RAL Report
RAL-94-092

# Partitioning Methods for Unstructured Finite Element Meshes

C Greenough and R F Fowler

August 1994

***Rutherford Appleton Laboratory*** Chilton DIDCOT Oxfordshire OX11 0QX

# Partitioning Methods for Unstructured Finite Element Meshes

C. Greenough and R.F. Fowler

1 March 1994


## Abstract

During the last few years there has been a rapid development of computer systems that use parallelism to achieve high computational performance. This has lead to a growth in research in the area of parallel computational methods. As part of this research, as in the field SIMD research, *old* algorithms have been revisited to assess their inherent parallelism. In the field of MIMD architectures the general belief held is that *divide and conquer* algorithms are most suitable for these types of machines. This has re-opened research into domain decomposition methods.

In this paper we review and compare a range of basic methods which can be used to partition finite element meshes. These have been implemented within a software tool called *ralpar* and tests made on a small set of 3D meshes. The comparisons are made in terms of the basic attributes of the partitions such as number of neighbours, number of interface nodes and number of cut edges in the underlying graph.

We also describe some of the extensions to these methods we have made and compare these with existing techniques together with some ideas of partition assessment based on software and architectural models.


---

Mathematical Software Group
Computational Modelling Division
Rutherford Appleton Laboratory
Chilton, Didcot
Oxfordshire OX11 0QX

# Contents

# 1 Introduction

The use of computing systems with some type of parallel architecture has grown significantly over the past few years. These systems are seen as the path by which sufficient computing power can be provided for accurate 3D simulations of complex phenomenon in fluid dynamics, stress analysis and electromagnetism. Such simulations typically use meshes or grids containing $10^5$ to $10^6$ nodes and may employ automatic grid refinement during the solution.

Many of the parallel machines being used are based on the MIMD form of parallelism where the memory of the machine is distributed over a network of processors. A consequence of this is that the program and its associated data must be distributed between these processors. In finite volume and finite element methods this leads to the problem of how to distribute large unstructured grids and meshes initially, and how to redistribute them subsequently, if refinement is made.

In section 2 we briefly discuss domain decomposition methods and related techniques for parallelisation of unstructured grid calculations. We then describe the set of partitioning methods that have been studied in this work. Ways of measuring the quality of the partitions are then considered, and finally we present and discuss some comparative results on 3D meshes.

# 2 Domain Decomposition Methods

The partitioning of finite volume and finite element meshes is fundamental to the use of computing systems with parallel architectures. This is true whether the numerical algorithm being used is explicit or implicit. At some point in the solution there will be a need to transfer data between processors. The goal of partitioning methods is to reduce to a minimum the communication cost of this information transfer whilst ensuring that the computational load for each processor is, as near as possible, equal. In many problems the load balance can be simply related to an equi-distribution of the number of grid points or finite elements between the processors and the communication cost to the number of node points on the subdomain interfaces.

The most practical method of formalising the division of a grid or mesh into a number of subdomains is by use of the connectivity graph associated with the mesh. Each vertex of the graph is associated with an element or volume and the edges of the graph depict the connectivity.

There is considerable interest in the mesh partitioning problem as can be seen in recent publications, e.g. [2], [10], [3]. Hendrickson [11] and Barnard [16] review the current situation with multi-level graph partitioning methods and demonstrate some of the potential of the approach.

One deficiency in the literature is the lack of references to mesh re-distribution on parallel systems. This will be important in the case of automatic mesh adaption. A recent publication which addresses this problem is a report by Diniz [6] on a distributed graph partitioning algorithm, though this was not available at the time of writing. Khan and Topping [18] have considered the problem using Genetic algorithms and neural networks and they present some interesting results for two dimensional meshes.

The basic objectives of Domain Decomposition algorithms (DD) are to:

- decompose the original problem into smaller subdomains,
- solve the original problem on the subdomains, and
- to somehow *patch* the subdomain solutions together to form the solution to the original problem.

In general the above must be repeated iteratively until some convergence criterion is satisfied.

The alternative approach, which is usually differentiated from true domain decomposition, is to retain the algorithm used in the serial solution, but to implement it in parallel. This still requires a partitioning of the mesh that assigns equal amounts of data to each processor and minimises communication costs of the interfaces.

Originally DD methods were used to enable the solution of stress analysis problems too large for computer memories. This was often termed *substructuring*. Many different algorithms have been developed over the last few years and indeed there is an annual conference on DD methods [4].

