Technical Report of
ADVENTURE Project
ADV-99-1 (1999)

PARALLEL DECOMPOSITION OF 100-MILLION DOF MESHES INTO HIERARCHICAL SUBDOMAINS

Hiroyuki TAKUBO and Shinobu YOSHIMURA
School of Engineering
University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-8656 Japan
e-mail: takubo@garlic.q.t.u-tokyo.ac.jp

Key words: Domain Decomposer, Domain Decomposition Method, Parallel/Distributed environment, Graph, Multilevel Partition

1. Introduction

This study describes the implementation and performance of a domain decomposer in a parallel/distributed environment, which can decompose finite element meshes with over 100-million DOFs into parts and subdomains of specified numbers for the hierarchical Domain Decomposition Method (DDM). METIS and ParMETIS which are graph-partition package were used as a basic domain decomposing algorithm and this system improved the quality of the decomposed meshes, i.e. the iteration number which was needed to reach convergence in a FEM analysis using the Domain Decomposition Method (DDM) was reduced.

As computer technology has progressed, the demand and the requirements for the Finite Element Method (FEM) have increased. In FEM analysis for more general (or more complex) problems, the DOFs needed to obtain sufficiently accurate results become extraordinarily large, e.g. 100-million or more. It is almost impossible for one processor to process a huge number of problems in regard to both calculation time and on board memory. Parallelization and distribution are therefore necessary in any FEM processes of pre-, main- and post-.

An ordinary FEM analysis with one processor approximately consists of three parts, as is shown in Figure 1 (a): pre-process (e.g. mesh generator), main-process (e.g. FEM-code), and post-process (e.g. visualizer, etc.). On the other hand, it is necessary for an analysis with DDM to decompose FEM meshes into subdomains and generate DDM data, *ex*. inner boundary conditions. The Domain Decomposer, therefore, is between the pre-process and the main-process in the ordinary FEM analysis flow, as is shown in Figure 1 (b).

The Domain Decomposer (as the name indicates) decomposes FEM meshes into subdomains. During this process, it must meet the two following requirements. First, the calculation load of each decomposed subdomain is as uniform as possible in order to avoid load imbalances and the concentration of communication among the processors in the DDM analysis. The second is to minimize the number of inner boundary conditions. DDM is an iterative method, and it is known that the fewer the number of inner boundary conditions there are, the faster the convergence should be obtained. Therefore, it is very important to appropriately choose which algorithm should be taken to decompose. METIS and ParMETIS were chosen for this purpose, and satisfactory results were obtained.

The entire system is implemented in C-language with its universal libraries, and MPI, message-passing libraries, is used for communication among processes. This system, therefore has high portability and can run in almost any UNIX-based parallel environment (e.g. massively parallel processors, work-station/personal-computer clusters, etc.).

2. Domain Decomposition Method (DDM)

As DDM is described in detail in the subsequent section, "4. MAIN PROCESSES", it will be explained briefly here.

DDM is one method which is used to parallelize FEM analysises. In FEM analysis using DDM, the FEM mesh is decomposed into a number of subdomains *a priori*. Each subdomain must have data about connectivity to its adjacent subdomains because the object which is being analyzed is originally continuous. In DDM, they are considered to be new boundary conditions, and are called 'Inner Boundary Conditions', in contrast to outer boundary, Dirichlet boundary, or Neumann boundary conditions, which are assigned to the original problem (Figure 2). As each subdomain can be analyzed with an FEM code, by distributing each subdomain to processors, whole domain can be anlysed in parallel. After all of the subdomains have been analyzed with FEM, the inner boundary conditions are examined. If they have converged, the stresses and strains on all nodes can be calculated. If not, the inner boundary conditions are modified and all of the subdomains are analyzed again. Therefore, DDM is an iterative method.

In DDM, as each analysis of the subdomains is done using an FEM, the vast properties of past FEM can be reused almost without modification. In addition, because

DDM is independent of any hardware, it can be used on many parallel systems, e.g. massively parallel processors, workstation/personal-computer clusters, etc.

