DOMAIN DECOMPOSER: A Software Tool for Mapping PDE Computations to Parallel Architectures

N. P. Chrisochoides‡
C. E. Houstis‡
E. N. Houstis‡
P. N. Papachiou‡
S. K. Kortesis*
J. R. Rice†‡

Abstract

Domain decomposition methods have proved to be an efficient approach for parallel processing of partial differential equations (PDEs) on parallel architectures. Their built in course grain parallelism makes them suitable for MIMD computing as a methodology to assure that the algebraic data are generated and distributed in different processors so that the processor workload is balanced and their synchronization/communication cost is kept minimum. These requirements can introduce serious computation costs since many times optimum workload balance and minimum synchronization/communication cost involve the solution of NP-hard problems. In this paper we outline a software infrastructure consisting of "fast" heuristics for determining "optimal" mapping of PDE data suitable for domain decomposition methods. Furthermore we describe a software system which assists the user to visualize and manipulate such mappings in the environment of parallel-ELLPACK system.

1 Introduction

The numerical solution of a partial differential equation (PDE) usually is represented by an approximate function defined over a given mesh of the

*University of Thessaloniki, Polytechnic School, Thessaloniki, GREECE

†This research was supported in part by AFOSR 88-0234, ARO grant DAAG29-83-K-0026, NSF grant CCF-8619817 and ESPRIT project GENESIS.

1 *INTRODUCTION*

PDE domain. This function is determined by solving a system of algebraic equations that depend on the discretization method used. For the solution of this set of equations with a parallel MIMD machine, a partitioning of the underlying computation and their allocation to individual processors is required so that nearly optimal speedup is achieved. Any optimal partitioning/allocation (mapping) strategy has the following objectives: (a) the workloads of all processors is balanced, and (b) the processor synchronization and communication cost is kept to a minimum. Partitionings of PDE computations satisfying the above goals are usually defined either on the algebraic data structures (discrete systems) or on the computational graph of the selected parallel PDE solver. Instances of such approaches are formulated and studied in [Fox 86], [Sada 87], [Pomm 90], [Ayka 88], and [Hous 87,90a,90b]. One of the most promising parallel approaches for the numerical solution of partial differential equations(PDEs) is the so-called domain decomposition (DD) methodology. Its basic idea can be applied either at the level of the PDE problem or at the underlying computation. We refer to the first formulation as "continuous" and the second one as "discrete" domain decomposition [Chris 91]. With the continuous DD approach the PDE problem is subdivided into a number of coupled "smaller" PDE problems defined on subdomains of the original PDE domain with appropriate interface conditions which depend on the solutions of the neighbor subdomain PDE problems. One advantage of this approach is its ability to use existing sequential algorithms on the individual subdomains. However the convergence of the local solutions to the global solution has been shown only for special elliptic PDE problems [Mari 88]. The implementation and performance of this approach is under investigation in [Chris 91] in the environment of //ELLPACK system. The "discrete" version of the domain decomposition approach has been extensively studied by many researchers and there is much data supporting its suitability for the parallel processing of PDEs. The basic idea of the method is to subdivide the discrete PDE geometric data and assign the corresponding computations in different processors. Both DD approaches require the "optimal" decomposition of the PDE domain(continuous or discrete) so that the mapping of the underlying computation on a parallel machine satisfies the above performance objectives. In this paper we review various geometry decomposition techniques and describe a software tool for supporting domain decomposition procedures which are based on partitionings of the discrete or continuous PDE geometric data. The actual experience reported upon here is for the discrete approach.

The geometric data structures to be used for the formulation of the geometry decomposition mapping strategies are the ones used by finite difference and finite element discretization procedures of PDE problems. The finite difference mesh or grid Ω_h is an orthogonal grid consisting of interior grid points $\Omega_h^0 \equiv \{(x0(i), y0(i), z0(k))\colon 1 \leq i \leq Nx,\ 1 \leq j \leq Ny,\ 1 \leq k \leq Nz\}$ and $\partial\Omega_h \equiv \{(xb(i),\ yb(i), zb(i))\}_{i=1}^{Nb}$ boundary points $(\Omega_h = \Omega_h^0 \cup \Omega_w^b)$. The finite element method (FEM) mesh consists of Ω_h, a set of elements $\{e_j\}_{j=1}^{NE}$ with nodes $\{n_i\}_{i=1}^{N}$. In the geometry decomposition strategy, we seek a partition of the underlying PDE computation determined by a decomposition (partitioning) of Ω_h into P (number of processing elements) nonoverlapping subdomains $\{D_i\}_{i=1}^{P}$ and then allocating them to the processors such that the following criteria are met.

(i) the subdomains have the same number of elements or grid points,

(ii) the interface among the subdomains is "small",

(iii) the number of adjacent subdomains to each subdomain is "minimum",

(iv) the subdomains are not disconnected,

(v) the communication requirements of the underlying computation on a given architecture (processor interconnection graph) are "minimum",

(vi) the synchronization among subdomain computations is kept low.

