1989

Automatic Load Balanced Partitioning Strategies for PDE Computations

N. P. Chrisochoides

C. E. Houstis

Elias N. Houstis
Purdue University, enh@cs.purdue.edu

S. K. Kortesis

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:
89-862

AUTOMATIC LOAD BALANCED PARTITIONING STRATEGIES FOR PDE COMPUTATIONS

N. P. Chrisochoides
C. E. Houstis
E. N. Houstis
S. K. Kortesis
J.R. Rice

CSD-TR-862
February 1989

AUTOMATIC LOAD BALANCED PARTITIONING STRATEGIES FOR PDE COMPUTATIONS*

N.P. Chrisochoides, C.E. Houstis†, E.N. Houstis,
S.K. Kortesis†, and J.R. Rice

Computer Sciences Department
Purdue University
CSD-TR-862
February, 1989

Abstract

In this paper we study the partitioning and allocation of computations associated with the numerical solution of partial differential equations (PDEs). Strategies for the mapping of such computations to parallel MIMD architectures can be applied to different levels of the solution process. We introduce and study heuristic approaches defined on the associated geometric data structures (meshes). specifically, we study methods for decomposing finite element and finite difference meshes into balanced, nonoverlapping subdomains which guarantee minimum communication and synchronization among the underlying associated subcomputations. Two types of algorithms are considered: clustering techniques based on sequential orderings of the discrete geometric data and optimization based techniques involving geometric or graphical metric criteria. These algorithms support the automatic mode of a geometry decomposition tool developed in the parallel ELLPACK environment which is implemented under X11-window systems. A brief description of this tool is presented.

---

† Visiting scholars from University of Crete and Thessaloniki, Greece. This research was supported in part by NSF grant CCF-8619817, AFOSR 88-0243, ARO grant DAAG29-83-K-0026, David Ross Grant 6901291 and ESPRIT project 1588.

AUTOMATIC LOAD BALANCED PARTITIONING STRATEGIES FOR PDE COMPUTATIONS

N.P. Chrisochoides, C.E. Houstis†, E.N. Houstis,
S.K. Kortesis† and J.R. Rice

Department of Computer Science
Purdue University
West Lafayette, IN 47907

Abstract

In this paper we study the partitioning and allocation of computations associated with the numerical solution of partial differential equations (PDEs). Strategies for the mapping of such computations to parallel MIMD architectures can be applied to different levels of the solution process. We introduce and study heuristic approaches defined on the associated geometric data structures (meshes). Specifically, we study methods for decomposing finite element and finite difference meshes into balanced, nonoverlapping subdomains which guarantee minimum communication and synchronization among the underlying associated subcomputations. Two types of algorithms are considered: clustering techniques based on sequential orderings of the discrete geometric data and optimization based techniques involving geometric or graphical metric criteria. These algorithms support the automatic mode of a geometry decomposition tool developed in the parallel ELLPACK environment which is implemented under X11-window systems. A brief description of this tool is presented.

1. INTRODUCTION

In this paper we consider the problem of partition and allocation of computations associated with the numerical solution of Partial Differential Equations (PDEs) into load balanced tasks requiring minimum synchronization and communication. The efficient solution of this problem is essential for the parallel processing of such computations. Its formulation can be based either on the geometric or the algebraic data structures associated with the PDE solution process or on the precedence graph of the PDE solver. In [Hous 88] we present a methodology for partitioning and allocation which is applicable to the computation graph of the PDE solution process. In this paper we formulate and analyze partitioning methodologies for computations defined on geometric data structures. The approach has been implemented as a special software tool in the *parallel ELLPACK* environment [Hous 89b]. The issue of automatic partitioning has been raised in [Fara 88] in conjunction with finite element computations on local memory machines and in [Fox 86a,b] for certain algebraic computations. Some partitioning strategies for nonuniform problems on multiprocessors are studied in [Berg 85]. In Section 2 we present several partitioning strategies and formulate various criteria that are used to determine them. A new algorithm based on geometric graph partitioning (GGP) ideas is developed and presented in Section 3, together with the description of existing clustering approaches. In Section 4 we present the performance evaluation of four automatic partitioning strategies. The results obtained indicate that the new GGP algorithm produces "optimum" partitions within reasonable time intervals. Finally, Section 5 contains a brief

† Visiting scholars from University of Crete and Thessaloniki, Greece. This research was supported in part by NSF grant CCF-8619817, AFOSR 88-0243, ARO grant DAAG29-83-K-0026, David Ross grant 6901291 and ESPRIT project 1588.

description of the geometry decomposition software tool implemented under *X*-windows, which allows the user to specify and manipulate domain decompositions interactively, as well as to display the automatically generated ones. A detailed description of this tool will appear in [Hous 89b].