The problem that arises when applying DDM to a parallel system is to maintain the load-balance among the processors, because the time required analyzing a subdomain is in general different for each one. In order to avoid a decrease in parallel efficiency which comes from an unbalanced load distribution, dynamic load distribution is commonly applied to the DDM. The load balance among processors could be maintained by decomposing the FEM meshes into a larger number of subdomains than that of the processors which were used in the DDM analysis. The 'Parent-Child model' is widely used (as is shown in Figure 3 (a)) to actualize dynamic load balancing. Here, however, the targets of authors to analyze are over 100 million DOF problems. When approaching problems with such a vast number of DOFs, it is actually impossible to keep all the data in the memory of one Parent-processor. In order to solve this problem, the authors introduced the 'Grand-Parent-Child model' (Figure 3(b)) and distributed data to each Parent process.

3. Domain Decomposer

The purpose of the Domain Decomposer is to decompose FEM meshes into subdomains for the DDM, i.e. the Domain Decomposer makes groups of elements which consist of the original FEM meshes as if those groups are to be fragments of the original meshes.

In general, the requirements for the Domain Decomposer are the following:

(1) Physical requirements: In general, there is a trade-off between speed and memory-use. In the present study, the authors emphasize an efficient use of memory rather than computation time reduction as to deal with extra-large scale meshes.

(2) Minimizing of the number of inner boundary conditions: As noted in 2.1, DDM is an iterative method. The total time for an analysis using DDM is in proportion to the number of iterations which is needed to obtain a convergence. On the other hand, it is well known that when analyzing the same FEM problem using DDM, the fewer inner boundary conditions there are, the fewer the number of iterations to reach a convergence. Therefore, minimizing the number of inner boundary conditions is very important in order to shorten the time for an FEM analysis using DDM.

(3) Load balancing among subdomains: As mentioned in 2.1, DDM is actualized using dynamic load balancing among processors in order to prevent a decline in the parallel efficiency. In dynamic load balancing, the load imbalances among subdomains can cause a decline in parallel efficiency at the end of every iteration. When the number of

subdomains of smaller size is much more than that of larger ones, communication from Child PEs to Parent PEs should be concentrated in the end of each iteration process, and then parallel efficiency reduces. In DDM, the communication between a Parent and a Child is invoked at every point when the child finishes an analysis of a subdomain. Therefore, when the communication time ratio to the calculation time becomes greater, a parallel efficiency declines. As a result, it is desirable to maintain the load balancing among subdomains.

There are many algorithms that can decompose FEM meshes into subdomains. The authors adopt METIS, which is a graph-partition package that meets the above requirements. About METIS and ParMETIS will be described in detail in a later section.

After decomposing the FEM meshes into subdomains, the domain decomposer makes inner boundary conditions, and rewrites data in the DDM format which the DDM system can analyze.

4. METIS and ParMETIS

METIS is graph-partition package which has been developed at the University of Minnesota by George Karypis and Vipin Kumar [Karypis and Kumar, 1995; Karypis and Kumar, 1996]. Its main purpose is to create a partition which is close to the answer of a given k-way graph partitioning problem.

The k-way partitioning problem of a graph involves making a specified number of partitions of vertices by grouping those which are adjacent to each other by the edges, while each partition has the same number of vertices and the edge-cut produced by the partitioning is minimal (see Figure 4). Here, an edge-cut is defined as the number of edges which are cut accompanying the partitioning. This problem is known as NP-complete and, when the number of vertices is adequate, it is impossible to solve. A solution to this problem involves making a partition which is as close to the answer as possible by relaxing certain restraints.

METIS uses a Multilevel Partitioning Method (MPM) as its algorithm in order to solve the k-way graph partitioning problem. The Multilevel Partitioning Method is an effective algorithm that can solve k-way partitioning problems for very large scale graphs in a relatively short amount of time with a high level of quality, or smaller remaining edge-cut balances of the vertices among the partitions. It consists of roughly three phases: coarsening, initial partitioning, and uncoarsening and refinement. In the coarsening phase, the original graph is coarsened into the graph with smaller number of vertices by collapsing a pair of vertices into a new heavier vertex. This phase is repeated several times until the number of vertices is small enough. In the initial partitioning phase, the coarsened graph is divided into a specified number of partitions. The number of graph become smaller, and the time needed to obtain the partitions is limited because of the coarsening phase. Finally in the uncoarsening and refinement phase, the partitioned graphs are uncoarsened by one coarsening level and refined to obtain a better partition. This phase is also repeated until the level reaches the original.

