

Parallel ELLPACK Elliptic PDE Solvers

E.N. Houstis, S.B. Kim, S. Markus, P. Wu, N.E. Houstis, A.C. Catlin, S. Weerawarana *

Department of Computer Sciences

Purdue University

West Lafayette, IN 47907, USA.

T. S. Papatheodorou

CTI

Greece.

March 27, 1997

Abstract

Parallel ELLPACK [35, 61] is a machine independent problem solving environment (PSE) that supports PDE (partial differential equations) computing across many hardware platforms. In this paper we review parallel methodologies based on the “divide and conquer” computational paradigm and their infrastructure for solving general elliptic PDEs. Particularly, we describe those that have been implemented and tested in the parallel ELLPACK PSE. Moreover, we describe two parallel frameworks that allow the *reuse* of the discretization part of the sequential elliptic PDE solvers. Numerical results indicate the effectiveness of the reuse frameworks implemented.

*This work was supported by NSF grants 9123502-CDA and 92022536-CCR, 620-92-J-0069 and ARPA grant DAAH04-94-G-0010.

Contents

1	Introduction	3
2	General Elliptic PDE Solvers	3
2.1	Parallelization Methodologies for “Legacy” Elliptic PDE Software	3
2.1.1	An Off-Line Reuse Parallel Methodology for “Legacy” PDE Software	4
2.1.2	A Parallel Framework for Building New PDE Software	6
2.1.3	An On-Line Reuse Parallel Methodology for “Legacy” PDE Software	6
3	Parallel Linear Algebraic Solvers for PDE Equations	7
4	The Performance of Three Parallel ELLPACK Solution Frameworks	7
5	Discrete Geometric Data Partitioning Strategies	8
5.1	General Mesh Partitioning Heuristics	9
5.2	Fast Grid/Mesh Partitioning Heuristics	10

1 Introduction

Computational models based on partial differential equation (PDE) mathematical models have been applied successfully over the last 50 years to study many physical phenomena and design a variety of artifacts. The overall quantitative and qualitative accuracy of these computational models in representing the physical situations or artifacts that they are supposed to simulate, depends very much on the computer resources available. The recent advances in high performance computing technologies have provided an opportunity to speed up significantly these computational models and increase dramatically their numerical resolution and complexity. The purpose of this paper is a) to review the various parallelization techniques proposed to speed up the existing computational PDE models, that are based on the “divide and conquer” computational paradigm and involve some form of decomposition of the geometric or algebraic data-structures associated with these computations and b) to formulate parallel methodologies that are capable of *reusing* parts of the sequential PDE solvers. For simplicity of this exposition we focus on computational models derived from elliptic PDE models and implemented in the Parallel ELLPACK PSE. Most of the parallelization techniques presented here are applicable to general semi-discrete and steady-state models. Specifically, we consider PDE models consisting of a PDE equation ($\mathbf{L}\mathbf{u} = \mathbf{f}$), defined on some region and subject to some auxiliary condition ($\mathbf{B}\mathbf{u} = \mathbf{g}$) on the boundary of ($\Omega (= \partial\Omega)$). It appears that one can formulate many (thousands) computational models to simulate the above general mathematical model, depending on the approximation technique selected to discretize the domain of definition, the PDE equation and boundary conditions, or the PDE approximate solution. In this article, we have selected to review parallel computational models based on the most popular discretization techniques, such as finite difference approximations of \mathbf{L} and \mathbf{B} and piecewise polynomial (finite element) approximation of the solution \mathbf{u} . In the parallel computational models considered, the continuous PDE problem is reduced to a distributed sparse system of linear equations. Depending on the type of the PDE operators \mathbf{L} and \mathbf{B} and the simplicity/regularity of the PDE region, the corresponding finite difference or finite element system of equations can be solved by general or rapid parallel algebraic solvers. Following, we discuss the various proposed parallel methodologies for the implementation of these two classes of PDE solvers on a virtual parallel machine environment. We have selected to cite the associated research results that have already led to some parallel implementations that are available, or very close to appear, in the form of commercial or public domain software.

2 General Elliptic PDE Solvers

The plethora of numerical elliptic PDE solvers can be distinguished and classified by the levels of grid(s)/mesh(es) used to approximate the continuous PDE model (i.e. **single** or **multi-level**), the refinement of the grid as a function of the discretization error in an intermediate computed solution (i.e. **static** or **dynamic (adaptive)**), and the implementation structure of the PDE software (i.e. **multi-segment** or **single-segment**). In this article we have selected to review the parallelization techniques proposed for single-level grid elliptic PDE solvers implemented in multi-segment (**general case**) and single-segment form (**rapid case**). Some of the parallelization approaches presented here are easily applicable to multi-level elliptic PDE solvers. An overview of parallel multi-level methods can be found in [8], [9], [48]. The parallelization of adaptive elliptic PDE solvers is a much harder problem. A discussion of the issues and results related to parallel adaptive techniques for elliptic, parabolic and hyperbolic problem can be found in [22] and [23]. The following discussion is focused on parallelization techniques that allow both to reuse existing (“**legacy**”) PDE software parts and provide a *template* or *framework* to build new parallel PDE software.