2. PARTITIONING STRATEGIES FOR PDE COMPUTATIONS

The numerical solution of partial differential equations usually is represented by an approximate function defined over a given mesh of the PDE domain. This function is determined by solving a system of algebraic equations that depends on the discretization method used. For the solution of this set of equations with a parallel MIMD machine consisting of many processor elements, a partitioning of the underlying computation is required. The strategy of allocating the partitions of the computations to processors affects the performance of the parallel processing. Any optimal partitioning/allocating strategy has the following objectives (a) the work loads of all processors are balanced, and (b) the processor synchronization and communication cost is kept to a minimum. Partitions of PDE computations that satisfy the above goals can be defined either on the geometric data structures (meshes) or algebraic data structures (discrete systems) or on the computation graph of the selected PDE solver. In this paper we consider partitioning strategies applied to fixed PDE meshes Ω_h, which are referred as *geometry decomposition* approaches. Instances of the other two approaches are formulated and studied in [Fox 86] and [Hous 88]. Throughout this paper we assume that Ω_h is a finite element mesh consisting of $\{n_i(x,y,z)\}_{i=1}^{N}$ nodes with $\{w_i\}_{i=1}^{N}$ connectivity and, $\{e_j(n_1, \ldots, n_k)\}_{j=1}^{NE}$ elements. The case of finite difference meshes can be handled in a similar way. In the geometry decomposition strategy, we seek a partition of the underlying PDE computation determined by a decomposition of Ω_h into P (number of processing elements) nonoverlapping subdomains $\{D_i\}_{i=1}^{P}$ such that

(i) each subdomain contains almost the same number of elements,

(ii) they are *"circular" or "spherical"* and connected, and

(iii) the number of interface nodes or interior boundary elements among subdomains is minimum.

It has been observed that a nearest neighbor allocation of such partitioning computations tends to be close to optimum. For the mathematical formulation of this partitioning problem, we introduce Euclidean and graphical metrics. Throughout we denote by $(\hat{x}_i,\hat{y}_i)$ the geometric center of a 2-D element e_i, $|D_k| = c_k$ the number of elements in this subdomain, D_k' the interface interior elements of D_k, C_{e_j} the subset of Ω_h consisting of all adjacent elements to e_j, and $\chi(e_i,e_j)$, the characteristic function $\{\chi(e_i,e_j) = 1$ if $e_i \in C_{e_j}$ and e_i and e_j are in different domains or $\chi(e_i,e_j) = 0$ otherwise$\}$. In the case of 2-dimensional meshes we define the mass center of each subdomain D_i to be the point

$$x_{D_k} = \frac{1}{c_k} \sum_{e_i \in D_k} \hat{x}_i, \quad y_{D_k} = \frac{1}{c_k} \sum_{e_i \in D_k} \hat{y}_i$$

and the Euclidean distance between two subdomains (D_k,D_ℓ) to be the distance between mass centers

$$d_{D_k,D_\ell} = \sqrt{(x_{D_k} - x_{D_\ell})^2 + (y_{D_k} - y_{D_\ell})^2}\ .$$

Furthermore, we denote by $r_k = \sqrt{|D_k|/\pi}$ the radius of the subdomain D_k. The topology of a finite element mesh Ω_h is represented by a graph $G(V,L)$ whose vertices V correspond to the elements of Ω_h and edges L correspond to adjacent pairs of elements. A P-way domain decomposition of Ω_h is viewed as a P-way graph decomposition of G. The problem of partitioning graphs has been studied by many researchers [Barn 82], [Dona 73], [Gold 84], [Chri 76]. In this paper we are particularly interested in the techniques presented in [Kern 70]. For the formulation of P-way graph partitioning techniques, we define the mass center CR_k of a subgraph $G_k(V_k,L_k)$ as follows:

$$CR_k = \{v_i \in V_k : \min \sum_{v_j \in V_k} [\rho(v_i,v_j)]^2\}$$

where $\rho(v_i,v_j)$ denotes the minimum path that connects the pair v_i, v_j in V_k. The distance of vertix $v_i \in V_k$ from CR_k is defined by the quantity

$$d_{i,CR_k} = \frac{1}{|CR_k|} \sum_{v_j \in CR_k} [\rho(v_i, v_j)]^2$$

while the distance between subdomains (subgraphs) $D_k(G_k)$ and $D_\ell(G_\ell)$ is defined to be

$$d_{D_k,D_\ell} = \frac{1}{|CR_k|\ |CR_\ell|} \sum_{\substack{v_i \in CR_\ell \\ v_j \in CR_k}} [\rho(v_i, v_j)]^2.$$

In the case of graphical metrics, the radius r_k of D_k is defined as the minimum d_{i,CR_k} for all vertices $v_i \in G_k$ for which the corresponding element belongs to the interface D_k' of the subdomain D_k.

