1994

EPPOD: A Problem Solving Environment for Parallel Electronic Prototyping of Physical Object Design

Poting Wu

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

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

Report Number:
94-043

EPPOD: A Problem Solving Environment for Parallel Electronic Prototyping of Physical Object Design

Poting Wu, Elias N. Houstis, and John R. Rice
Computer Sciences Department
Purdue University
West Lafayette, IN 47907

CSD-TR-94-043
June, 1994

EPPOD: A problem solving environment for parallel electronic prototyping of physical object design

Poting Wu, Elias N. Houstis and John R. Rice

Department of Computer Sciences
Purdue University
West Lafayette, Indiana 47907, U.S.A.
e-mail: wu@cs.purdue.edu

Technical Report CSD-TR-94-043
June, 1994

Abstract

One of the next *"Grand Challenges"* for computer applications is the creation of a system for the design and analysis of physical objects. This system will provide accurate computer simulations of physical objects coupled with powerful design optimization tools to allow prototyping and final design of a broad range of items. We refer to such software environment as **Electronic Prototyping for Physical Object Design (EPPOD).** The deep challenges in building such systems is in software *"integration"*, in utilizing *"massively parallelism"* to satisfy their large computational requirements, in incorporating *"knowledge"* into the entire electronic prototyping process, in creating *"intelligent"* user interfaces for such systems, and in advancing the *"algorithmic infrastructure"* needed to support the desired functionality. In this paper we address the issues related to parallel processing of the computationally intensive components of the EPPOD system and present an architecture of a EPPOD system for the *optimum* design of physical "parts" on message passing parallel machines. The parallel methodology adopted to map the underlying computations to parallel machines is based on the "optimal" decomposition" of continuous and discrete geometric data associated with the physical object. One of the main goals of this methodology is the ***reusability*** of existing software parts while implementing various components of the EPPOD system on parallel computational environments.

This work was supported by NSF grants 9123502-CDA and 9202536-CCR, AFOSR F49620-92-J-0069 and PRF grant 6902003.

CONTENTS

1. Introduction

One of the next "Grand Challenges" for computer applications is creating a system for the design and analysis of physical objects. Example applications include mechanical design, biomedical design, and so on. These systems will provide accurate computer simulations of physical objects coupled with powerful design optimization tools to allow prototyping and final design of a broad range of items. When such systems become a reality, they will have an even greater impact than systems for electronic design, one of the great achievements of computing technology of the past decade. Our studies indicate that the computer science problems arising from such systems can be surmounted. The deep challenges in systems integration, in obtaining enough computing power, in devising competent geometric tools, and so on can be met. It is clear that enough physical phenomena are well understood so that a useful, accurate, and broadly applicable system for physical design can be created. Thus, the challenge is to incorporate this knowledge into a high level design system where the design parameters (e.g., shapes, materials, construction techniques, environment conditions) are specified rapidly and accurately, and then optimized. It is reasonable to expect the appearance of powerful physical design and analysis systems within five or ten years. Throughout we refer to this system as **Electronic Prototyping for Physical Object Design (EPPOD)** [Wu 93c]. We believe that EPPOD is one of the grand challenges for computer science which will require *massively computational power* and *"smart" software* for its realization.

The system must provide both automatic optimization techniques and the ability of the designer to direct the variations and optimization of design parameters. The system must allow new knowledge about physical phenomena to be incorporated so that it can grow in capability as science advances. Among the benefits of a prototyping system for physical design include *faster designs, less cost, higher quality and better performance.* It sounds almost too good to be true, but one can document a variety of example applications ranging from the mundane design of aluminum cans to the highly refined technology of jet engines, where design times are cut by a factor of 10 or 100, where design costs are reduced by similar factors, or where very mature devices have been improved in performance enough to justify retooling large production facilities. As powerful and versatile prototyping systems become available many organizations must adopt their use or accept economic and technological obsolescence.

In the past ten years, there have been many examples of a handful of engineers designing a complete computer system of novel architecture, from scratch. In two years they completed the design, ordered the parts, assembled them, and expected them to work properly right at the start. Such amazing efforts have been successful because of electronic design systems, and hint at the vast potential of electronic prototyping in other fields. Electronic design systems are a critical factor, perhaps *the* critical factor, in the rapid evolution of computer products.

Following we list some of the important components of an EPPOD system. The EPPOD process starts with the user specifying the initial geometric schematic of the object. For this a **geometric modeler** is used to represent and manipulate geometric shapes of all kinds. It allows geometric information to be readily exchanged and geometric design activities to be carried out in isolation. Some of these objects represent mechanical objects whose position changes with time. For this we need a **discrete mechanics** subsystem to simulate objects moving due to forces, constraints, collisions, and contacts of all kinds. Each object might characterized by physical properties that

change according to its interaction with the outside world. The description of **continuous physical phenomena** is usually done by mathematical models involving partial differential equations, and the models are categorized as *steady state, structural stresses and strains, evolutionary, wave phenomena, shock waves, combustion, vibration, etc.* Manufacturing of physical objects usually impose design and cost objectives and constraints. For the satisfiability of the various design objective functions and constraints we need an optimization subsystem The designer must be able to "input", "control"' and "see" the physics and geometry directly and in informative ways without complex programming. Thus the system must have an **interactive GUI user interface**.The computation work for design is enormous by today's standards and success depends on exploiting very high performance computers using parallel and distributed computing techniques. Therefore we need a **parallel methodology** preferably capable of reusing existing EPPOD software parts and running seamlessly on various parallel machines. Expert assistance is needed to aid the designer in everything from selecting the right models to checking the correctness of a design; from selecting computational methods to exploiting parallelism and assessing the validity of the computed results. The development of **domain specific expert systems** technology is necessary. The software design must allow for a highly modular, flexible system that incorporates both old and new software parts or systems, it must evolve naturally as technology changes. **A component based software methodology** must be developed. Specialized (targeted) design systems should be built to exploit the special properties and context of a targeted application.

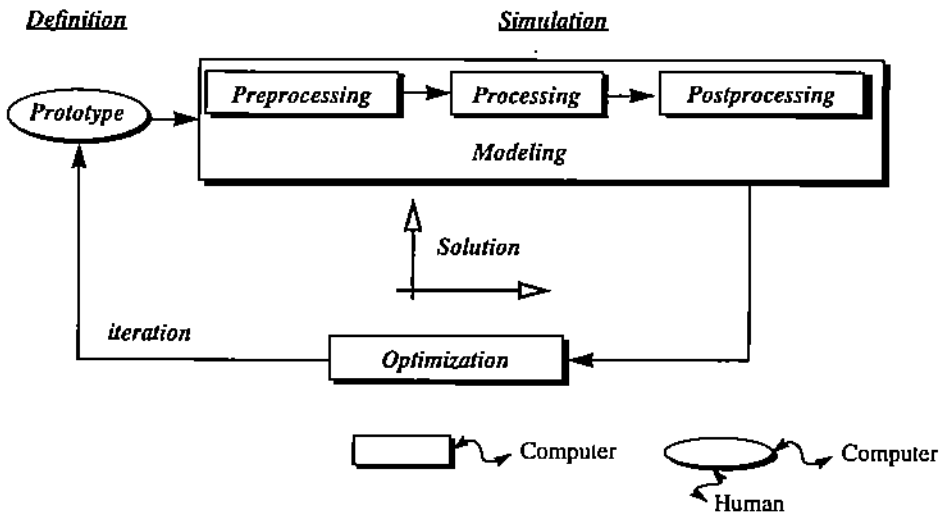


Figure 1: An abstract view of the EPPOD system for the design of physical parts that brings the conception, geometry modeling, continuous physics modeling, analysis, and optimization stages of design into an unified *concurrent* framework.

