Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 751030, 6 pages
http://dx.doi.org/10.1155/2013/751030

Research Article

Mesh Partitioning Algorithm Based on Parallel Finite Element Analysis and Its Actualization

Lei Zhang,[1] Guoxin Zhang,[1] Yi Liu,[1] and Hailin Pan[2]

[1] *State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, China Institute of Water Resources and Hydropower Research, Beijing 100038, China*
[2] *Beijing Institute of Water, Beijing 100044, China*

Correspondence should be addressed to Lei Zhang; zhangl@iwhr.com

Received 18 July 2013; Accepted 4 September 2013

Academic Editor: Zhiqiang Hu

In parallel computing based on finite element analysis, domain decomposition is a key technique for its preprocessing. Generally, a domain decomposition of a mesh can be realized through partitioning of a graph which is converted from a finite element mesh. This paper discusses the method for graph partitioning and the way to actualize mesh partitioning. Relevant softwares are introduced, and the data structure and key functions of Metis and ParMetis are introduced. The writing, compiling, and testing of the mesh partitioning interface program based on these key functions are performed. The results indicate some objective law and characteristics to guide the users who use the graph partitioning algorithm and software to write PFEM program, and ideal partitioning effects can be achieved by actualizing mesh partitioning through the program. The interface program can also be used directly by the engineering researchers as a module of the PFEM software. So that it can reduce the application of the threshold of graph partitioning algorithm, improve the calculation efficiency, and promote the application of graph theory and parallel computing.

1. Introduction

Currently, numerical simulation has become an important means to solve complex scientific and engineering problems. The scientific and technical workers in many fields have developed specialized CAE calculation program, offering solution to a variety of practical problems. However, as the problems arising from simulation become more complex, many professional softwares fail to meet the current demand in computation speed, data accuracy, and computation scale. With increasing the speed and scale of calculation as the primary objective, high-performance computing has become a fundamental solution to this problem. A great number of commercial CAE softwares have developed the parallel computing function, mostly by performing parallel computing based on domain decomposition. Domain decomposition refers to the coarse-grained decomposition of the entire computational domain. Therefore, the effectiveness and efficiency of domain decomposition have a great influence on calculation softwares.

2. Graph Partitioning Method

The optimal partitioning of irregular and nonstructural graphs is the key to efficient and scientific simulation on high-performance parallel machine. For finite element computing, especially that on distributed storage system, the physical meshes need to be mapped onto each processor to ensure the consistency of the number of mesh cells and the minimal exchange of information between the processors. Therefore, a proper way is to define the mesh as a graph and then perform graph partitioning [1]. The purpose of graph partitioning is to ensure the same size and minimal edgecut for each subdomain. It can be defined by giving a weighted undirected graph $G = (V, E)$, with a weight for each vertex and edge. In K-path partitioning of a graph, vertex set V is divided into k nonintersecting subdomains or subsets, making the sum of the weights of all the vertices of these subdomains roughly equal. For those edges whose vertices are distributed in different domains, their sum of weight needs to be minimized.

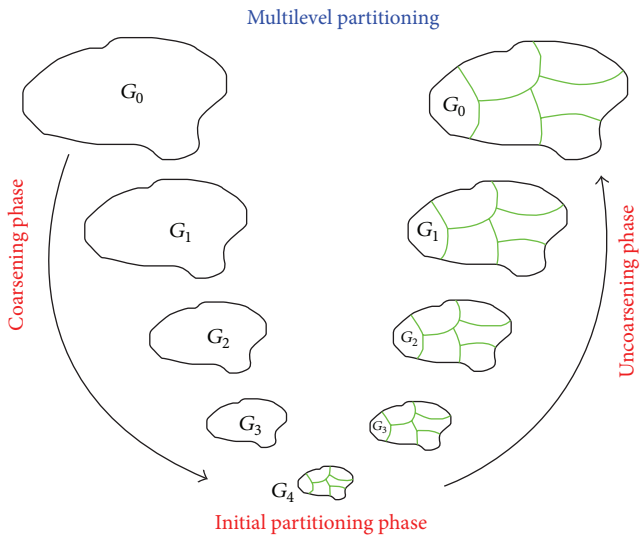

FIGURE 1: Multilevel partitioning process.