3. AUTOMATIC LOAD BALANCED GEOMETRY PARTITIONING STRATEGIES

In this section we formulate and analyze various heuristic domain decomposition techniques for finite element meshes. Specifically, we identify and study two classes of methods. The first class consists of partitioning strategies which are based on some global ordering of elements which minimize the bandwidth or envelope of the discrete PDE system. The other class involves optimization based methods using Euclidean or graphical metrics.

3.1 Clustering techniques

The simplest load balanced decomposition strategy is to group the first NE/P elements of the mesh into subdomains, assuming some a priori global ordering of the mesh elements. Figure 3.1 depicts a domain partitioning based on an ordering of elements along vertical mesh lines. Although this ordering scheme leads to banded systems with minimum bandwidth, it produces partitions whose subdomains are "non-spherical", sometimes disconnected and with lengthy interior interfaces.

A second, more sophisticated clustering approach is based on the Cuthill and McKee ordering scheme. This approach tries to determine subdomains D_k around some specific element e_i such that $\sum\chi(e_i,e_j) > 0$ over all elements $e_j \in C_{e_i}$. If the cardinality of $|D_k| < c_k$, then D_k is enlarged recursively by including the elements $C_{e_k} - (C_{e_k} \cap D_k)$ where $e_k \in D_k$ until $|D_k| = c_k$. This expansion of the Cuthill-McKee ordering ensures to some degree local minimization of subdomain interfaces while it forces two consecutive subdomains to be adjacent. This ensures the global minimization of the subdomain interface lengths. Our experiments indicate that this scheme leads to partitioning with disconnected subdomains (Figure 3.1b). Different

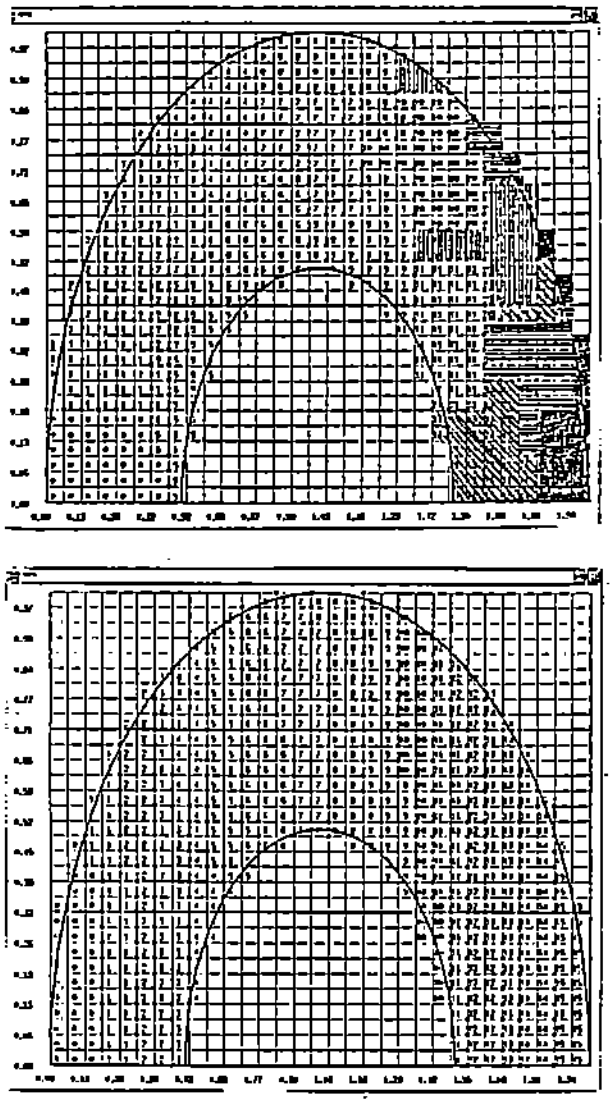

Figure 3.1 Two P-way partitions of a discretized semi-annulus domain using the clustering approach based on the natural ordering and Cuthill-McKee ordering of the mesh elements.

implementations of the above strategy are presented in [Fahr 88] and [Hous 89b].