The methods mainly differ in the following ways:

- the partitioning of the domain (with or without overlap regions)
- the way in which they solve the subdomain problems (exactly or inexactly)
- the way in which they construct the problem for the interfaces (from the PDE or algebraically from the matrix)

Not all these approaches are suitable for all problems. The method must in general be matched to the problem. The two main approaches are characterised by the way the subdomains are constructed, namely, overlapping or non-overlapping. The main overlapping approach is based on the Schwarz alternating procedure [14] or variants of this.

The non-overlapping approach decomposes the domain into non-overlapping subdomains with lower dimensional interfaces. In a two-dimensional problem the interface will be one-dimensional. The reduced interface operator is usually not local and requires subdomain solutions. The most common approach is to form the Schur complement matrix of the interface.

In both cases a partitioning of the mesh is required that will minimise communication costs. In this paper we only consider non-overlapping partitions.

## 3 Mesh Partitioning Techniques

A large number of methods of mesh partitioning have been developed. Most rely on the topology of the finite element mesh either directly or through processing the connectivity graph. A number of the techniques are:

- Coordinate bisection (Fox, [9])
- Greedy (Farhat & Wilson, [7])
- Bandwidth minimisation (Malone, [13])
- Inertial bisection
- MINCUT (Kernighan & Lin, [12])
- Recursive Graph Bisection (Williams, [17])
- MINGRAPH (Hu and Blake, [3])
- Spectral bisection (Simon *el al*, [15])
- Simulated annealing (Baiardi *et al*, [1])
- Multi-level methods (Barnard & Simon, [16])

We outline some of these methods below. Full implementation details will be given in [20, 21].

### 3.1 Greedy methods

The algorithm of Farhat [7] is a "Greedy" method which gathers elements about a seed point for each domain. It is based on assigning a weight to each node equal to the number of elements attached to it. In a problem with $N$ elements and $p$ processor we require $N_p = N/p$ elements assigned to each processor. The algorithm proceeds as follows:

1. Form nodal weighting array
2. do $p = 1$ until $N_p$

    2.1 Locate node $n$ with next minimum available weight
    2.2 Label all elements attached to this node; they initiate the list of elements of the $p$th subdomain
    2.3 Recursively add to this list the elements adjacent to those already labeled, starting always from the first labeled element in the previous step.
    2.4 Remove used elements from list and update nodal weights.

This process is repeated until the complete mesh is partitioned.

Many variations of this method are possible. A Modified Farhat method has been proposed by Fowler *et al* [19]. This is based on the observation that at each cycle the algorithm selects a new minima as its next starting point. It is clear from the structure of finite element meshes that at each cycle of this algorithm there are going to be a number of nodes with the minimum connection weight. The Modified algorithm seeks to try a limited number of these alternative starting points at each cycle of the algorithm. The number of trials has to be limited because the total number of choices increases rapidly with the number of partitions.

### 3.2 Bandwidth minimisation

This method was proposed in [13] by Malone. An arbitrary finite element mesh is fully defined by specifying the element types, the coordinates of the nodes and a list of the nodes associated with each element. From the node association a connection matrix C is formed. C is defined by:

$$C_{ij} = 1 \tag{1}$$

if nodes $i$ and $j$ are connected, otherwise

$$C_{ij} = 0 \tag{2}$$

The half-bandwidth $m$ of the matrix C is defined by

$$m = \max_{C_{ij} \neq 0} \{|i - j|\} \tag{3}$$

The connection matrix C presents the sparsity distribution of the system matrices that will results in the finite element assembly.

The Malone method first requires a bandwidth minimisation on the nodal numbering. There are a number of well known algorithms to do this. Then the list of elements is sorted according to the lowest node number within each element. The partition into $p$ parts is than made be dividing this ordered list into the required number of sections.

A simple variation on this method is to use an algorithm for *profile* minimisation, rather than bandwidth minimisation. It is shown in [20] that in many cases this gives slightly superior results to bandwidth minimisation. Another possibility is to use the Malone method in a recursive bisection manner as described in the next section.

## 3.3 Recursive Bisection Methods

Recursive bisection methods seek to divide the connection graph in half at each step. This is usually easier than trying to do $p$-sections. The methods have two main advantages:

- each subproblem is easier than the general problem, and
- there is some natural parallelism in the method.

In many methods a scale quantity $s_e$ is associated with each graph vertex $e$, which is often called a *separator field.* By evaluating the median $S$ of $s_e$ the graph is split according to $s_e$ being greater or less than $S$. However, any partitioning method, including the Greedy and Bandwidth ones above, can be used recursively if desired. Strict recursive bisection is limited to $2^n$ partitions for integer $n$.

