussion of estimating the cost of substructures in the iterative partitioning process and empirical rules are given below.

Fig. 2 shows the proposed iterative mesh partitioning process. A powerful and effective mesh partitioning algorithm is needed as the kernel algorithm for mesh partitioning in the *Partitioning Phase*. This kernel algorithm may be one of the algorithms present in the literature and is assumed to use the two common optimization criteria presented in Section 2. In any case, the algorithm should satisfy the following requirements.

(1) The algorithm should permit assignment of different weightings to different elements and nodes of the mesh and be capable of accounting for these weightings in mesh partitioning.

(2) The algorithm should be able to start from an assigned initial partitioning result before the

mesh is further repartitioned.

(3) The algorithm should efficiently produce good partitions such that the time taken for the entire iterative mesh partitioning process is bounded within a reasonable fraction of the actual parallel finite element computations.

Many state-of-the-art partitioning algorithms and packages have been proposed, e.g., Chaco (1994), RST (Hsieh *et al.* 1995), JOSTLE (Walshaw 1999), and ParMETIS (Karypis *et al.* 2002). This work selects the JOSTLE-MS (multilevel static) and JOSTLE-MD (multilevel dynamic) methods respectively, in the JOSTLE graph partitioning package for the initial mesh partitioning and repartitioning. JOSTLE is selected not only because it is competitive on its quality of partitioning, but also because it allows for tuning a partitioned result through the adjustment on vertex and edge weights. However, it should be noted that other partitioning algorithms with similar functionality (e.g., ParMETIS) are also good candidates for the proposed iterative approach. In this work, the effectiveness of the proposed iterative strategy is evaluated by comparing the final repartitioning result with the initial partitioning result. In addition, Hsieh *et al.* (1995) have discussed mapping from a mesh partitioning problem to a graph partitioning problem. This work employs the communication graph approach and initializes the weights of all vertices and edges in the graph to a unique value of 100.

An indicator E^i is used in the *Evaluating Phase* to indicate a time cost of the parallel finite element analysis. The indicator E^i is defined as follows:

$$E^i = \max(E^i_{1,k}, \text{ for each substructure } k) + E^i_2 \tag{9}$$

in which

$$E^i_{1,k} = c_1 \times (I^i_{1,k})^2 + c_2 \times (I^i_{2,k})^2 \tag{10}$$

$$I^i_{1,k} = \frac{N^i_{IN,k}}{\mathbf{AVG}(N^i_{IN,j})} \tag{11}$$

$$I^i_{2,k} = \frac{N^i_{EN,k}}{\mathbf{AVG}(N^i_{EN,j})} \tag{12}$$

$$E^i_2 = \left(\frac{N^i_{EN}}{N^0_{EN}}\right)^3 \tag{13}$$

The notation **max**(.) denotes the maximum value in parenthesis, while **AVG**(.) denotes the average value of all substructures in parenthesis. Superscripts i denote the ith iteration in mesh partitioning, while j denotes the jth substructure. The superscript 0 means the initial partitioning (the first iteration). Factors E^i_1 and E^i_2 respectively correspond to the computation loads of the static condensation of each substructure and the interface solution time among substructures. $N^i_{IN,k}$ and $N^i_{EN,k}$ denote the internal nodes and interface nodes of substructure k in mesh partitioning iteration i. N_N and N^i_{EN} denote the total number of nodes and total interface nodes of iteration i, respectively. N_P denotes the total number of substructures. I_1 and I_2 indicate the number of internal and interface nodes of each substructure, respectively. These values are normalized by the average in each substructure. Eq. (10) shows that the matrix condensation time of each substructure is presumed to depend on the number of internal and interface nodes in the substructure. Constants c_1 and c_2 in Eq. (10) are the weight coefficients of terms I_1 and I_2, respectively. The value I_1 is raised to the power of 2 because the operation count of matrix condensation is generally proportional to the number of degrees of freedom to the power of 1 to 3 (usually depending on the sparseness of the global system matrix). For example, the factorization of a matrix associated with a 2-D $\sqrt{n}$ by $\sqrt{n}$ regular grid (similar to a five-point-finite-difference grid or a 2-D regular frame structure mesh) has been shown to require at least $\boldsymbol{O}(n^{1.5})$ operations (George and Liu 1981). The number of interface nodes is also assumed to affect greatly the computation loads of static condensation because it constrains the matrix renumbering of the substructure (That is, the interface degrees of freedom should be renumbered after the internal ones). The value I_2 is raised to the power of 2, as shown in Eq. (10). More constraints on matrix renumbering generally lead to worse renumbering results. However, the effects of the constraints on the computation loads of matrix condensation can not easily be evaluated precisely, especially when the substructures are modeled as irregular meshes. Eq. (13) shows that the solution time for interface degrees of freedom is proportional to the cube of the number of interface nodes. The result is based on the fact that the condensed matrix is nearly always a dense matrix, and requires about $\boldsymbol{O}(N_{EN}^{i\;3})$ operations to factorize.