METIS can partition a graph in a shorter amount of time with a higher level of quality when compared with other multilevel partitioning schemes [Gondran, 1997].

When viewing the FEM meshes as a type of graph, METIS can be used as a domain composer (see Figure 5). From this perspective, the partitions produced via METIS correspond to the domains, and the edge-cut of the graph to the number of the inner boundary conditions, i.e. by decomposing the FEM meshes with METIS, high quality domains which satisfy the requirements in 2.2 can be obtained. ParMETIS is the parallel version of METIS. Its main purpose, its algorithm, and the quality of the partitioned graphs are almost the same as that of METIS.

5. Domain Decomposer with METIS for Father-Child Model DDM

5.1 Method

The authors made a domain decomposer for the Parent-Child Model DDM, and tested its performance by ensuring the efficiency of METIS as an algorithm for a domain decomposer. Its input is a FEM mesh, and its output is the DDM meshes for the Parent-Child Model DDM.

As the output is for a Parent-Child Model DDM, the size of the FEM meshes can be at most one-million or slightly higher. It runs on only one processor and is not parallelized/distributed.

5.2 Analysis flow

The processes of the system from the input to the output consist of three phases :

(1) Pre-Filter phase:

The input FEM mesh is converted into the corresponding graph because METIS can only partition graphs. Because the DDM which is used here only regards domains as the assembly of elements, the graph is converted in an element-oriented way, i.e. the elements of the FEM mesh correspond to the vertices of the graph and the element-element connectivities are corresponded to the edges.

(2) Partitioning phase:

The graph which was converted in the Pre-Filter phase is partitioned by METIS.

(3) Post-Filter phase:

The graphs partitioned in the previous phases are regenerated into subdomains, or meshes which correspond to DDM. In addition for DDM analysis, the inner boundary conditions are generated and added to each subdomain. The final output is arranged into one file.

5.3 Implementation

All codes including the METIS libraries are written in ordinary C, and can be compiled and build on almost any UNIX system with a C compiler and linker.

5.4 Results and discussion

Table 1 is a comparison between the new and the old system which has been used up to this point, which uses a Recursive Bisection Method (RBM) with the geometric data on FEM meshes [Shioya, 1995]. The new system with METIS decomposes an FEM mesh with one-million DOFs into the same number of subdomains with fewer inner boundary conditions in a shorter time than the old system. Because METIS and the other part of the present system use memory independently, the total memory needed is more than that for the old system. The load balances among the subdomains show the ratio between the maximum and minimum time needed to analyze each subdomain with an FEM. Using METIS, load balancing is superior to that in the old system. Another advantage of using METIS is that it can decompose FEM meshes into a specified number of subdomains.

The above results confirm that METIS is a highly efficient domain decomposer.

6. Parallel/Distributed Domain Decomposer for Grand-Father-Child Model DDM

6.1 Method

The efficiency of METIS as an algorithm for a domain decomposer is shown in Section 3, and this section describes the extension of this system to the parallel/distributed environment.

The ultimate objective of this system is to decompose FEM meshes that are over 100 million DOFs. When processing meshes with such huge DOFs, the memory limitation becomes a serious problem. For example, the size of one million DOFs FEM mesh is about 30 Mbytes, and that of 100 million DOFs is about 300 Mbytes. The required DOFs for the FEM analysis will keep increasing, and it is obvious that it will be difficult to store all the data on the memory of one processor. Moreover, the time needed to decompose them increases over linear to the DOFs. It is therefore necessary to build a parallelized/distributed system.

A distributed system is also required in DDM analysis, and the output of this system is for a Grand-Parent-Child model DDM, i.e. the original FEM mesh is decomposed into several parts each of which consists of subdomains. Each part corresponds to a Parent, which makes it necessary to create data about which parent owns the inner boundary condition.

