Journal for Geometry and Graphics
Volume 1 (1997), No. 1, 23–29.

Supporting CAE Parallel Computations with IE-Graph Solid Representation

M. Flasiński[1], **R. Schaefer**[1,2], **W. Toporkiewicz**[3]

[1]*Institute of Computer Science, Jagiellonian University, ul. Nawojki 11, PL 30-072 Cracow, Poland*

[2]*Computer Center, Cracow University of Technology,*
[3]*PhD student, Cracow University of Technology,*
ul. Warszawska 24, PL 31-155 Cracow, Poland

Abstract. An algorithm providing a decomposition of the lumped structure body into arbitrary parts having minimal interface has been introduced. The presented approach may be applied in parallel domain decomposition methods for mechanical analysis. Optimal partitioning is provided before the computational mesh is generated on the basis of its prescribed node density distribution. A new unambiguous solid representation is utilized.

1 Motivation

Because of technical and commercial limitations one processor-one memory computer is hardly applicable for solving complicated engineering problems. In order to gain a better performance computer architectures evolve towards multiprocessors systems with shared memory or arrays of computers linked with a fast network. In order to use a heterogeneous distributed environment of a computer network flexibly, hardware development is followed by intensive evolution of basic software (e.g. PVM, Linda, EMERALD). The successful utilization of multiprocessor computer installations forces us to parallelize most complicated problems that appear in the discrete mechanical analysis of structures.

One of progressive technologies in this area is PCD-SBS (Preconditioned Conjugate Gradient tailored with Subdomain By Subdomain preconditioner) domain decomposition method, which may be used to solve huge FE-equations issuing from linear or stepwise linearized problems (see [3]). The general idea of the PCD-SBS method is comprehended in partitioning of the whole FE-mesh into several connected parts. Degrees of freedom are numbered along the interfaces and then consecutively inside interiors of particular subdomains. The resulting stiffness matrix has the arrow-form (see Fig. 1). Upper bound coefficients are related to degrees of freedom laying on interfaces, and diagonal blocks to coefficients that come from degrees of freedom located inside interiors of subdomains. The PCG process contains two types of operations:

ISSN 1433-8157/$ 2.50

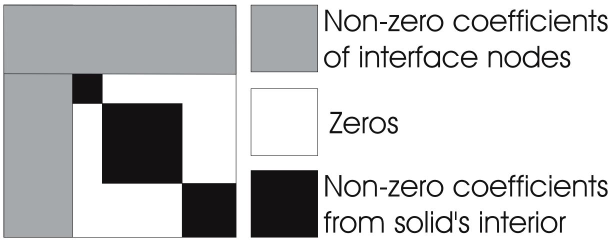

Figure 1: The arrow-form stiffness matrix resulting from domain decomposition

- a computation of inverses of diagonal blocks, which is performed once,
- a sequence of products of above inverses and parts of upper stripe and a specific preconditioner, which are provided parallely in each iteration step.

A rough analysis of PCD-SBS algorithm leads us to the conclusion that complexity of an elementary task is proportional to the square, or to the cube of a diagonal block's dimension, and that the number of iterations grows when partitioning is more fine and the upper bound becomes broader. Taking into account the above discussion, the problem of the optimal domain partitioning may be formulated as follows:

Decompose the structure's domain into subdomains containing prerequisited number of degrees of freedom, minimizing simultaneously the number of degrees of freedom on interfaces.

During the entire process of parallel computations the stage of a domain decomposition is one the of critical points. The main obstacle now seems to be lack of algorithms that would allow effective partitioning of a domain into subparts, before the FE-computation mesh is generated. It is partially due to the model used for representing a structure. If such representation does not provide enough topological information, partitioning becomes very complicated. So the choice of representation is crucial for constructing effective decomposing algorithm. We would like to introduce a solid representation which may solve the problem.

2 IE-Graph Solid Representation

The IE-graph representation scheme is a one-to-one mapping between a specific subset of lumped structures and a subset of indexed, edge-unambiguous graphs (see [2]). The scheme under consideration is based on the new solid taxonomy which describes each structure as the finite union of so-called simple solids (SS). Simple solids can be obtained from the arbitrary given set of basic constructive solids (BCSs) by a sequence of subtracting operations called features.

