ABSTRACT: In this work, an iterative mesh partitioning approach is proposed to improve the efficiency of parallel substructure finite element computations. This approach employs an iterative strategy with a set of empirical rules that are derived from results of numerical experiments on a number of different finite element meshes. Two test examples are presented to demonstrate the effectiveness of the approach. The results indicate that the proposed approach can effectively improve the efficiency of parallel substructure finite element computations.

KEYWORDS: mesh partitioning, parallel finite element computations, parallel substructure method.

1. INTRODUCTION

Parallel substructure method has been one of the most popular approaches for parallel finite element computations. The method first partitions the structure into a number of substructures and assigns each substructure to a separate processor. Then, the processes of matrix condensation [1] of substructure interior degrees-of-freedom is performed independently and concurrently within each substructure without any inter-process communication. Finally, a parallel algorithm is often used to solve the condensed set of equations associated with unknowns along substructure interfaces. This approach reduces the cost of inter-process communication and, therefore, is usually more efficient than the outright use of parallel solver for the solution of the entire set of uncondensed equations.

To achieve high parallel efficiency of the parallel substructure method, it is important that the finite element mesh of the structure is partitioned in such a way that, among processors (or substructures), computational workloads are well balanced and inter-process communication is minimized. Although many mesh partitioning algorithms have been proposed in the literatures [2], most of them use only simple and general assumptions in their optimization process for load balancing without taking into account the characteristics of parallel solution algorithms. Therefore, it has been found that the mesh partitions produced by these algorithms usually do not lead to optimal parallel efficiency for a specific solution algorithm [3,4].

In this work, an improved mesh partitioning approach is proposed to obtain better mesh partitions for parallel substructure finite element computations. This approach uses 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. In addition, two test examples are presented to demonstrate the effectiveness of the present approach.
2. OPTIMIZATION CRITERIA IN MESH PARTITIONING

Most of the mesh partitioning algorithms proposed in the literature use the following two criteria for optimization in their heuristic approaches: (1) balance of the number of elements among substructures (or processors) and (2) minimization of the number of interface nodes among substructures. It is well known that the more the workload is balanced among processors, the less the waiting time is for those processors with lighter workloads. Therefore, the first criterion is mainly for the balance of computational loads (in this case, static condensation) among substructures, assuming the workload is proportional to the number of elements. The second criterion is for the reduction of both the inter-process communications and the size of the linear equations associated with the interface degrees-of-freedom. The minimization of the size of the linear equation can help improve the overall parallel efficiency because it is usually difficult for a parallel equation solver to achieve high efficiency.

However, many researches have shown that the workload of each substructure is usually not just proportional to the number of elements [3,4]. Mesh partitioning based on the above criteria often does not produce partitions that give a good balance of computational loads among processors. As a result, the overall efficiency of parallel analysis is not optimized. To achieve a better balance of workloads among substructures, a better optimization criterion is needed. However, the workload related to the matrix condensation of each substructure depends on several factors, such as nodal numbering, nodal adjacencies, and matrix sparsity [5], and can not be easily predicted using a simple formula.

On the other hand, from the authors’ experiences on parallel substructure finite element computations, it has been found that if a substructure has more interface nodes, it should have fewer elements to achieve a better balance of workloads among substructures. Therefore, this work proposes an iterative approach with a set of empirical rules to take into account the number of interface nodes within each substructure for balance of substructure workloads. The approach is discussed in details in Section 4.

3. COMPUTING ENVIRONMENTS

In this work, the proposed mesh partitioning approach is implemented in an object-oriented mesh partitioning environment, called MPE++ [6], in which several popular algorithms have already been implemented for mesh partitioning. In addition, a graph partitioning package, called METIS (version 2.0) [7] has also been embedded in MPE++. It should be noted that METIS randomly permutes the adjacency list of each vertices of a graph before partitioning it, and the mesh partitioning results may be different by using different compilers or operating systems. An object-oriented parallel finite element program, called PFE++ [8], is slightly modified and used to perform the parallel finite element analyses. In the modified version of PFE++, the modified Cholesky decomposition [5,9] is used for substructure condensation and the sparse matrix techniques are employed for matrix operations and storage [10]. All substructure computations are performed concurrently. However, instead of using a parallel equation solver, the present PFE++ performs sequential matrix factorization on a single processor to solve the condensed set of equations associated with the interface degrees-of-freedom. In addition, an MPI [11] based package, called PPI++ [12], is used by PFE++ to handle all message-passing tasks among processors.

A Pentium-166 PC running the Windows98 operating system is used for mesh partitioning. A PC cluster consisting of six 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 the speed of 100 Mbps.