In the *Tuning Phase*, the weight of each element in the kth substructure is multiplied by a factor of F_k^i as shown below, to promote redistribution of elements from substructures with larger E_1^i to substructures with smaller E_1^i.

$$F_k^i = \left[\frac{E_{1,k}^i}{\mathrm{mean}(E_1^i)}\right]^{\omega} \tag{14}$$

$$\mathrm{mean}(E_1^i) = \left(\prod_{k=1}^{N_P} E_{1,k}^i\right)^{\frac{1}{N_P}} \tag{15}$$

Eqs. (14) and (15) show that the tuning factor F_k^i is normalized by the geometric mean of E_1^i of all substructures to prevent the weights of elements becoming very large or very small after several iterations. The coefficient ω is used to relax the tuning factor ($\omega < 1$) to prevent violent changes in mesh repartitioning in the following iterations.

5. Examples

Several finite element meshes with different shapes are used and actual parallel finite element analyses are performed on partitions of these meshes, to investigate the effectiveness of the proposed mesh partitioning approach. In the iterative mesh partitioning, both c_1 and c_2 (see Eq. (10)) are set to 0.5, while the coefficient ω (see Eq. (14)) is set to 0.8. This section presents some consistent results obtained from these numerical studies. A 2-D finite element model consisting of Q4 elements is partitioned into three substructures to show how the mesh partitioning varies during the iterations. A 3-D finite element model is then partitioned into four substructures by the proposed iterative mesh partitioning, to demonstrate the relationship between indicator E and the parallel finite element time, using different mesh partitioning results produced through different iterations.

