Computación y Sistemas Vol. 4 No. 2 pp. 94-105
© 2000, CIC - IPN. *ISSN 1405-5546* Impreso en México

Distributed Solution of Simulation-Based Optimization Problems on Networks of Workstations

Thomas Barth[1], Barnd Freisleben[2], Manfred Grauer[1] and Frank Thilo[2]
[1]Departament of Information Systems, University of Siegen
[2]Departament of Electrical Engineering and Computer Science, University of Siegen
Hölderlinstr. 3, D-57068 Siegen, Germany
E-mail: {barth, grauer}@fb5.uni-siegen.de, {freisleb, thilo}@informat1k.uni-siegen.de

Article received on February 15, 2000; accepted on August 23, 2000

Abstract

The paper deals with the distributed solution of simulation-based nonlinear constrained optimization problems in engineering on a network of workstations. This type of nonlinear optimization and control problems in engineering can be generally characterized as non-convex and non-smooth. Additionally, the involved simulation introduces certain numerical "noise" to the solution process. These characteristics lead to very time consuming solution processes, because of long-running computations for simulation and the difficulties in finding a global optimum. The resulting requirements for the corresponding software architecture and software integration of the simulation and optimization components are discussed. The implementation of this software architecture is presented using the OpTiX system for optimization, the finite-element package FEFLOW for simulation, and the WINNER *resource management system for load distribution in networks of workstations. Furthermore, the concepts of a distributed optimization algorithm are described and preliminary results of its application to industrial applications from water engineering are used to show the feasibility of the proposed system for distributed optimization.*

Keywords: Distributed optimization, computational engineering, simulation-based optimization, direct optimization methods, load balancing, scalability

1 Introduction

Most optimization problems in engineering cannot be formulated analytically. For constrained optimization and control problems in engineering it is typical that the objective function and/or the constraints are highly nonlinear due to the underlying mathematical model in form of FEM-(Finite-Element-Method)-based solution approaches; sometimes these problems even have mixed-integer decision variables. Examples of typical engineering problems are design optimization problems in the aircraft industry (Hönlinger et al. (1998), Hörnlein and Stettner (1998), Schneider et al. (1999)), facility optimization in the water industry (Dandy et al. (1996), Jonoski et al. (1997)), crashworthiness optimization (Stander (1999)), and design problems (Eschenauer and Grauer (1999)) in the automotive industry. Due to the FEM-simulations, the assumptions on convexity and smoothness of objective and constraint functions are not valid any more. Therefore, optimization algorithms with local convergence properties are not applicable. Furthermore, a global optimum of the problem must be found. These characteristics lead to an excessive computation time for the solution of a problem of this kind, making non-sequential solution approaches inevitable. Non-sequential algorithms can be used to reduce computation time by performing time consuming computations in parallel, and they are also necessary to apply hybrid approaches to determine a global optimum (Boden and Grauer (1995)).

The computation of solutions to this class of problems typically requires to perform numerically complex FEM-simulations which quite often have to be repeated many times during the course of a mathematical optimization. In practice, optimization and FEM-simulation are usually separate software systems without the possibility to mutually "call" each other as subroutines. Therefore, it is necessary to couple them as two "black boxes", because source-code level integration is in general not possible.

Typically, the optimization module requests a FEM-simulation for every evaluation of the objective function and/or the constraints of the optimization problem. The computation time of a single FEM-simulation depends on the complexity of the simulated model and may range from a few seconds up to several hours. Many hundreds or thousands of these FEM-simulations lead to very long computation times, a sequential strategy on a sequential hardware architecture cannot cope with. The distribution of the entire optimization process on parallel computing hardware is inevitable. Instead of using "traditional" parallel hardware (MPP, vector computer), networks of workstations (NOW) can be used as a "virtual" parallel computer for distributed applications. This distribution can either be realized via parallel FEM-simulation and/or parallel optimization algorithms. In both cases, data-parallel as well as task-parallel approaches for FEM-simulation and/or optimization are appropriate. Our focus is put on task-parallelism in FEM-simulation due to the fact that domain decomposition methods executable on networks of workstations are not common in todays FEM-simulation systems. Especially in the case of optimization algorithms, an approach integrating data-parallelism and task-parallelism is preferred. A step towards this integration is presented in this paper by using several FEM-simulators (task-parallel) for the computations requested by an optimization algorithm (data-parallel). This optimization algorithm is a direct method using only the value of the objective function. The integration of task- and data-parallelism is realized by using a set of solutions (data-parallel) which are evaluated in parallel by multiple instances of a FEM-simulator (task-parallel). To utilize the computational resources of a network of workstations, the optimization algorithm must be designed in a way that almost all of the evaluations can be computed independently on the available workstations. A minimum sequential component of the algorithm is desired to improve scalability and efficiency. These properties enforce the usability of a distributed algorithm on a network of workstations.

The efficient use of networks of workstations as an economical hardware platform for such distributed scientific computing tasks (Livny and Raman (1998); Warren et al. (1997)) requires a resource management system (Pruyne and Livny (1995); Becker et al. (1995)) which should facilitate the computation of time-consuming processes on lightly loaded workstations. With an adequate load distribution strategy for a network of workstations, a reduction of the total computation time can be achieved. This is accomplished by selecting the most appropriate workstation for execution of a process instead of making a random selection. However, at the same time an interactive user on a selected host should be protected from an excessive loss of computational power due to the disturbance resulting from "foreign" load. Strategies for adaptive load distribution on a network of workstations have to take both of these requirements into account.

In this paper, a software architecture for supporting the distributed computation of simulation-based optimization problems in engineering is presented. This architecture provides basic functionality for interface management and synchronization between the functional components as well as resource management. The implementation consists of the OpTiX[1] system for optimization, and FEFLOW[2] as an example for a FEM-simulation package in engineering. Resource management in a network of workstations is handled by the WINNER system. To utilize the potential computational resources of a network of workstations with a non-sequential optimization algorithm, the concepts of a distributed direct optimization algorithm are proposed. Experimental results for optimizing a mathematical test problem to demonstrate both the properties of the algorithm and the efficiency of load distribution are discussed, and the computation of an industrial problem from water engineering is presented to show the feasibility of our approach for practical engineering problems.