Four methods which have been implemented in this way are as follows:

### 3.3.1 Coordinate bisection

In the coordinate bisection method the $(x, y, z)$ coordinates of the centroid of an element are used to define the position of that element. The coordinates are then sorted into order in each coordinate direction. The bisection is made by dividing the graph into two about the median in one direction.

This process can then be repeated on the subgraphs to generate 4,8,16,... partitions. Typically the first cut will be made in the $x$ direction, followed by $y$, $z$, $x$ and so on. A variation of this, which we have investigated, is to attempt to cut the graph in each of the 3 directions at every bisection step. We then select the cut direction that gives the fewest interface nodes between domains. We refer to this as Cost-Geometric bisection.

The above type of geometric bisection uses the cartesian coordinate directions. However in general there is no reason for the mesh to "align" with these directions and the results are not invariant to rotation of the mesh. An alternative is to use the principal axes of inertia of the mesh, determined by assuming unit mass at each vertex. Again a number of variations are possible, such as making the cut on the axis of lowest inertia or trying all 3 axes to find the one with lowest interface node cost.

### 3.3.2 Graph bisection

The graph bisection algorithm tries to find the maximum diameter of the connectivity graph. This can be done by a few iterations of labeling levels of the graph about a seed point and then setting the new seed point to the last labeled vertex. The basic step of the graph partitioning then is then to divide the vertices of the graph according to their level numbers from the last seed point.

The results obtained are dependent on how the connectivity graph is defined. One possibility is to say that two elements are connected if they share a common face (or, in 2D, a common edge). In [3] this is called the edge communication graph. Another definition is that two elements are connected if they share a common node, and [3] refers to this as the *true* communication graph. Which is more appropriate may depend on the details of the computational method. We allow either definition to be used.

### 3.3.3 Spectral bisection

Below is a simple example of the Spectral Bisection method. A full explanation can be found in e.g. [2]. The mesh and graph are:

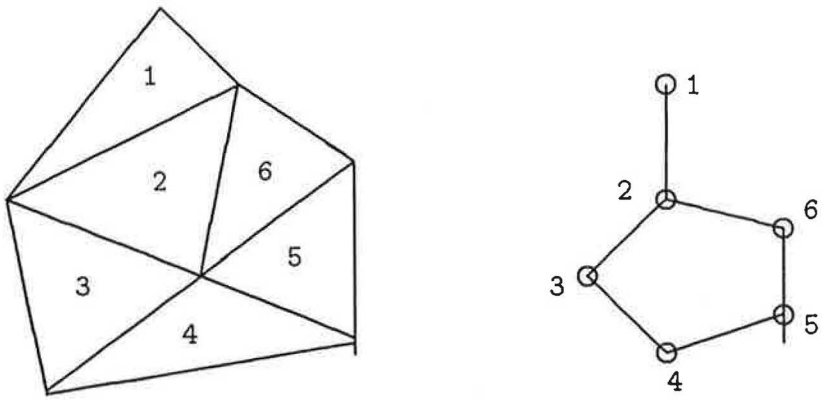


Figure 1: A Simple Finite Element Mesh with Associated Graph

The Laplacian of the connection graph $L(G)$ is given by:

$$l_{ij} = \begin{cases} -1 & \text{if } (v_i, v_j) \epsilon E \\ \deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

This simple mesh lead to the following Laplacian matrix:

$$L(G) = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 0 & 0 & -1 & 2 \end{bmatrix} \tag{5}$$

Starting from the requirement to minimise the number of cut edges in the graph bisection, it can be shown [2] that the eigenvector associated with the second smallest eigenvalue is a good candidate for use as a separator field.

The second eigenvalue and associated eigenvector in this case are:

$$\lambda_2 = 0.6972 \tag{6}$$

$$\mathbf{x}_2 = \begin{bmatrix} -6.606 & -2 & 1 & 3.303 & 3.303 & 1 \end{bmatrix} \tag{7}$$

On the basis of this we can partition graph the as:

$$P_1 = \{1, 2, 6\} \text{ and } P_2 = \{3, 4, 5\}$$

or

$$P_1 = \{1, 2, 3\} \text{ and } P_2 = \{4, 5, 6\}$$

Either choice is valid. For realistic problem sizes an efficient solution of the sparse eigenvalue problem is essential.

### 3.3.4 Kernighan and Lin methods

The MINCUT algorithm of Kernighan and Lin [12], can be used to improve an existing bisection of the graph. It does this by repeatedly swapping vertices between domains in an ordered manner, and then selecting the best result as the starting point for the next iteration. The function that this method tries to minimise is the number of cut edges in the graph, as in the spectral method. Vertices are selected for swapping according to their effect on the cost function, with those that decrease it most taken first. Moves that increase the cost function can be accepted, if a lower minimum occurs later in the iteration. Careful implementation of the method is necessary to prevent it becoming excessively expensive for large meshes.