ParMETIS can partition graphs which are distributed to processors. However, ParMETIS has a drawback, which is that it can decompose graphs only into the number of partitions which is equal to that of the processes in the MPI environment. In a DDM analysis of over 100 million DOFs, as the number of subdomains should be more than several thousands, it is impossible to decompose. The hierarchical decomposing method (see Figure 6) was used to overcome this obstacle. In the first step, the original FEM mesh is decomposed into parts with ParMETIS. Each part corresponds to the part of a Parent in an analysis using Grand-Parent-Child model DDM. In the next step, each part is decomposed into subdomains using METIS on each processor independently. In this way, huge FEM meshes can be decomposed into thousands of subdomains.

The input of this system is an FEM mesh (quadratic tetrahedron), and its output is the DDM meshes for the Grand-Parent-Child Model DDM.

6.2 Analysis flow

There are roughly five phases from input to output. Each phase is parallelized and distributed with MPI libraries.

(1) Pre-Filter phase:

This phase is almost the same as that of the system in Section 3. The input FEM mesh is converted into a graph: however, as the next partitioning-1 phase is by ParMETIS, the generated graphs are also distributed to processors.

(2) Partitioning-1 phase:

The graphs which were converted in the Pre-Filter phase are partitioned into parts by ParMETIS.

(3) Converter phase:

The global data of the original FEM mesh are localized in each partitioned part, and the inner boundary conditions between the parts are generated.

(4) Partitioning-2 phase:

The localized graphs are partitioned by METIS. Although each process of the METIS is independent, they run on each processor at the same time in a parallel fashion.

(5) Post-Filter phase:

The graphs partitioned in the previous phases are regenerated into subdomains, or meshes which correspond to Grand-Parent-Child model DDM. As for the analysis with DDM, the inner boundary conditions are generated and added to each subdomain. The final outputs are arranged into the number of files which is equal to that of the processors.

6.3 Implementation

All codes including the libraries of METIS and ParMETIS are written in C-language. As a communication tool among processors, MPI (Message-Passing Interface) is used in all codes. ParMETIS is also built using MPI for communication. MPI is well known as a standard specification for message-passing libraries, and is supported on many type of parallel hardware, massively parallel processors, workstation clusters, PC clusters, therefore, these systems can be compiled and built on almost any parallel computing environment.

6.4 Results

Using this system, FEM meshes were decomposed into parts and subdomains on a PC cluster with 25 processors which consisted of DEC Alpha 533MHz with 256Mbyte memory. The actual number of processors used is dependent on the number of Parents which will be used in the analysis with DDM. FEM meshes with about 100-million DOFs can be decomposed in two hours with this system. As an example, the appearance of decomposition of a Pantheon-model FEM mesh with two-million DOFs is shown in Figure 7.

7. Conclusion

A hierarchical domain decomposing system on parallel/distributed environments was developed. METIS was used and its efficiency as an algorithm for decomposing meshes was demonstrated. Using this system, an FEM mesh with over 100-million DOFs was decomposed into parts and subdomains for Grand-Parent-Child DDM in a few hours.

References

Shioya, R. (1995), Massively Parallel Computing for Large Scale Finite Elements, Doctoral Thesis, University of Tokyo, Japan.

Gondran, N. (1997), Comparison of Three Prepartitioning Packages, hpc profile THE NATIONAL PUBLICATION FOR HIGH PERFORMANCE COMPUTING APPLICATIONS AND TECHNIQUES, 16, pp.3-6.

Karypis, G. and Kumar, V. (1995), A fast and high quality multilevel scheme for partitioning irregular graphs, Technical report TR 95-035, Department of Computer Science, University of Minnesota, USA.

Karypis, G. and Kumar, V. (1996), Multilevel k-way Partitioning Scheme for Irregular Graphs, Technical report TR 96-064, Department of Computer Science, University of Minnesota, USA.

Karypis, G. and Kumar, V. (1996), Parallel multilevel k-way partitioning scheme for irregular graphs, Technical report TR 96-036, Department of Computer Science, University of Minnesota, USA.