An elementary (simple) feature is created by sweeping a prescribed contour of the sweeping bases (SB) set. Both BCSs and SB elements are represented as graph nodes. Sweeping parameters (e.g. depth, SB position and size with respect to the BCS initial face, number of the initial face) constitute the labeled graph edges.

The solid representation graph provides very important topological information about relationships between features carved inside a BCS, as well as about relationships between simple solids that may be glued together creating a complex solid. This information is contained in edge labels. From them we learn about one wall adjacency of features, their catenation along edges or interaction when their volumes intersect.

The labels indicate which walls of a BCS are affected by the particular feature. An example of a solid and its representation graph is shown in the Figures 2a and 2b respectively.

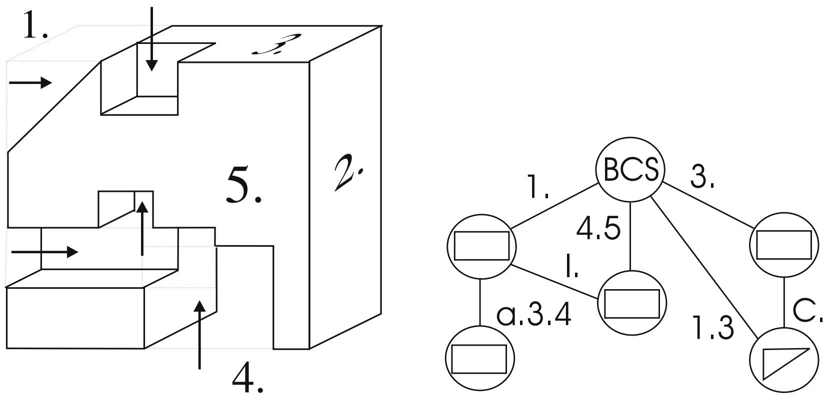

Figure 2: Example of a solid and its representation graph

Graph nodes are labeled either with BCSs or with features. The label 'S.2.1' means that two volumes are glued together with their first and second wall respectively. Labels that consist only of numbers indicate which walls are affected by the particular feature. Label 'I' indicates interaction and 'C' catenation of features. Labels like 'a.3.4' inform that two features are adjacent with their third and fourth walls.

Information provided by node and edge labels is of topological nature. The particular geometry of BCSs and features is stored in graph attributes (see [2]).

Although this representation has been invented to describe mechanical parts, it is of a descriptive power that allows one to represent structures. Another advantage is its uniqueness. Thanks to this, we may be sure that the partitioning selected in effect of the optimization process is really optimal.

3 Formulation of the optimization problem

3.1 Preliminary assumptions

For the sake of simplicity let us assume that the only BCS is a prism, features are also prismatic, and normal to walls. Expanding these assumptions does not provide much difficulty but forces us to consider many particular cases that make understanding of the general idea harder. We have a complex solid (CS) that consists of several simple solids (SS). In terms of a graph representation it means, that there are several graph nodes labeled with BCS-indicators and linked with edges indicating catenation of solids.

Solving an engineering problem of mechanical analysis of structure generally consists of two phases:

- a generation of the computational mesh,
- a calculation of the solution in the mesh points.

We suggest to provide the partitioning before the mesh is generated basing on some synthetic information about the node distribution. This assumption frees us from multiple mesh generation during the optimal partitioning creation. We assume in particular that the degrees of freedom density function, that indicates how many mesh nodes should be generated in the particular place of the volume is given.

Both of above solving stages are comparably expensive in terms of required computer performance. So we can additionally gain considerable time advantage paralleling also mesh generation.

3.2 The set of admissible controls

Let A indicate the set of admissible partitioning of a complex solid Ω. In our situation these are all such partitionings where volumes constituting Ω are cut along planes. Let $P_t \in A$ be an admissible partitioning. Then

$$P_t = \{\Omega_i\},\ i = 1 \ldots n, \quad \bigcup_{i=1}^{n} \Omega_i = \Omega, \quad \int_{\partial\Omega_i \cap \partial\Omega_j} dx = 0 \text{ for } i, j = 1, \ldots, n,$$

and Ω_i are closed subdomains of Ω, and each consists of one or more simple solids.