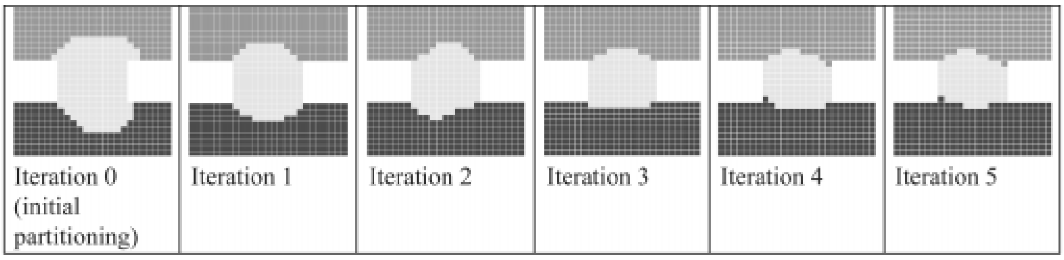

Fig. 3 Variations of mesh partitioning results

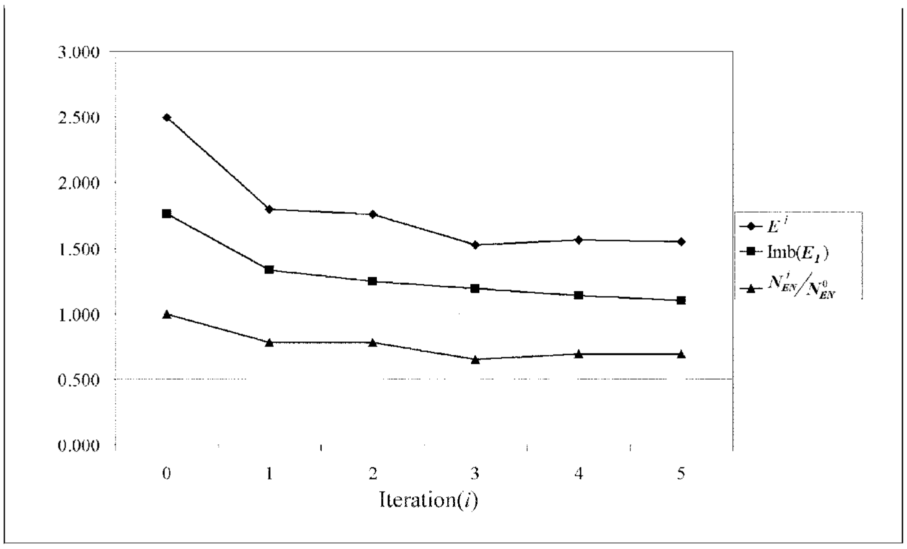

Fig. 4 Information about the mesh partitioning results

Finally, several 3-D finite element models are used and actual parallel finite element analyses are performed, to demonstrate the effects of the proposed iterative mesh partitioning.

Fig. 3 shows the variations of mesh partitioning results during the iterations using a 2-D finite element model consisting of Q4 elements. The middle substructure has more interface nodes than others and gradually becomes smaller (with less internal nodes) during iterations. Fig. 4 presents some statistical information on the mesh partitioning results during the iterations. **Imb**(E_1) in Fig. 4 denotes the ratio of the maximum to the minimum of indicator E_1 among substructures (with the value one representing good balance). Each substructure in the initial partitioning has almost the same number of elements because each element weight is set to a unique value (namely, 100). However, the indicators of substructure workload E_1 are not balanced because the middle substructure has more interface nodes than the others. The number of interface nodes in the middle substructure is twice that of the upper or lower substructure. The proposed iterative approach reduces the size of the middle substructure to balance E_1 among substructures, by increasing the element weights in the middle substructure and decreasing the weights in the upper and lower substructures. As the middle substructure shrinks, it has fewer internal nodes than the other

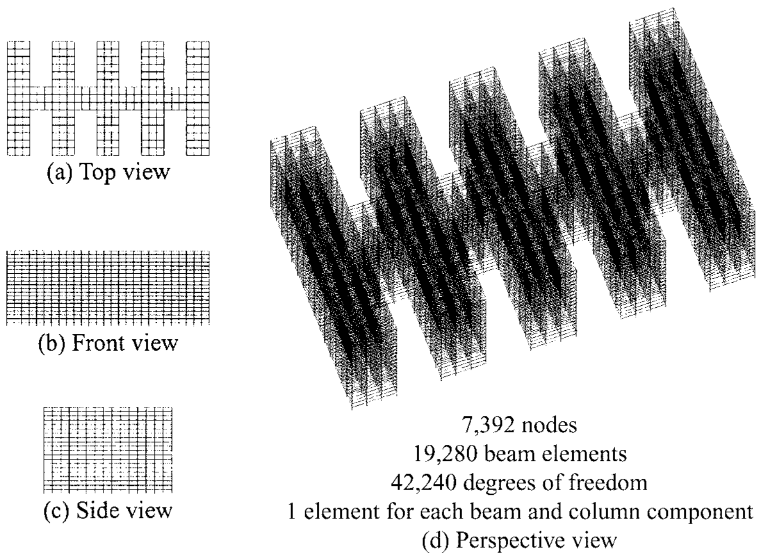

Fig. 5 Finite element model of H20

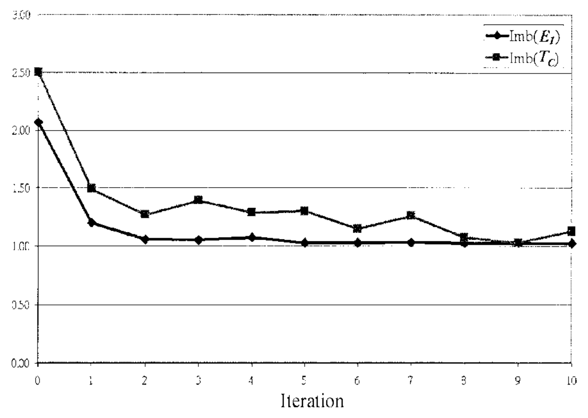

Fig. 6 Comparison between the imbalanced factor of indicator E_1 and substructure condensation time using model H20