The starting configuration may be totally random or some ordered partition. Hu and Blake [3] use the Graph Bisection method to generate a starting configuration (MINGRAPH). This and the random start method have been implemented along with a Greedy variation. This starts by assigning all vertices to one domain and then uses the ordered swapping technique of the MINCUT method to achieve balance. This works because swaps are always made from the partition with most vertices to the one with least. In this paper refer to this as RKLG (Recursive Kernighan and Lin with Greedy start) though MINGREEDY might be more appropriate.

## 3.4 Element and domain weighting

An important extension to the partitioning methods discussed above is the inclusion of element weighting. This can be required if the computational work associated of each element is not the same. This is the case when using a mixed element mesh as the amount computation will depend on the number of nodes in an element.

The user can specify the weight as a real value for each element, or in terms of the number of nodes in an element. Then each method is modified so that instead of trying to produce $p$ partitions of $N/p$ nodes each, it looks for partitions with weight as close as possible to:

$$W = \frac{\sum_i w_i}{N} \tag{8}$$

where $w_i$ is the weight of the element $i$.

Another extension that is becoming important for computations on heterogeneous workstation clusters is that of domain weighting. It may be that one machine is more powerful than the others and hence should have a larger share of the work. This can also be included in the above scheme.

# 4 Performance Assessment

There are a number of factors that can be used to assess the performance of these methods:

- The cost of the method in computational time.
- The cost of the method measured as a function of the effect of the partition on the complete solution process.
- To what degree can the method be parallelised?

The first of these is not difficult to measure. A complexity analysis coupled with timings from a number of trial meshes will allow some objective measures to be made. Care must be taken to ensure that the trial meshes contain a representative selection of geometries and connection patterns.

The second type of assessment is far more difficult since the methods used to parallelise the application program will depended on the computational methods being used to solve the physical problem. The parallelisation techniques will have their own requirements on the properties of the partitions. In this paper we present some results for the number of interface nodes that are generated by each method. This measure will be important if an algorithm that solves the interface problem is being used. We also look at the average number of neighbouring domains for each processor which is important if communication start up time is significant.

The final criterion is very important if methods involving adaptive refinement are going to be used in the application.

## 5 Some Test Examples

Three meshes have been used in testing and assessing the partitions produced by the different methods. These meshes were provided by Bertin & Cie within the IDENTIFY Esprit project.

Although the initial testing of the methods was performed using two-dimensional meshes, these are all three-dimensional meshes using a range of element types. The first, referred to as JET, is illustrated in Figure 2 and is a composed of 360 hexahedral elements with 548 nodes.

Figure 2: The JET test mesh. Only the surface mesh is shown.

The second mesh (COUDE), is composed of tetrahedral elements and prisms (6 noded elements), with 399 nodes and 902 elements. This is illustrated in Figure 3.

The final example, which is more representative of the sort of mesh used in real engineering applications, is that of an engine cylinder. This mesh has 23446 hexahedral elements and 26573 nodes. The mesh, referred to as TUBU, is shown in Figure 4. This mesh is still small compared to many current requirements, but does represent the expected level of complexity.

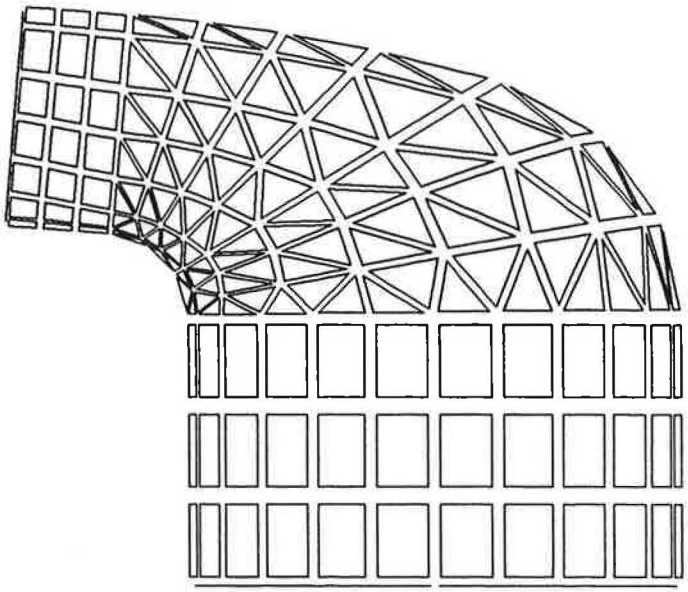

Figure 3: The COUDE mesh. A mixed mesh of tetrahedra and prisms.