2.1 Parallelization Methodologies for “Legacy” Elliptic PDE Software

There is significant state-of-the-art “legacy” software for elliptic and parabolic PDEs. It represents hundreds of man years of effort which will be unrealistic to expect to be transformed by “hand” (in the absence of parallelizing compilers) on some virtual or physical parallel environment. The “legacy”

<i>Name</i>	<i>Reference</i>
ELLPACK	[52]
//ELLPACK	[34, 35]
FIDISOL	[55]
VECFEM	[30]
CADSOL	[54]
PDEONE	[60]
PDECOL	[43]
PDE TWO	[59]
MGGHAT	[49]

Table 1: PDEpack: Public domain “legacy” PDE software

software can be classified in two large classes. The first class contains customized PDE software for specific applications. An example of a such software is PICES. This software is usually difficult to be adopted for the simulation of an alternative application. The second class contains PDE software that supports the numerical solution of well defined mathematical models which can be used easily to support the simulation of multiple applications. The first class tends to be application domain specific, thus the parallelization efforts and results appear in many diverse sources. In this article we review the parallelization techniques proposed for the second class of PDE software. Table 1 lists some of the public domain “legacy” software that is available in the **parallel ELLPACK** system [36]. It’s worth reminding the reader that the majority of the code of each PDE system is implementing the geometric and the PDE model discretization phases. This tends to be the most knowledge intensive part of the code. The rest of the code deals with the solution of the discrete finite difference or finite element equations. This phase is well understood and many alternative solution paths exist. We review those efforts that have already been implemented in the form of software. In Table 2 we summarize the above observations and in its last column we estimate the parallelization effort needed to convert or re-implement the components of the “legacy” PDE code into some parallel environment “by hand” From this analysis, it is clear that any parallel methodology that attempts to reuse the PDE discretization software parts is well justified. Following, we describe three parallel methodologies that are based on some “optimal” partitioning of the discrete PDE geometric data structures (i.e. grids and meshes). Figure 1 depicts these three decompositions approaches for a two dimensional region and message passing computational paradigm. The two left most paths in Figure 1 depict methodologies that support the reuse requirement. The third path provides a framework to develop new customized parallel code for the discretization part of the PDE computation. All three approaches assume the availability of parallel linear solvers implemented on distributed algebraic data structures obtained through an “optimal” partitioning of the corresponding PDE geometric data structures. In this article we have selected to review compatible linear parallel solvers that correspond to a parallel implementation of existing sequential counterpart on the assumed distributed data structures. Next, we elaborate on these approaches and indicate the required infrastructure.

2.1.1 An Off-Line Reuse Parallel Methodology for “Legacy” PDE Software

Figure 1a depicts an off-line approach, referred to as M^+ , which assumes that the discretization of the PDE model is realized by an existing sequential “legacy” PDE code, while it goes off-line to a parallel machine to solve the system of discrete equations. For the parallel solution of the discrete PDE equations, a decomposition of the sequentially produced algebraic system is required. It can be either *implicitly* obtained through a decomposition of the mesh or grid data or *explicitly* specified by the user. Then, the partitioning system is down-loaded on the parallel machine. This is the most widely used methodology, since it allows for the preservation of the most knowledge intensive part of the code and for speeding