The methods for graph partitioning include geometric method [2, 3], spectral method, combination method [4], multilevel method [5–7], dynamic repartitioning method, parallel graph partitioning, and multiconstraint and multiobjective partitioning. Multilevel methods (e.g., multilevel recursive bisection and multilevel K-path partitioning method) are the newly developed techniques. On the basis of multilevel ideas, this method consists of three stages: coarsening, initial partitioning, and multilevel optimization, as shown in Figure 1 [8]. In the coarsening stage, some vertices are merged to form the coarsening graph of the next level. This process is repeated until the coarsening graph is small enough; then initial partitioning is performed on the smallest graph, because the small size of the coarsening graph implies short time of execution. In the stage of refining and optimizing from the most coarse to the most refined graph, optimization is performed in the coarsening graph at each level. G_0 is the initial graph input, while $G_i + 1$ is the next level of coarsening graph of G_i.

The *k-way* graph partitioning problem is defined as follows. Given a graph $G = (V, E)$ with $|V| = n$, partition V into k subsets, $V_1, V_2, \ldots, V_k$ such that $V_i \cap V_j = \emptyset$ for $i \neq j$, $|V_i| = n/k$, and $\cup_i V_i = V$, and the number of edges of E whose incident vertices belong to different subsets is minimized. A k-way partition of V is commonly represented by a partition vector P of length n, such that for every vertex $v \in V$, $P[v]$ is an integer between 1 and k, indicating the partition at which vertex v belongs. Given a partition P, the number of edges whose incident vertices belong to different subsets is called the *edge-cut* of the partition. Multilevel method has high robustness. In terms of partitioning speed and quality, multilevel approach is superior to spectral method [9]. Multilevel recursive bisection has already been actualized in many software packages, such as Chaco, METIS, and Scotch. Multilevel k-path partitioning of the graph has also been actualized in METIS and Jostle [10]; recently, some improvement of the multilevel graph partitioning algorithms are presented, for example, multilevel spectral clustering algorithms [11], which improve the partitioning quality and minimize partitioning time. Some other new algorithms like isoperimetric algorithm and spectral rounding are also have their own advantages [12, 13].

For large-scale graph partitioning (above the level of tens of millions), the internal memory of a single processor is insufficient. For the parallel computing for self-adaptive problem, repartitioning needs to be done for the graph. In the serial part, the graphs need to be gathered to a single processor. This would become a serious bottleneck affecting the computing efficiency of the entire process. Thus, parallel graph partitioning is especially important. At present, parallel graph partitioning mainly targets the parallelization of geometric method, spectral method, and multilevel method. Geometric method is easy to be parallelized, but usually a parallel sorting algorithm is needed. By contrast, spectral method and multilevel method are more difficult to be parallelized. The execution time for these methods is equal to the time needed for the parallel vector multiplication for the randomly distributed matrix. For the initial graph after initial partitioning, the times needed for the parallel multilevel method and spectral method are reduced to the time needed for the vector multiplication for an already partitioned matrix. Generally, static graph partitioning does not yield initial partitioning, while high-quality initial partitioning is obtained in repartitioning. Therefore, the time for the parallel execution of self-adaptive problem is shorter than the time for static graph partitioning. Thus, initial partitioning needs to be done with geometric partitioning method, followed by multilevel repartitioning to improve parallel efficiency. At present, both ParMetis and Jostle have actualized parallel multilevel partitioning [14]. Now, an OpenMP based implementation of the Metis algorithms has been developed [15].