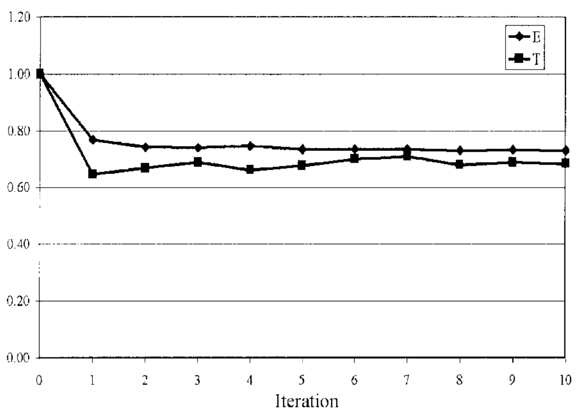

Fig. 7 Comparison between indicator E and parallel finite element analysis time T using model H20

substructures, resulting in a better balance of indicators E_1 among substructures (that is **Imb**(E_1)). The total number of interface nodes decreases during iterations (that is, the indicator E_2 decreases). Overall, the indicator E is reduced during the iterative procedures.

In the following test, a 3-D finite element model (named H20, as shown in Fig. 5) is partitioned into four substructures by the proposed iterative mesh partitioning, to show the relationship between indicators and the condensation times in actual parallel finite element analysis, and that between indicator E and the total parallel analysis time. The mesh partitioning results of the first ten iterations are used. T_C in Fig. 6 denotes the substructure condensation time in actual parallel finite element analysis. In this case, both E_1 and T_C become balanced during mesh partitioning iterations.