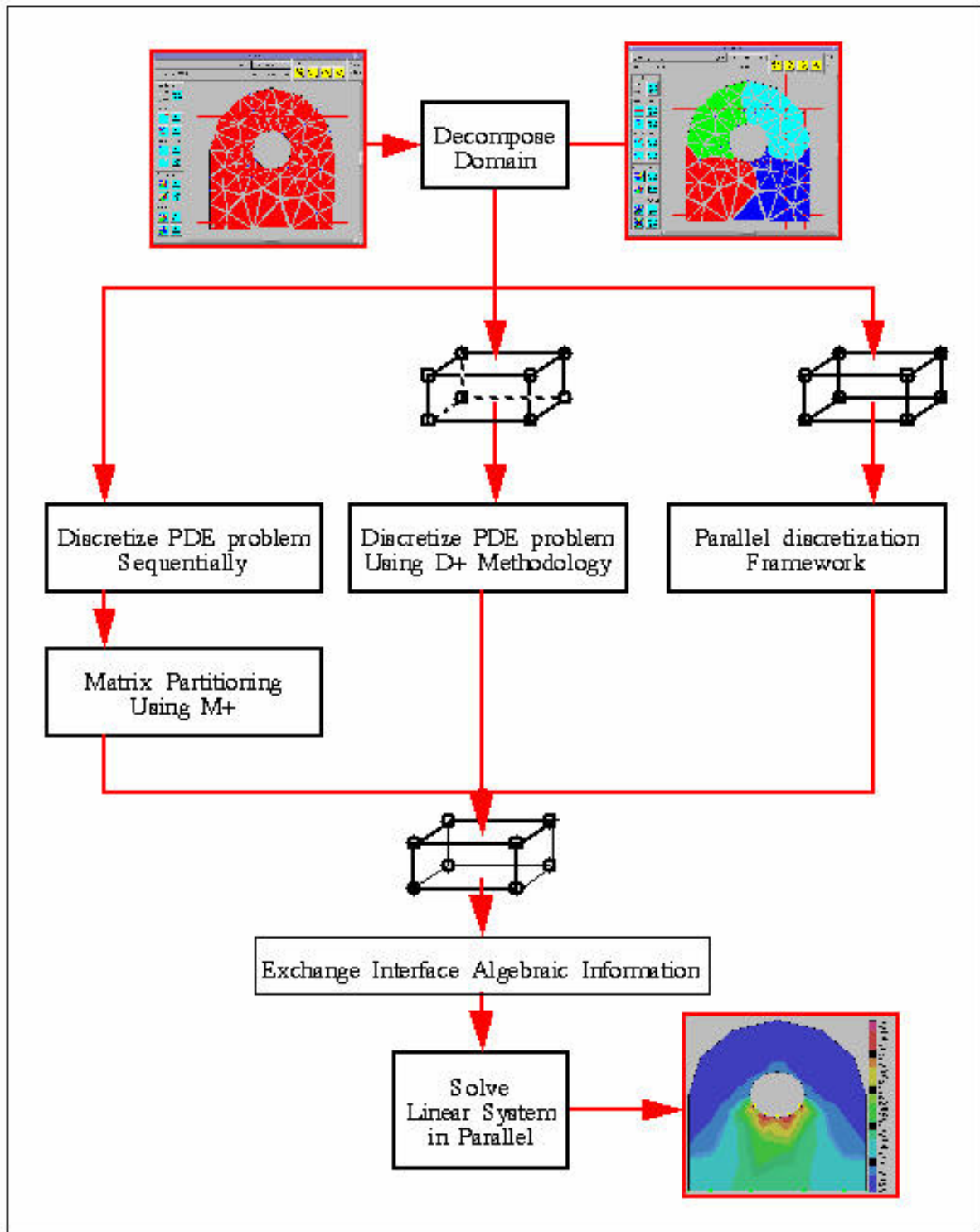


Figure 1: Three domain decomposition based parallel methodologies for elliptic PDEs. The left path (a) depicts an off-line parallel approach for solving the sequentially generated PDE equations, the center path (b) depicts an on-line non-overlapping domain decomposition approach capable of reusing existing discretization PDE software, and right path (c) depicts a framework for developing new parallel PDE software.

<i>Components</i>	<i>Computational Intensity</i>	<i>Knowledge Intensity</i>	<i>Parallelization effort</i>
Geometric discretization	$O(N)$	Very High	Significant
PDE model discretization	$O(N)$	Very high	Significant
Solution	$O(N^\alpha), 1 < \alpha \leq 3$	Well understood - High	Relative easy
Solution graphical display	$O(N)$	High	Specialized hardware

Table 2: The complexity of the elliptic PDE software parts and an estimate of the parallelization effort needed to implement them in some parallel environment, where N denotes the size of the discrete problem

up the most computationally intensive one. The obvious disadvantage of this approach is the memory bottleneck of the sequential server. To address this problem various off-line pipeline techniques have been proposed. The current version of the parallel ELLPACK system includes a software tool to support this methodology for the “legacy” software listed in Table 2. The tool is self contained and can be used for any PDE software and virtual parallel machines supported by standards such as MPI [29]. The input to this tool consists of the linear system and a partitioning of the associated matrix. The partitioning of the matrix problem can be obtained either explicitly by decomposing the matrix graph or implicitly by decomposing the discrete geometric data (i.e. mesh or grid). A comprehensive overview of the explicit matrix partitioning techniques and their performance evaluation can be found in [15]. Earlier results on the mapping of matrix system computations to parallel machines are reported in [27] and [3]. In section 4, we review a number of geometry partitioning strategies used to implicitly decompose the PDE matrix problem.

2.1.2 A Parallel Framework for Building New PDE Software

Figure 1c corresponds to a framework for developing customized PDE software. It is defined by a set of pre-defined decomposed geometric and algebraic data structures and their interfaces. The decomposition of the PDE data structures is chosen so that the underlying computations are uniformly distributed among processors and the interface length is minimum. Later, we review the proposed geometric and matrix decompositions to support this framework. This parallel framework has been used by many researchers to implement PDE based applications [1], [20], [31], [46], [50], [58]. Also, this framework was used for developing general PDE software [36]. The parallel PDE solvers implemented on the above framework are distinguished primarily by the way they handle the interface equations and unknowns. An overview of the various parallel solution strategies proposed for handling the interface and interior equations can be found in [18]. The simplest of these parallel strategies calls for the implementation of efficient sequential algebraic solvers on the framework data structures through the use of parallel sparse BLAS [11] that employ message passing primitives to exchange or accumulate interface quantities and carry out matrix-vector and vector-vector operations. The advantage of this approach is the fact that no new theory is required. Such parallel PDE solvers based on certain instances of finite difference and finite element schemes for elliptic PDEs can be found in //ELLPACK system [36], [35]. These PDE solvers are described in [40] together with their performance evaluation. This study indicates that they can deliver significant speedups even for moderate size problems.

2.1.3 An On-Line Reuse Parallel Methodology for “Legacy” PDE Software

In Figure 1b we illustrate a third methodology for developing parallel PDE software that supports the reuse of existing PDE codes and attempts to address the shortcomings of the previous two. It is referred to as D^+ . The basic idea of this approach is to use the mesh/grid decomposition to define a number of auxiliary PDE problems that can be discretized independently using the “legacy” PDE codes. Depending on the PDE operator and the approximation scheme used appropriate continuous *interface conditions* must be selected to guarantee that the parallel on-line generated system of equations is complete and equivalent (apart from round-off error) to the sequential discrete algebraic system. In some instances

a data exchange among processors might be required to complete the system of algebraic equations. A software environment that supports the D^+ approach for elliptic PDEs is available in //ELLPACK system.