3.2 Optimization based techniques

We now consider P-way balanced partitions which optimize certain objective functions. Specifically the problem of determining a load balanced P-way partition with minimum interface length is reduced to the minimization of the communication or cut cost function

$$\frac{1}{2}\sum_{k,l=1}^{P}\sum_{e_i\in D_k}\sum_{e_j\in D_l}\chi(e_i,e_j) \qquad (3.1)$$

provided $|D_k| = c_k > 0$, $\Omega_h = \bigcup_{k=1}^{P} D_k$ and $D_k \cap D_l = \varnothing$ for all subdomains. The number of feasible solutions of a P-way partitioning is prohibitively large [Kern 70] even for a moderate number P of subdomains. A good alternative is to determine semi-optimal solutions using fast heuristics. It appears that the most efficient heuristic strategy for partitioning graphs is the so-called Kernighan-Lin (Ker-Lin) approach [Kern 70]. According to this technique, a given feasible solution is improved with respect to the minimization of the objective function (3.1) by interchanging the pair of elements $(e_i,e_j) \in D_k \times D_l$ such that the so-called *profit function*

$$f(e_i,e_j) = 2\sum_{e_\lambda\in C_{e_i}}\chi(e_i,e_\lambda) - |C_{e_i}| + 2\sum_{e_\mu\in C_{e_j}}\chi(e_\mu,e_j) - |C_{e_j}| - 2\chi(e_i,e_j)$$

is maximum. Although the method is capable of determining a "good" local minimum as Figure 3.2 indicates, its time complexity is significantly large.

In order to guarantee that a P-way partition satisfies the criteria (i) to (iii), we introduce the following profit function for elements e_i, e_j in D_k and D_l respectively

$$F(e_i,e_j) = w\left[\left[\left\lceil\frac{d_{i,CR_k}}{r_k}-1\right\rceil-\left\lceil\frac{d_{j,CR_k}}{r_k}-1\right\rceil\right] + \left[\left\lceil\frac{d_{i,CR_l}}{r_l}-1\right\rceil-\left\lceil\frac{d_{j,CR_l}}{r_l}-1\right\rceil\right] + f(e_i,e_j).$$

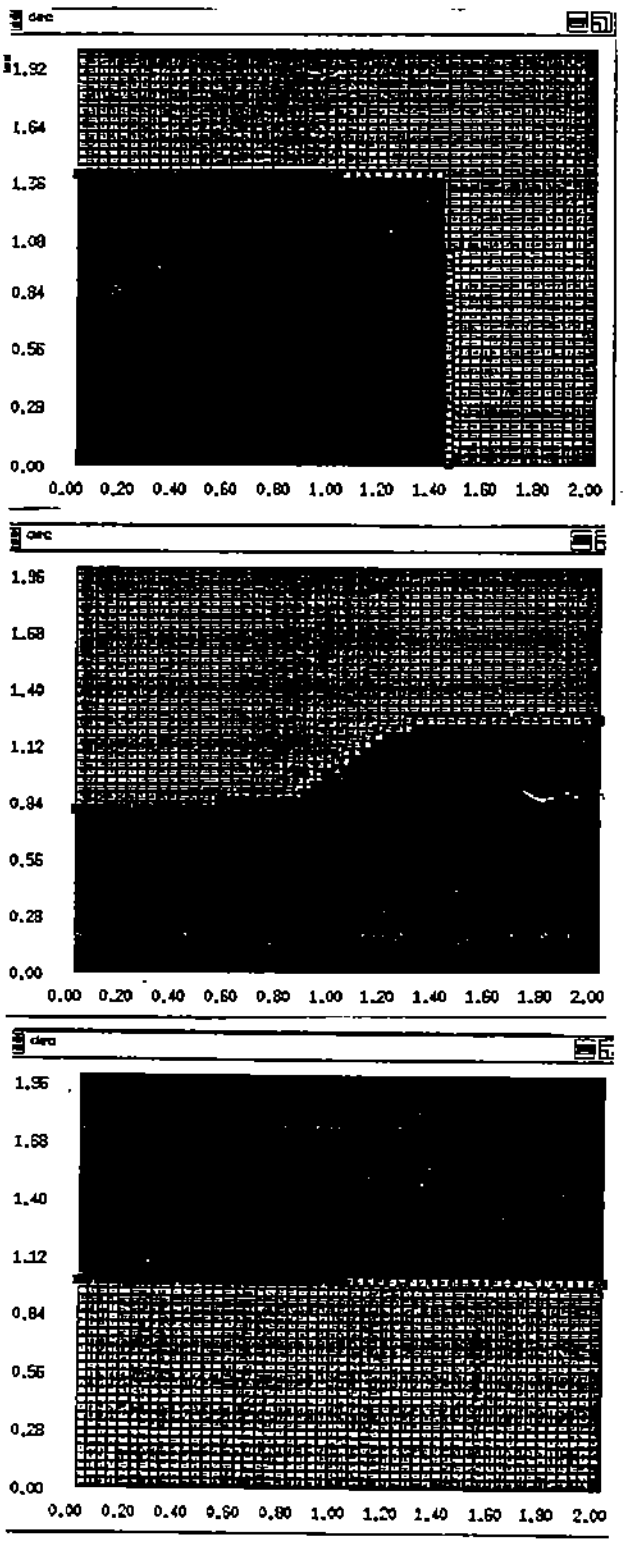

Figure 3.2 2-way partitions of a discrete rectangular domain. (a) Cuthill-McKee partition, (b) Ker-Lin partition, (c) GGP solution (optimal solution).