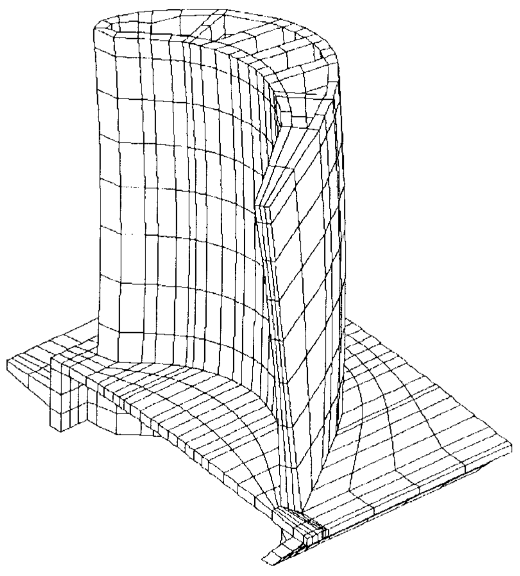

Fig. 8 Finite element model of BLADE

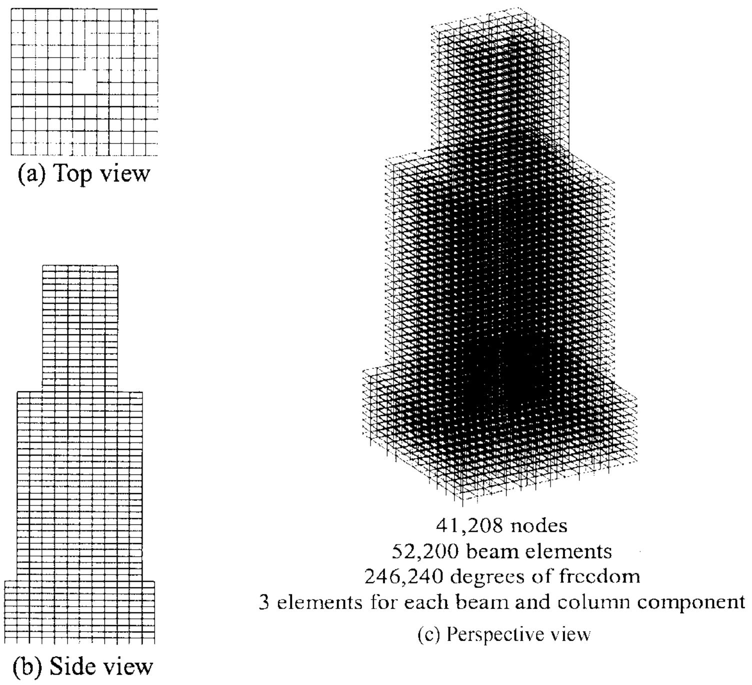

Fig. 9 Finite element model of HIGH60-1

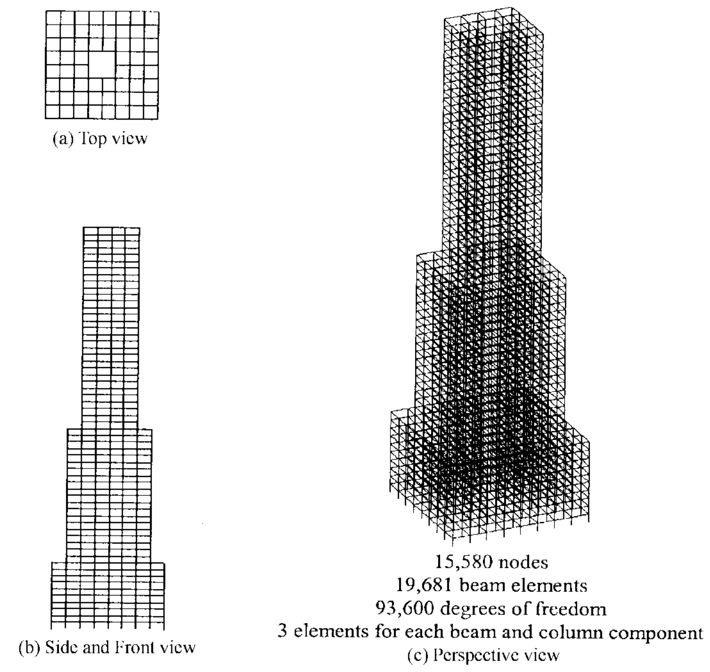

Fig. 10 Finite element model of HIGH60-2

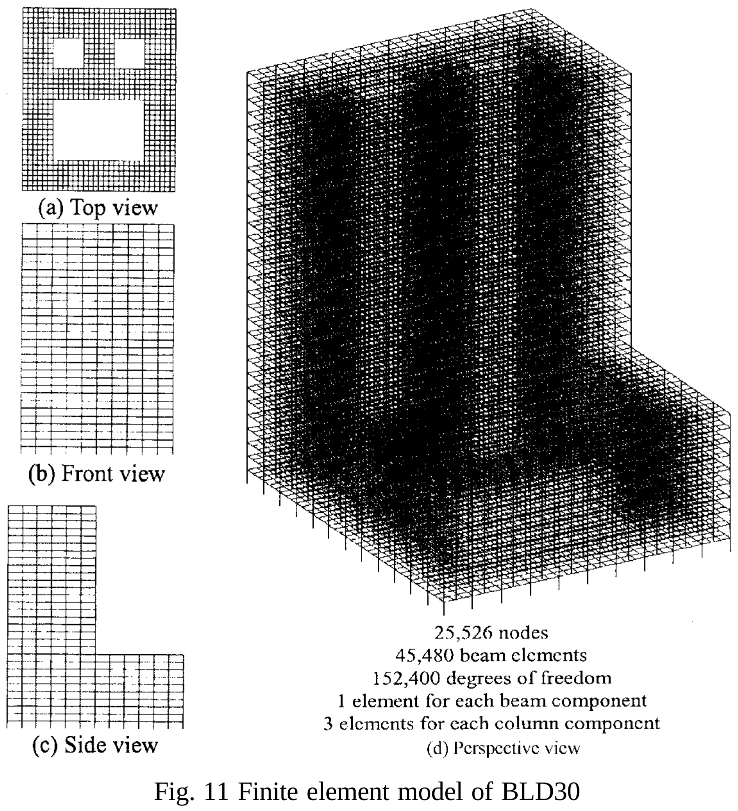
(a) Top view
(b) Front view
(c) Side view
25,526 nodes
45,480 beam elements
152,400 degrees of freedom
1 element for each beam component
3 elements for each column component
(d) Perspective view

Fig. 11 Finite element model of BLD30

Table 1 Indicators of initial partitions and best-*E* partitions

	T_{MP}	Iteration of best *E* (*i*)	E_1			E^i/E^0	N_{EN}^i / N_{EN}^0
			$\text{Imb}(E_1^0)$	$\text{Imb}(E_1^i)$	$\frac{\max(E_1^i)}{\max(E_1^0)}$		
BLADE	0.9	3	2.05	1.11	0.68	0.75	0.95
HIGH60-1	8.8	2	2.16	1.26	0.84	0.69	0.78
HIGH60-2	3.3	2	2.08	1.16	0.82	0.65	0.73
BLD30	7.8	2	2.61	1.38	0.80	0.80	0.93