3 Parallel Linear Algebraic Solvers for PDE Equations

All three parallel methodologies depicted in Figure 1, assume the existence of efficient parallel linear sparse solvers implemented on a set of distributed algebraic data-structures. It appears that there are many alternative parallel solvers that have been proposed and studied in the literature. Their detail exposition is beyond the scope of this article. Some review articles already exist on this subject. Instead, we discuss and reference those that are already available in the form of software and have been tested for the parallel solution of PDE equations. One class of such solvers consists of the classical stationary iterative methods. A software system realizing these solvers on portable message passing interface for solving sparse systems arising from finite element and difference approximations is the //ITPACK [39]. The system consists of seven modules implementing SOR, Jacobi-CG, Jacobi-SI, RSCG, RSSI, SSOR-CG and SSOR-SI under different indexing schemes [41] and it is integrated in the //ELLPACK system. The code is based on the sequential version of ITPACK which was parallelized by utilizing a subset of sparse BLAS routines [40]. The interfaces of the parallel modules, the assumed data structures and its performance on several MIMD machines are presented in [40]. The system has been proven to be very efficient for elliptic PDEs. Another class of iterative solvers are based on preconditioning conjugate gradient (PCG) method. Several realizations exist in public domain. One such system is the MPPCGPACK. Its modules are described in [56] together with their performance evaluation on 1024 nCUBE II machine. The software is commercially available through *Scientific Associates Inc.* It contains parallel PCG based solvers for symmetric and non-symmetric systems for MIMD machines. The system is integrated in //ELLPACK. Another PCG based software system for solving systems of sparse linear algebraic equations methods on a variety of computer architectures is reported in [37]. This software is designed to give high performance with nearly identical user interface across different scalar, vector and parallel platforms as well as across different programming models such as shared memory, data parallel and message passing programming interfaces. A template for the implementation of PCG methods on a variety of machines is realized by the PIM 1.1 software system [16]. In this template, the user is expected to customize the basic matrix-vector and vector-vector operations needed in the algorithm on the intended target parallel environment.

4 The Performance of Three Parallel ELLPACK Solution Frameworks

In this section we attempt to estimate the overhead of the three parallel frameworks depicted in Figure 1. For this we have selected the following elliptic PDE problem.

The temperature distribution on a two dimensional slice of a reactor with a steel dome and concrete base (Figure 2) is computed. The inside surface of the dome is initially 450°K and the ambient temperature around the reactor is 80°K. It is assumed that no heat is lost through the bottom surface of the dome or base. Since the problem is symmetric, we consider only the right half of the slice. We want to find the temperature T such that,

$$\nabla \cdot k \nabla T = 0, x \in \Omega$$

Here k is the thermal conductivity, $k = 30.62$ in Ω_1 (steel) and $k = 0.79$ in Ω_2 (concrete). The boundary conditions for the interior and exterior surfaces are as follows:

$$\begin{aligned} \nabla T \cdot \mathbf{n} &= 0 \\ T &= 450 \\ -k \nabla T \cdot \mathbf{n} &= 0.7(T - 80) \end{aligned}$$

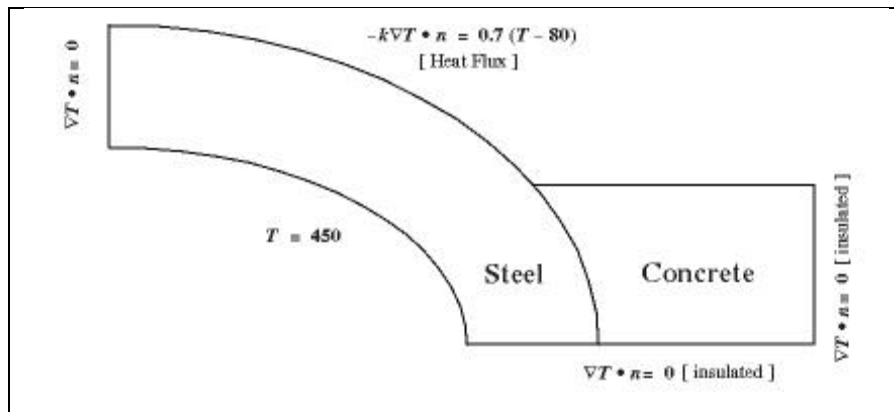


Figure 2: Steady-state heat diffusion in a reactor

Reuse Approach	Machine Configuration					
	1	2	4	8	16	32
D+ (on-line)	1.03	1.03	1.03	1.04	1.05	1.06
M+ (off-line)	1.03	2.03	3.96	7.63	14.20	26.77

Table 3: The discretization time ratios of the "on-line" and "off-line" reuse BiLinFEM implementations with respect to customized parallel BiLinFEM for different machine configurations.

where \mathbf{n} is the exterior unit outward normal to $\partial\Omega$.

This problem was discretized by a linear finite element method (FEM) based on triangular meshes and the system of the discrete equations was solved by JACOBI-CG parallel ITPACK solver [39]. Table 1 shows the ratios of the discretization D^+ and M^+ times with respect to the customized parallel linear FEM method (BiLinFEM).

The data in Table 1 suggest that the D^+ BiLinFEM reuse approach is as good as the customized one. In the case of M^+ reuse BiLinFEM, it is clear that the overhead depends on the performance of the off-line computational server used. Its obvious advantage is that it requires zero-programming.