Throughout we describe geometric based strategies defined on finite element meshes. The case of finite difference meshes can be handled in a similar fashion. It is easily seen that criteria (i) and (ii) can be modeled by a constrained optimization problem. Let $\chi(e_i, e_j)$ represent the processor (subdomain) adjacency of the element nodes e_k and e_j, that is

$$\begin{aligned} \chi(e_i, e_j) &= 1 \quad \text{if } e_i \text{ and } e_j \text{ are adjacent and in different subdomains} \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

Then the optimal decomposition using criteria (i) and (ii) of the FEM mesh Ω_h is the one that minimizes the cost function

$$\frac{1}{2} \sum_{k,\ell=1}^{P} \sum_{e_i \in D_k} \sum_{e_j \in D_\ell} \chi(e_i, e_j) \tag{2.1}$$

subject to the constraint

$$|D_k| = N/P \quad k = 1, \ldots, P. \tag{2.2}$$

This cost function models the communication requirements of the application assuming a uniform communication cost between adjacent elements.

Criteria (iii) and (iv) are usually imposed during the solution of (2.1)-(2.2) [Chris 89] by seeking solutions that optimize certain additional functions known as profit functions. The fifth criterion can be modeled by the following unconstrained minimization problem

$$\min_m \frac{1}{2} \sum_{i=1}^{P} \sum_{j=1}^{P} c(D_i, D_j) d(m^{-1}(D_i), m^{-1}(D_j)) \tag{2.3}$$

where m is the mapping of subdomains to processors, c the interface length between D_i and D_j and $d(m^{-1}(D_k), m^{-1}(D_\ell))$ is the distance between the two processors assigned to D_k and D_ℓ as measured in the interconnection network of the machine. Finally, the reduction of the synchronization requirements of the subdomain computations can be accomplished by grouping the various synchronization points. This technique is known as *coloring.* Elements are given a color and elements of each color are distributed to all the processors; then they process the same color concurrently. Between the processing of two colors the needed values from the latest colors are exchanged.

Although these five criteria can be imposed independently of each other, some times it makes sense to combine them with appropriate weights [Flow 88], [Fox 88], [Will 90], [Hous 90]. The optimal geometric mapping problem can be equivalently formulated and solved on the so called *topological graph* of the finite element mesh. Throughout this paper we denote this graph by $G_M(V_M, E_M)$ where each vertice V_M of the graph is one of the elements $\{e_j\}$ of the mesh and the edges E_M of the graph correspond to adjacent elements. In this formulation, the problem of P-way mesh partitioning in load balanced subdomains is equivalent to obtaining a P-way partitioning of the topological graph $G_M(V_M, E_M)$. Regardless of its formulation, the above stated problem of partitioning/allocation geometric data structures is NP-hard [Flow 88]. Thus most of the proposed solutions are, at least, "nearly" optimal and are obtained by powerful heuristic techniques. In this paper we give a brief description of a generic software tool, *Domain Decomposer*, for supporting "continuous" or "discrete" geometry decomposition methods[Chri 89] and describe its software infrastructure. Specifically, in

Sections 2 and 3, we review the more promising partitioning and allocation techniques for mapping discrete geometric data associated with PDE discretization methods onto parallel architectures. Section 4 describes the functionality of the domain decomposer.

2 Geometry Based Partitioning Strategies

Most of the heuristics proposed for the partitioning phase can be classified into three large groups. The first group includes the *cluster* techniques whose main idea is to sort the geometric or topological mesh data in some direction and then partition the resulting sequence of elements in P-ways. The second group consists of the *deterministic optimization* techniques. Their idea is to find "semi"-optimal feasible solutions in linear time. These methods tend to terminate in some local minimum value without the ability to move out of them. A more expensive alternative is to use a *stochastic* optimization technique, which tends to locate the global optimum "most" of the time. In [Hous 90] we have implemented simulated annealing and neural network approaches to the mesh partitioning problem. Next we give a short description of the algorithms currently supported by the domain decomposition tool.

2.1 Clustering techniques

Farhat [Fahr 88] proposed a method for ordering the topological data of a mesh. The underlying idea of his scheme is equivalent to the well known Cuthill-McKee method of ordering as applied to order finite element meshes so that the corresponding linear algebraic system of equations has minimum bandwidth and profile. According to this technique one finds all the unlabeled neighbors of element (vertix) i and labels them in order of increasing connectivity (degree). We refer to this method as *CM-cluster.* Another naive way for splitting FEM meshes is to sort some geometric data of the mesh (i.e., coordinates of vertices, coordinates of sector origin of the elements, coordinates of the centroid of the elements) and subdivide the sorted lists in sublists of length N/P. The same idea can be also applied to the corresponding topological data of the mesh. These sorting algorithms are referred throughout this paper as $1 \times P$ *geometric/topological partitioning* and $P \times Q$ *geometric/topological partitioning* algorithms.

2.2 Deterministic optimization techniques

The problem of partitioning geometric meshes into load balanced subdomains can be formulated on the topological graph of the mesh. This is equivalent to disconnecting a graph into nearly equally sized subgraphs by cutting the minimum number of edges. This problem can be formulated as an optimization problem by defining an objective function whose minimum value corresponds to the optimal partition of the mesh. In order to satisfy the load balancing constraint among the subdomains, the objective function has two components; one which is minimized when there are no edges by the partition (minimum communication), and the other is minimized when an equal number of nodes is assigned to each subdomain. These two subobjectives tend to *compete* with one another in that the first goal is satisfied when the nodes are uniformly distributed across the specified number of subgraphs, while the second goal is met (trivially) by collapsing all the nodes into a single partition.