Let $\rho \in C(\Omega)$ be the function which describes the spatial density of FE degrees of freedom (see [1]) related to nodes of FE mesh that will be generated.

3.3 Decomposition indicators

There are two factors according to which we perform a decomposition. The first one is the number of degrees of freedom on the interface of catenated volumes that we call *interface capacity* I_t :

$$I_t = \sum_{i,j=1}^{n} \int_{\partial\Omega_i \cap \partial\Omega_j} dx, \quad \Omega_i, \Omega_j \in P_t\,.$$

The index t indicates that the interface capacity refers to a particular partitioning. The other factor is the *volume capacity* V_t :

$$V_i^t = \int_{\Omega_i} \rho dx, \quad \Omega_i \in P_t$$

It expresses the number of degrees of freedom in the interior of Ω_i, where $\Omega_i \in P_t$.

3.4 The goal of optimization

Let $\{\widehat{V}_1, \ldots, \widehat{V}_n\}$ constitute the volume capacity decomposition that is best suited to the current performance distribution. We intend to find a partitioning $P_t \in A$ that generates $\{V_1^t, \ldots, V_n^t\}$ sufficiently close to the prescribed $\{\widehat{V}_1, \ldots \widehat{V}_n\}$ and that offers possibly small interface capacity I_t.

3.5 Discussion of the problem formulation

The choice of the set of admissible partitionings was dictated by the requirement of simplicity. Decomposition itself can not put much overhead on the entire computation and in case of planes parallel to walls, it is easy to detect whether the particular feature falls entirely into one part of the divided volume or is intersected, etc. However, in our opinion this solution is acceptable.

We assume that load balancing is the leading factor of a decomposition. It is good, if a minimal interface accords with load balancing but it may happen that this second requirement forces us to choose a dividing plane with a nonoptimal interface capacity.

It must be mentioned here that the decomposition problem formulated this way is in a general case NP-complete, because it requires checking all possible sequences of decomposition. For instance, if we have to decompose a volume into three, mutually unequal parts, we do not know whether to start from subtracting the biggest or the smallest one, etc. We must test all possible combinations. Fortunately there are approximating strategies available that allow to check only one sequence of partitioning and provide a suboptimal solution.

4 The suggested course of decomposition

The search for the optimal decomposition would start from checking, whether some of the simple solids constituting the CS fit into volume capacities specified in the load balancing requirements. Such solids could be, at the very beginning, assigned to particular computers. However, at most cases, partitioning of a SS is unavoidable. It is easy to notice, that the only nontrivial remaining problem is to divide a volume into two parts according to the given proportion. For instance, if we need to decompose a volume into three parts in proportion 1:1:2, we may, at first, divide it into two equal parts, and then repeat this operation for one of them. All further considerations refer to dividing a volume into parts according to the given proportion.

Search for the optimal plane dividing a volume would be carried along three mutually normal directions. It is necessary to test especially these planes that include at least one feature wall – either the begin or the end of a feature along the given direction, because along the whole length of a feature the loss of the interface area that it introduces is constant. Moreover, such planes offer a very good proportion of interface capacity to the subtracted volume capacity what makes us consider them as possibly optimal surfaces of division. On each of the three directions we would choose the plane that provides the minimal interface capacity and volume capacities of divided parts according to the load balancing requirements. Then from these three results the best one would be elected.

5 Possible optimization strategies

There are many optimization strategies that deal with dividing a volume. Let us introduce two of them. They differ in the functional to be minimized and in the way that load balancing requirements are involved. Because of previously motivated reasons, formulation of strategies is restricted to the case of partitioning a SS into two subparts subtracting one volume. The functional I indicates now the interface capacity of the dividing plane, and V stands for volume capacity of the subtracted part.

$$\textit{Strict strategy:} \quad \min_{V_{min} \leq V \leq V_{max}} I(V)$$

where V_{min} and V_{max} express load balancing requirements. In this strategy the constraints with regard to the load balancing are strict; the volume capacity V of the subtracted solid must fit exactly into the given interval.

$$\textit{Flexible strategy:} \quad \min_{0 \leq V \leq V_W} a_1 I^2(V) + a_2 (V - V_D)^2$$

where V_D is the desired capacity of a subtracted volume, V_W is the node capacity of the entire volume and a_1, a_2 are tuning factors, that say whether we insist more on minimizing the interface or on balancing the load. The penalty function $a_2(V - V_D)^2$ grows with the difference between the subtracted volume's capacity and the desired capacity.