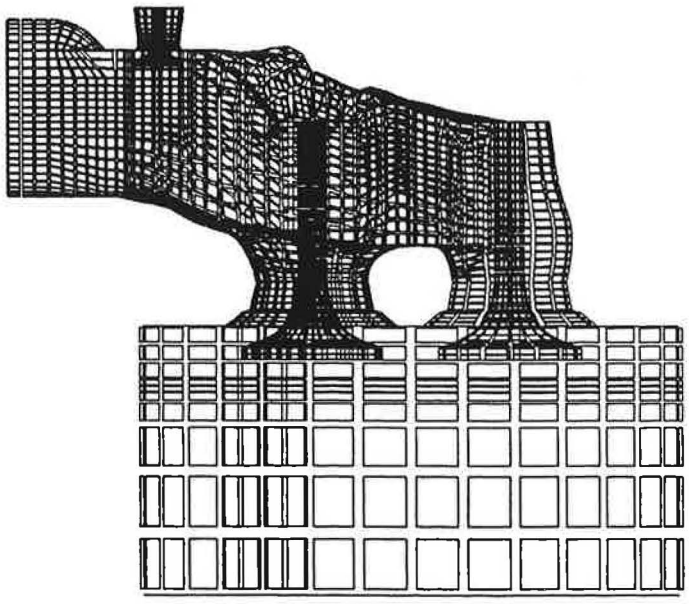

Figure 4: The TUBU engine cylinder mesh.

# 6 Partitioning Results

To compare the methods, each of the test meshes has been partitioned, using a selection of techniques, into $N = 2^n$ parts for $n$ from 2 to 8.

In addition to the methods which have implemented ourselves, we have looked at the software package CHACO [5], from Sandia National Laboratories, which implements the spectral bisection method of mesh partitioning.

At the present time we have only obtained results using spectral bisection on the TUBU mesh. The CHACO package allows the user a range of options and the results reported here refer to the multilevel version of the spectral method. In this approach, the mesh is "coarsened" by allowing certain neighbouring elements to merge. This reduces the size of the eigenvalue problem that has to be solved for bisection. After the bisection on the coarse mesh has been made the results are propagated up to the real mesh. Since the bisection on the fine mesh may now be less good than if the standard spectral method had been used, an extra refinement is made using the Kernighan and Lin algorithm [12].

Table 1 gives the results for the cost function for the JET mesh along with the average number of neighbour domains for each domain. Similar results are given in tables 2 and 3 for the COUDE and TUBU meshes. The cost value is the number of interface nodes generated.

| | Interface node cost of Method | | | | |
|---|---|---|---|---|---|
| Partitions | Farhat | Malone | Geo-bis. | Cost-Geo. | Inertia |
| 2 | 71 | 51 | 64 | 55 | 64 |
| 4 | 144 | 152 | 111 | 112 | 111 |
| 8 | 252 | 357 | 218 | 205 | 231 |
| 16 | 339 | 440 | 326 | 290 | 331 |
| 32 | 413 | 472 | 380 | 385 | 395 |
| 64 | 476 | 505 | 472 | 457 | 476 |
| 128 | 513 | 527 | 521 | 507 | 525 |
| 256 | 531 | 535 | 533 | 526 | 534 |
| | Average number of neighbours | | | | |
| Partitions | Farhat | Malone | Geo-bis. | Cost-Geo. | Inertia |
| 2 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 4 | 2.50 | 1.50 | 2.50 | 3.00 | 2.50 |
| 8 | 3.50 | 1.75 | 5.50 | 5.25 | 5.00 |
| 16 | 7.13 | 3.63 | 6.50 | 6.50 | 6.25 |
| 32 | 9.38 | 6.63 | 8.00 | 8.38 | 7.75 |
| 64 | 9.67 | 6.03 | 10.65 | 10.03 | 10.59 |
| 128 | 11.78 | 9.00 | 10.72 | 13.22 | 10.86 |
| 256 | 10.33 | 9.24 | 10.37 | 13.64 | 10.46 |

Table 1: Cost and neighbour results for partitioning the JET mesh.