The simplest graph partitioning problem is the 2-way one, in which the graph nodes are divided between two partitions (subdomains), D_1 and D_2. This optimization problem can be mathematically stated as finding the minimum of the following objective function:

$$E(\mathbf{S}) = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j\neq i}}^{N} C_{ij}(1 - S_j)S_i \quad - r \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j\neq i}}^{N} (1 - S_j)S_i \tag{3.1}$$

where

$$S_i = \begin{cases} 0 & \text{if} \quad e_i \in D_1 \\ 1 & \text{if} \quad e_i \in D_2, \end{cases}$$

C denotes the adjacency of the graph G_M and r is a *repulsion* coefficient which determines the relative weight of the two competing sub-objectives. The formulation of the P-way partition problem involves a more complicated cost function

$$E(\mathbf{S}) = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j\neq i}}^{N} \sum_{k=1}^{P} C_{ij}(1 - S_{jk})S_{ik} \quad - r \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j\neq i}}^{N} \sum_{k=1}^{P} (1 - S_{jk})S_{ik} \tag{3.2}$$

where

$$S_{ik} = \begin{cases} 0 & \text{if} \quad e_i \notin D_k \\ 1 & \text{if} \quad e_i \in D_k. \end{cases}$$

Observe that the P state variables $S_{i1}, \ldots, S_{iP}$, are associated with each element, e_i, but only the one corresponding to the subdomain in which e_i is allocated can be non-zero. Another formulation is to consider the following cost function

$$E(\mathrm{S}) = \sum_{k=1}^{P} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} C_{ij}(1 - S_{jk})S_{ik} + r \sum_{k=1}^{P} \left(\sum_{i=1}^{N} S_{ik} \right)^2 \tag{3.3}$$

The second term is chosen to equalize the size of each subdomain. Its form implies that the minimum is obtained for equal values of $|D_k|$.

The domain decomposer has a library of various P-way algorithms for minimizing the above cost functions. It includes the well known Kernighan-Lin heuristic [Kern 70], [Chris 89] technique for minimizing the cut-cost of the mesh graphs, assuming the solution satisfies the constraints of the balanced partitioning. The idea of this approach is to identify an improved feasible solution by interchanging elements among the subdomains that optimize a profit function. We have developed a P-way partitioning algorithm with a modified profit function based on Kernighan-Lin's idea of selecting improved feasible solutions [Chris 89]. We refer to this as the *GGP* (geometric graph partitioning) algorithm. A recursive variation of this algorithm based on a modified 2-way Kernighan-Lin algorithm was also developed [Chri 89] and named *GGP-rec*; this heuristic is also called *orthogonal recursive bisection* [Fox 86], [Will 90]. These algorithms require an appropriate initial feasible solution, which can be selected by the user out of the set of predetermined initializations. Another approach for solving (3.1) is the eigenvalue method [Bopp 87]. It turns out that minimizing the cost function (3.1) over the subspace $L = \{\mathrm{S} \in \mathbf{R}^{\mathbf{N}} \colon \sum_{i=1}^{N} S_i = N/2\}$ is equivalent to determining the largest eigenvalue of $C + D$, where $D = \mathrm{diag}(r)$.

2.3 Stochastic optimization techniques

A significant advance in optimization was made in 1983 with the invention of simulated annealing (SA) [Kirk 83]. This technique models the optimization variables in a problem as if they were a collection of atoms slowly being cooled into a ground state, corresponding to the optimal solution problem. SA has the advantage over gradient descent techniques in that *thermal noise* can perturb the evolution of the solution to prevent it from becoming stuck in local minima. SA has been used in [Flow 88], [Fox 86] and [Will 90] to minimize the cost function (3.3). Hopfield neural networks [Hopf 76] constitute another avenue for solving discrete combinatorial problems. These networks involve many simple computing units (or artificial neurons) which objective is to minimize an energy function associated with the optimization problem. In [Hous 90] we consider various artificial neural networks (ANN) for solving (3.1). We are currently studying a new approach called mean field annealing to study the partitioning problem which is a combination of ANN and SA.