It is easy to see that the maximization of F forces the interchanging of elements outside the "radius" of D_k, D_t which are closer to the current interface of the two selected subdomains. Next, we describe a 2-way optimization based algorithm. Its generalization to a P-way optimization is called *GGP-recur* and is obtained by applying the same algorithm recursively. The Cuthill-McKee algorithm and the 2-way GGP have been combined to produce a P-way partition algorithm called *CM-GGP*. A global non-recursive P-way partitioning geometry graph algorithm is under development and it will appear elsewhere.

/* An automatic 2-way domain decomposition algorithm */

Assume an initial decomposition $\Omega_k = D_1 \cup D_2$ with $|D_1| = c_1$, $|D_2| = c_2$ and interfaces D_1', D_2'.

Step 1: Compute the characteristics of the initial decomposition (center of mass, distance, communication-cost or cut-cost).

Step 2: Determine a pair $(e_i,e_j) \in D_1' \times D_2'$ such that the profit function $F(e_i,e_j)$ is positive or maximum and e_i or e_j has not been considered in the previous interchanges with $f(e_i,e_j) \leq 0$.

Step 3: **If** the value of the profit function F is positive and the number of interchanges with nonpositive F values is less than the given limit (*max-int*), **then** update D_1, D_2 and repeat the algorithm with the updated decomposition as the initial decomposition

If the value of the profit function F is non-positive, the number of unsuccessful interchanges is greater than the given limit (*max-int*) and the distance d_{D_1,D_2} increases or cut-cost has been reduced **then** set the number of unsuccessful interchanges equal to 0, update D_1 and D_2 and repeat the algorithm. **OTHERWISE** terminate.

The data in Figure 3.3 imply that the new profit function forces interchanges of elements among subdomains D_1 and D_2 that increase their distance while reducing the objective function (3.1).

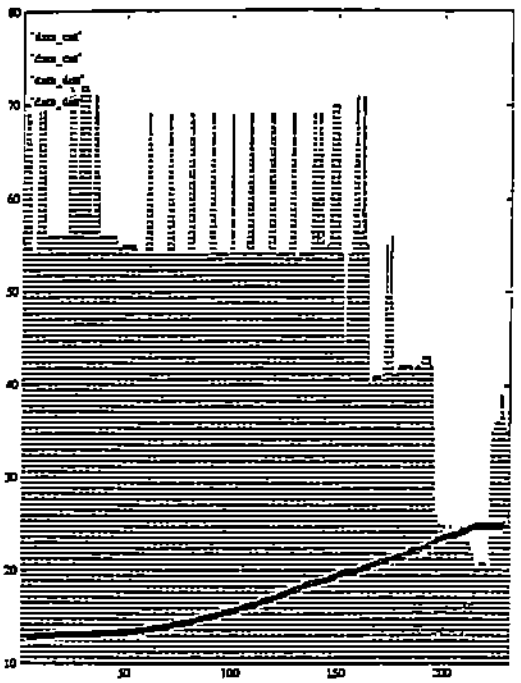

Figure 3.3 The values of the cut-cost function and the distance $d_{D_1D_2}$, during the execution of the GGP-algorithm for an L-shape domain. A mesh with 406 elements was used.

4. PERFORMANCE OF GEOMETRY PARTITIONING STRATEGIES

In this section we present the results of some preliminary experiments to measure the degree of satisfiability of the partitioning criteria (i) to (iii) and the time complexity of various geometry decomposition approaches. We have implemented five load balancing algorithms. The two are clustering algorithms based on the natural and Cuthill-McKee orderings which are called *Nat-clust* and *CM-clust*, respectively. The other three are graph partitioning type algorithms which try to optimize certain cut-cost functions guided

by certain profit functions involving Euclidean metrics. Their implementation using graphical metrics is under way. These are the Kernighan Lin (*Ker-Lin*), the recursive algorithm based on the 2-way algorithm described in Section 3 (*GGP-recur*) and the GGP-alg algorithm. Figure 4.1 presents the communication requirements or interface lengths for the semi-annulus discrete domain of Figure 3.1 obtained using the four algorithms. In the case of Ker-Lin and GGP-recur algorithms the initial feasible partition used is the *P*-way solution of *CM*-clust. The data indicate clearly that the *GGP*-recur solution is quantitative and qualitative "closer" to the optimum.