3. Mesh Partitioning and Graph

Parallel finite element computing needs to partition the mesh are based on two principles: ensuring the same node number in each subdomain and minimizing the node number on the interface between subdomains. The purpose of the two principles is to ensure the load balancing of each processor and the minimal communication volume between processors. Mesh partitioning by means of graph partitioning first requires using the graph to simulate the computation structure of the mesh, that is, converting the mesh to the graph. Each node of the mesh can be programmed as the vertex in the graph, with the connecting line between nodes as the edge of the graph. The graph obtained in this way is called nodal graph. Another practice is that each mesh cell can be processed to correspond to a vertex of the graph so that when two cells share a face or an edge, the side exists between the corresponding vertices. This graph is called dual graph. These two graphs correspond to the partitioning based on mesh node and mesh cell, respectively. Figure 2 shows the two graphs.

In the past two decades, graph partitioning methods have witnessed rapid development, giving rise to many graph partitioning software packages, such as Metis, ParMETIS, Chaco,

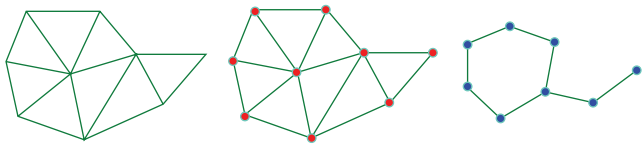

Figure 2: Graph for the mesh.

Jostle, Party, and Scotch. Besides, ParMETIS and Jostle have also actualized graph partitioning. Among these software packages, the most representative ones are METIS and its parallel version ParMETIS. METIS is a procedure set that can actualize serial graph partitioning, finite element mesh partitioning, and calculate the fill-reducing ordering of sparse matrix. The actualized algorithms include multilevel recursive bisection, multilevel K-path partitioning algorithm, and multiconstraint algorithm. METIS has offered the function to achieve corresponding purposes. Moreover, according to the number of partitions, different functions can be selected to speed up the partitioning. It has also provided the function for the conversion of mesh to the function of nodal graph and dual graph together with the corresponding function and procedure for graph partitioning. As a parallel library based on MPI, ParMetis can actualize nonstructural graph and mesh partitioning and implement various algorithms for the fill-reducing ordering of sparse matrix. Besides extending the functions provided by Metis, ParMetis also contains the programs suitable for AMR (adaptive mesh refinement) computing and large-scale numerical simulation, so it can actualize graph partitioning, mesh partitioning, graph repartitioning, partition encryption, and matrix reordering. The algorithm actualized in ParMetis is based on parallel multilevel K-path partitioning method, self-adaptive repartitioning algorithm, and parallel multiconstraint algorithm.

4. Mesh Partitioning Based on Metis and ParMetis

4.1. Data Structure for Metis and ParMetis. Metis has provided some programs, including partnmesh and partdmesh, the programs of partitioning the mesh by converting the mesh to nodal graph and dual graph. By calling the two programs in accordance with the corresponding data entry format, mesh partitioning based on nodes and cells can be carried out. The data file format is shown in Figure 3(a). The first line represents the total number and type of the cells. Each line following it represents the node number making up each cell. Metis supports four cell types, namely, triangular, tetrahedral, hexahedral, and quadrangular cells, which are respectively denoted by 1, 2, 3, and 4. Meanwhile, programming can be carried out on the function provided by Metis to store the data in the CSR format (compressed storage format).

ParMetis is mainly a function library. In order to actualize graph partitioning, the user needs to program and call ParMetis function. The required data structure for the graph is DCSR (distributed compressed storage format). As for the graph shown in Figure 5, the data format on the three processors is as follows in Figure 4.

Later, ParMETIS_V3_PartKway function can be called to perform graph partitioning. If the information about the vertex coordinates of the graph is available, the parallel graph partitioning can be carried out with ParMETIS_V3_PartGeomKway and ParMETIS_V3_PartGeom function. As they both use the coordinate-based multiconstraint K-path partitioning algorithm and space-filling curve, their computing speed is faster than that of ParMETIS_V3_PartKway, but the partitioning quality of ParMETIS_V3_PartGeom is much poorer. ParMetis provides the function ParMETIS_V3_PartMeshKway for direct parallel partitioning of the mesh. This function quickly converts the mesh to dual graph. Then, it calls ParMETIS_V3_PartKway function and ParMETIS_V3_PartGeomKway function to perform parallel partitioning. Since the data structure for ParMETIS_V3_PartMeshKway is distributed, the distributed storage of the mesh cell information is necessary, but without any need for information of node coordinates. Besides, ParMetis has also provided the function for parallel subdivision as well as the parallel function for matrix ordering. The latter is of great practical significance in directly solving the sparse linear equations.