Table 2 Elapsed time of parallel finite element analyses using four processors (in seconds)

	$\frac{\max(T_C^i)}{\max(T_C^0)}$	$\frac{T_I^i}{T_I^0}$	T_{Total}		
			T_{Total}^0	T_{Total}^i	$\frac{T_{Total}^i}{T_{Total}^0}$
BLADE	0.54	1.04	46.7	35.9	0.77
HIGH60-1	0.49	0.47	560.2	283.0	0.51
HIGH60-2	0.77	0.41	97.0	50.3	0.52
BLD-30	0.56	1.16	211.5	174.0	0.82

(See Fig. 6). Fig. 7 shows indicator *E* and total elapsed time *T* normalized by their values according to the initial partitioning. The result shows that *E* and *T* are approximately equal. Both *E* and *T* decline and converge to smaller values during iterative partitioning.

In the following tests, four finite element models (namely, BLADE, HIGH60-1, HIGH60-2, BLD30, as shown in Figs. 8 to 11 are used to demonstrate the effectiveness of the proposed iterative mesh partitioning approach. A maximum of three iterations are performed for each model. If *E* rises during the iterative partitioning, the preceding partitioning result (with minimal *E*) is chosen in parallel substructure finite element analysis. Each model is partitioned into four substructures, separated using only original partitioning (without iteration) and the proposed iterative partitioning (with five iterations). In the iterative mesh partitioning, the result corresponding to the lowest (best) indicator *E* is selected to perform parallel finite element analysis.

Table 1 lists various indicators of the initial partitioning results and the partitioning results with best (smallest) *E* values (iteration *i*) from among ten iterations. T_{MP} denotes the execution time for the iterative partitioning with ten iterations. Indicators E_1 among substructures become balanced in all of these models. **max**(E_1) among substructures and the indicator *E* both become small. In addition, the numbers of interface nodes among substructures is also reduced in all the test models even though the proposed iterative approach is originally designed to balance indicator E_1 (representing a balance of the overall number of internal and interface nodes).

Table 2 shows the elapsed time of the actual parallel finite element analyses using four processors. The superscript *i* denotes the iteration with the best indicator *E* over ten iterations. The value **max**(T_C) denotes the maximal elapsed time for matrix condensation among substructures; T_I denotes the time required for solving the interface degrees of freedom, while T_{Total} denotes the total elapsed time for parallel finite element analysis. The results show that the proposed iterative mesh partitioning approach reduces the matrix condensation time. The time required to solve the interface degrees of freedom is also reduced significantly in two of the four tests. Overall, the proposed approach efficiently reduces the total elapsed time for parallel finite element analysis.

6. Conclusions

This work has presented an iterative mesh partitioning approach to produce partitions that result in more efficient parallel substructure finite element computations, as shown by tested examples. The proposed iterative approach not only improves the balance of computational loads among substructures, but also reduces the total number of interface nodes.

More finite element meshes of different shapes should be considered to further test the effectiveness of the proposed approach. Further study should be conducted into the parameters and rules derived empirically from results of numerical experiments. The use of a different kernel partitioning algorithm (other than the JOSTLE algorithm used in this work) in the iterative process can be considered. Moreover, the application of this iterative approach to parallel solution algorithms other than the parallel substructure algorithm may be studied.

Acknowledgements

This work is an extension of a previous study presented in the Seventh East Asia-Pacific Conference on Structural Engineering & Construction (Hsieh *et al.* 1999) and was sponsored by the National Science Council, Executive Yuan, Republic of China under the Grant No. NSC 88-2211-E-002-018. The authors would like to thank Prof. C. Walshaw of University of Greenwich for providing the JOSTLE graph partitioning library, and Dr. C. Ashcraft of Boeing Shared Services Group for providing the MSMD object for matrix ordering.