2.4 Parallel ELLPACK partitioning algorithms

In this section we list the partitioning algorithms which are under development in the *parallel ELLPACK* system [Hous 90d]. Their analysis and performance evaluation is reported in [Chri 89, 90, 91]. We have implemented four basic types of heuristics for partitioning FEM meshes. They include $1 \times P$ strips, $P \times Q$ lattices, and 2-way recursive bisection and P-way partitionings. The $1 \times P$-way partitions are obtained by sorting the x (or y or z) coordinates of the centroid of the elements and subdividing the sorted list in groups of NE/P elements. In this case, the assignment of the subdomains to an array of processors is the identity. We refer to this algorithm as GEO $1 \times P$. A $P \times Q$-way partitioning is obtained by sorting the coordinates of the element centroid in x and y directions. We refer to these techniques as GEO $P \times Q$. Variations of it have been proposed in [Prom 90], [Sada 88] and Simulog's system. Another important class of heuristics included in this library are the so-called orthogonal recursive bisection (ORB) techniques based on different 2-way partitioning heuristics. We have implemented the following techniques: ORB methods based on 2-way Kernighan-Lin (ORB_KL) [Kern 70], geometry graph partitioning (ORB_GGB), [Chri 89], Artificial Neural Networks (ORB_ANN) [Hous 90], the eigenvalue method (ORB_GE) [Bopp 87], simulated annealing (ORB_SA) [Flow 88], [Will 90], and inertia axis[1] or mass center of the mesh method [ORB_I] [Will 90]. Finally, the library includes two P-way heuristics based on CM-cluster and GGP heuristics. The implemented algorithms are listed in Table 1 with their acronyms and short descriptions.

[1]Unpublished manuscript, SIMULOG/INRIA, Sophia Antipolis, France.

Table 1: FEM partitioning algorithms

Name	Description
GEO $1 \times p$	1-D strips
GEO $p \times q$	2-D strips
ORB-E	Eigenvalue Ortho. Rec. Bisection [Bopp 87]
ORB-M	Mass Center ORB [Will 90]
ORB-I	Inertia Axis ORB[1]
ORB-KL	Kernighan-Lin ORB [Kern 77]
ORB-GGP	Modified K-L ORB [Chri 89]
ORB-ANN	Neural Net [Hous 90]
ORB-SA	Simulated Anealing
GGP	P-way Geometric Graph Part [Chri 89]
SA	P-way SA [Will 90]
CM-Clustering	Cuthill-Mckee [Fahr 88]

3 Domain Decomposition Allocation Strategies

In this section we address the problem of assigning processors to subdomains. For this purpose we let $G_A(V_A, E_A)$ to represent the interconnection graph of the architecture. Recall that the topological graph of the mesh in G_M (V_M, E_M). Without loss of generality, we can assume that the size of the two graphs is the same ($|V_M| = |V_A|$). For general topological graphs the graph allocation problem is equivalent to the minimization of one of (2.3) or the cost function

$$\max_{i,j} c(D_i, D_j) d(m^{-1}(D_i), m^{-1}(D_j)) \tag{3.1}$$

where m is the mapping, c is the adjacency matrix of the graph G_M and d is the distance (shortest path) of the two processors in the graph G_A. One can replace c in (3.1) by the interface length between the two subdomains D_i, D_j or the communication requirements between them. The choice of the cost function depends on the programming type (asynchronous, synchronous) [Fahr 89] and the selection of the coefficient c is influenced by the level of abstraction assumed in the solution of the mapping problem. Some rather crude solutions to the assignment problem employ a projection of the G_A graph to well-known, simpler graphs I_A (i.e., array, mesh). In these cases, the allocation is usually part of the partitioning process due to the simplicity of the machine interconnection graph. A rather general approach is to project both G_M and G_A graphs to the same space (i.e., Euclidean space) and solve the assignment problem for the projected graphs. Some projection techniques to Euclidean space are described in [Fuku 84], [Chri 90]. Figure 1 depicts such projections for a semi-annulus discretized domain. Here the assignment problem is determined by minimizing the following cost function

$$\min_m \sum_i \| (x_i^M, y_i^M) - (x_{m^{-1}(i)}^A, y_{m^{-1}(i)}^A) \|_2^k \qquad (3.2)$$

where $k = 1, 2$ and (x_i^M, y_i^M), (x_i^A, y_i^A) are the coordinates of the graph nodes (G_M, G_A) projected in Euclidean space.

We have implemented several techniques for solving the above minimization problems. They are classified as *explicit*, *implicit* or *naive*. The naive approaches include the RANDOM algorithm where the assignment is done at random and the SHIFT algorithm which assigns each subdomain D_i to processor $(i-1)$ of the G_A graph. For the explicit solution of the optimization problem (2.3), we have used and tested several algorithms [Hana 72], [Weil 71], [Carp 80], and [West 83], the one selected for parallel ELLPACK is called EXPLICIT_H. In this heuristic $c(D_i, D_j)$ models the interface length of the two subdomains. We use two implicit approaches based on subdomain exchange among processors and greedy procedures to achieve goals (2.3) or (3.1) [Goto 81], [Chri 90]. We refer to these algorithms as *SUBD-EXCH* and *GREEDY*, respectively.

Table 2 summarizes the allocation algorithms we have implemented in the Domain Decomposer. The analysis and performance of these algorithms is reported in [Chris 90, 91]. Stochastic optimization techniques for the allocation problem have been considered explicitly or implicitly by several authors. A review of these methodologies is presented in [Erca 89].