The cost results for the TUBU mesh are illustrated in Figure 5. It can be seen that, in terms of the cost function, the spectral bisection and Kernighan and Lin (RKLG) methods give the best results. For the case of 2 or 4 partitions the spectral cost is about half that of the Farhat (greedy) and the cost-geometric bisection methods. For larger numbers of partitions the relative advantage of spectral

| | Interface node cost of Method | | | | |
|---|---|---|---|---|---|
| Partitions | Farhat | Malone | Geo-bis. | Cost-Geo. | Inertia |
| 2 | 36 | 49 | 56 | 50 | 66 |
| 4 | 106 | 122 | 144 | 94 | 114 |
| 8 | 168 | 235 | 190 | 149 | 195 |
| 16 | 227 | 299 | 256 | 216 | 270 |
| 32 | 289 | 345 | 301 | 285 | 315 |
| 64 | 330 | 365 | 350 | 329 | 353 |
| 128 | 360 | 374 | 375 | 358 | 377 |
| 256 | 382 | 385 | 388 | 382 | 387 |
| | Average number of neighbours | | | | |
| Partitions | Farhat | Malone | Geo-bis. | Cost-Geo. | Inertia |
| 2 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 4 | 2.50 | 1.50 | 3.00 | 3.00 | 2.50 |
| 8 | 4.75 | 3.00 | 5.50 | 5.25 | 5.25 |
| 16 | 7.38 | 5.13 | 7.00 | 6.50 | 7.25 |
| 32 | 9.06 | 9.19 | 8.94 | 8.38 | 8.63 |
| 64 | 10.53 | 12.47 | 10.06 | 10.03 | 10.09 |
| 128 | 11.13 | 12.63 | 10.28 | 13.22 | 11.16 |
| 256 | 9.98 | 11.15 | 9.98 | 13.64 | 10.13 |

Table 2: Cost and neighbour results for partitioning the COUDE mesh.

bisection decreases. With 512 partitions the spectral method gives a cost only 7% better than Farhat, though of course the domains are very small at this point.

In Figure 6 we show the average number of neighbouring domains for each method on the TUBU mesh, again as a function of the number of partitions.

It can be seen that the Malone method (profile minimisation) generally gives the lowest number of neighbouring domains, though at the cost of generating large interfaces due to its slicewise decomposition. The spectral bisection method is not as good as the Malone method in this respect, though better than many of the others. For larger numbers of partitions the RKLG method is better than the spectral method on this measure.

## 7 Conclusions

The best domain partitioning method will depend both on the mesh used and on the particular application the decomposition will be employed in. For the TUBU mesh, which is the test example most closely representing the expected meshes for real engineering applications, we have seen that the best results, in terms of our cost function, are given by the multilevel spectral bisection algorithm used in the CHACO package and the RKLG method. However, the simpler cost-geometric bisection and Farhat methods give results that are only slightly worse for larger numbers of partitions.

This comparison is valid if the parallel computation is dependent on the number of interface nodes that are generated, as is often the case. However, if the parallel machine used has a very high start up time for communications, or very expensive multi-hop communication, then minimising the number of neighbour domains may be an advantage. In this case the Malone method would be the one of

| Partitions | Interface node cost of method | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Farhat | Malone | Geo-bis | Cost-Geo | Inertia | RGraB | Spectral | RKLG |
| 2 | 660 | 699 | 1315 | 825 | 1102 | 741 | 377 | 474 |
| 4 | 2040 | 2156 | 3899 | 1879 | 3578 | 1931 | 983 | 1103 |
| 8 | 3610 | 4824 | 4890 | 3185 | 5035 | 3826 | 2202 | 2352 |
| 16 | 5279 | 9309 | 6688 | 4947 | 7197 | 5605 | 3660 | 3753 |
| 32 | 7760 | 18390 | 10645 | 7199 | 10961 | 7894 | 6103 | 5997 |
| 64 | 9990 | 23270 | 12244 | 9594 | 13399 | 10901 | 8741 | 9286 |
| 128 | 12885 | 24706 | 15557 | 12474 | 16641 | 13839 | 11582 | 11985 |
| 256 | 15984 | 25245 | 20169 | 15774 | 20723 | 16884 | 14686 | 14700 |
| 512 | 19096 | 25627 | 21909 | 19124 | 22800 | 19921 | 17906 | 17408 |

| Partitions | Average number of neighbours | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Farhat | Malone | Geo-bis. | Cost-Geo. | Inertia | RGraB | Spectral | RKLG |
| 2 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 4 | 3.00 | 1.50 | 2.50 | 2.00 | 2.50 | 2.00 | 2.00 | 3.00 |
| 8 | 3.75 | 1.75 | 4.75 | 3.25 | 5.50 | 3.00 | 3.00 | 3.50 |
| 16 | 5.63 | 1.88 | 6.00 | 5.38 | 6.63 | 4.25 | 4.50 | 4.50 |
| 32 | 8.13 | 2.56 | 7.81 | 6.44 | 8.50 | 6.31 | 5.81 | 5.38 |
| 64 | 9.25 | 4.38 | 9.31 | 8.34 | 9.72 | 8.47 | 7.75 | 6.78 |
| 128 | 11.19 | 7.34 | 10.16 | 9.81 | 10.81 | 10.73 | 9.08 | 8.20 |
| 256 | 12.43 | 10.36 | 12.29 | 11.00 | 13.50 | 12.31 | 10.73 | 9.98 |
| 512 | 13.21 | 12.13 | 13.51 | 12.32 | 14.31 | 13.52 | 11.83 | 11.93 |