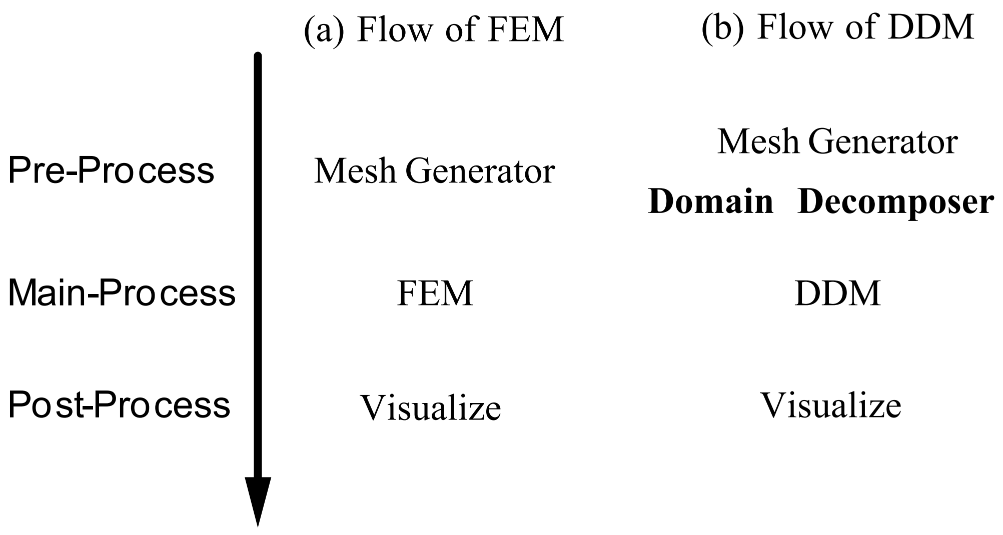


Figure 1 Flow of FEM and DDM

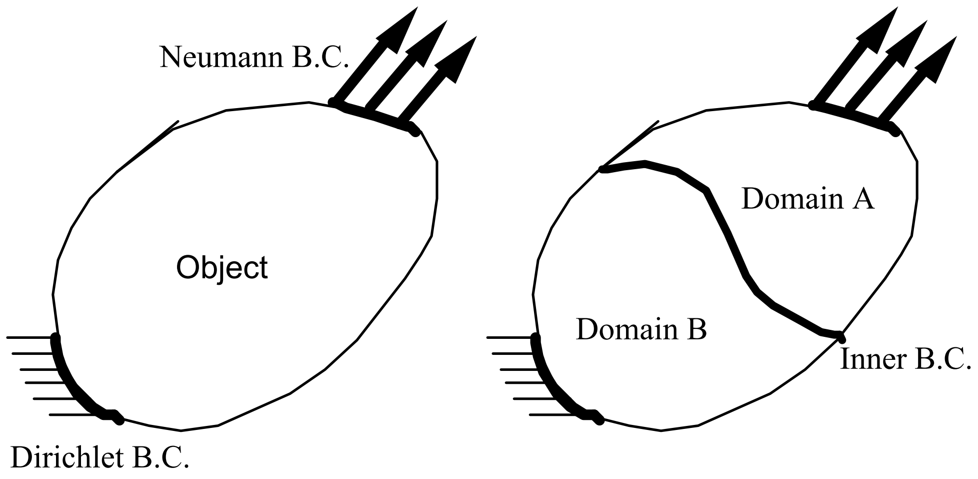
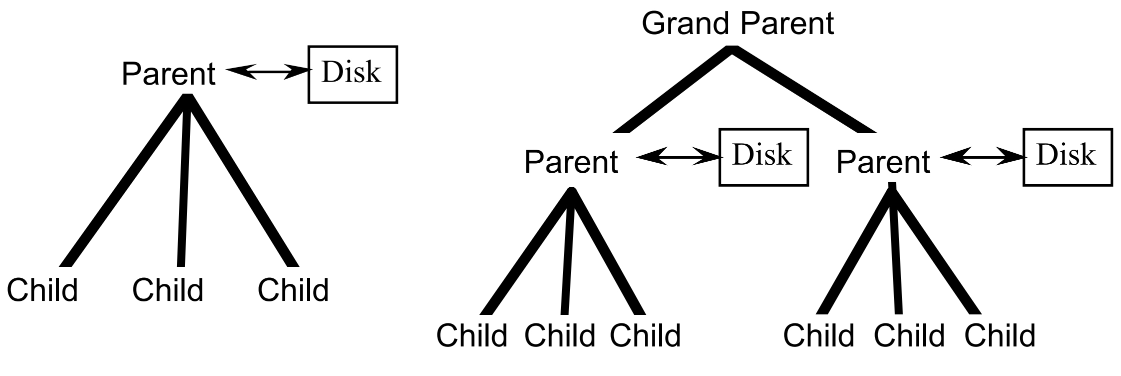