4. IMPROVED MESH PARTITIONING APPROACH

An iterative approach with a set of empirical rules is proposed in this work to take into account the number of interface nodes within each substructure for balance of substructure workloads. After
partitioning the finite element mesh using one of existing mesh partitioning algorithms, this approach adjusts the weights of the elements in each substructure to promote redistribution of elements from substructures with more interface nodes to substructures with fewer interface nodes in the subsequent partitioning steps of the iterative process. The element weights of each substructure are adjusted mainly based on the ratio between the number of interface nodes of the substructure and the minimum number of interface nodes among all substructures. This rule for weight adjustment and other rules for termination of the iterative process are obtained empirically from results of many numerical experiments conducted by the authors. More discussions on the iterative partitioning process and the empirical rules are given below.

Figure 1 shows the process of the proposed iterative mesh partitioning approach. In the Partitioning Phase, a powerful and effective mesh partitioning algorithm is needed as the kernel algorithm for mesh partitioning. This kernel algorithm can be one of the existing algorithms available in the literature and is assumed to use the two common optimization criteria described in Section 2. However, it should satisfy the following two requirements:

1. The algorithm should permit assignment of different weighting values to the elements and nodes of the mesh and is capable of accounting for these weighting values in mesh partitioning.
2. The algorithm should produce reasonably good partitions in such an efficient way that the time for the entire iterative mesh partitioning process is bounded within a reasonable fraction of actual parallel finite element computations.

In this work, the METIS graph partitioning algorithm with Sorted Heavy-Edge Marching (SHEM) coarsening, Graph Growing Partitioning followed by Boundary Kernighan-Lin (GGPKL), and Boundary Greedy Kernighan-Lin Refinement (BGKLR) [13] is selected as the kernel partitioning algorithm for the iterative approach. The discussions on mapping from a mesh partitioning problem to a graph partitioning problem can be found in [14], and are not repeated here because of limited space. In this work, the communication graph approach [14] is used and the weights of all the vertices and edges in the graph are all initialized to the integer value of one.

In the Evaluating Phase, the ratio of the maximum number and the minimum number of the interface nodes among all substructures ($R_{BN}$) is calculated. If the value of $R_{BN}$ is smaller than (or equals to) a threshold value, the iterative process is terminated. Otherwise, the process continues to the Tuning
Phase and then enters the next iteration of partitioning. In addition, the total iteration number should be limited so that the timesaving in parallel computations due to better mesh partitioning results would not be offset by excessive time spent in mesh partitioning. In this work, the threshold value of 1.5 is used and the maximum number of iteration is set to 2.

In the Tuning Phase, to promote redistribution of elements from substructures with more interface nodes to substructures with fewer interface nodes, the weight of each element in the $i^{th}$ substructure is multiplied by a factor of $F_i$ as shown below:

$$F_i = \frac{\alpha \times N_{BN,i}}{\text{MIN}(N_{BN,i} \text{ for } i = 1, N_P)}$$

in which, $N_{BN,i}$ denotes the number of interface nodes in the $i^{th}$ substructure; $N_P$ denotes the total number of substructures; $\text{MIN}(.)$ denotes the minimum value in the parenthesis; $\alpha$ is a scaling factor. In this work, $\alpha = 2$ is used.

5. DEMONSTRATION EXAMPLES

In this work, to investigate the effectiveness of the proposed mesh partitioning approach, several finite element meshes of different shapes have been used and actual parallel finite element analyses have been performed on partitions of these meshes. Consistent results have been obtained from these numerical studies and two of them are presented in this section.

Figures 2 and 3 show the finite element meshes of a 20-story building and a turbine blade, respectively. The mesh of 20STORY (see Fig. 2) is partitioned into 3 substructures (i.e., $N_P = 3$), while the mesh of BLADE (see Fig. 3) is partitioned into 4 substructures (i.e., $N_P = 4$). Actual parallel finite element computations are performed on the partitions (of these two meshes) obtained from each partitioning step of the entire iterative partitioning process. The parameters used to control the iterative process and the timing statistics about mesh partitioning and parallel computations are recorded for each iterative step, as shown in Tables 1 and 2. In the tables, $N_{ITR}$ denotes the number of iterations performed in the proposed approach. As described earlier in Section 4, the maximum number of iterations is set to 2 in this work. In the case that $N_{ITR}$ equals zero, the mesh is partitioned only once and simply by the original METIS algorithm. It is listed here for comparison between mesh partitioning with and without the iterative process. $N_{BN}$ denotes the total number of interface nodes among substructures. $T_{MP}$ denotes the elapsed time spent in mesh partitioning, and $T_{PFEA}$ denotes the elapsed time spent in linear-static parallel finite element analysis on the partitioned mesh. $T_{SOLVING}$ denotes the elapsed time for solving the interface degrees of freedom. It should be noted that $T_{PFEA}$ includes $T_{SOLVING}$ and different computers are used for computations of mesh partitioning and parallel finite element analyses (see Section 3). In addition, $T_{PFEA+MP}$ denotes the sum of $T_{PFEA}$ and $T_{MP}$.
Figure 2 Finite element mesh of 20STORY  Figure 3 Finite element mesh of BLADE