In this paper we address the issues related to parallel processing of the computing bound components of the EPPOD system and present an architecture and implementation of a problem solving environment (**PSE**) for the *optimum* design of physical "parts" on message passing parallel machines. Figure 1 depicts the instance of EPPOD that we are currently implementing on distributed memory machines.

2. A parallel electronic prototyping process

One of the technical challenges in building an EPPOD system is the harvesting of computer power offered by the massively parallel machines and networks of powerful desktop computing engines and its exploitation in realizing the EPPOD system. Fortunately, most of the software/ algorithmic components of EPPOD are inherently parallel or they can run in some concurrent mode. Fortunately, there is already a significant amount of high quality sequential software that can implement several of the EPPOD components. Thus, one research challenge is to identify parallel methodologies that are capable of **reusing** most of this existing software. This issue is addressed later on. In Figure 2 we redefine the EPPOD system depicted in Figure 1 from the parallel point of view and indicate the components that we have targeted for parallelization. These components are associated with the **modeling** and **global optimization** phases.

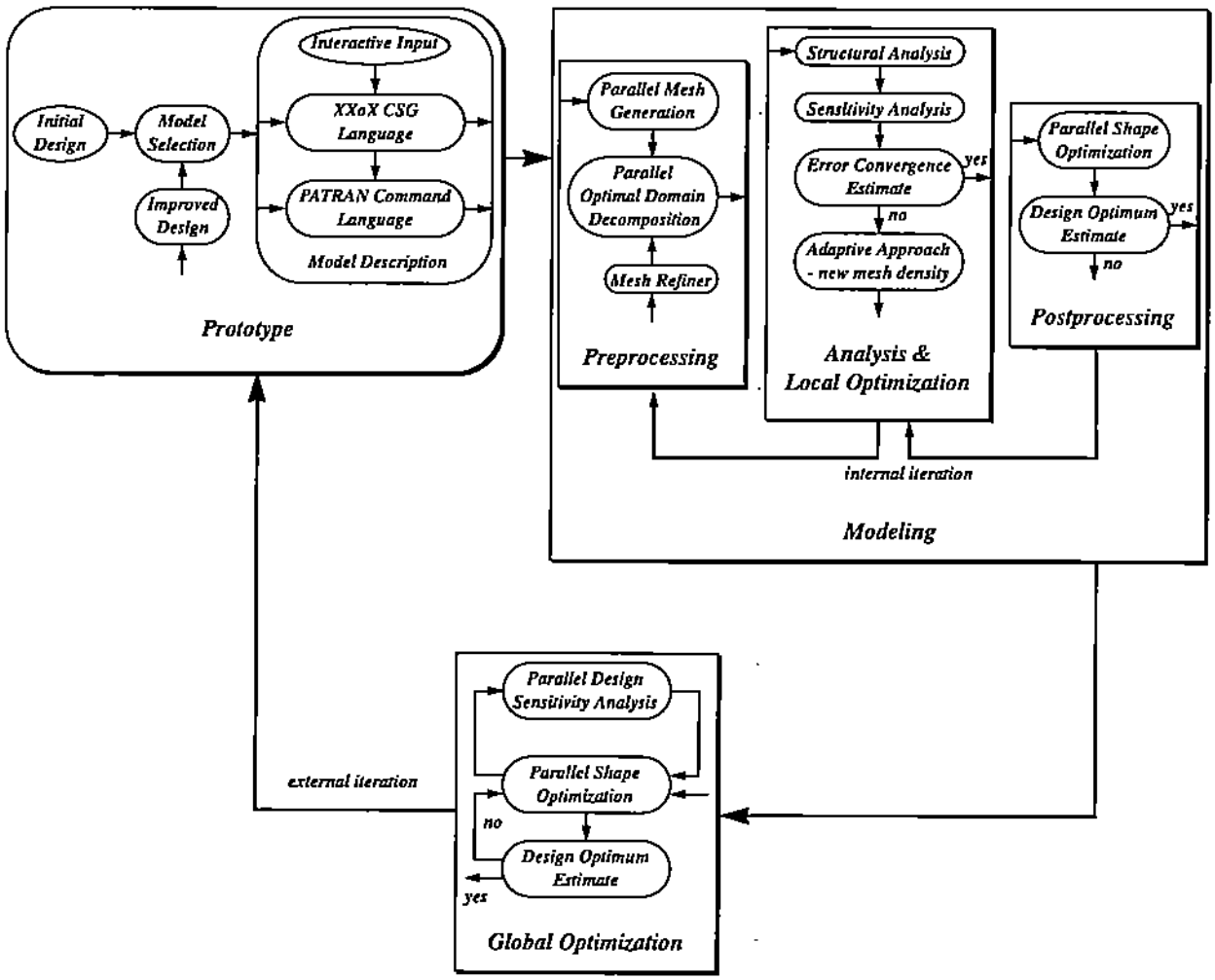


Figure 2: The parallel version of the EPPOD system depicted in Figure 1.

3. The software architecture of EPPOD system for mechanical parts

The new generation of software that supports large scale component based applications will be characterized by seamless dynamic integration of its components. Moreover, all the user interactions with the system will be done through at least a GUI interface with some "computationally intelligent" support for determining the user defined parameters [Hous 92, 94]. Figure 3 depicts the software components needed for the optimal design of a mechanical part together with an iconic view of its output. Each of these components is controlled and supported by a GUI user interface and appropriate run time libraries. All of them except mesh splitting are used to support the various phases of the electronic prototyping process. Mesh splitting is used to support our **reuse** parallel methodology. In the following sections we describe various subsystems selected for the implementation of these components and the parallelization of some of them.