Figure 2 Boundary conditions



(a) Parent-Child Model

(b) Grand-Parent-Child Model
(The number of Parents can be more than two.)

Figure 3 Two kinds of DDM models

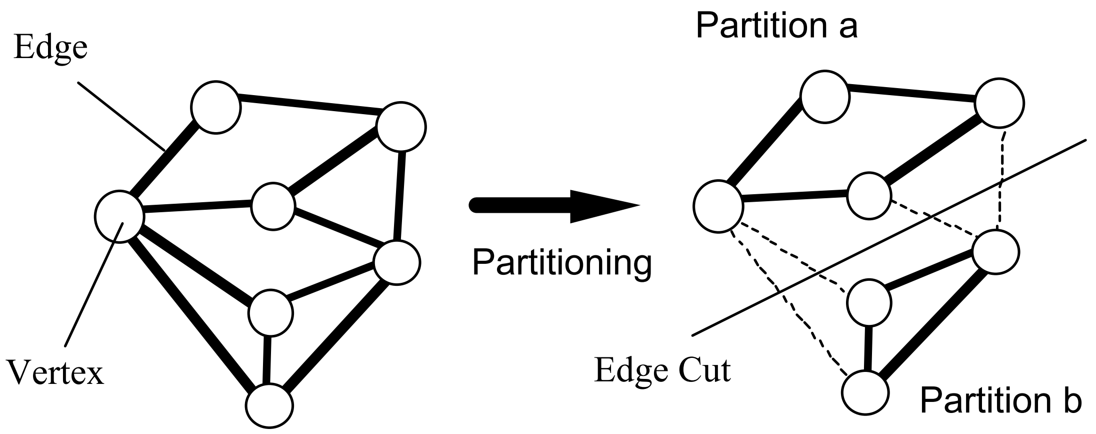


Figure 4 k-way partitioning problem of a graph

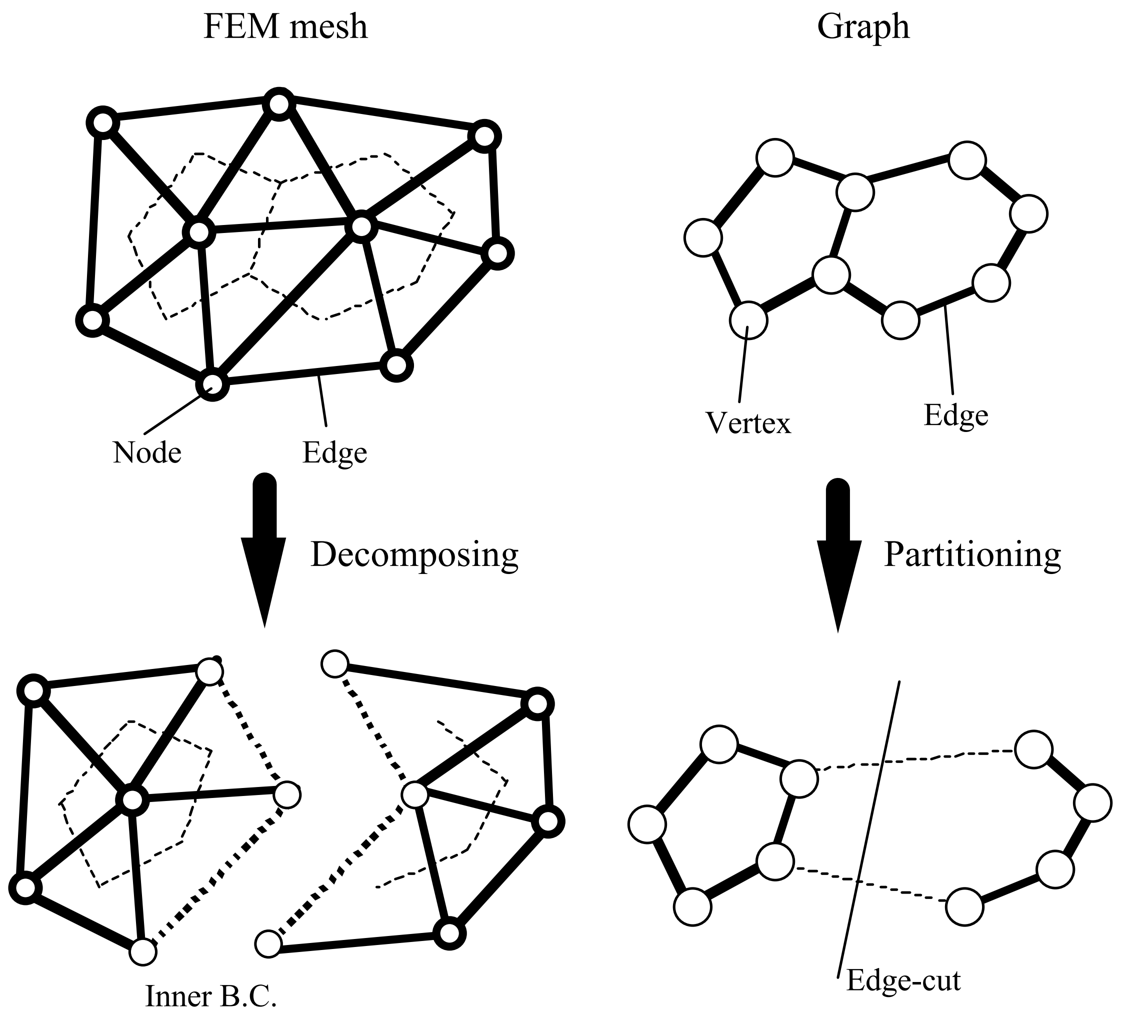


Figure 5 Correspondence between an FEM mesh and a graph

Table 1 Comparison between the old and new systems

	Old System (RBM)	New System (MPM)
Execution Time(min)	28	20
Memory Usage (MBytes)	40	72
Number of Inner B.C.	527,982	487,323
Ratio of Min/Max Computations of Subdomains	30 - 40	3
Others		Num. of Subdomains can be Specified