4 Domain Decomposer

We have built an interactive environment called *DecTool* (short for Domain Decomposer Tool) to help with domain decomposition. An example display is shown in Figure 2. DecTool provides facilities for both automatic (using predefined algorithms), and manual decomposition of a given 2-D or 3-D discrete domain. This interactive environment is written using the 4th release of the X11 toolkit known as ATHENA Widgets. DecTool consists of

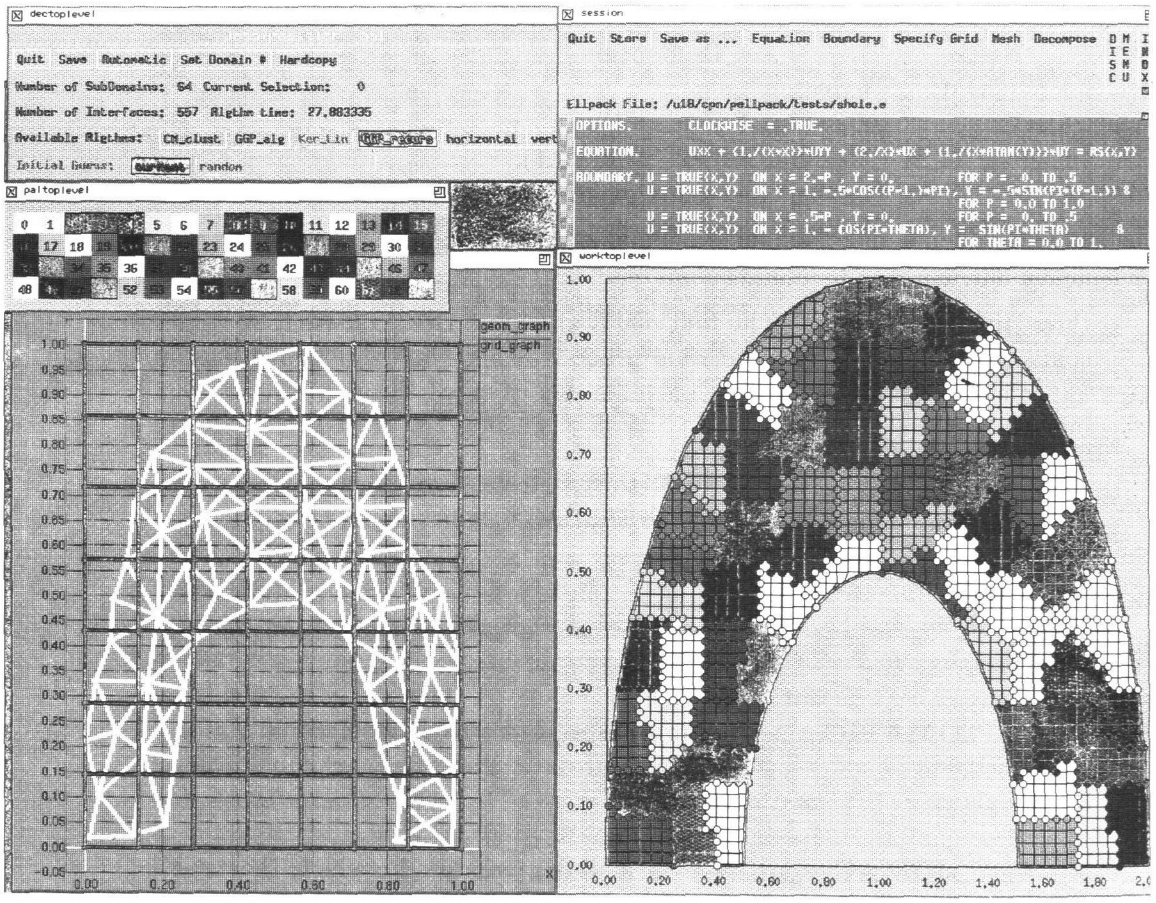
Table 2: Domain decomposition allocation strategies

Name	Description
RANDOM	naive
SHIFT	$D_i \rightarrow i-1$ processor
EXPLICIT_H	Munkres algorithm for (3.1)
SUBD_EXCH	implicit algorithm for (3.1) or (3.2)
GREEDY	implicit algorithm for (3.1) or (3.2)

three different windows. The first one is the basic DecTool window, which controls the domain decomposition process. This window is shown in the upper left corner of Figure 2. Control is implemented through a set of four buttons.

QUIT:	Signals to exit from the tool and return to parallel ELLPACK environment (Fig. 3).
SAVE:	An output file is produced which contains the description of the last decomposition of the domain.
AUTOMATIC:	Invokes a specified automatic decomposition algorithm from a library of available algorithms.
SET DOMAIN #:	Invokes a dialog window in which the user specifies the number of subdomains (processors).

In the basic window, there are three additional widgets for invoking the library decomposition techniques and specifying the appropriate initializations. Furthermore, this window displays the interface length of the generated automatic decomposition and decomposition execution time in seconds. The decompositions are displayed and manipulated in another window. Each subdomain is colored differently, the interface nodes are displayed as colored circles or squares. The colors indicate the assignment of subdomains (processes) to processors with a color map (color palette) displayed in a different window. The user can modify an automatic decom-


dectoplevel
Quit Save Automatic Set Domain # Hardcopy
paltoplevel
geom_graph
grid_graph
session
Quit Store Save as ... Equation Boundary Specify Grid Mesh Decompose
worktoplevel

```
Number of SubDomains: 64 Current Selection: 0
Number of Interfaces: 557 Algthm time: 27.883335
Available Algthms: CM_clust GGP_alg Ker_Lin [✓ GGP_recurs] horizontal vert
Initial Guess: [✓ current] random
```