5 Discrete Geometric Data Partitioning Strategies

The parallel methodologies considered in this article are based on some decomposition of the PDE discrete geometric data (i.e. grid or mesh). Without loss of generality, we discuss the various proposed decomposition strategies in terms of finite element meshes. The conversion to the grid case is straight forward. The formulation and implementation of this phase of the proposed parallel methodologies is often done at the topological graph of the finite element mesh $G = (V, E)$ of a domain Ω , where V denotes the set of elements or nodes and E is the set of edges of G that represent the connectivity of the vertices V with its neighbors. Throughout, we denote by d_v the number of adjacent vertices to each vertex v of V , $D = \{D_i\}$ the partitioning of G or domain Ω , N_s the set of subdomains in D , and $|V| = N_n$ or N_e the size of the mesh or graph G . The mesh decomposition is usually defined in terms of several optimality criteria [14], [18], [63]. They include *load balancing* (the subdomains are of almost equal size), *minimum interface length* (minimum number of common edges or nodes between subdomains), *minimum subdomain connectivity* (minimum number of neighbor subdomains), *minimum bandwidth of the local matrix problem*, and *optimal aspect ratios of the subdomains* (local matrix problem well conditioned). The problem of graph partitioning based only on the first two criteria has been extensively studied and found to be "hard". Thus, most of the proposed partitioning strategies are approximate (i.e. heuristic) in nature. These heuristics have been found to be very costly even for moderate size PDE problems [40]. Two "fast" alternative strategies have been formulated for grid [40] and mesh [63] respectively which we

review later.

5.1 General Mesh Partitioning Heuristics

First, we discuss a set of well known and tested heuristics for the automatic partitioning of meshes subject to the above listed optimality criteria and review some software tools available to assist the PDE geometric data decomposition.

Neighborhood Search Schemes The first class consists of heuristics that are based on some neighborhood search scheme utilizing the connectivity information of the mesh graph G . For these schemes, the partitioning of G is equivalent to the construction of a traversal tree from graph G . Two well known neighborhood search schemes are based on *depth-first search* and *breadth-first search* [4]. If the traversal order scheme remains fixed for the entire mesh graph G , then the searching strategy is called *strip-wise*. In case the traversal order is allowed to change after the formulation of each subdomain D_i , then the search is called *domain-wise* [17]. The optimality of these searching strategies depends on the starting vertex. It is usually selected as the one with minimum degree of connectivity that usually coincides with a boundary node or element. It has been observed that this selection effects the *bandwidth* (w) of the coefficient of the associated finite element matrix. Moreover, it has been shown [19] that the maximum partitioning interface C is given by the relation $C = (N_s * w) / N_n$. Another set of neighborhood search heuristics are the ones used for bandwidth reduction of a matrix. They have been used by several researchers [26], [28], [45], [51] to solve the mesh partitioning problem. One of the common disadvantages of the neighborhood searching strategies is that they often produce disconnected subdomains. One way to prevent this from happening is to follow a traversal order that is based on the degree of connectivity of the graph G . A well known such ordering scheme is the so called Reverse Cuthill-McKee [10], [25]. Other graph-based mapping heuristics and their performance are presented in [3] and [53]. Various implementation of these heuristics for finite element meshes and grids together with their performance evaluation are reported in [2], [11], [12], [13], [17], [63].

Spectral Search Heuristics According to these search schemes the vertices V are visited (sorted) in the order defined by the size of the components of an eigenvector or combination of eigenvectors of the Laplacian matrix $L(G)$ of the graph G . The elements $L_{i,j}(G)$ of $L(G)$ are defined [5] to be +1 if $(v_i, v_j) \in E$, $-d_i$ if $i = j$, and 0 otherwise. These approaches depend on the choice of the eigenvector(s) of $L(G)$. Fiedler [21] observed that the second eigenvector of L represents a good measure of the connectivity of the graph G . A recursive implementation of this search scheme, referred to as recursive spectral bisection (RSB) based on Fiedler's eigenvector was introduced in [57]. RSB was found to be computationally very expensive. To improve its performance a multilevel version of RSB was developed [6]. Other spectral heuristics combining several eigenvectors with quadrisection and octasection implementations are proposed and discussed in [32]. The performance of spectral heuristics is presented in [5], [6], [32], [63].

Coordinate Axis Splitting This is another class of enumerative schemes whose main characteristic is that they ignore the connectivity information of the mesh graph G . They are based on coordinate sorting and partitioning along cartesian, polar, and symmetric inertial axis of the graph G . A comprehensive evaluation of these heuristics is reported in [63]. Following we review the underlying ideas of these strategies.

Cartesian axis splitting: In these schemes the cartesian coordinates of the mesh nodes or the element center of mass are sorted and split along each axis. There exist non-recursive and recursive implementations in both strip-wise and domain-wise form [44], [63]. In the case of recursive schemes the bisection direction can vary for each recursive step. One can choose this direction by splitting along the longest expansion which can be easily determined [44]. An alternative implementation is the one that compares the communication cost of the produced partitioning in both possible directions and chooses the one corresponding to less cost.

Polar/spherical axis splitting: The basic idea is similar to cartesian axis splitting schemes. In the polar/spherical axis partitioning schemes, the sorting of the coordinates of nodes or element center of mass is done along r , θ , and z/α axis. In addition to the available options in cartesian axis splitting schemes, the origin point can be selected as either center of inertia or center of mass. An implementation of this

scheme is described in [44]. Due to the periodicity of the cartesian to polar coordinates transformations, these schemes can produce, with high probability, disconnected subdomains. In [63] an implementation of this scheme is reported that avoids the above shortcoming by appropriate angle shifting.

Inertia axis splitting: This scheme first computes the main symmetry axis from the node coordinates of the mesh or the coordinates of element mass [44]. Then, it splits the domain into several subdomains along this axis. It repeats this step until the predefined number of subdomains is reached. The symmetry axis is obtained by computing the eigenvector corresponding to the largest eigenvalue of the inertia matrix $I = A^T A$ [62] where A is the matrix of the mesh coordinates. Implementations of these schemes and their variations are presented in [18], [44] and [63].