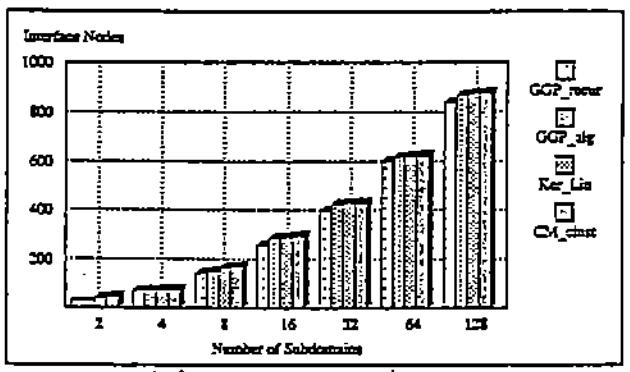

Figure 4.1 The interface lengths of five *P*-way partitions of a 2500 element mesh in the semi-annulus domain obtained with the one clustering and the three graph partitioning algorithms.

In Figure 4.2 we present the number of interface nodes as a function of the mesh size, for 2-way partitions obtained by two optimization based algorithms (Ker-lin, GGP-alg). Figure 4.3 shows the time complexity of these algorithms measured in terms of the number of interchange elements required for different meshes. The results in Figure 4.2 and 4.3 are the average values over a population of five domains assuming random initial partition. These results indicate that the GGP solution is closer to the optimum partition as defined by criteria (i) to (iii). Furthermore, the time complexity measured implies that the "better" algorithm (GGP) is the faster one.

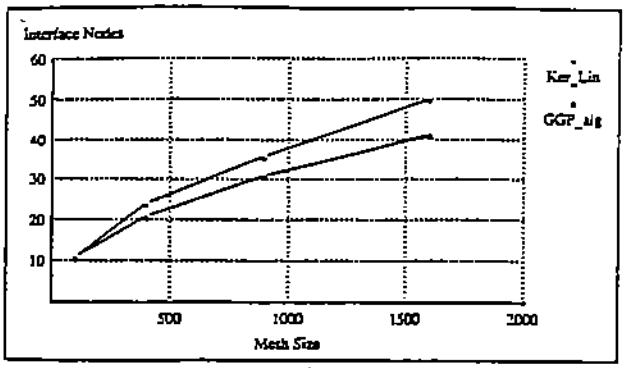

Figure 4.2 The average number of interface nodes for two 2-way optimization based partitions for different meshes over five different domains.

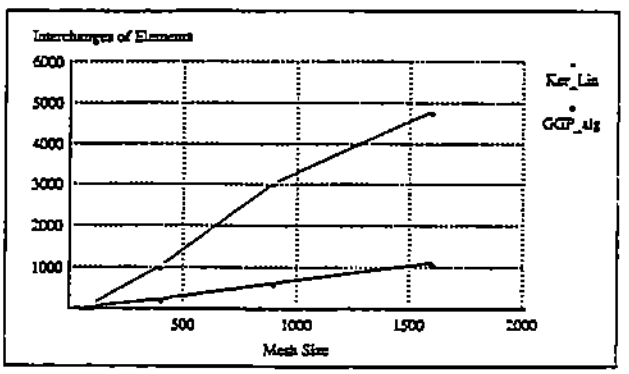

Figure 4.3 Average time complexity of 2-way optimization based methods for different size meshes over five different domains.

Finally the data in Figures 4.4 and 4.5 support the conclusion that the proposed GGP algorithm produces "better" partitions among the clustering and optimization based algorithms.

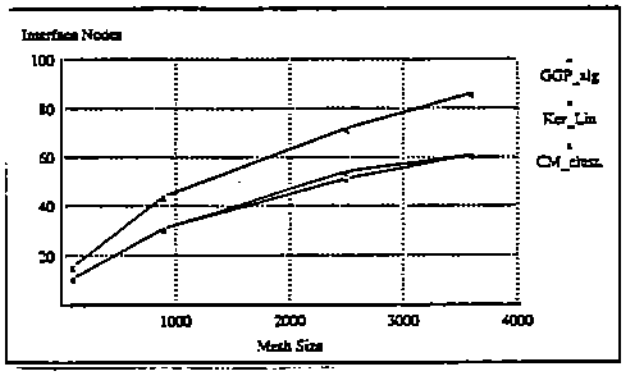

Figure 4.4 The number of interface nodes as a function of the mesh size obtained by the clustering algorithm CM-clust and the two optimization based algorithms (Ker-Lin, GGP-alg) with initial partition the CM-clust solution.

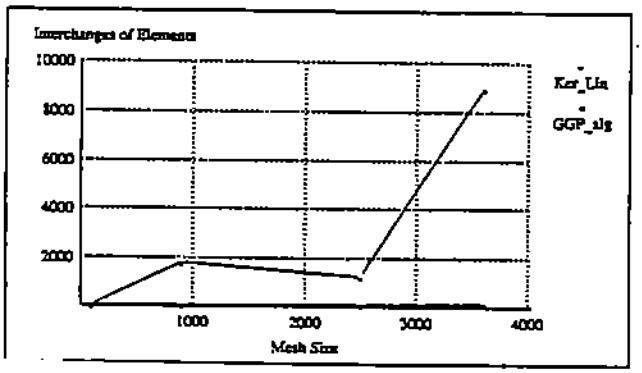

Figure 4.5 The time complexity of Ker-Lin and GGP-alg algorithms with the initial partition of the CM-clust solution.