The paper is organized as follows. In section 2 three optimization problem classes relevant in engineering are formally introduced and some of their typical applications are listed. In section 3, a software architecture for simulation-based optimization is presented. In section 4, the components of an implementation of the architecture are described. The concepts of a distributed direct optimization algorithm are described in section 5. In Section 6, experimental results for selected applications are discussed. Section 7 concludes the paper and discusses areas for future research.

[1]OpTiX is a trademark of Co.Com Ltd., Berlin, Germany
[2]FEFLOW is a trademark of WASY Ltd., Berlin, Germany

2 Classes of Optimization Problems in Engineering

Three classes of mathematical nonlinear constrained optimization and control engineering problems will be presented. In this section $f(\mathbf{x}) : I\!R^n \to I\!R$ denotes the objective function of an n-dimensional optimization problem. The m equality constraints are given by $g_j(\mathbf{x}) = 0, \forall j = 1, \ldots, m$ and k inequality constraints by $h_p(\mathbf{x}) \leq 0, \forall p = 1, \ldots, k$.

- The Problem of Optimal Design (POD)
  The class of optimal design or facility optimization problems can be stated as $\{\min f(\mathbf{x}) \mid \mathbf{x} \in \mathcal{U}_{POD}\}$ with the feasible domain:

$$\mathcal{U}_{POD} = \{\mathbf{x} \in I\!R^n \mid \mathbf{g}(\mathbf{x}) = \mathbf{0}, \mathbf{h}(\mathbf{x}) \leq \mathbf{0}\} \qquad (1)$$

  This is the class of nonlinear constrained static optimization problems.
- The Problem of Optimal Control (POC)
  Next, the class of optimal control problems is defined. The vector of time-dependent input variables of the control system consists of $\mathbf{z}(t)$ of input variables and $\mathbf{u}(t)$ of q decision variables. The output vector is denoted by $\mathbf{y}(t)$ and the state vector is given by $\mathbf{s}(t)$. The dynamic behaviour is described by a system of differential equations $\dot{\mathbf{x}} = \varphi(\mathbf{z}(t), \mathbf{s}(t), \mathbf{y}(t), \mathbf{u}(t), t)$ with start conditions and the given time horizon $M = [t_0, t_e]$. This control problem can – under certain conditions – be solved using the same algorithms as for solving (POD), if it is transformed to a discrete time problem (Veliov (1997)). This discrete problem forms the set $\mathcal{U}_{POC}$ of l feasible discrete controls:

$$\begin{aligned}\mathcal{U}_{POC} = \{\mathbf{u} \in I\!R^{q*l} \mid \\ &\mathbf{g}(\mathbf{z}(t), \mathbf{s}(t), \mathbf{y}(t), \mathbf{u}(t)) = \mathbf{0}, \\ &\mathbf{h}(\mathbf{z}(t), \mathbf{s}(t), \mathbf{y}(t), \mathbf{u}(t)) \leq \mathbf{0}, \\ &\dot{\mathbf{x}} = \varphi(\mathbf{z}(t), \mathbf{s}(t), \mathbf{y}(t), \mathbf{u}(t), t), \text{and} \\ &\mathbf{x}(t_0) = \mathbf{x}_0, t \in M\}.\end{aligned}$$

  The optimal control problem is then:

$$\min \int_{t_0}^{t_e} \phi(\mathbf{z}(t), \mathbf{s}(t), \mathbf{y}(t), \mathbf{u}(t), t)dt \quad \mathbf{u}(t) \in \mathcal{U}_{POC} \}$$

  The time-discretization of the decision variables transforms a problem from class (POC) to a higher $(q * l)$ dimensional problem of the class (POD).
- The Problem of Feedback Optimization (PFO)
  In contrast to the optimal control problem POC, the decision variables of a problem belonging to the class of feedback optimization problems do not depend directly on time. The time dependence is introduced via a feedback of the time-dependent output variables $\mathbf{y}(t)$. Thus, the decision variables can be defined as $\mathbf{u}(\mathbf{y}(t))$ and the set of feasible solutions is:

$$\begin{aligned}\mathcal{U}_{PFO} = \{\mathbf{u}(\mathbf{y}(t)) \in I\!R^q \mid \\ &\mathbf{g}(\mathbf{z}(t), \mathbf{s}(t), \mathbf{y}(t), \mathbf{u}(\mathbf{y}(t))) = \mathbf{0}, \\ &\mathbf{h}(\mathbf{z}(t), \mathbf{s}(t), \mathbf{y}(t), \mathbf{u}(\mathbf{y}(t))) \leq \mathbf{0}, \\ &\dot{\mathbf{x}} = \varphi(\mathbf{z}(t), \mathbf{s}(t), \mathbf{y}(t), \mathbf{u}(t), t), \text{and} \\ &\mathbf{x}(t_0) = \mathbf{x}_0, t \in M\}.\end{aligned}$$

The optimization problem in this case is:

$$\left\{ \min \int_{t_0}^{t_e} \phi(\mathbf{z}(t), \mathbf{s}(t), \mathbf{y}(t), \mathbf{u}(\mathbf{y}(t)), t)dt \mid \mathbf{u} \in \mathcal{U}_{PFO} \right\}.$$

With the decision variables $\mathbf{u}(\mathbf{y}(t))$ being not directly depending on time, this problem class can also be solved with the same algorithms as problems (POD) and (POC).

Industrial applications of these different types of optimization problems in various engineering domains are demonstrated e.g. for water engineering (Grauer et al. (1999)), for the automotive industry (Boden (1996); Weinert (1994)), in aircraft design and for metal forming processes (Grauer and Barth (1999)), and in chemical engineering (Grauer et al. (1978)).