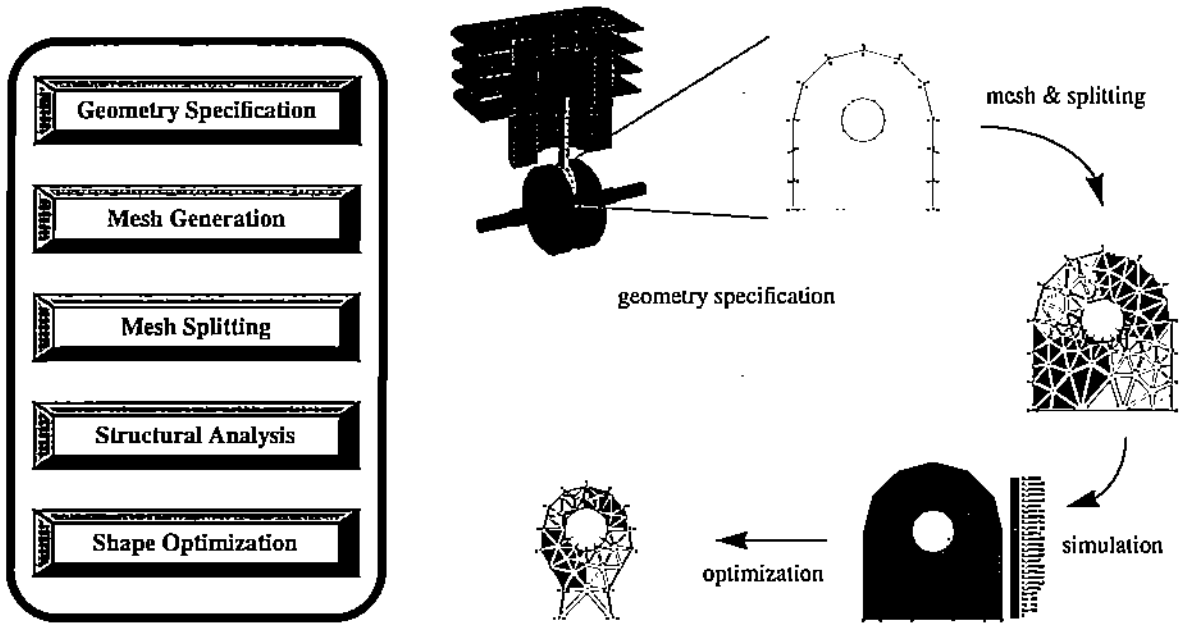


Figure 3: The tools of a problem solving environment and their output for the electronic prototyping of mechanical parts implemented on distributed memory machines.

4. Geometry specification tool

Any design and simulation of a physical object starts with the user completely specifying the problem on an assumed initially geometry. There are many geometry modelers in the market differing primarily in the representation scheme assumed. In the EPPOD system we have integrated XXoX and PATRAN geometry modeling systems. **XXoX** is a solid modeling system based on the CSG representation of solid objects. It consists of geometry and graphics libraries [XoXE 92], [XoXR 92] and an X-window interactive user interface [Wu 93b]. In the XXoX environment one can create 3-D primitives and 2-D outlines of cross-sections, manipulate the geometry by orienting, combining, cutting, and deforming the objects. Figure 4 displays the description of an engine part in XXoX language and its graphical view.

```
cylinder1 = cylinder(0,0,-0.52,1.56,1.04)
cylinder2 = cylinder(0,0,-0.52,1.04,1.04)
cylinder3 = cylinder(0,0,0,0.5,5.4)
box1 = box(-0.26,-0.52,1.04,0.52,1.04,6.6)
rod = rotate(cylinder1-cylinder2,0,0,0,0,1,0,-90)
rod = rod | box1 | translate(scale(rod,1,0.5,0.5),0,0,8.16)
rod = rod | translate(rotate(cylinder3,0,0,0,0,1,0,-90),2.6,0,8.16)
```

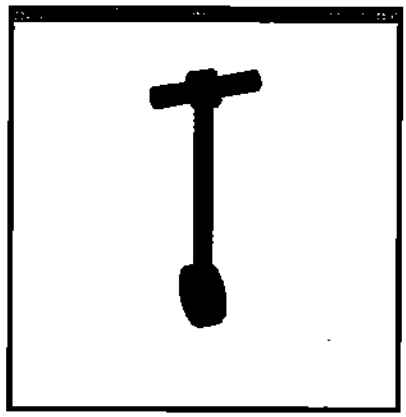


Figure 4: The description of an engine part in the XXoX language and its graphical view using XoX graphics routines.

PATRAN is a general purpose 3-D mechanical computer aided engineering software package that uses interactive graphics to link engineering design, analysis and results evaluation functions. Its solid geometry editor is based on surface representation. The detail description of this engineering PSE can be found in [Patran 92].

5. Parallel mesh generation and splitting tool

For the simulation of the physical behavior of a geometric object, one needs to approximate the object by a grid or mesh depending on the selected simulation technique. Unfortunately, the discretization of 3-D objects is a difficult mathematical problem. Moreover, the requirement to adapt the mesh in certain subregions complicates even more the mesh generation process. The current mesh generation technology is restricted in many respects and requires significant human interaction. In addition the computational requirements are prohibitively large, especially for 3-D objects and non-linear physical phenomena; this often involves rapidly changing protyping where the mesh has to be recomputed several times. Thus, the parallel implementation of this component is well justified. Recent studies [Chri 91, 94] and [Farh 93] have shown that the mesh can be used as an intermediate mechanism to map the underlying computations to parallel machines. This parallelization approach is called **geometry splitting** or **domain decomposition** and it requires the decomposition of the mesh into load balanced subregions with small interface lengths. This mesh decomposition problem is known to be NP-complete and finding a near optimum solution is done through appropriate heuristics. Therefore, its parallelization is recommended. It appears that there is a natural interrelation between the mesh generation and its decomposition. By exploiting this interaction, we are able to implement both the mesh and decomposition geometry preprocessing components in an integrated way on message passing machine computational

environment. Figure 5 depicts the five mesh generation steps for a 2-D mechanical part and a 4-processor machine configuration.

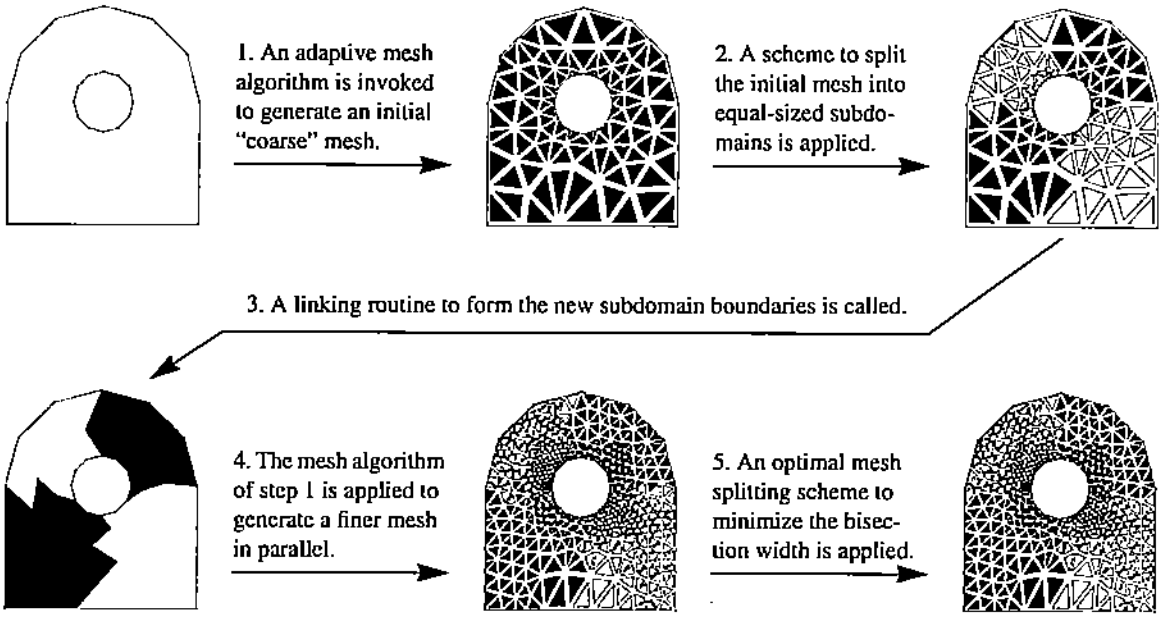
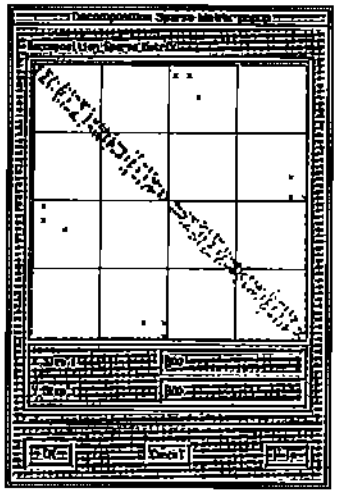