5. GEOMETRY DECOMPOSITION

We have built an interactive environment called *DecTool* to help with domain decomposition. An example display is shown in Figure 5.1. The environment provides facilities for both automatic (using a predefined algorithm), and manual decomposition of a domain. This interactive environment is written using the Toolkit from the third release of X11 windows. Its detailed description is given in [Hous 89b].

DecTool consists of three different windows. The first one is the basic DecTool window which initially appears on the screen (left center in Figure 5.1). It is used to control DecTool using the following three buttons.

DONE:	Signals to exit from the tool. After exiting, an output file is produced which contains the description of the last decomposition of the domain in a predefined format.
AUTOMATIC:	Invokes the automatically decomposed algorithm.
MANUAL:	Allows the user to specify the decomposition explicitly using the mouse.

In this basic window, there are three additional widgets (X Toolkit jargon), for input of editable parameters of the tool. The first, NUMBER OF SUBDOMAINS, specifies the number of subdomains for the decomposition. The user specifies the number of subdomains before using the AUTOMATIC button. When the MANUAL button is used, an estimate of the number of subdomains is entered, which should not be less than the final number of the subdomains. The second widget, MODE, provides two options, SUBDOMAIN and INTERFACE. Every click inside the rectangle changes the mode of the tool. Mode refers to what the user is planning to accomplish, i.e., specify subdomains or specify the assignment of the interface grid points. The last widget is one that shows the currently selected subdomain. If there is none, NONE is displayed, otherwise the color of the selected subdomain and its number are displayed.

The second window displayed is a color palette. A color can be assigned to each subdomain and this assignment is displayed on the palette window. This window is also used to interactively select (change) the currently selected subdomain by clicking inside the color rectangle that corresponds to the subdomain selected.

The third window displays the domain, which is defined in terms of a set of mesh lines and a boundary line. In case there is a decomposition, it is displayed using the color assignment specified by the palette window. This window can also be used to edit interactively, (by using the mouse). the current decomposition. It is also possible to specify a decomposition from scratch by selecting the MANUAL option.