3 Software Architecture for Distributed Simulation-Based Optimization

The previously presented problem classes and their solution processes yield various requirements for an integrated software environment supporting the solution of simulation-based optimization problems in engineering. Therefore, the computationally expensive numerical solution process and the necessity to integrate FEM-simulation packages with optimization essentially affect the software design. The following tasks for software engineering can be identified:

- **Wrapping of Legacy Systems**
  Mostly, numerical FEM-simulation software packages are written in programming languages like FORTRAN or C. To follow the object-oriented paradigm in software engineering, it is useful to integrate these software systems by wrapping them. This means to design and implement classes which transform a call to an interface method of that class to a call to the legacy software. If there is no application-level interface available, this "call" may be identical to the start of the program with appropriate parameters, e.g. in the form of an input file for a program.
- **Interface Management**
  Both the optimization and simulation software must provide interfaces to set and retrieve data. For instance, the optimization software must be able to set parameters of the FEM-simulation model according to certain values of decision variables of the optimization problem. Similarly, after a FEM-simulation run (i.e. the evaluation of constraints for a vector of decision variables) is completed, the optimization must obtain values for constraints from the simulation model. At present, most FEM-simulation systems, e.g. for structural mechanics, aerodynamics or aeroelastics (Cifuentes (1989); Schweiger et al. (1996)), provide only a file-based interface. Access to data of the simulation model via a programming interface is usually not possible.
- **Synchronization**
  If the optimization algorithm has requested the evaluation of a solution vector from the FEM-simulation system, the algorithm has to wait for the completion of the FEM-simulation. Vice versa, the FEM-simulation software has to wait for the next request after finishing the current one. Thus, between these two components at least two processes have to be synchronized, and the data exchange has to be coordinated.
- **Distributed Computation**
  For the distributed computation of the optimization and simulation tasks, networked high-performance workstations are a preferable platform (Boden (1996); Grauer and Barth (1997)), at least from an economic point of view. The coupled optimization/simulation system should be shielded from platform-specific hard- and software matters, such as inter-process/object communication, by an intermediate software layer (middleware).

A system architecture for coupling optimization and FEM-simulation software systems according to the above requirements is shown in Fig. 1. The two lower layers provide (platform-independent) functionality to start the individual components of the system and to provide the communication between them. These two layers implementing the middleware can be realized using an object-oriented approach like the Common Request Broker Architecture (CORBA, OMG (1998)) but alternatively, using an implementation of the non-object-oriented Message Passing Interface standard (MPI) (Gropp et al. (1994)), or the Parallel Virtual Machine (PVM) (Geist (1994)), is also possible. Besides this functionality, the two lower layers also offer functionality for load distribution.

The layer above implements the interface management functions to provide the basis for application-level communication, i.e. exchange of values for decision variables and constraints between the components. This layer encapsulates the application-specific interface, whether it is file-based or a programming interface, and makes a common interface for data exchange available, e.g. by creation and/or transformation of files. Furthermore, synchronization between the components will be handled in this layer.

Components like the pre- and postprocessor have their own (graphical) user interfaces. The topmost layer has to provide a user interface for the convenient formulation of the optimization problem (e.g. by allowing nodes of a finite element model representing constraints or decision variables to be selected graphically, or by enabling the user-friendly specification of the objective function) and probably additional visualization techniques for the results of the optimization. As a whole, this layer should present the components of a coupled optimization and FEM-simulation system consistently, and initiate and control the data flow between distributed components: from model generation in the preprocessing stage and FEM-simulation/optimization to the visualization of the optimization results in the postprocessing stage.

4 Components for Distributed Simulation-Based Optimization

In this section, the particular components required to solve the applications described in section 6. are presented. For the optimization, OpTiX (Brüggemann and Grauer (1991); Boden (1996); Brüggemann (1997)) is used to specify an optimization problem and a (distributed) strategy for the solution. As an example for a commercial FEM-simulation code in engineering, FEFLOW (Diersch (1998)) is integrated as the FEM-simulation component in the software architecture. FEFLOW provides simulation functionality for subsurface flow- and transport processes in groundwater. WINNER (Arndt et al. (ALV, 1998); Arndt et al. (PDCS, 1998); Arndt et al. (1999)) offers resource management facilities for a network of workstations.

- The Distributed Optimization Environment OpTiX
  OpTiX is an optimization environment supporting the entire process from problem formulation to distributed solution of a problem on networked workstations. The problem of optimizing a mathematical function can be formulated in a formal mathematical language. The

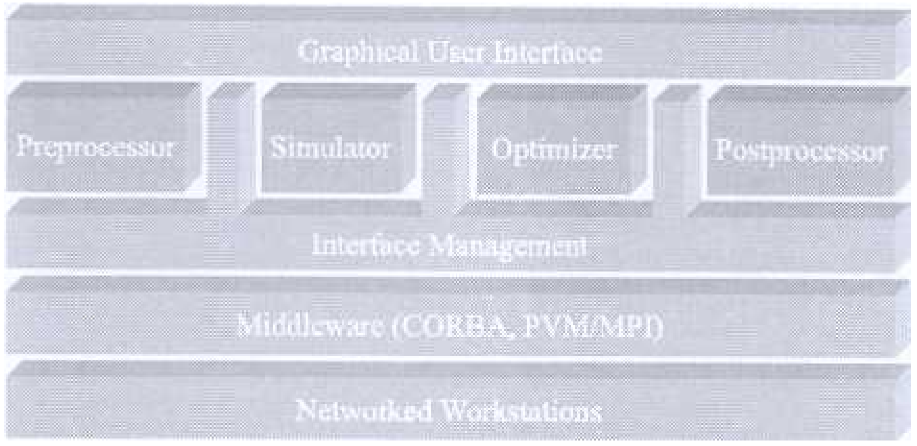


Figure 1. The software architecture for a distributed problem solving environment with coupled FEM-simulation and optimization.

combination of a problem and an appropriate optimization algorithm can be computed on any of the workstations in the network. The parallel solution of optimization problems is supported by allowing to (graphically) design so called "visual optimization schemes", i.e. manager/worker schemes for the parallel solution of a decomposed problem formulation or a hybrid approach where different algorithms solve the same problem in parallel and exchange their best solution. The object-oriented design of OpTiX offers the following benefits:

- flexibility concerning the problem formulation,
- integration ("wrapping") of non-object-oriented implementations of optimization algorithms,
- transparent management of the distributed computations across a network of workstations, and
- a graphical user interface for the formulation and the control of the solution of an optimization problem.