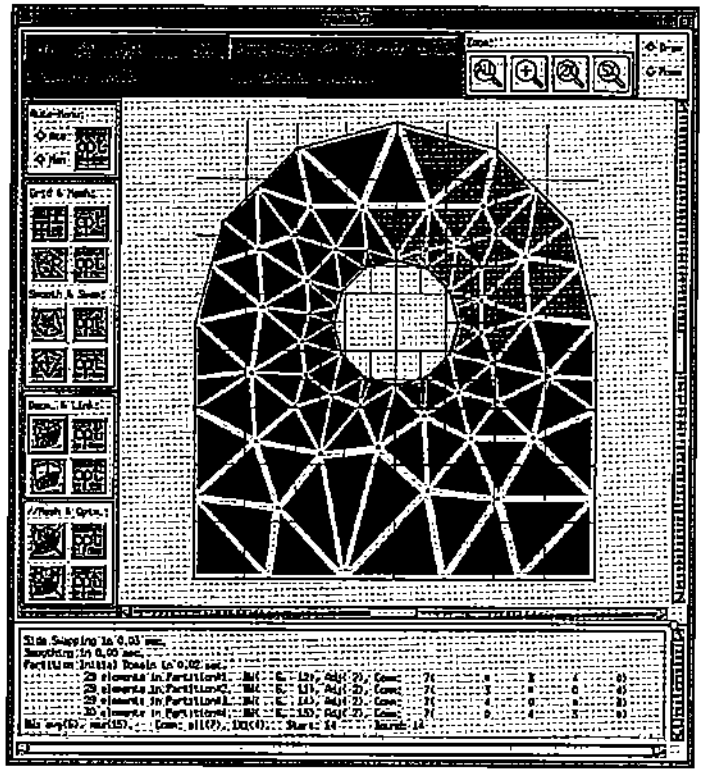
Figure 5: A five step methodology for parallel mesh and mesh-decomposition.

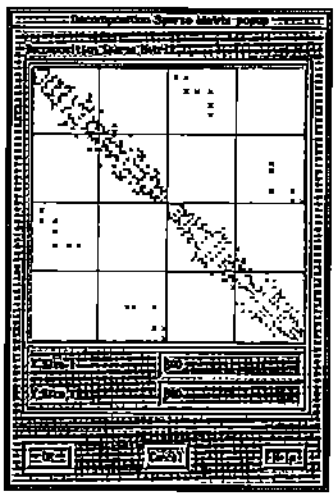
The GUI interface consists of several sections corresponding to *file and window management menu bar, visualization and manual mode button panel, command area, grid/mesh generation button panel, decomposition button panel, optimization button panel,* and *mesh/decomposition display area.*

For the decomposition of the mesh several algorithmic options exist, see [Wu 93a]. All these algorithms are static, sequential and operate off line from the rest of the computation. One of the unique aspects of our implementation of the decomposition process is its coupling with the parallel mesh generation. The details of this methodology, algorithmic infrastructure and its performance are reported in [Wu 93a]. Figure 6 displays two meshes with decompositions and the structure of the corresponding algebraic systems of the corresponding finite element equations.



Triangular Element Mesh Generation & Element-Wise Domain Decomposition





Triangular Element Mesh Generation & Node-Wise Domain Decomposition

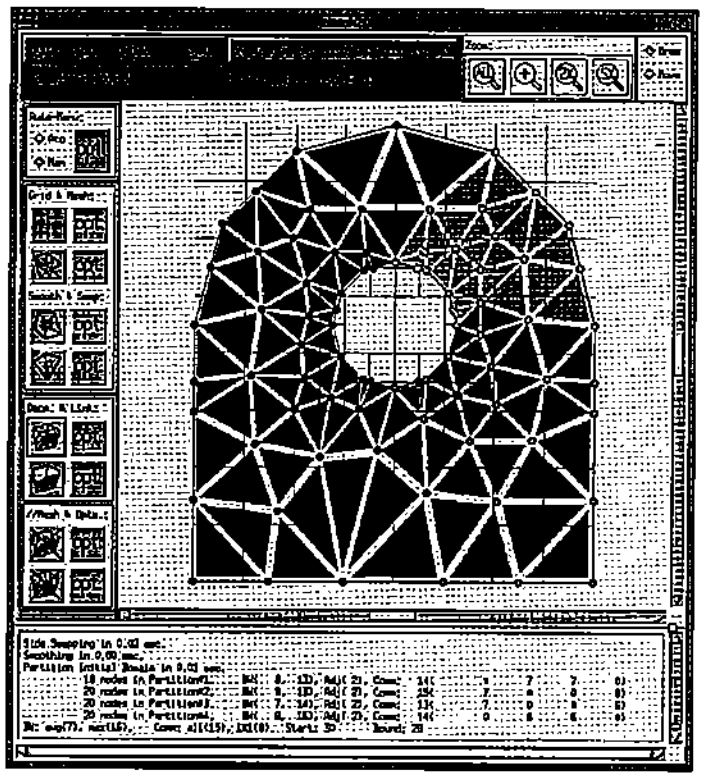


Figure 6: Examples windows of the GUI for mesh generation and splitting. The left windows show the matrix structure of the underlying discrete model while the right windows show the mesh and its splitting. The examples here use two different numerical discretizations for the same part.

6. Domain decomposition framework for structural analysis

We have already noted that there is a lot of high quality sequential software for electronic prototyping that can be utilized to support the building of customized EPPOD problem solving environments.