4.2. Actualization and Test of Partitioning. Targeting the function provided by ParMetis, the author wrote the interface program for ParMETIS_V3_PartMeshKway and converted the mesh to graph after parallel reading of the data file. Then parallel partitioning is finally actualized. Take a finite element mesh as an example. For a mesh with a cell number of 810,000 and a node number of 836,381, Metis function and ParMetis function are respectively called to partition the mesh. Data show that the partitioning results of Metis and ParMetis are consistent, as displayed in Table 1 (ParMetis results are calculated with 2CPU).

Metis (including ParMetis) uses two types of objective function. One has the minimal edgecut, while the other has the minimal ncommunications volume, which is represented by the sum of the numbers of domains to which the boundary points belong or the total number of times of data transmission. For the partitioning of finite element mesh, the two objective functions produce generally consistent results. The above test was controlled by the objective of minimizing the cut edge. Table 1 shows the minimal cut edge number. In the dual graph, the edgecut, when reflected in the finite element mesh, is equivalent to the common edge or common face between cells. Table 2 shows the comparison of the partitioning time of Metis and ParMetis. It can be seen from Table 2 that the partitioning time for over 800,000 cells is within seconds, which is insignificant compared with one-step elastic finite element computing time which is about 350 seconds. In addition, the increase in the number of subdomains does not have a great impact on partitioning time. Meanwhile, a great amount of "I/O" time in the domain partitioning is consumed. The time for ParMetis to convert the mesh to the dual graph (Mesh 2 Dual) accounts for over

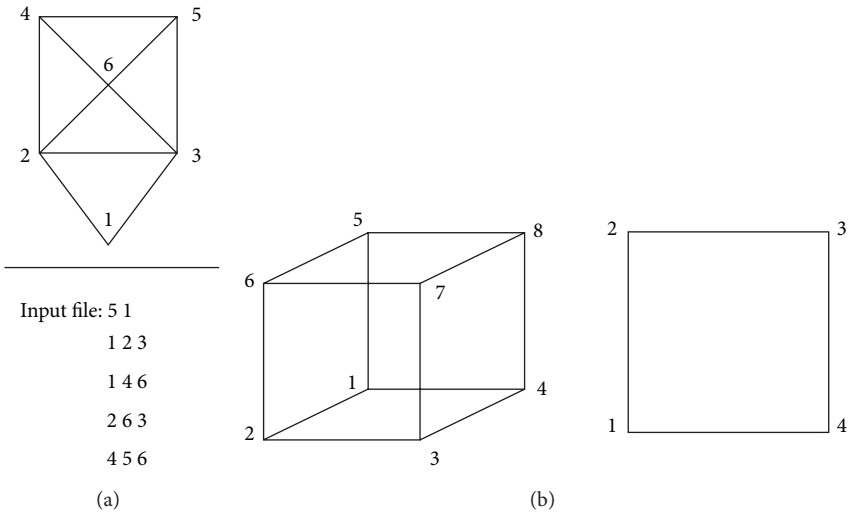

Figure 3: Data file and node numbering format.

Table 1: The results for multilevel recursive bisection algorithm.

	Subdomain 1	Subdomain 2	Subdomain 3	Subdomain 4	Cut edge number
Nodal graph	209095	209095	209095	209096	22002
Dual graph	202500	202500	202500	202500	21242

Processor 0 :
xadj 0 2 5 8 11 13
adjncy 1 5 0 2 6 1 3 7 2 4 8 3 9
vtxdist 0 5 10 15

Processor 1 :
xadj 0 3 7 11 15 18
adjncy 0 6 10 1 5 7 11 2 6 8 12 3 7 9 13 4 8 14
vtxdist 0 5 10 15

Processor 2 :
xadj 0 2 5 8 11 13
adjncy 5 11 6 10 12 7 11 13 8 12 14 9 13
vtxdist 0 5 10 15

Figure 4: Data format on three processors.