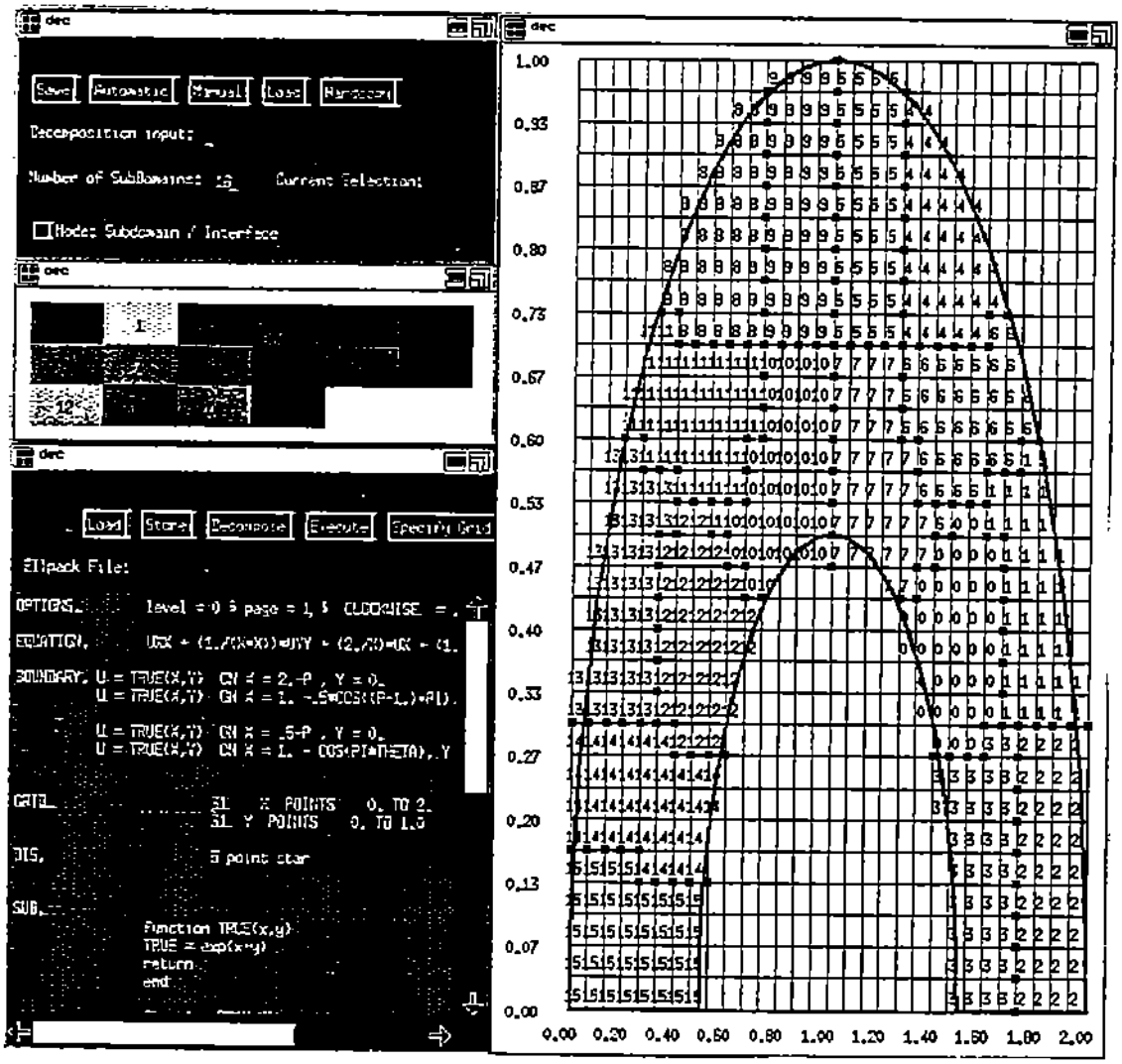

Figure 5.1 Example of the geometry decomposition tool DecTool for an annular region and 16 subdomains. The two DecTool windows are on the left side, the bottom left is for parallel ELLPACK and the right one is the display of the decomposed finite element mesh.

6. REFERENCES

[Barne 82] E.R. Barnes, "An algorithm for partitioning the nodes of a graph", *SIAM Journal Algebraic and Discrete Methods*, Vol. 3, pp. 541–550, Dec. (1982).

[Berg 85] M.J. Berger and S. Bokhari, "A partitioning strategy for non-uniform problems as multiprocessors", ICASE Report No. 85–55, Nov. (1985).

[Chris 76] N. Christofides and P. Brooker, "The Optimal partitioning of graphs", *SIAM Journal of Applied Mathematics*, Vol. 30, pp. 55–69 (1976).

[Donat 73] W.E. Donath and A.J. Hoffmann, "Lower bounds for the partitioning of graphs", *IBM Journal of Research and Development*, Vol. 17, pp. 420–425, Sept. (1973).

[Farh 88] C. Farhat, "A simple and efficient automatic FEM domain decomposer", Computers and Structures, Vol. 28, pp. 579–602 (1988).

[Fox 86a] G.C. Fox, "A graphical approach to load balancing and sparse matrix vector multiplication on the hypercube", in *Numerical Algorithms for Modern Parallel Computer Architectures*, IMA Vol. Math. App., 13 (M. Schultz, ed.), Springer-Verlag, pp. 37–61 (1987).

[Fox 86b] G.C. Fox, "A review of automatic load balancing and decomposition methods for the hypercube", in *Numerical Algorithms for Modern Parallel Computer Architectures*, IMA Vol. Math. App., 13 (M. Schultz, ed.), Springer-Verlag, pp. 63–76 (1987).

[Garey 79] M.R. Garey and D.S. Johnson, "Computers and Intractability", W.H. Freeman and Company, (1979).

[Goldb 84] M.K. Goldberg and R. Gardner, "On the minimal cut problem", *Progress in Graph Theory*, (1984).

[Hous 88] C.E. Houstis, E.N. Houstis, J.R. Rice, S.M. Samartzis and D.L. Alexandrakis, "The algorithm mapper: A System for modeling and evaluating parallel applications/architectures pairs", in *Fourth Generation Mathematical Software Systems* (Houstis, Rice and Vichnevetsky, eds.), North-Holland (1989), to appear.

[Hous 89a] E.N. Houstis, T.S. Papatheodorou and J.R. Rice, "Parallel ELLPACK: An expert system for the parallel processing of partial differential equations", *Math. Comp. Simul.*, Vol. 31 (1989), to appear.

[Hous 89b] E.N. Houstis, P.N. Papachiou, J.R. Rice and M.S. Samartzis, "Domain decomposer: A software tool for partitioning PDE computations based on geometry decomposition strategies", Purdue University Technical report, in preparation.

[Kern 70] B.W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs", *The Bell System Technical Journal*, Feb. (1970), pp. 291–307.

[Pirkt 83] Lennart, Pirktl, "On the use of cluster analysis for partitioning and allocation computational objects in distributed computing systems", *Computer Science and Statistics: The Interface*, James E. Gentle (ed.), North-Holland, pp. 361–364, (1983).