The first two aspects imply the existence of an abstract layer between the optimization algorithms and the problem formulation. This interface reduces algorithms to their essential attributes (stopping criteria, maximal iteration number etc.) and optimization problems to the objective function, constraints etc. These abstractions enable the combination of any problem with any algorithm (if appropriate) for its solution. Instances of algorithms can be implemented in C, C++ or FORTRAN, problem formulations can be analytical or simulation-based. The distributed computation of an optimization problem must be transparent to the user. Therefore, the user must have the opportunity to graphically design an optimization strategy (parallel, sequential) which is then employed on the network of workstations without further interaction with the user.

- The Groundwater Simulation System FEFLOW
FEFLOW (Finite Element Subsurface Flow and Transport Simulation System) is a 2D/3D simulator based on the finite-element method. Systems of equations for three-dimensional problems can be solved using direct or iterative methods. FEFLOW provides different solvers for this purpose.
For optimization purposes, the interface to the finite-element model is very important. Nodes of this FE-model represent locations of decision variables and constraints, and the optimization must have access to these nodes to set and get values. Besides the traditional interface based on files, the Interface Manager (IFM) of FEFLOW enables the loading of libraries (shared objects, dynamic link libraries) at runtime (Gründler (1997)). Furthermore, IFM provides a bi-directional interface: callbacks transfer control to external code at certain points in the FEFLOW-internal cycle (e.g. before or after every time step in the simulation). This allows control over the simulation, for instance to wait for a request using synchronous communication. Likewise, the interface provides access to the FE-model with functions e.g. to retrieve the hydraulic head in a certain node of the FE-model.
- The Resource Management System WINNER
The resource management system WINNER has been designed for typical Unix NOW environments, consisting of a central server and several workstations. It provides a basic load distribution service and some supplementary tools to enable users to efficiently execute different kinds of sequential and parallel applications.
The various tasks of the WINNER system are performed by three kinds of *manager* processes: system managers, node managers, and job managers (see Fig. 2) The system manager is the central server process of a WINNER network. Its duties include (a) collecting the load information of all respective workstations, (b) managing the currently active jobs, and (c) assigning hosts to job requests. On every host participating in a WINNER network, a node manager performs the tasks related to its machine. It periodically measures the host's utilization and reports it to the system manager. Furthermore, node managers are responsible for starting and controlling WINNER processes on their nodes, like e.g. reducing process priorities to protect console users (Arndt et al. (ALV, 1998)).
System and node managers run as daemon processes. In contrast, job managers are invoked by users to execute sequential and parallel jobs. Thus, job managers are part of WINNER's user interface. WINNER provides a variety of different job managers. Among them are: wrun for transparent remote execution of interactive applications, a parallel and distributed *make*-utility (wmake), and wpvm which supports parallel applications based on PVM. Their common duties are (a) acquiring resources from the system manager, (b) starting processes on the acquired nodes via the respective node managers, and (c) controlling the started processes.
Whenever a job manager requests resources for executing processes, the system manager selects those hosts which can currently provide the highest performance. This performance is estimated by taking both the base

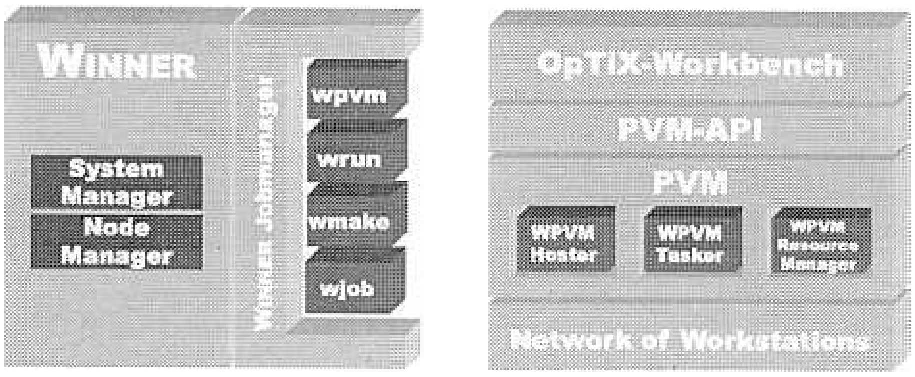


Figure 2. WINNER system structure (left) and scheme of integrating WINNER/wpvm-components with PVM and OpTiX (right).

speed of each workstation and its current load situation into account. Any additional resource constraints specified by the job manager (like a minimum amount of main memory) might reduce the set of suitable hosts. Workstations which are actively being used by console users are generally not used at all.

WINNER provides a job manager (wpvm) for being able to schedule PVM tasks. This job manager automatically selects a number of appropriate hosts for the PVM virtual machine (usually a tedious task when performed manually), starts the user's parallel application, and improves PVM's scheduling by utilizing WINNER's load distribution mechanism.

In order to avoid changes to the PVM system and to application programs, WINNER follows the approach introduced along with the CARMI system (Pruyne and Livny (1995)) for integrating a resource management system into PVM. This is achieved by registering several processes into the PVM infrastructure (for details see Arndt et al. (ALV, 1998)): The so-called PVM resource manager is the most important of these processes. Its main task is to replace PVM's default round-robin scheduling by WINNER's workload scheduling schemes. The hoster and tasker processes are responsible for starting the PVM daemons and PVM user processes, respectively, thus enabling wpvm to start and control all processes of the PVM application via the WINNER interfaces (Fig. 2, right).

Once invoked, a parallel application's master task will presumably spawn some PVM child processes (via pvm_spawn()), e.g. the OpTiX workbench spawns manager and worker processes (see section 6.). The default PVM scheduler uses a simple round robin scheme which yields good results in the case of unloaded, uniformly performing, single-processor workstations only. When using wpvm, the task placement decisions for these processes are automatically redirected to the resource manager which uses WINNER for choosing a suitable host for each task. This results in an enhanced scheduling scheme which respects the number of processors, different base speeds and the current load situation of each participating workstation.

The static structure of the coupling between FEM-simulation and optimization software in the case of OpTiX and FEFLOW is depicted in Fig. 3. The boxes represent distinct – maybe nested – components of the system, arrows between them specify a dependency, i.e. that the optimization components on the right side of the Fig. 3 use the components OpTiX and PVM. The FEM-simulator package contains the FEM-simulation system itself and the built-in interface management to external modules. This external module communicates with the optimizer package via PVM. The optimizer package comprises the problem description as well as the optimization algorithm. These parts are taken from the OpTiX system which is used in this context as a library without the graphical user interface for problem formulation etc. WINNER replaces the built-in PVM load distribution strategy. Therefore, all packages using PVM utilize WINNER implicitly.