Table 3: Cost and neighbour results for partitioning the TUBU mesh. RGarB is recursive graph bisection and RKLG is recursive Kernighan and Lin with a "greedy" start.

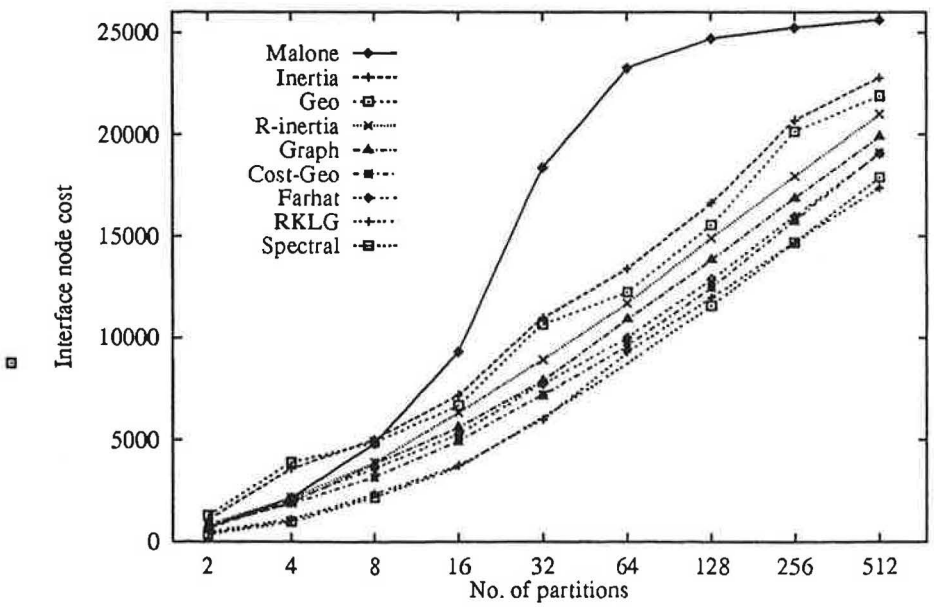


Figure 5: Cost as a function of number of partitions for the TUBU mesh.

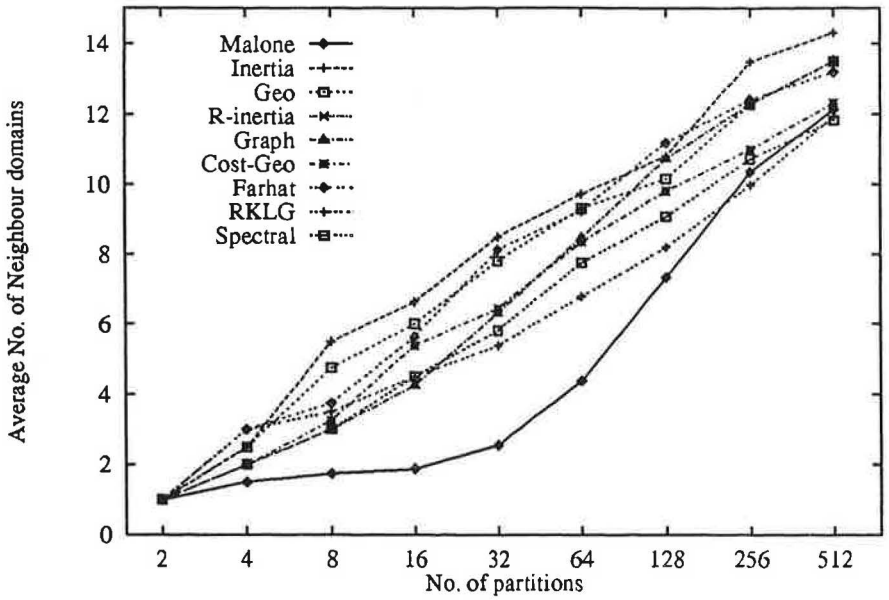


Figure 6: Average number of neighbouring domains as a function of number of partitions for the TUBU mesh.

choice.