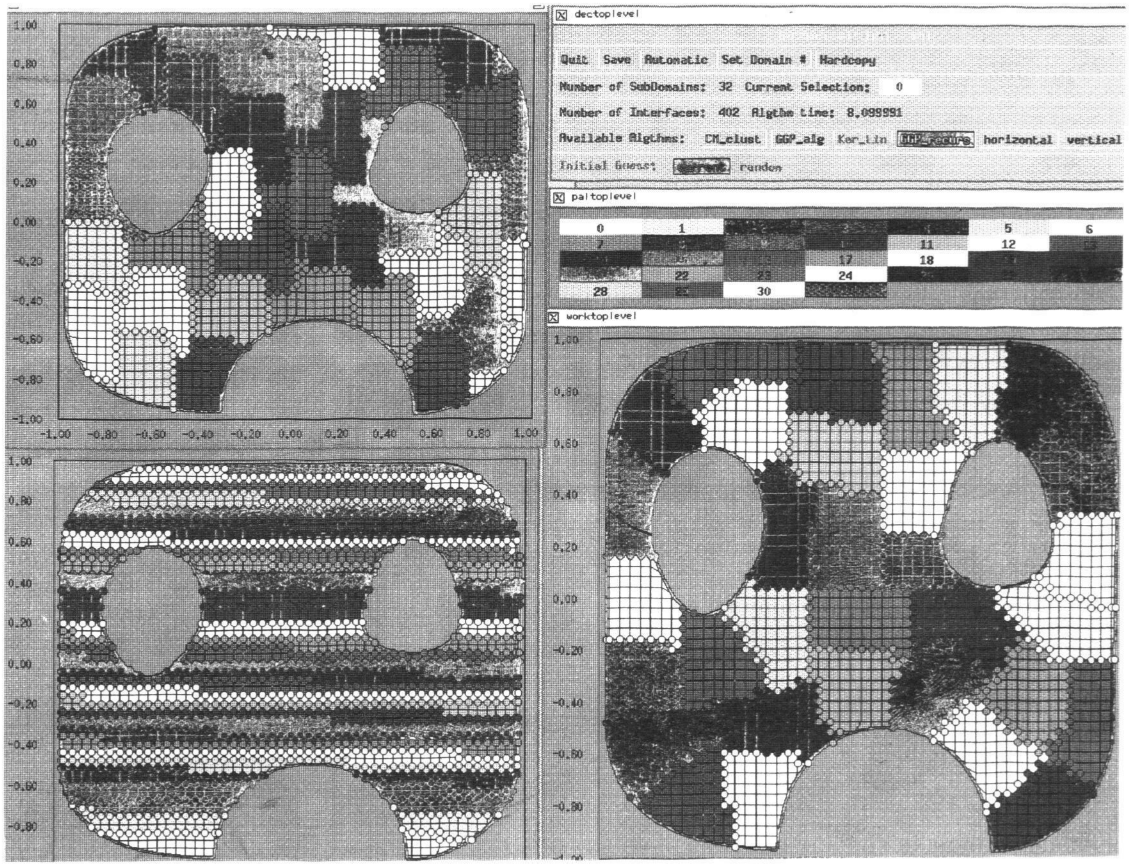
```
Ellpack File: /u18/cpn/pellpack/tests/shole.e
OPTIONS.          CLOCKWISE  = .TRUE.

EQUATION.         UXX + (1./(X*X))*UYY + (2./X)*UX + (1./(X*ATAN(Y)))*UY = RS(X,Y)

BOUNDARY. U = TRUE(X,Y)  ON X = 2.-P , Y = 0.             FOR P =  0. TO .5
          U = TRUE(X,Y)  ON X = 1. -.5*COS((P-1.)*PI), Y = -.5*SIN(PI*(P-1.)) &
                                                          FOR P = 0.0 TO 1.0
          U = TRUE(X,Y)  ON X = .5-P , Y = 0.             FOR P =  0. TO .5
          U = TRUE(X,Y)  ON X = 1. - COS(PI*THETA), Y =  SIN(PI*THETA)       &
                                                          FOR THETA = 0.0 TO 1.
```

Figure 1: Displays a decomposition of a semi-annulus domain, together with the projected G_M and G_A graphs into Euclidean space.

position by clicking the left button on a specific color in the palette and an element of the mesh. By holding down the middle button of the mouse, an entire set of elements or interface nodes can be recolored. These mouse operations can also be used to construct a decomposition manually.