5 A Distributed Direct Optimization Algorithm

The characteristics of optimization problems in engineering (constraints, non-convexity, non-smoothness, no reliable derivative information) raise the need for non-sequential, global, and direct optimization methods. One popular approach to introduce parallelism to a basically sequential algorithm is implementing parallel versions of frequently used operations, e.g. linear algebra operations on (sufficiently large) matrices (Bertsekas and Tsitsiklis (1989); El–Rewini and Lewis (1998)). This (fine-grained or data-parallel) technique enables the reuse of existing sequential code on parallel hardware with a certain speedup. One essential drawback of this approach is the limitation of this speedup according to Amdahl's law which restricts the theoretically achievable speedup due to the sequential component of the code. Hence, this kind of "parallel" algorithm is not likely to be scalable, i.e., to utilize an increasing number of resources (CPUs or hosts in a network of workstations) effectively. Consequently, inherently parallel, scalable algorithms are necessary which utilize any additional computational resources and adapt their solution strategy to the given resources and the problem dimension.

The proposed distributed direct optimization algorithm is based on the concepts of both the simplex-based parallel direct search method (Dennis and Torczon (1991); Kearsley et al. (1993)) for bound-constrained and the Complex Box method (Box (1965); Grauer (1987)) for constrained nonlinear optimization. Additionally, the problem of finding the global optimum is tackled by a hybrid approach combining the more global simplex/complex methods with (parallel) local search strategies to overcome the weakness of relatively slow local convergence of simplex/complex methods. The basic idea of the algorithm is the adaptation of the search strategy according to problem size and resources. To achieve this, different parameters of the algorithm are provided which determine the degree of parallelism, i.e., the number of parallel constraint and objective function evaluations per iteration and the multiplicity of search directions.

Both previously mentioned methods are based on geomet-

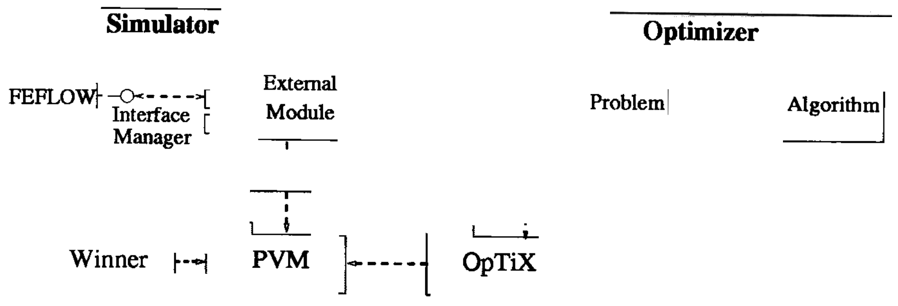


Figure 3. Components of the coupled simulation and optimization software (as UML component diagram)

rical operations (reflection, contraction) performed on the vertices of an (at least) $(n+1)$-dimensional polyhedron in the n-dimensional solution space of the optimization problem. The basic parameters are the size $s \geq n+1$ of the polytope, the number of solutions e which are modified using reflection and contraction, and the "look ahead" factor which controls the number of new solutions l generated from one solution. Hence, the number of new solutions per iteration is $l * e$ if no contraction operations are performed and $2 * l * e$ otherwise. Additionally, the point on which the solutions are reflected can also be varied. Reflection on the best solution restricts the search, whereas the reflection on the center of gravity of the polytope allows a search in all s directions given by the polytope and the center of gravity. In Fig. 4, various alternatives are illustrated. Different settings of the aforementioned parameters yield different search strategies.

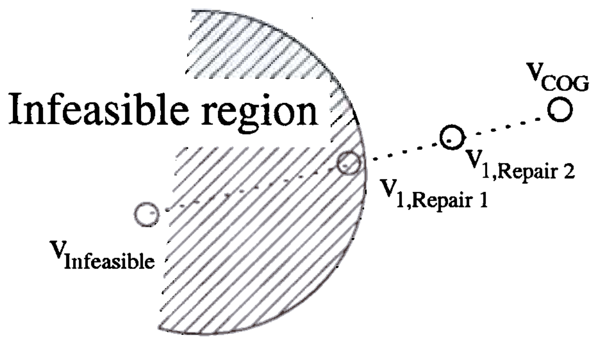


Figure 5. Repair of the infeasible solution v_1 using a parallel binary search along the direction towards the center of gravity v_{COG}. The first repair step yields the infeasible solution $v_{1,Repair1}$, the second (and final) step the feasible solution $v_{2,Repair2}$.

In case of constrained optimization problems there is an additional computational effort for an unpredictable number of repair operations to move infeasible solutions into the feasible region. This repair is basically a parallel binary search along the line between the center of gravity of feasible solutions and the solution to be repaired (see Fig. 5).

The relation of problem dimension n to available resources p can be used to distinguish between different strategies:

- $n \ll p$
  Offers highest possible degree of parallelism. e and l can be adjusted either to emphasize multidirectional search ($e > l$) or search preferably along the direction of the center of gravity. A common value for the polytope size is $s = 2 * n$.
- $n \approx p$
  The degree of parallelism is limited and the size of the polytope should be reduced to $n+1$. The values for e and l can be set according to $p \approx 2 * l * e$. The search direction should be restricted by using the best solution as the center of reflection.
- $n \gg p$
  Similar to the previous strategy, the degree of parallelism is further reduced. In the extreme case only the worst solution ($e = 1$) can be reflected on the best solution with a look ahead $l \approx p/2$.