The strict strategy fits better for environments where computers differ much in their performance. Then it is important not to overload weaker machines. The flexible strategy will demonstrate its efficiency in a well balanced environment where computers have a considerable 'safety margin' of performance.

6 Computational solutions

The search for the optimal decomposition would start from checking whether some of the simple solids constituting the CS fit into volume capacities specified in the load balancing requirements. Such solids could be, at the very beginning, assigned to particular computers. But in most of the cases, partitioning of a SS is unavoidable. For the purpose of volume partitioning, it is convenient to extract from the Solid Representation Graph a list of positive and negative volumes, i.e. BCSs and features, that constitute the Complex Solid. In this list three sequences of volumes would be built, according to the coefficients of their characteristic points along the X-, Y and Z-direction.

This approach may decrease the number of tests which are necessary in order to determine which features fall entirely into divided parts of the particular BCS, and which are intersected. For instance, when checking along the X-axis encounters a feature with the minimal X-coefficient bigger than that of the division plane, then without further tests it is known that all subsequent features in the list belong entirely to the other half of the divided BCS. It turns out that volume partitioning is in fact reduced to a variation of a sorting algorithm, which offers a very acceptable computational complexity.

The optimization of this division is a harder task, because it requires the integration of the node density function. To do this, some variation of the Monte Carlo method seems suitable, because it does not require the computation of the volume boundary. It allows to perform integration knowing only the geometry of the BCS and features, without taking care for their intersections, adjacency or catenation. This shifts the reconstruction of the solid boundary to the end of the entire decomposition process. Reconstruction of the solid boundary is performed by the host which accepts the particular part of the CS for calculations.

The search for the optimal plane dividing a volume would be carried along three mutually normal directions. The suggested approach is based on bisection. At the first step the volume capacity of the entire SS would be calculated. In case it does not meet the load balancing requirements, the testing division plane would be placed at the half of its length. This process would iterate on volume parts until a satisfactory outcome is obtained. As stop condition for this process may serve some combination of difference of minimized functional value and step length in subsequent iterations. In this way – along each of the three mutually normal directions – the plane would be figured out that provides the minimal value of the optimized functional. After the decomposition is made, the remaining task is the reconstruction of the SRG (Solid Representation Graph) structure. In general, it consists in relinking graph edges, and/or changing their labels. It is not complicated but requires the consideration of several special cases.

7 General conclusions

- While using the parallel approach in engineering computations, an effective decomposition of the problem's domain is crucial. Due to the IE-graph representation, a relatively simple algorithm provides a suboptimal solution (optimal in the restricted class of partitionings) of this problem. We would like to emphasize the great advantages of the IE-graph representation like descriptive power and uniqueness of volume's description.
- An approximation strategy was introduced. As it was mentioned, finding the optimal decomposition is an NP-complete problem. What we suggest is a sequential decomposition of a volume and an optimization of interface capacity at each step, but for simpler

cases checking all possible decomposition sequences may prove acceptable.

- The strategy of a decomposition should be adjusted to a distribution of computer performance over a network. Two strategies (strict and flexible) were formulated that fit to variously balanced environments.
- Only a scheme of the algorithm was presented. There are still many technical problems to be solved, like an effective computation of volume and interface capacity functions, for instance.

References

[1] G. Ciarlet: *The Finite Element Method for Elliptic Problems.* North Holland 1978.

[2] M. Flasinski: *Use of graph grammars for the description of mechanical parts.* Computer-Aided Design **27**, No. 6, 403-433 (1995).

[3] M. Papadrakakis, S. Bitzarakis: *Domain Decomposition PCG methods for serial and parallel processing.* Computing Systems in Engineering 1995.

Received November 26, 1996, final form June 12, 1997