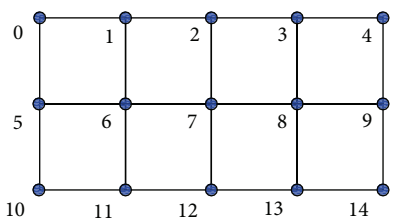

Figure 5: A simple graph.

one half of the total time for the partitioning process, while the time for the same conversion accomplished by Metis is already included as part of the "partitioning" time.

It can be seen from Table 2 that the total computing time that ParMetis spends on the 256 partitions with 2 CPUs displays obvious advantages over Metis, the efficiency reaching 76.5%. As a matter of fact, the parallel partitioning does show obvious time advantages for small-and medium-sized graphs, especially the static graphs. This is because the time for graph partitioning itself is very insignificant in the entire computing process, especially when the internal memory of a signal processor can meet the partitioning needs. For small-and medium-sized problems, parallel graph partitioning is not so necessary considering the time consumption, but for multiphysical, multistage, and multimesh simulations, where the data size is huge, parallel partitioning will display superiority. Parallel graph partitioning is quite effective in solving the problems of inadequate internal memory of a single processor or the I/O bottlenecks caused by repartitioning.

Figure 6 shows the comparison of the total computing time that ParMetis spends on partitioning the above-mentioned 8100 million mesh cells to 256 domains with different number of CPUs. According to the figure, parallel computing still has obvious advantages, but the efficiency is not so high, mainly because the mesh size is small, the total computing time is not long, and the communication time of parallel computing accounts for a large share. Figure 7 is a diagram for parallel partitioning of a arch dam and foundation system.

5. Conclusion

In this paper, the authors discuss the way to actualize partitioning methods and mesh partitioning and introduce

Table 2: The comparison of the partitioning time of Metis and ParMetis (second).

Partition number		4	16	64	256
Metis	I/O	2.350	2.350	2.400	2.370
	Partition	3.730	3.750	4.130	4.700
	Total time	**6.080**	**6.100**	**6.530**	**7.070**
ParMetis (2CPU)	Mesh2Dual	2.559	2.568	2.567	2.571
	Partition	1.533	1.663	1.81	2.047
	Total time	**4.092**	**4.231**	**4.385**	**4.618**

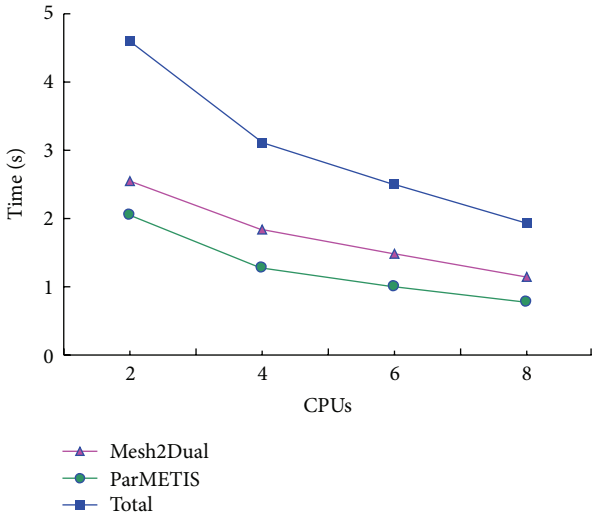

Figure 6: Time of 256 partitionings on different CPUs.