The algorithm used for the solution of the subsequently presented optimization problems comprises the following steps:

1. **Initialization**
   The starting polytope with s randomly generated solution vectors is built and the constraints are evaluated. Infeasible solutions are repaired using a parallel binary search directed towards the center of gravity of feasible solutions.
2. **Exploration**
   The e worst solutions are reflected on the center of gravity. A reflection factor $\alpha < 1$ indicates a move (contraction) of the solution towards the center of gravity, $\alpha > 1$ indicates an expansion beyond the center of gravity. Each of these reflections is performed l times in parallel (s. Fig. 4). Infeasible solutions are repaired using the parallel binary search from the initialization phase. This generates $e*l$ new solutions from which the s best are selected for the next iteration.
3. **Local search**
   When the exploration is terminated (e.g. after the maximum number of iterations), a parallel local search starts from the best solution. It computes p (p number of hosts) random solutions in a sphere with radius r around the best solution. The radius is reduced if the local search fails to find a better solution. Infeasible solutions are rejected instead of repaired as in the previous phases. The local search stops after a given number of iterations or when the improvement is less than a given ϵ.

6 Applications and Results

In this section, basic characteristics of the proposed distributed optimization algorithm will be demonstrated by applying the algorithm to mathematical test problems and optimization problems from groundwater engineering. The benefits of adaptive load distribution using WINNER and OpTiX will also be shown with the distributed optimization of the mathematical test problem.

6.1. Mathematical Test Problem

In order to demonstrate the scalability of the algorithm, the well-known mathematical benchmark function based on the Rosenbrock function (Schittkowski (1980)) was used as a benchmark. The n-dimensional Rosenbrock problem is defined as

$$\min f(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100 * (x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right.$$

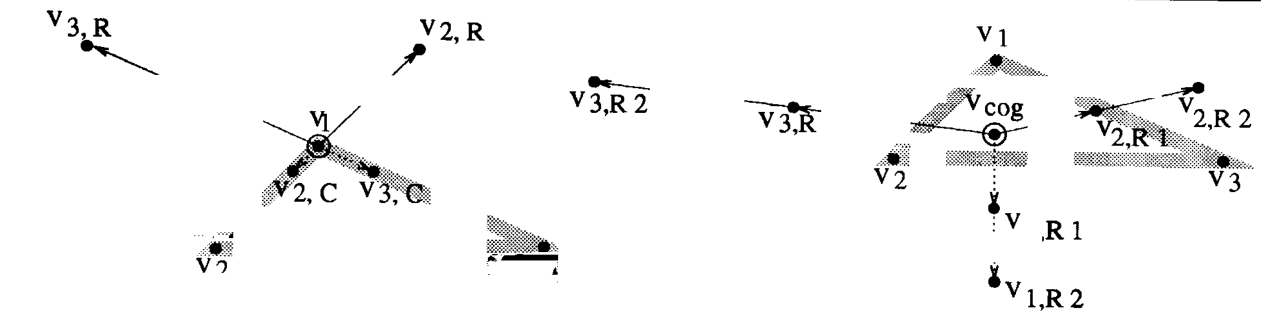


Figure 4. Parallel reflection and contraction of a 2-dimensional polytope with $s = 3$ solutions. At the top, the $e = 2$ worst solutions v_2 and v_3 are reflected $l = 1$ times on the best solution v_1 with factor 2 (solutions $v_{2,R}$ and $v_{3,R}$) and a contraction is performed with factor 0.75 (solutions $v_{2,C}$ and $v_{3,C}$). Beneath, all solutions are reflected $l = 2$ times with factor 2 (solutions $v_{2,R1}$, $v_{2,R2}$, $v_{3,R1}$, and $v_{3,R2}$) on the center of gravity v_{COG}

with $-3 \leq x_i \leq 3, \quad \forall i = 1, \ldots, n.$

The minimum is known to be $f(\mathbf{x}) = 0$ for $x_i = 1, \forall i = 1, \ldots, n$.

In our context this problem was only used as a nonlinear optimization problem whose problem dimension can be easily varied. In contrast to optimization problems in engineering, the evaluation of the objective and constraint function is negligible in terms of computation time. To "emulate" the conditions of engineering problems, the evaluation of the objective function was artificially delayed to three seconds each. This approximates the relation of two to three orders of magnitude between computation and communication time in the case of engineering problems. Assuming this relation, the communication cost for distributed computation is almost irrelevant. The test problems were considered as being solved when the best solution in the polytope was less than 10^{-6}. The starting points of the optimization runs were automatically generated symmetrically around the optimum to assure convergence and a comparable quality of the solutions for different problem dimensions. These tests are not intended to demonstrate any kind of global convergence property of the algorithm, but they serve to illustrate the scalability of the algorithm. All tests were performed on a heterogeneous network of Sun workstations ranging from 75MHz SparcStation 10 to 296MHz Ultra 30 connected by 10Mbit Ethernet or 155Mbit ATM. The algorithm's parameters l and e were held constant for all tests yielding a constant amount of work (i.e. evaluations of the objective functions) for the solution of each of the Rosenbrock problems. For the solution of the previously described mathematical optimization problem, the local search phase of the algorithm was not performed because it adds almost constant time to the computation time which is irrelevant when studying scalability properties.

In Fig. 6, the computation times for the solution of the 10-dimensional Rosenbrock problem are shown for an increasing degree of parallelism, i.e. increasing number of hosts in the workstation network. As expected, this curve is very close to the theoretically achievable linear speedup. The nature of the algorihm implies this behaviour because almost all computations within one iteration can be performed in parallel, which guarantees good utilization of any additional resources.

The computation times for increasing problem size and degree of parallelism are depicted in Fig. 7. For comparison, the computation times of the sequential Complex Box algorithm are also shown.

In Fig. 8, the relative speedup of the proposed method is shown. The speedup decreases for increasing dimension of the optimization problem.