Deterministic optimization heuristics The mesh partitioning problem can be formulated as a constrain or unconstrained optimization problem. This set of heuristics is applied to solve these associated optimization problems. The basis of most of them is the so called Kernighan and Lin algorithm ($K - L$) [38]. Several variation of this algorithm have been proposed in order to incorporate most of the mesh partitioning criteria. A detail review of these class of strategies together with the description of an efficient improvement of the $K - L$ algorithm for mesh/grid can be found in [14]. Some of these heuristics are available in the //ELLPACK system [36].

Stochastic optimization heuristics Another class of heuristics proposed for the approximation of the solution of the partitioning optimization problem is based on stochastic techniques such as simulated annealing [42] and Hopfield neural networks [33]. Their application and modification for the partitioning of finite element mesh graph has been studied by several authors [7], [24], [47], [62]. Although these techniques tend to generate more accurate solutions to the mesh partitioning problem, they also tend to be computationally very intensive [47], [62]. Similar computational behavior has been observed for the neural based heuristics.

Software tools for mesh/grid decomposition Several tools have been developed to incorporate the above algorithmic infrastructure. The //ELLPACK system has a graphical tool (DOMAIN DECOMPOSER) that allow users to obtain and display automatic decompositions by a variety of heuristics for two and there dimensional meshes/grids. The user can either modify interactively these decompositions or specify his own. A description of an earlier version of this system is presented in [12]. The current version supports both element and node wise partitionings using most of the heuristics described above. This tool is completely integrated with the parallel ELLPACK problem solving environment so it supports all the parallel discretization and solution modules currently available in //ELLPACK library. A similar tool has been developed by *Simulog Inc.* [44]. This tool is integrated with their own three dimensional finite element mesh generator and flow mechanics code. The Simulog tool allows the user to view the automatically obtained mesh decompositions and it uses primarily coordinate axis decomposition strategies. A third domain decomposition tool is the TOP/DOMDEC [18]. It offers several heuristic decomposition algorithms including Greedy, RCM, recursive RCM, principal inertial, recursive inertial, recursive graph bisection, RSB and MRB. The user interface includes three-dimensional graphics, a parallel simulator, and an output function with parallel I/O data structures.

5.2 Fast Grid/Mesh Partitioning Heuristics

It has been observed that the decomposition of fine meshes can be very costly. In [40], it is reported that a 64-way MRSB partitioning of 150x150 finite difference grid of an L-shaped domain requires half of the time to solve the corresponding 5-point difference equations obtained from the discretization of a model PDE problem using Jacobi-CG on a single processor. Instead of solving the exact partitioning problem, it is proposed in [40] to extend the matrix problem in the entire rectangular overlaying grid used to generate the actual grid, by an identity matrix and solve the modified problem in parallel using the decomposition of the overlayed rectangular grid. This method is referred to as an **encapsulation** approach. Numerical results indicate that this approach outperforms all the ones that are based on the partitioning of the exact grid [40]. Unfortunately, this approach can not be generalized for finite element meshes. A natural “fast” alternative for mesh decomposition is to integrate the mesh generation and the partitioning steps [63] and implement them in parallel. This is natural, since most of the mesh generators already use some form of coarse domain decomposition as a starting point. The numerical results reported in [63] suggest

that this parallel integrated approach can result in the significant reduction of the data partitioning overhead.

References

- [1] A mimd implementation of a parallel euler solver for unstructured grids. *The Journal of Supercomputing*, 6(0):117–137, V. Venkatakrishman and H. D. Simon and T. J. Barth.
- [2] M. Al-Nasra and D. T. Nguyen. An algorithm for domain decomposition in finite element analysis. *Computers and Structures*, 39(3/4):227–289, 1991.
- [3] C. Aykanat, F. Ozguner, F. Ercal, and P. Sadayappan. Iterative algorithms for solution of large sparse systems of linear equations on hypercubes. *IEEE Transactions on Computers*, 37(12):1554–1568, December 1988.
- [4] S. Baase. *Computer Algorithms: Introduction to design and analysis*, pages 145–207. Addison-Wesley, Reading, MA, 1988.
- [5] S. T. Barnard and H. D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. In *Sixth SIAM Conference on Parallel Processing for Scientific Computing*, pages 711–718, 1993.
- [6] S.T. Barnard and H.D. Simon. A fast multilevel implementation of recursive spectral bisection for partiioning unstructures problems. Technical Report RNR-92-033, revised April 93, pp. 1-25., NASA Ames Research Center, 1993.
- [7] H. Byun, E. N. Houstis, and S.Kortesis.
- [8] T. Chan and R. S. Tuminaro. A survey of parallel multigrid algorithms. In A. Noor, editor, *Parallel Computations and their Impact on Mechanics*, volume AMD-Vol. 86, pages 155–170. Amer. Soc. Mech. Engr., Dec. 1987.
- [9] T.F. Chan and Y. Saad. Multigrid algorithms on the hypercube multiprocessor. *IEEE Trans. on Computers*, C-35(11):969–977, November 1986.
- [10] W. M. Chan and A. George. A linear time implementation of the Reverse Cuthill-McKee algorithm. *BIT*, 20(0):8–14, 1980.
- [11] N. P. Chrisochoides, M. Aboelaze, E. N. Houstis, and C. E. Houstis. The parallelization of some level 2 and 3 blas operations on distributed-memory machines. In *Advances in Computer Methods for Partial Differential Equations*, pages 127–133. IMACS, 1992.
- [12] N. P. Chrisochoides, C. E. Houstis, S. K. Kortesis, E.N. Houstis, P. N. Papachiou, and J. R. Rice. Domain decomposer: A software tool for mapping pde computations to parallel machines. In R.Glowinski et al., editor, *Domain decomposition methods for partial differential equations IV*, pages 341–357. SIAM Publications, 1991.
- [13] N. P. Chrisochoides, C.E. Houstis, S.K. Kortesis E.N. Houstis, and J.R. Rice. Automatic Load Balanced Partitioning Strategies for PDE Computations. In E.N. Houstis and D. Gannon, editors, *Proceedings of International Conference on Supercomputing*, pages 99–107, Crete-Greece, 1989. ACM Press.
- [14] N. P. Chrisochoides, E. N. Houstis, and J. R Rice. Mapping algorithms and software environments for data parallel pde iterative solvers. *Journal of Distributed and Parallel Computing*, 21(0):75–95, 1994.
- [15] C.Pommerell, M.Annaratone, and W. Fichtner. A set of new mapping and coloring heuristics for distributed-memory parallel processors. *SIAM J. Sci. Stat. Comput.*, 13(1):194–226, 1992.