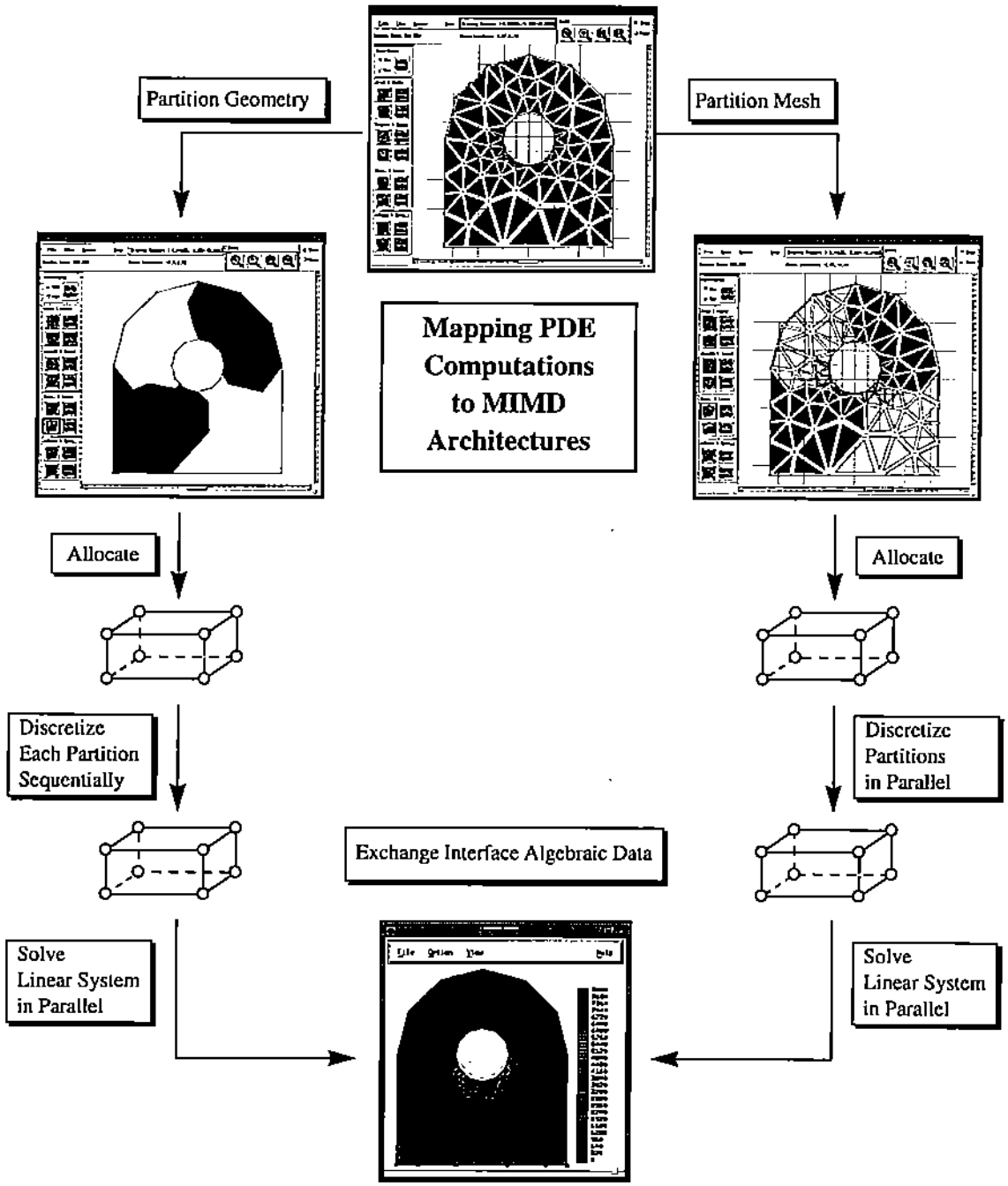


Figure 7: A geometry based parallel methodology capable of ***reusing*** existing sequential parts without modification.

Thus, it is necessary to develop a parallel methodology that utilizes (**reuses**) these software without significant modification. The parallel mesh and mesh splitting infrastructure together with the parallel algebraic solvers we have developed [Kim 94] allow us to implement a REUSE parallel methodology which is based on the parallel "optimum" mesh splitting approach defined above. This approach is described in Figure 7 and can be implemented either on the continues decomposition of the mathematical model governing the physics of the object or the discretized data structures of the numerical method used to approximate the solution of the continuous model. We have created a software framework or template that implements the above methodology which allows the user to call any sequential code to carry out the discretization of the mathematical model defined in each subdomain with appropriate conditions on the interface which guarantee that the interface equations are equivalent to the interior equations of the global mesh for the interface elements or nodal points. These interface conditions depend on whether one uses element-wise or node-wise mesh splitting. In some instances (i.e., node-wise splitting) an additional communication step is required to complete the generation of the algebraic equations. This approach allows the utilization of the discretization part of the code whose generation requires high level knowledge of the mathematical models involved and their efficient approximation.

7. Parallel shape optimization framework

Shape optimization based electronic prototyping is part of any future scenario for intelligent manufacturing [Ding 86], [Haft 86]. Unfortunately, the resulting optimization problems can be prohibitively large. Moreover, it is known that most of the optimization problems belong to the class of "hard" problems. Thus, seeking parallel methods for their solution is well justified [El 91], [Sikio 88]. The idea of divide and conquer is already used to approximate the solution of large optimization problems sequentially. This suggests that it might be more efficient to divide the system into several smaller subsystems, optimize them locally and in parallel, and then approximate the global solution by implementing some form of global optimization on the interfaces of the subsystems. This approach is referred to as the **two-level scheme**. In general, an optimization problem involving many variables and constraints cannot be decomposed into independent subproblems which can be independently optimized. However, the above described problem decomposition approach yields good approximations to the global minimum while allowing the parallel application of shape optimization on local subsystems. The effectiveness of the two-level scheme comes from the inherent parallelism in modeling physical objects. The analysis and shape optimization is implemented using the parallel mesh and mesh splitting tool and their algorithmic infrastructure. For the shape optimization problem we are developing two level semi-optimal algorithms based on the local and global mesh and decomposition data. Figure 8 indicates the two-level approach in relation with the rest of the electronic prototyping components.

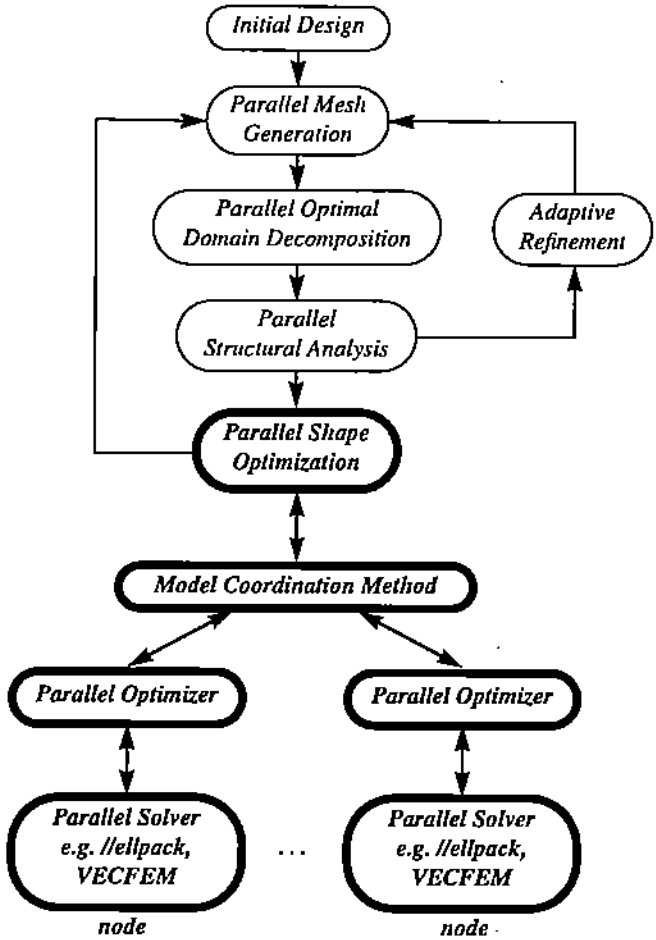


Figure 8: The framework for the two-level parallel optimization scheme within EPPOD based on the Model Coordination method.