The efficiency E = Speedup/Number of hosts (Kumar et

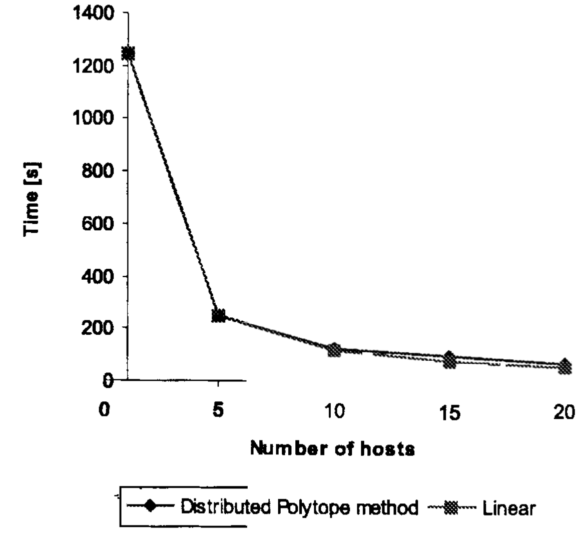


Figure 6. Computation times of the distributed Polytope method solving the 10-dimensional Rosenbrock problem with different degrees of parallelism compared to theoretical computation times assuming linear speedup.

al. (1994)) is a measure for utilization of resources. Fig. 9 depicts the efficiency of the algorithm for different problem dimensions and varying degree of parallelism. It can be observed that the efficiency decreases with increasing number of hosts. Generally, higher dimensional problems yield a lower efficiency for all evaluated numbers of hosts.

To investigate the benefits of load distribution, results from solving a decomposed formulation of the Rosenbrock problem are presented in the following. In earlier publications, the advantages of distributed approaches to the solution of mathematical optimization problems were demonstrated (Boden and Grauer (1995)). These promising results suggest a reduction of computation time of up to one order of magnitude. The decomposed Rosenbrock problem was solved by a combination of OpTiX for the computational part and PVM respectively WINNER/wpvm for the distribution of optimization processes. While OpTiX originally uses PVM for the start of its optimization processes and communication between them, it has been now combined with WINNER/wpvm. Therefore, the PVM strategy for starting processes on remote machines was replaced by the strategy of the wpvm-jobmanager. No changes have to be made either on the OpTiX- or on the WINNER/wpvm-side. The fol-

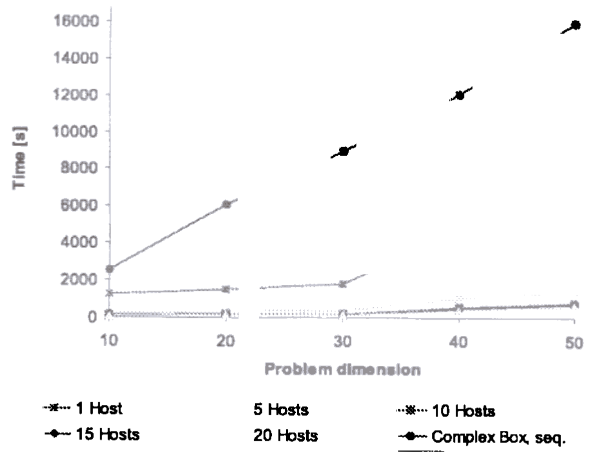


Figure 7. Comparison of computation times of the distributed Polytope method solving 10- to 50-dimensional Rosenbrock problems with different degrees of parallelism compared to computation times of the sequential Complex Box algorithm.

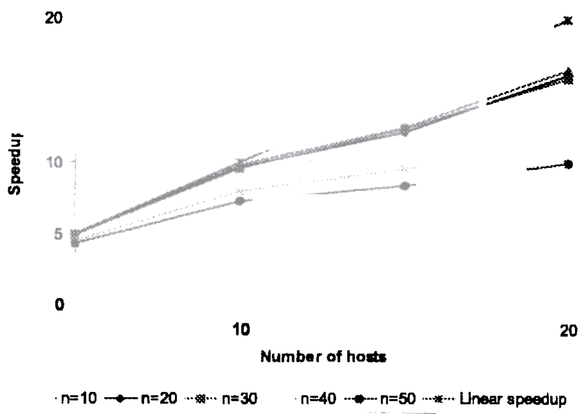


Figure 8. Relative Speedup of the distributed Polytope method for the solution of 10- to 50-dimensional Rosenbrock problems on 5 to 20 hosts compared to linear speedup.

lowing application shows that load distribution using WINNER/wpvm reduces the total solution time of a decomposed optimization problem in comparison to the built-in PVM host selection strategies.

The decomposed formulation of the Rosenbrock problem makes use of the approach that computation time is reduced if several (sub-)problems with a smaller dimension than the original n-dimensional problem are solved, even if the solutions of the subproblems must be combined for the solution of the original problem. In the case of the Rosenbrock function, the expression $100 * (x_{i+1} - x_i^2)^2$ prevents the independent solution of a subset of the sum, because indices i and $i+1$ occur in every addend. Therefore, a manager/worker scheme can be applied, in which the worker processes compute a solution for decision variables which are non-decision variables in the manager process and vice versa. These worker processes can be computed in parallel alternately to the computation of the manager process. Using this approach, the 100-dimensional Rosenbrock function can be decomposed into two worker processes with 33 decision variables and one worker process with 32 variables. The manager process uses these 98 decision variables as non-

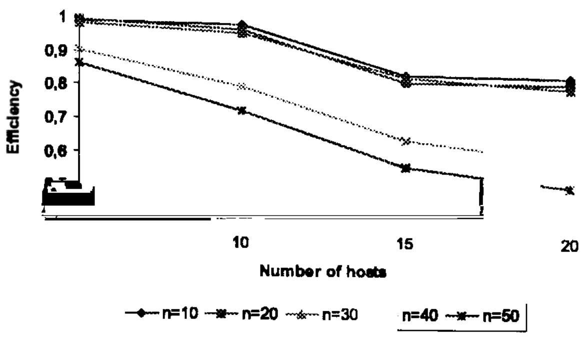


Figure 9. Efficiency of the distributed Polytope method for the solution of 10- to 50-dimensional Rosenbrock problems on 5 to 20 hosts.

decision variables and computes a solution for the remaining two decision variables, which yields the solution of the original 100-dimensional problem.

The parallel computation of the manager/worker scheme leads to four optimization processes, whereas the three worker processes run in parallel, and the manager process starts its computation when all workers have finished. Both the worker and manager problems were solved using an implementation of the Quasi Newton algorithm (Brüggemann and Grauer (1991)) with identical parameters.