- [16] R.D. da Cunha and T.R. Hopkins. PIM 1.1: The parallel iterative methods package for systems of linear equations users guide (FORTRAN 77 version). Technical report, Computing Laboratory, University of Kent at Canterbury, 1993.
- [17] C. Farhat. A simple and efficient automatic fem domain decomposer. *Computers and Structures*, 28(5):579–602, 1988.
- [18] C. Farhat and H. D.Simon. TOP/DOMDEC - a software tool for mesh partitioning and parallel processing. Technical Report RNR-93-011, pp.1-28, NASA Ames Research Center, 1993.
- [19] C. Farhat and M. Lesoinne. Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics. *International Journal for Numerical Methods in Engineering*, 36(0):745–764, 1993.
- [20] C. Farhat and E. Wilson. A new finite element concurrent computer program architecture. *Int. J. for Numerical Methods in engineering*, 24(0):1771–1792, 1987.
- [21] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(100):619–633, 1975.
- [22] J. Flaherty, M. Benantar, and M.S. Krishnamoorthy. Coloring Procedures for Finite Element Computation on Shared-Memory Parallel Computers. In A. K. Noor, editor, *Adaptive, Multilevel, and Hierarchical Computational Strategies*, volume 157, pages 435–490, New York, 1992. ASME.
- [23] J. Flaherty, M. Benantar, C. Ozturan, M. S. Shephard, and B. K. Szymanski. Parallel computation in adaptive finite element analysis. In C.A. Brebbia and M.H. Aliabadi, editors, *Chapter 7: Adaptive Finite Element and Boundary Element Methods*, London, 1993. Elsevier Applied Science.
- [24] G.C. Fox. A review of automatic load balancing and decomposition methods for the hypercube. In M. Schultz, editor, *IMA Institute*, pages 63–76. Springer-Verlag, 1986.
- [25] A. George and J. W. H. Liu. Algorithms for matrix partitioning and the numerical solution of finite element systems. *SIAM J. Numer. Anal.*, 15(2):297–327, 1978.
- [26] A. George and J. W. H. Liu. An implementation of a pseudoperipheral node finder. *ACM Transactions on Mathematical Software*, 5(3):284–295, 1979.
- [27] A. George and J. W. H. Liu. Algorithms for matrix partitioning and the numerical solution of finite element systems. *SIAM J. Numer. Anal.*, 15(2):297–327, April 1978.
- [28] N. E. Gibbs and J. W. G. Poole. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM J. Numer. Anal.*, 13(2):236–250, 1976.
- [29] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press, October 1994.
- [30] L. Gross, C. Roll, and W. Schoenauer. Vecfem for mixed finite elements. Technical Report Interner Bericht Nr. 50/93, Rechenzentrum der Universitat Karlsruhe, December 1993.
- [31] G.Yagawa, N. Soneda, and S.Yoshimura. A large scale finite element analysis using domain decomposition method on a prallel computer. *Computers and structures*, 38(5/6):615–625, 1991.
- [32] B. Hendrickson and R. Leland. An improved spectral load balancing method. In *Sixth SIAM Conference on Parallel Processing for Scientific Computing*, pages 953–961, 1993.
- [33] J. J. Hopfield. Neural networks and physical systems with emergent collective abilities. *Proc. Natl. Acad. Sci. USA*, 79:2554–2558, 1982.