References

Ashcraft, C., Pierce, D., Wah, D.K. and Wu, J. (1999), "The reference manual for SPOOLES, Release2.2: An object oriented software library for solving sparse linear systems of equations", Boeing Shared Services Group, USA. (available from http://www.netlib.org/linalg/spooles/).

Farhat, C. (1988), "A simple and efficient automatic FEM domain decomposer", *Comput. Struct.*, **28**(5), 579-602.

Geng, P., Oden, J.T. and van de Geijn, R.A. (1997), "A parallel multifrontal algorithm and its implementation", *Comput. Method Appl. Mech. Eng.*, **149**, 289-301.

George, A. and Liu, J.W.H. (1981), *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Inc., New Jersey, U.S.A.

Han, T.Y. and Abel, J.F. (1984), "Substructure condensation using modified decomposition", *Int. J. Numer. Meth. Eng.*, **20**(11), 1959-1964.

Hendrickson, B. and Leland, R. (1994), "The Chaco user's manual: Version 2.0", *Sandia Tech Report SAND94-2692*, Sandia National Laboratories, U.S.A.

Hsieh, S.H., Paulino, G.H. and Abel, J.F. (1995), "Recursive spectral algorithms for automatic domain partitioning in parallel finite element analysis", *Comput. Method Appl. Mech. Eng.*, **121**, 137-162.

Hsieh, S.H., Paulino, G.H. and Abel, J.F. (1997), "Evaluation of automatic domain partitioning algorithms for parallel finite element analysis", *Int. J. Numer. Meth. Eng.*, **40**(6), 1025-1051.

Hsieh, S.H., Yang, Y.S., Cheng, W.C., Lu, M.D. and Sotelino, E.D. (1998), "MPE++: An object-oriented mesh partitioning environment in C++", *Proc. 6th Asia-Pacific Conference on Structural Engineering & Construction*, January 14-16, 1998, Taipei, Taiwan.

Hsieh, S.H., Yang, Y.S. and Tsai, P.L. (1999), "Improved mesh partitioning for parallel substructure finite element computations", *Proc. 7th East Asia-Pacific Conference on Structural Engineering & Construction*, August 27-29, 1999, Kochi, Japan, 123-128.

Karypis, G., Schloegel, K. and Kumar, V. (2002), "ParMETIS parallel graph partitioning and sparse matrix ordering library Version 3.0", Technical report, Department of Computer Science/Army HPC Research Center, MN, U.S.A.

Kernighan, B.W. and Lin, S. (1970), "An efficient heuristic procedure for partitioning graphs", *The Bell System Technical J.*, **49**, 291-307.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983), "Optimization by simulated annealing," *Science*, **220**, 671-680.

Message Passing Interface Forum. (1994), "MPI: a message-passing interface standard", *Int. J. Supercomputer Applications*, **8**(3/4), 159-416.

Nour-Omid, B., Raefsky, A. and Lyzenga, G. (1987). "Solving finite element equations on concurrent computers", in Noor, A.K. (ed.), *Parallel Computations and Their Impact on Mechanics*, New York, U.S.A.

Simon, H.D. (1991), "Partitioning of unstructured problems for parallel processing", *Computing Systems in Engineering*, **2**(2/3), 135-148.

Vanderstraeten, D. and Keunings, R. (1995), "Optimized partitioning of unstructured finite element meshes", *Int. J. Numer. Meth. Eng.*, **38**, 433-450.

Vanderstraeten, D., Farhat, C., Chen, P.S., Keuning, R. and Ozone, O. (1996), "A retrofit based methodology for the fast generation and optimization of large-scale mesh partitions: Beyond the minimum interface size criterion", *Comput. Method Appl. Mech. Eng.*, **113**, 25-45.

Walshaw, C. (1999), "Serial jostle library interface: Version 1.1.8," Technical report, School of Computing and Mathematical Sciences, University of Greenwich, London, UK.

Walshaw, C., Cross, M. and Everett, M. (1997), "Parallel dynamic graph partitioning for adaptive unstructured meshes", *J. Parallel and Distributed Computing*, **47**(2), 102-108.

Yang, Y.S. and Hsieh, S.H. (1997), "Some experiences on parallel finite element computations using IBM/SP2", *Proc. 7th KAIST-NTU-KU Tri-Laterial Seminar/Workshop on Civil Engineering*, Kyoto, Japan, December 1-3, 1997, 63-68.