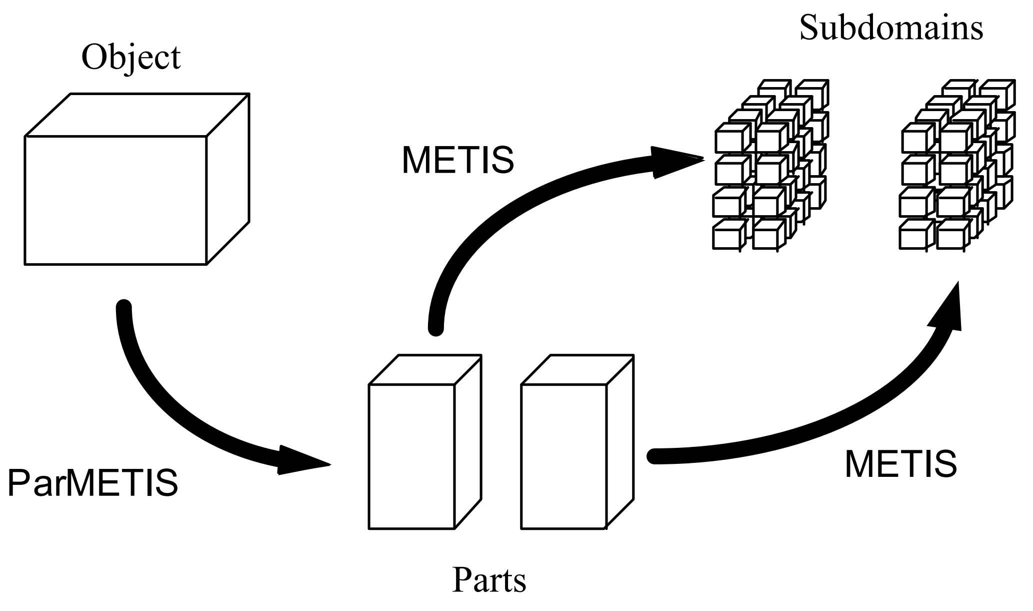


Figure 6 Hierarchical decomposition

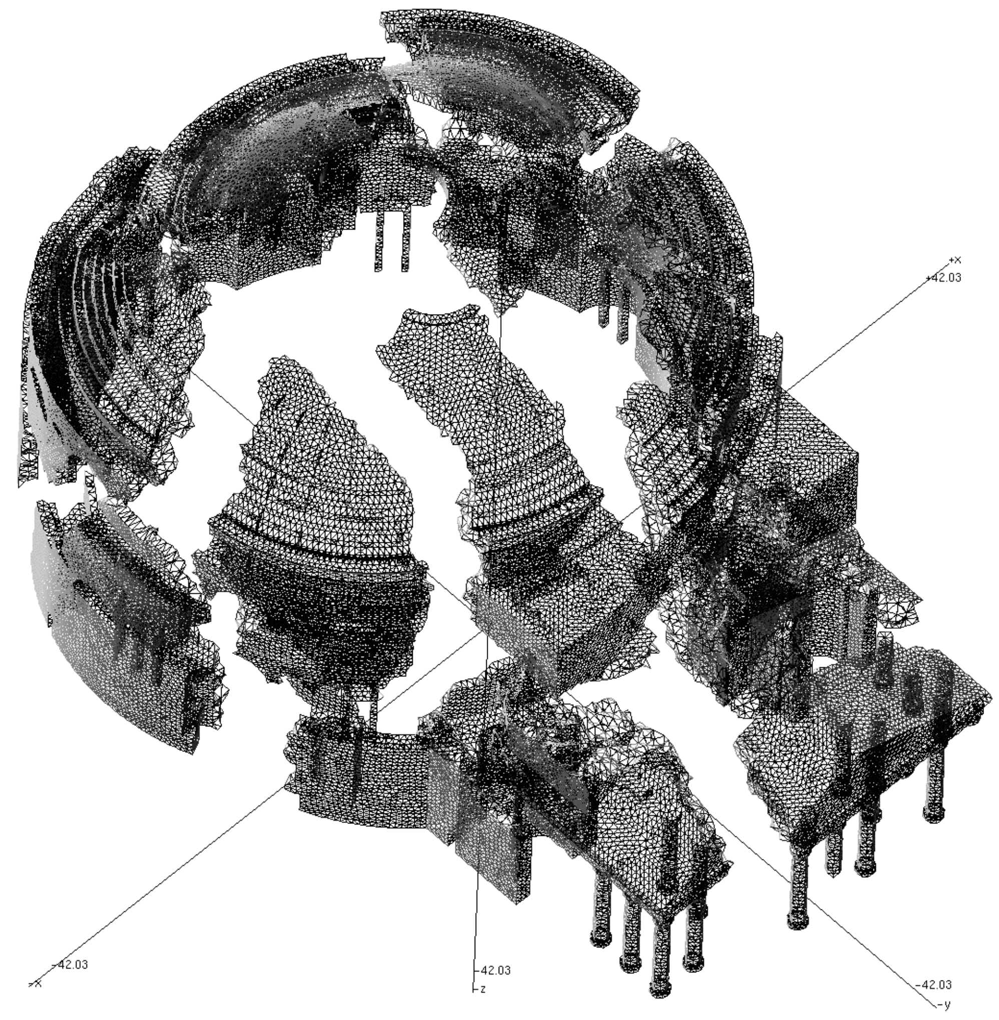


Figure 7 The Pantheon-model with Two-million DOFs (divided into 16)