- [34] E. N. Houstis, T. S. Papatheodorou, and J. R. Rice. Parallel ELLPACK: An expert system for the parallel processing of partial differential equations. In *Intelligent Mathematical Software Systems*, pages 63–73. North-Holland, Amsterdam, 1990.
- [35] E. N. Houstis and J. R. Rice. Parallel ellpack: A development and problem solving environment for high performance computing machines. In P. W. Gaffney and E. N. Houstis, editors, *Programming Environments for High-Level Scientific Problem Solving*, pages 229–241. North-Holland, 1992.
- [36] E. N. Houstis, J. R. Rice, N. P. Chrisochoides, H. C. Karathanasis, P. N. Papachiou, M. K. Samartzis, E. A. Vavalis, K. Y. Wang, and S. Weerawarana. Ellpack: A numerical simulation programming environment for parallel mimd machines. In D. Marinescu and R. Frost, editors, *International Conference on Supercomputing*, pages 96–107, Amsderdam, June 1990. ACM Press NY.
- [37] W. Joubert and G. F. Carey. Pcg: A software package for the iterative solution of linear systems on scalar, vector and parallel computers. In *Proceedings of 14thIMACS World Congress on Computational and Applied Mathematics*, volume 1, pages 247–250, Atlanta, 1994.
- [38] B. W. Kernigham and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, February 1970.
- [39] S. Kim, E. N. Houstis, and J. R. Rice. Parallel stationary iterative methods and their performance. In D. Marinescu and R. Frost, editors, *INTEL supercomputer users group conference*, San Diego, 1994.
- [40] Sang-Bae Kim. Parallel Numerical Methods for Partial Differential Equations. Ph.D. Thesis, 1993, Department of Mathematics, Purdue University. Technical Report CSD-TR-94-000, pp.1-00, Purdue University, Computer Science, 1993.
- [41] D. Kinkaid, J. Respass, and R. Grimes. Algorithm 586: Itpack 2c: A fortran package for solving large linear systems by adaptive accelerated iterative methods. *ACM Tran. Math. Soft.*, 8(0):302–322, 1982.
- [42] S. Kirkpatrick, J. C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
- [43] N.K. Madsen (Lawrence Livermore Laboratories) and R.F. Sincovec (Kansas State University). Algorithm 540: Pdecol, general sollocation software for partial differential equations. *ACM Transactions on Mathematical Software*, 5(3):326–351, September 1979.
- [44] M. Lorient and L. Fezoui. Mesh-splitting preprocessor. Technical Report unpublished manuscript, Simulog Inc, 1989.
- [45] J. Malone. Automated mesh decomposition and concurrent finite element analysis for hypercube multiprocessor computers. *Computer Methods in Applied Mechanics and Engineering*, 79(0):27–58, 1988.
- [46] J.G. Malone. Automated mesh decomposition and concurrent finite element analysis for hypercube multiprocessor computers. *Computer methods in applied mechanics and engineering*, 70(0):27–58, 1988.
- [47] N. Mansur. Physical optimization algorithms for mapping data to distributed-memory multiprocessors, phd thesis. Technical Report CRPC-TR92229, Syracuse University, August 1992.
- [48] O. A. McBryan, P. O. Frederickson, J. Linden, A. Schueller, K. Solchenbach, K. Stueben, C.-A. Thole, and U. Trottenberg. Multigrid methods on parallel computers - a survey of recent developments. *Impact Comput. Sci. Eng.*, 3:1–75, 1991.

- [49] W. F. Mitchell. Adaptive refinement for arbitrary finite element spaces with hierarchical bases. *J. Computational and Applied Math.*, 36:65–78, 1991.
- [50] B. Nour-Omid, A. Raefsky, and G. Lyzenga. Solving finite element equations on concurrent computers. In A. Noor, editor, *Parallel computations and their impact on mechanics*, pages 209–226, 1987.
- [51] S. Pissanetsky. *Ordering for Gauss elimination: Symmetry matrices, Sparse Matrix Technology*, pages 94–158. Academic Press, Orlando, 1984.
- [52] J. R. Rice and R. F. Boisvert. *Solving elliptic problems using ELLPACK*. Springer-Verlag, 1985.
- [53] P. Sadayappan and F. Ercal. Nearest-neighbor mapping of finite element graphs onto processor meshes. *IEEE Transactions on computers*, C-36(12):1408–1424, December 1987.
- [54] M. Schmauder, R. Weiss, and W. Schoenauer. The cadsol program package. Technical Report Interner Bericht Nr. 46/92, Rechenzentrum der Universitat Karlsruhe, April 1992.
- [55] W. Schoenauer, E. Schnepf, and H. Mueller. The fidisol program package. Technical Report Interner Bericht Nr. 27/85, Rechenzentrum der Universitat Karlsruhe, December 1985.
- [56] J.N. Shadid and R.S. Tuminaro. Sparse iterative algorithm software for large scale MIMD machines: an initial discussion and implementation. *Concurrency: practice and experience*, 4(6):481–497, 1992.
- [57] H. D. Simon. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*, 2(2/3):135–148, 1991.
- [58] P. Le Tallec, Y. H. De Roeck, and M. Vidrascu. Domain decomposition methods for large linearly elliptic three-dimensional problems. *J. Computational and Applied Math.*, 34:93–117, 1991.
- [59] David K. Melgaard (Kansas State University) and Richard F. Sincovec (Boeing Computer Services Company). General software for two-dimensional nonlinear partial differential equations. *ACM Transactions on Mathematical Software*, 7(1):106–125, March 1981.
- [60] Richard F. Sincovec (Kansas State University) and Niel K. Madsen (Lawrence Livermore Laboratories). Software for nonlinear partial differential equations. *ACM Transactions on Mathematical Software*, 1(3):232–260, September 1975.
- [61] S. Weerawarana, E.N. Houstis, A.C. Catlin, and J.R. Rice. //ellpack: A system for simulating partial differential equations. In *Proceedings of IASTED International Conference on Modelling and Simulation*, 1995. to appear.
- [62] R. Williams. Performance of dynamic load balancing algorithms for unstructured mesh calculations. *Concurrency: Practice and experience*, 3(5):457–481, 1992.
- [63] P. Wu and E. N. Houstis. Parallel mesh generation and decomposition. *Computer Systems in Engineering*, (CSD-TR-93-075, pp. 1-49), 1994.