For the formulation of the two-level semi-optimal scheme, we consider the general optimization problem of choosing the variables $\{X\}$ such that

$$Z = F(\{X\}) \Rightarrow \min.$$

$$\{h(\{X\})\} = \{0\}$$
$$\{g(\{X\})\} \leq \{0\}$$

$$\{X^L\} \leq \{X\} \leq \{X^U\}$$

where $F(\{X\})$ is the objective function and $\{h(\{X\})\}$ and $\{g(\{X\})\}$ are sets of equality and inequality constraints where $\{X^L\}$ and $\{X^U\}$ are the lower and upper bound vectors of $\{X\}$. The decomposition of the optimization problem is carried out, first by converting the problem into a two-level form with separate and distinct tasks assigned to each level. That is, we split apart the variables and constraints in each subdomain which do not interact with others (the non neighbors) to form the lower-level units. Then, we choose the dependent variables, called *coordinating variables*, which correspond to the interface variables and form the higher-level unit which determines the overall (global) system optimum. In general, the lower-level and the higher-level

problems are solved iteratively. There are two different ways to convert a given problem into two-level hierarchical optimization problems. They are referred throughout as the *model coordination* and the *goal coordination* methods [Kirsch 75]. We have selected to implement the model coordination approach in EPPOD.

For this, we partition the vector $\{X\}$ into two subvectors

$$\{X\}^T = (\{S\}^T, \{T\}^T)$$

where $\{S\}$ is called the subvector of *coordinating variables* between the subdomains and $\{T\}$ is called the subvector of *subdomain variables*. Furthermore, we decompose $\{T\}$ into n subvectors $\{T\}=(\{T_1\}...\{T_n\})$ where $\{T_i\}$ represents the subdomain variables associated with the i-th subdomain and n is the number of subdomains. According to this two-level scheme the components of the original problem can be written in the following form

$$Z = F(\{X\}) = \sum_{i=1}^{n} F_i\Big(\{S\}, \{T_i\}\Big)$$

$$\{h\} = \begin{matrix} \{h_1(\{S\}, \{T_1\})\} \\ \cdots \\ \{h_i(\{S\}, \{T_i\})\} \\ \cdots \\ \{h_n(\{S\}, \{T_n\})\} \end{matrix} \qquad \{g\} = \begin{matrix} \{g_1(\{S\}, \{T_1\})\} \\ \cdots \\ \{g_i(\{S\}, \{T_i\})\} \\ \cdots \\ \{g_n(\{S\}, \{T_n\})\} \end{matrix}$$

and the original problem can be restated as

$$Z = \sum_{i=1}^{n} F_i\Big(\{S\}, \{T_i\}\Big) \Rightarrow min$$

$$\{h_i(\{S\}, \{T_i\})\} = \{0\} \qquad i = 1, \ldots, n$$
$$\{g_i(\{S\}, \{T_i\})\} \leq \{0\} \qquad i = 1, \ldots, n$$

$$\{S^L\} \leq \{S\} \leq \{S^U\}$$

$$\{T_i^L\} \leq \{T_i\} \leq \{T_i^U\} \qquad i = 1, \ldots, n$$

where $\{S^L\}$, $\{S^U\}$, $\{T_i^L\}$, and $\{T_i^U\}$ are the lower and upper bound vectors for the decomposed subproblems.

8. A realization of EPPOD system and numerical examples

We are currently building an instance of EPPOD system for the design of physical parts. Figure 9 indicates the flow diagram of the optimum design process realized and the interactions of the various components in the EPPOD system. The components have been implemented in C and FORTRAN languages while their GUI user interfaces are built using Motif and X-windows tools. The integration of the various components is done by using the client/server paradigm.