Since in most engineering applications the linear algebra will dominate the solution time it is important that the effect of a particular partition on this element of the solution be assessed. The simple cost function implemented in our software goes some way towards this assessment but a more sophisticated cost function is required. It is necessary to consider both the algorithm and the parallel machine it is to be used on.

Although considerable work has been done on static partitioning methods there are still further areas to be explored. However, with meshes becoming larger and larger, the need is for dynamic data allocation methods. The parallel versions of some of the partitioning techniques outlined in this paper may be the starting point of such algorithms.

## Acknowledgments

This work was supported in part by the European Commission through the ESPRIT project IDENTIFY, number 6753. Test meshes were supplied by Bertin et Compagnie, France. We wish to thank Bruce Hendrickson and Robert Leland for making the CHACO package available to us.

## References


[1] F Baiardi and S Orlando: "Strategies for a massively parallel implementation of simulated annealing", *Springer-Verlag Lecture Notes in Comp. Sci.* 366,273 (1989).

[2] C Walshaw and M Berzins: "Dynamic load-balancing for PDE solvers using adaptive unstructured meshes", School of Computer Studies, University of Leeds, 1992

[3] YF Hu and RJ Blake: "Numerical Experiences with Partitioning Unstructured Meshes", Daresbury Laboratory Report, DL/SCI/P865T, March 1993

[4] T Chan (ed): "Proceedings of the 2nd International Symposium on Domain Decomposition Methods for PDEs", SIAM Publications, Philadelphia, (1988).

[5] B Hendrickson and R Leland: "The Chaco User's Guide, Version 1.0", SANDIA Report SAND93-2339, November 1993, SANDIA Laboratories.

[6] P Diniz, B Hendrickson, R Leland and S Plimton: "A new parallel distributed graph partitioning algorithm", Technical Report, Sandia National Laboratories, Albuquerque, NM, 1993 (*In preparation*)

[7] C Farhat, W Wilson and G Powell: "Solution of Finite Element Systems on Concurrent Processing Computers" *Engineering with Computers*, **2**, 157–165, (1987).

[8] RF Fowler, BW Henderson and C Greenough: "Grid Partitioning Techniques", IDENTIFY Report, RAL.93.8, December 1993.

[9] GC Fox: *Numerical Algorithms for Modern Parallel Computers*, Springer-Verlag, Berlin (1988).

[10] G Reeve: "The CAMAS domain decomposition tool", EUROPORT–1 Workshop on Unstructured Grids, GMD, Schloss Birlinghoven, Germany (1994).

[11] B Hendrickson and R Leland: "A Multilevel Algorithm for Partitioning Graphs", Technical Report SAND93-1301, Sandia National Laboratories, Albuquerque, NM, 1993

[12] B Kernighan and S Lin: "An efficient heuristic procedure for partitioning graphs", Bell System Technical Journal, 29 (1970), pp. 291-307.

[13] JG Malone: "Automatic Mesh Decomposition and Concurrent Finite Element Analysis for Hypercube Multiprocessor Computer", Comp. Meth. in Applied Mechanical Eng., **70**, 27–58 (1988).

[14] HA Schwarz: "Uber einige Abbildungsauf-gaben", *Ges. Math. Abh.* **11** 65–83 (1869).

[15] A Pothen, HD Simon and KP Liu: "Partitioning Sparse Matrices with Eigenvectors of Graphs", Report RNR-89-009, NASA Ames Research Center, July 1989.

[16] ST Barnard and HD Simon: "A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems", in Proc. 6th SIAM Conf. Parallel Processing for Scientific Computing, SIAM, pp. 711–718, 1993

[17] RD Williams: "Performance of Dynamics Load Balancing Algorithms for Unstructured Mesh Calculations", Concurrency, Practice and Experience, 3, No.5, 457–482 (1991).

[18] AI Khan and BHV Topping: "Subdomain generation for parallel finite element analysis", Computing Systems in Eng., Vol. 4, Nos. 4-6, pp. 473-488 (1993).

[19] RF Fowler, BW Henderson, and C Greenough, "Initial Experiences in Porting a Three-Dimensional Semiconductor Device Modelling Program to the Intel iPSC/860", Rutherford Appleton Laboratory Report, RAL-92-090 (1992).

[20] C Greenough and RF Fowler: "Partitioning Methods for Unstructured Finite Element Meshes", Rutherford Appleton Laboratory Report, (*In preparation*).

[21] C Greenough and RF Fowler: "RalPar – The RAL Mesh Partitioning Program", Rutherford Appleton Laboratory Report, (*In preparation*).