Number of SubDomains: 32 Current Selection: 0

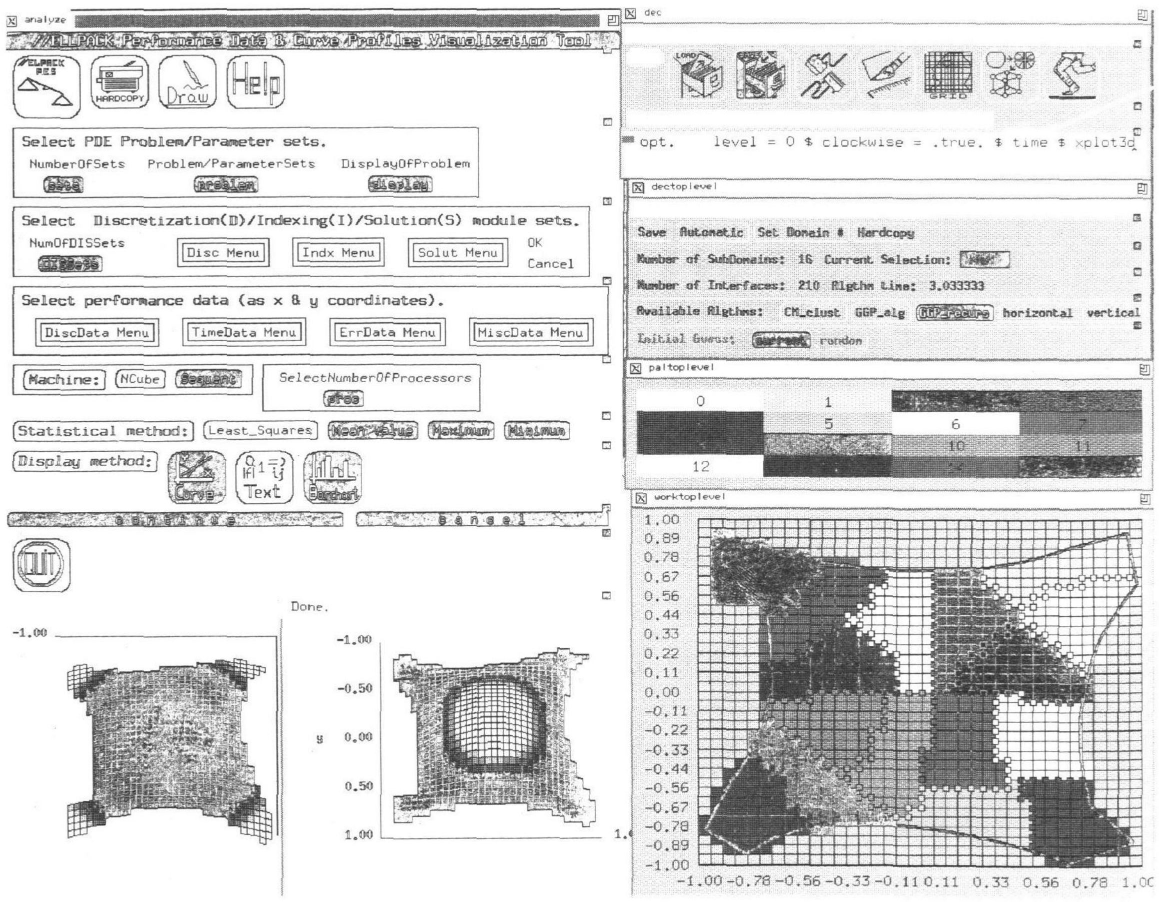
Number of Interfaces: 402 Algthm time: 8.099991

Available Algthms: CM_clust GGP_alg Ker_Lin [✓ GGP_recurs] horizontal vertical

Initial Guess: [✓ current] random

Figure 2: Example of the geometry decomposition tool *DecTool* for a domain with holes and 32 subdomains. The two control windows are on the upper right side. The three other windows from top left to bottom right display the CM_cluster, GEO $1 \times P$ and ORB_GGP 32-way partitionings.

In the case of 3-D meshes, the tool will be able to display nodal assignments to subdomains obtained by the heuristics of Tables 1 and 2. We are currently developing the third version of this tool whose interface will facilitate the mapping algorithms listed in Table 2. Furthermore it will generate additional per:ormance data (i.e imbalance measurements, degree of the computational g-aph, e.t.c) and will use models to predict the speedup of the underlying computation [Chri 91].



```
opt.      level = 0 $ clockwise = .true. $ time $ xplot3d
```

```
Number of SubDomains: 16  Current Selection:
Number of Interfaces: 210  Algthm time: 3.033333
```

Figure 3: An instance of the *Parallel Ellpack* Environment

References

[Ayka 88] C. Aykanat, F. Ozgu, F. Ercal and P. Sadayappan, Iterative algorithms for solution of large sparse systems of linear equations on hypercubes, IEEE Transactions on Computers, vol. 37, no. 12, 1554–1568, 1988.

[Bopp 87] R.B. Boppana, Eigenvalues and graph bisection: An average case analysis, In the 28th Annual Symp. Foundation, Computer Science, 280–285, 1987.

[Carp 80] G. Carpaneto and P. Toth, Solution of the assignment problem, ACM Trans. on Database Software, Vol. 6, No 1, 104–111, March 1980

[Chri 89] N.P. Chrisochoides, C.E. Houstis, E.N.Houstis, S.K. Kortesis and J.R. Rice, Automatic load balanced partitioning strategies for PDE computations, Proceedings of International Conference on Supercomputing (E.N. Houstis and D. Gannon, eds.), Iraklion-Greece, ACM publications, 99–107, 1989.

[Chri 90] N.P. Chrisochoides, C.E. Houstis, E.N. Houstis, Geometry based mapping strategies for PDE computation, CER-90-16, Computer Science Department, Purdue University, West Lafayette, IN 47907.