From Tables 1 and 2, the following observations can be made:

1. The iterative mesh partitioning approach proposed can effectively reduce the elapsed time of parallel analyses ($T_{PFEA}$). This also means that the iterative process in the approach helps to improve the balance of workloads among processors and, therefore, increase the efficiency of parallel finite element analysis. Even if the iteration is performed only once, the reduction on $T_{PFEA}$ can be quite significant.

2. The elapsed time spent in each mesh partitioning within the iterative process (i.e., $N_{ITR} > 0$) is usually less than that spent in the initial mesh partitioning (i.e., $N_{ITR} = 0$).

3. Although $T_{MP}$ increases as $N_{ITR}$ increases, the total time spent in both mesh partitioning and parallel analysis ($T_{PFEA+MP}$) are still reduced by the proposed iterative approach. In these two examples, the savings in time are up to about 20% and 28%, respectively.

4. As the iteration proceeds, both the values of $R_{BN}$ and $N_{BN}$ decrease. This means that the proposed approach can also improve the balance of the number of interface nodes among substructures and reduce the total number of interface nodes. Because $N_{BN}$ is proportional to the size of the condensed system equations to be solved, the decrease of $N_{BN}$ often leads to the reduction on the elapsed time for solving the interface degrees of freedom ($T_{SOLVING}$).

<table>
<thead>
<tr>
<th>$N_{ITR}$</th>
<th>$R_{BN}$</th>
<th>$N_{BN}$</th>
<th>$T_{MP}$ (sec.)</th>
<th>$T_{PFEA}$ (sec.)</th>
<th>$T_{SOLVING}$ (sec.)</th>
<th>$T_{PFEA+MP}$ (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.0</td>
<td>242</td>
<td>4.4</td>
<td>79.6</td>
<td>18.5</td>
<td>84.0</td>
</tr>
<tr>
<td>1</td>
<td>2.0</td>
<td>242</td>
<td>6.8</td>
<td>61.8</td>
<td>18.5</td>
<td>68.6</td>
</tr>
<tr>
<td>2</td>
<td>1.5</td>
<td>218</td>
<td>9.7</td>
<td>57.1</td>
<td>14.1</td>
<td>66.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$N_{ITR}$</th>
<th>$R_{BN}$</th>
<th>$N_{BN}$</th>
<th>$T_{MP}$ (sec.)</th>
<th>$T_{PFEA}$ (sec.)</th>
<th>$T_{SOLVING}$ (sec.)</th>
<th>$T_{PFEA+MP}$ (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.2</td>
<td>617</td>
<td>1.2</td>
<td>86.3</td>
<td>22.3</td>
<td>87.5</td>
</tr>
<tr>
<td>1</td>
<td>1.7</td>
<td>597</td>
<td>2.3</td>
<td>75.2</td>
<td>23.4</td>
<td>77.5</td>
</tr>
<tr>
<td>2</td>
<td>1.6</td>
<td>515</td>
<td>3.4</td>
<td>59.4</td>
<td>21.2</td>
<td>62.8</td>
</tr>
</tbody>
</table>

6. CONCLUSIONS

An iterative mesh partitioning approach has been proposed in this work to produce partitions that lead to better efficiency of parallel substructure finite element computations. From all the examples tested in this work (although more tests are still needed), the results indicate that the proposed approach can effectively improve the efficiency of parallel substructure finite element computations. In addition, 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 used to further test the effectiveness of the proposed iterative approach. Further study should be also conducted on the parameters and rules that are needed in the approach and derived empirically from results of numerical experiments. In addition, the use of a different kernel partitioning algorithm (i.e., other than the METIS algorithm used in this work) in the iterative process can be investigated. Moreover, the application of this iterative approach to improvement of other parallel solution algorithms (other than the parallel substructure algorithm) can be studied.
ACKNOWLEDGEMENT

This work is sponsored by the National Science Council, Executive Yuan, Republic of China under the Grant No. NSC 88-2211-E-002-018.

REFERENCES