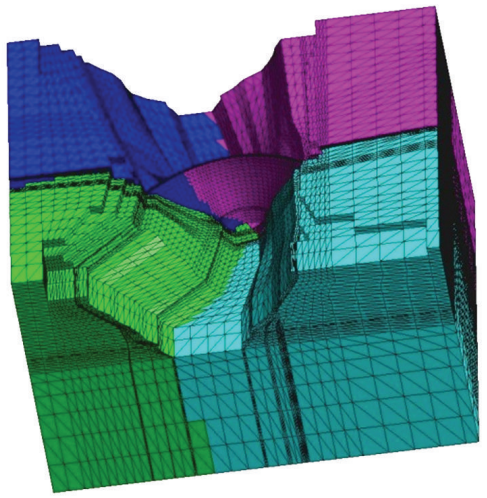

Figure 7: Diagram for parallel partitioning.

relevant softwares for parallel partitioning algorithm. The data structure and key functions of Metis and ParMetis are analyzed. The partitioning program is compiled based on these key functions. Moreover, detailed test on typical mesh partitioning is conducted. The results show that ideal partitioning results will be yielded by actualizing domain decomposition by the programming based on Metis and ParMetis. Parallel partitioning is mainly for the simulation of large-computing scale, large-data size, and multiphysical, multistage, and multimesh simulations, while for static finite element method of general size, serial partitioning is enough.

Conflict of Interests

The authors do not have any conflict of interests with the content of the paper.

Acknowledgments

This work was financially supported by the National Natural Science Foundation of China (51209235), the National Basic Research Program of China (973 Program) (2013CB035904, 2013CB036406), the National "Twelfth Five-Year" Plan for Science & Technology Support (SQ2013BAJY4138B02), the Governmental Public Industry Research Special Funds for Projects of MWR (201201050), and Special Research Funds of IWHR (1361/1353/1169/1309).

References

[1] Z. Lei, *High Performance Computing Method and Its Application in Hydraulic Structure Analysis*, Hohai University, Nanjing, China, 2008.

[2] M. J. Berger and S. H. Bokhari, "A partitioning strategy for nonuniform problems on multiprocessors," *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 570–580, 1987.

[3] M. T. Heath and P. Raghavan, "A Cartesian parallel nested dissection algorithm," *SIAM Journal on Matrix Analysis and Applications*, vol. 16, no. 1, pp. 235–253, 1995.

[4] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, 1970.

[5] C. Walshaw and M. Cross, "Mesh partitioning: a multilevel balancing and refinement algorithm," *SIAM Journal on Scientific Computing*, vol. 22, no. 1, pp. 63–80, 2000.

[6] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[7] G. Karypis and V. Kumar, "Multilevelk-way partitioning scheme for irregular graphs," *Journal of Parallel and Distributed Computing*, vol. 48, no. 1, pp. 96–129, 1998.

[8] G. Karypis and K. Schloegel, *PARMETIS: Parallel Graph Partitioning and Sparse Matrix Ordering Library*, Version 4.0, University of Minnesota, Minneapolis, Minn, USA, 2013.

[9] G. Karypis and V. Kumar, "A parallel algorithm for multilevel graph partitioning and sparse matrix ordering," *Journal of Parallel and Distributed Computing*, vol. 48, no. 1, pp. 71–95, 1998.

[10] C. Walshaw and M. Cross, "JOSTLE: parallel multilevel graph-partitioning software—an overview," in *Mesh Partitioning Techniques and Domain Decomposition Techniques*, pp. 27–58, Civil-Comp Press, 2007.

[11] T. F. Kong, *Multilevel Spectral Clustering: Graph Partitions and Image Segmentation*, Massachusetts Institute of Technology, 2008.

[12] L. Grady and E. L. Schwartz, "Isoperimetric partitioning: a new algorithm for graph partitioning," *SIAM Journal on Scientific Computing*, vol. 27, no. 6, pp. 1844–1866, 2006.

[13] D. A. Tolliver and G. L. Miller, "Graph partitioning by spectral rounding: applications in image segmentation and clustering," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, pp. 1053–1060, June 2006.

[14] J. J. Dongarra, I. Foster, G. Fox, K. Kennedy, and L. Torczon, *Sourcebook of Parallel Computing*, Morgan Kaufmann Publishers, 2003.

[15] D. LaSalle and G. Karypis, "Multi-threaded graph partitioning," in *Proceedings of the 27th IEEE International Parallel & Distributed Processing Symposium*, 2013.