[Chri 91] N.P. Chrisochoides, On the mapping of PDE computations to distributed memory machines, Purdue University Ph.D. Thesis, 1991.

[Fhar 88] C. Farhat, A simple and efficient automatic FEM domain decomposer, Computers and Structures, vol. 28, 579–602, 1988.

[Fhar 89] C. Farhat, On the mapping of massively parallel processors onto finite element graphs, Computers and Structures, vol. 32, no. 2, 347–353, 1989.

[Flow 88] J. Flower, S. Otto and M. Salana, Optimal mapping of irregular finite element domains to parallel processors, In Parallel Computers and Their Impact on Mechanics, AMD vol 86, (Ed. A.K. Noor), The American Society of Mech. Engineers, New York, 239–250, 1988.

[Fox 86a] G.C. Fox, A graphical approach to load balancing and sparse matrix vector multiplication on the hypercube, Proceedings of IMA initiate, (Ed. M. Schultz), Springer-Verlag, 37–51, 1986.

[Fox 86b] G.C. Fox, A review of automatic load balancing and decomposition methods for the hypercube, Proceedings of the IMA Institute, (Ed. M. Schultz), Springer-Verlag, 63–76, 1986.

[Fuku 84] Kunio Fukunaga, Shoichiro Yamada, Harold S. Stone , and Tamotsu Kasai " A presentation of hypergraphs in the Euclidean space." IEEE Trans. Vol. C- 33 No. 4, 364–366, April 1984

[Goto 81] Satoshi Goto " An efficient algorithm for the two-dimensional placem ent prob lem in electrical circuit layout" IEEE Trans. on Circuits and Systems, Vol. CAS-28 Jan., 12–18, 1981

[Hana 72] M. Hanan and J. M. Kurtzberg, A review of the Placement and Quadratic Assignment Problems, SIAM Review, Vol 14, No 2, 324–342, April 1972

[Hopf 76] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci., USA, vol. 79, 2554–2558, 1982.

[Hous 90a] C.E. Houstis, Module allocation of real-time applications to distributed systems, IEEE Trans. on Software Engineering, vol. 6, no. 7, July 1990, pp. 699–709.

[Hous 90b] 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 application/architecture pairs, in *Intelligent Mathematical Software Systems*, North-Holland, 87–101, 1990

[Hous 90c] E.N. Houstis, S.K. Kortesis and H. Byun, A workload partitioning strategy for PDEs by a generalized neural network, CSD-TR-934, Computer Science Department, Purdue University, W. Lafayette, IN, May 1990.

[Hous 87] C.E. Houstis, E.N. Houstis and J.R. Rice, Partitioning PDE computations: Methods and performance evaluation, Parallel Computing 5, 141–163, 1987.

[Hous 90d] E.N. Houstis, J.R. Rice, N.P. Chrisochoides, H.C. Karathanasis, P.N. Papachiou, M.K. Samartzis, E.A. Vavalis, Ko Yang Wang and S. Weerawarana, //ELLPACK: A numerical simulation programming environment for parallel MIMD machines, In: *Supercomputing* 90, (Ed. J. Sopka), ACM Press, 97–107, 1990.

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

[Kirk 83] S. Kirkpatrick, C. Gelatt and M. Vecchi, Optimization by simulated annealing, *Science*, vol. 220, no. 4598, 671–680, 1983.

[Mari 88] L.D. Marini and A. Quarteroni, An iterative procedure for Domain Decomposition Methods : A finite element approach, First International Symposium on Domain Decomposition Methods for Partial Differential Equations. (ed's R. Glowiniski, G.H. Golub, G.A. Meurant, and J. Periaux), SIAM, 129–143, 1988.

[Morr 87] R. Morrison and S. Otto, The scattered decomposition for finite elements, Journal of Scientific Computing, vol. 2, no. 1, 59–76, 1987.

[Pomm 90] C. Pommerell, M. Annaratone and W. Fichtner, A set of new mapping and coloring heuristics for distributed-memory parallel processors, Proceedings of Copper Mountain Conference on Iterative Methods, (Ed. T.M. Manteuffel), Vol. 414, 1–27, 1990.

[Sada 87a] P. Sadayappan and F. Ercal, Cluster-partitioning approaches to mapping parallel programs onto a hypercube, Proceedings of International Conference on Supercomputing (E. Houstis, T.S. Papatheodorou and C. Polychronopoulos, eds.), Athens-Greece, Springer-Verlag, 476–497, 1987.

[Sada 87b] P. Sadayappan and F. Ercal, Nearest-neighbor mappings of finite element graphs onto processor meshes, IEEE transactions on computers, vol. 36, no. 12, 1408–1420, 1987.

[Weil 71] R. Weil An Extension of the Munkres Algorithm for the Assignment Problem to Rectangular Matrices, Communications of ACM, Vol. 14, 802–804 1971

[West 83] David H. West Approximate Solution of the Quadratic Assignment Problem, ACM Trans. on Mathemat ical Software, Vol. 9, No 4, 461–466, 1983

[Will 90] R.D. Williams, Performance of dynamic load balancing algorithms for unstructured mesh calculations, Concurrent Supercomputing Facility, Cal Tech unpublished manuscripts, 23 pages, 1990.