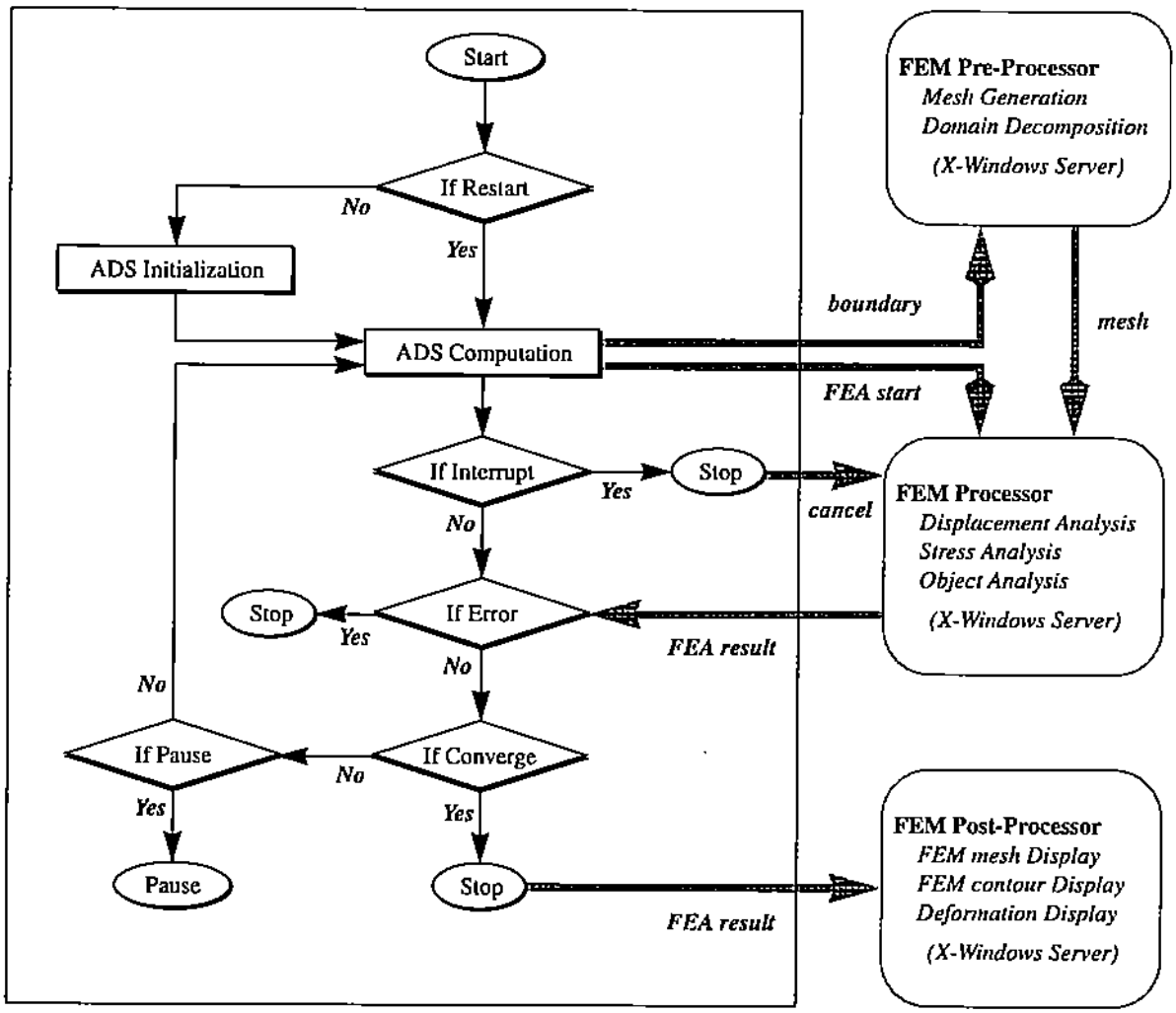


Figure 9: The flow diagram of the implemented EPPOD system. ADS (Automated Design Synthesis) is an optimization subsystem [Vand 85, 86, 89], [Roger 86] used to implement the parallel shape optimization phase. FEM and FEA stands for finite element method and analysis respectively.

In this section we display the optimum design solutions of two engine parts. On the left of Figure 10 we display the initial design, meshing, and decomposition of the original shape of an engine rod connector together with the output of the finite element analysis (i.e., deformation and stress diagrams). On the right of Figure 10 we display the same data after solving the shape optimization problem defined in the middle of the Figure 10. The goal of the chosen objective function is to minimize the area of the part under certain displacement and geometric constraints.

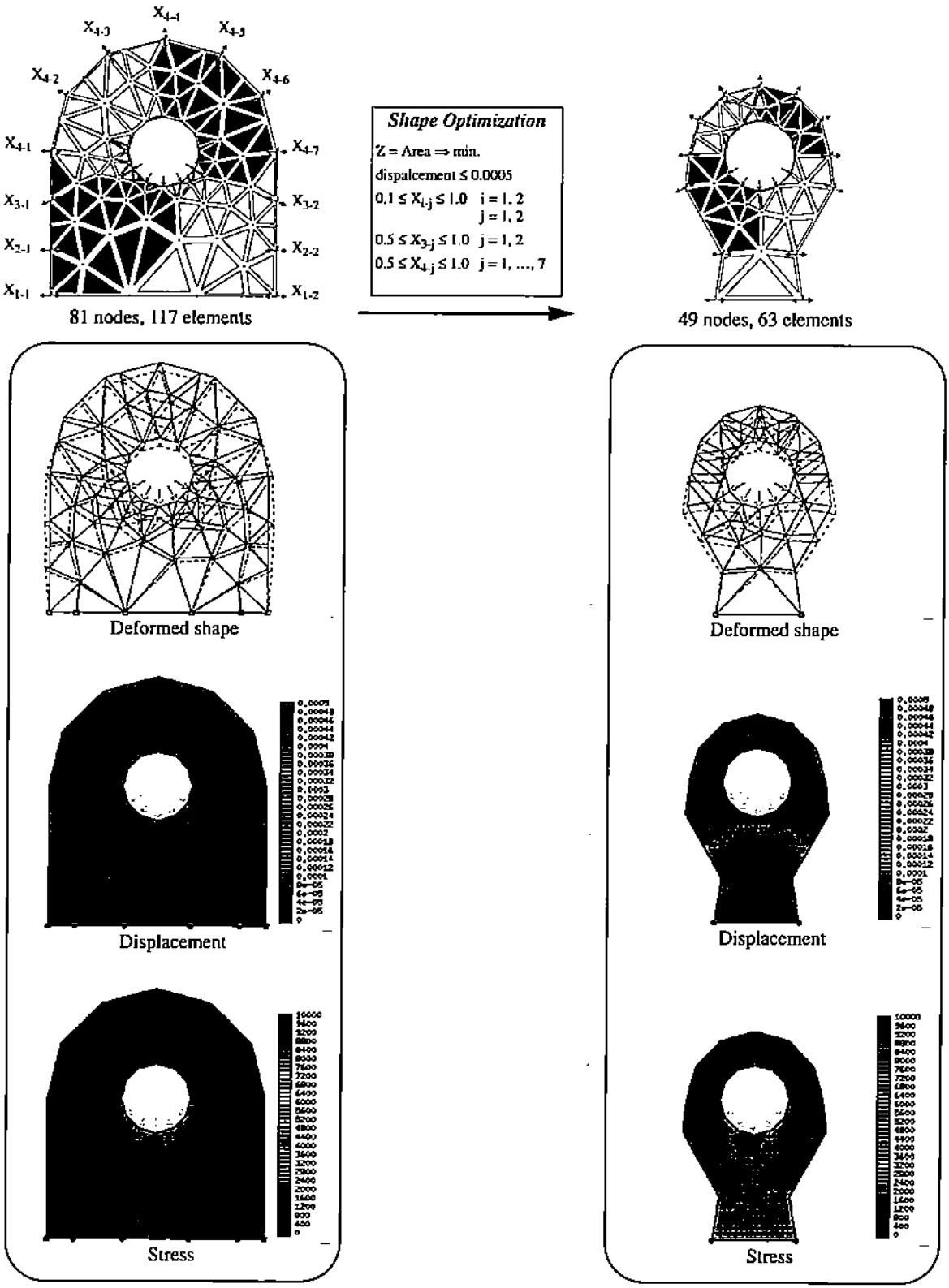


Figure 10: Parallel electronic prototyping of an engine rod connector.

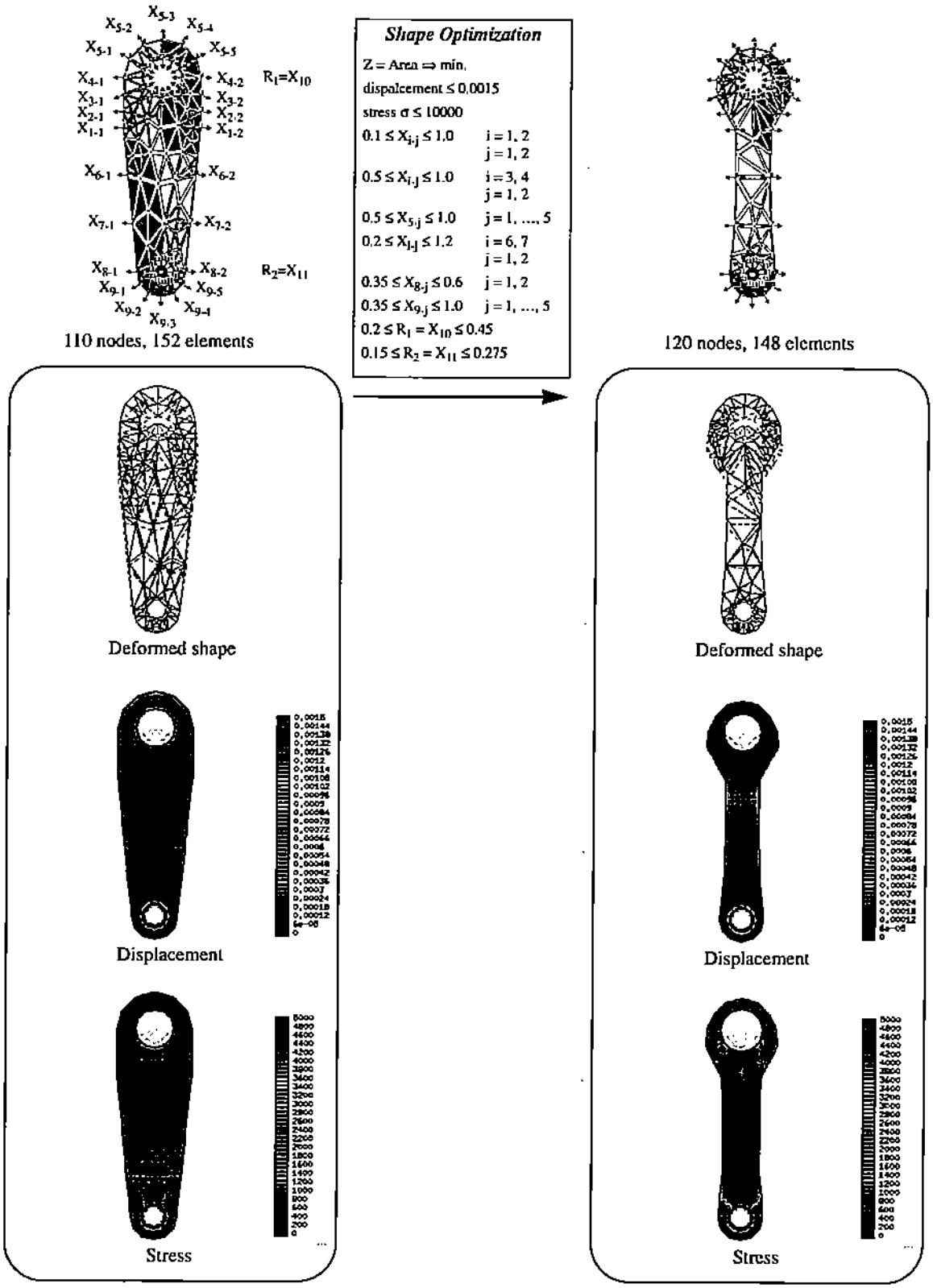


Figure 11: Parallel electronic prototyping of a torque arm.

In Figure 11 we display the specifications of the electronic prototyping process for a torque arm together with the definition of a shape optimization problem whose objective is to minimize the part's area while various stress and geometric constraints are satisfied. The optimized shape of the arm is also shown.

9. References

[Chri 91] N. P. Chrisochoides, E. N. Houstis and C. E. Houstis, "Geometry based mapping strategies for PDE computations", *Proceedings of the International Conference on Supercomputing*, Cologne-Germany, (June 1991) 128-135.

[Chri 94] 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**, (1994), 75-95.

[Ding 86] Yunliang Ding, "Shape optimization of structures: A literature survey", *Computers & Structures* **24 - 6**, (1986) 985-1004.

[El 91] M. E. M. El-Sayed and C. K. Hsiung, "Design optimization with parallel sensitivity analysis on the CRAY X-MP", *Structural Optimization* **3**, (1991) 247-251.

[Farh 93] Charbel Farhat and Michel Lesoinne, "Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics", *International Journal for Numerical Methods in Engineering* **36**, (1993) 745-764.

[Haft 86] Raphael T. Haftka and Ramana V. Grandhi, "Structural shape optimization - A survey", *Computer Methods in Applied Mechanics and Engineering* **57**, (1986) 91-106.

[Hous 92] E. N. Houstis and J. R. Rice, "Parallel ELLPACK: A development and problem solving environment for high performance computing machines", *Programming Environments for High-Level Scientific Problem Solving* (P. W. Gaffney and E. N. Houstis, Eds), North-Holland, (1992) 229-241.

[Hous 94] E. N. Houstis, C. E. Houstis, J. R. Rice, and S. Weerawarana, "PYTHIA: A computationally intelligent paradigm to support smart problem solving environments for PDE based applications", to appear.

[Kim 94] Sang Bae Kim, E. N. Houstis, and J. R. Rice, "Parallel stationary iterative methods and their performance", *Proceedings of the INTEL supercomputer users group conference* (Dan Marinescu and R. Frost, Eds), San Diego, (June 1994), to appear.

[Kirsch 75] Uri Kirsch, "Multilevel approach to optimum structural design", *Journal of the Structural Division*, ASCE **101 - ST4**, April 1975, 957-974.

[Patran 92] *PATRAN, A Division of PDA Engineering - PATRAN Plus User Manual, Vol. 1 & 2*, PDA Engineering, PATRAN Division.

[Roger 86] James L. Rogers and Jean-Francois M. Barthelemy, "An expert system for choosing the best combination of options in a general purpose program for automated design synthesis", *Engineering with Computers* **1**, (1986) 217-227.

[Sikio 88] Efthimios S. Sikiotis and Victor E. Saouma, "Parallel structural optimization on a network of computer workstations", *Computers & Structures* **29-1**, (1988) 141-150.

[Vand 86] Garret N. Vanderplaats and Hiroyuki Sugimoto, "A general-purpose optimization program for engineering design", *Computers & Structures* **24-1**, (1986) 13-21.

[Vand 89] G. N. Vanderplaats, "Effective use of numerical optimization in structural design", *Finite Elements in Analysis and Design* **6**, (1989) 97-112.

[Vand 85] Garret N. Vanderplaats, Hirokazu Miura, and Mladen Chargin, "Large scale structural synthesis", *Finite Elements in Analysis and Design* **1**, (1985) 117-130.

[Wu 93a] Poting Wu and E. N. Houstis, "Parallel dynamic mesh generation and domain decomposition", *Technical Report, Purdue University, Department of Computer Sciences*, CSD-TR-**93-075**, (October 1993) 1-49.

[Wu 93b] Poting Wu and E. N. Houstis, "XXoX: An interactive X-window based user interface for the XoX solid modeling library", *Technical Report, Purdue University, Department of Computer Sciences*, CSD-TR-**93-015**, (January 1993) 1-47.

[Wu 93c] Poting Wu and E. N. Houstis, "Parallel electronic prototyping of physical objects", *Technical Report, Purdue University, Department of Computer Sciences*, CSD-TR-**93-026**, (April 1993) 1-48.

[XoXE 92] *SHAPES Geometric Computing System - Geometry Library Reference Manual (C Edition)*, XOX Corporation, (1992).

[XoXR 92] *SHAPES Geometric Computing System - Graphics Library Reference Manual (C Edition)*, XOX Corporation, (1992).