In Tab. 1, the results of 10 runs under different load situations of the optimization are compared. On hosts **thales**, **pythagoras** and **euklid**, a FEM-simulation generated CPU load, the hosts **aristoteles**, **kepler**, **cusanus** and **avenarius** were idle. This scenario of hosts with and without load was used to simulate "real life" conditions in a network of workstations. The difference between the results of the different runs under the control of PVM can be explained as follows: since the subproblem solved by worker 3 is the computationally most expensive problem, the host selected to run this process determines the total computation time, because the manager process has to wait for all of the worker processes. The PVM strategy leads to considerably longer computation times if it selects a host already loaded. This selection is prevented by WINNER/wpvm as described in section 4.

6.2. A Groundwater Engineering Problem

The following optimization problem from groundwater engineering is an example for the class (POD). It is a minimization problem with limits on the acceptable rise of groundwater level. The problem can be stated as follows. The process of building a sluice in a local port in a city causes an increasing infiltration of surface water and therefore a rise of the groundwater level. To protect the tree population in a nearby park, this rise of the groundwater level has been restricted to 0.1 meters in each of five observation points. The objective function of the minimization problem is the sum of the quantity of water – as a measure for the operational costs – four pumps extract from the area in order to lower the groundwater level.

Formally, the optimization problem has four decision variables (the individual quantities of extracted water per day of four pumps), five (implicit) constraints (upper bounds of groundwater level in five observation points) and four (explicit) constraints (technical restrictions of the pumps).

The problem was solved using the proposed software architecture with the FEM-simulation system FEFLOW and the Polytope algorithm described in 5. as the optimization algorithm. The algorithm was performed on a varying number of hosts. In Table 2, the hereby computed solutions are

Table 1. Total computation times for the solution of the decomposed Rosenbrock problem using PVM and wpvm strategies for load balancing.

Hosts	Total time using PVM strategy [s]	Hosts	Total time using wpvm strategy [s]
ar-eu-ke-th	476.92	av-ar-cu-ke	274.73
eu-ke-th-cu	266.92	av-cu-ar-ke	269.65
ke-th-cu-py	425.09	ar-cu-av-ke	270.89
th-cu-py-av	268.73	av-ar-cu-ke	269.43
cu-py-av-ar	277.55	ar-cu-ke-av	279.40
py-av-ar-eu	469.18	ar-av-cu-ke	268.03
av-ar-eu-ke	267.78	cu-ar-av-ke	268.84
ar-eu-ke-th	466.58	cu-av-ar-cu	269.52
eu-ke-th-cu	268.80	av-cu-ar-ke	270.74
ke-th-cu-py	478.66	ar-cu-ke-av	278.37
Average time (abs.)	366.62	Average time (abs.)	271.96
Average time (rel.) %	100	Average time (rel.) %	74.18

compared with the reference solution and the best solution computed by the sequential Complex Box method.

The results show that all computed solutions yield an improvement of approximately 25%, but algorithms terminate with different solutions for different numbers of hosts. This is a result of the different degree of parallelism which affects the generation of the starting polytope and hence the whole solution process. This explains also the superlinear (relative) speedup when increasing the number of hosts from one to four. Generally, almost identical quality of the solution can be obtained in substantially less time using the distributed algorithm. It must be critically remarked that efficiency decreases for increasing degree of parallelism. This lack of performance should be rectified by an improved adaptation of the search strategy (i.e. adaptation of the factors l and e of the distributed Polytope method) to the number of hosts.

7 Conclusions

In this paper, an approach for the distributed computation of numerically complex, long running optimization problems in engineering applications on a network of workstations was presented. The basic functionality of the proposed architecture includes interface management, synchronization and resource management. As an instance of this architecture, a prototypical implementation was realized using the OpTiX system for optimization and the finite-element package FEFLOW for simulation; adaptive load distribution on a network of workstations was performed by the WINNER resource management system. With this prototype, the optimization of a mathematical test problem as well as industrial problems from water engineering were obtained. The results demonstrate the scalability properties of the Distributed Polytope method necessary for efficient exploitation of available resources in a network of workstations. The newly developed Distributed Polytope method is based on the ideas of (parallel) simplex-based algorithms (e.g. (Kearsley et al. (1993))). In contrast to the "traditional" strategy to parallelize an existing optimization algorithm by introducing parallel oeprations (e.g. parallel linear algebra operations Chio et al. (1995); Blackford et al. (1997)), this algorithm is designed inherently parallel to minimize the sequential component which reduces possible scalability. Therefore, the method integrates a parallel constraint-handling technique to repair infeasible solutions. Also, first steps towards a detailed scalability analysis are made (s. Barth et al. (2000)).

There are several issues for future work. For example, the prototype should be improved with respect to the computation of different classes of optimization problems in engineering – e.g. control problems – involving different kinds of FEM-simulation code, e.g. used in aircraft design. As indicated by the performance results, the efficiency of the distributed algorithm has to be improved. The adaptation of its search strategy to problem size and available resources must be integrated to utilize additional resources more effectively, possibly iso-efficient (Kumar et al. (1994)). This adaptation can be realized in terms of varying polytope size and the parameters for reflection and contraction. Finally, as an alternative for distributed computing based on PVM message passing, the use of CORBA (equipped with adaptive load distribution) as a middleware could be envisaged (Barth et al. (1999)).

Acknowledgements

This research is partially funded by the German Federal Ministry of Science and Technology under contract FKA/FUEGO 28/05/1998. Furthermore, the authors would like to thank H. Diersch, S. Kaden and I. Michels from WASY Ltd., Berlin for their cooperation and support.

References

Anderson, T. E., Culler,D. E., Patterson, D. A. and the NOW Team, "A Case for NOW (Networks of Workstations)",*IEEE Micro*, 15(1), 1995, pp. 54-64.

Arndt, O., Freisleben, B., Kielmann, T., Thilo, F., "Dynamic Load Distribution with the WINNER System", In: *Proceedings Workshop Anwendungsbezogene Lastverteilung (ALV'98)* Munich, Germany, SFB 342/01/98, pp. 77-88, 1998.

Arndt, O., Freisleben, B., Kielmann, T., Thilo, F., "Scheduling Parallel Applications in Networks of Mixed Uniprocessor/Multiprocessor Workstations", *Proceedings Parallel and Distributed Computing Systems (PDCS98)*, Chicago, 1998, pp. 190-197.

Arndt, O., Freisleben, B., Kielmann, T., Thilo, F., "Batch Queueing in the WINNER Resource Management System", In: *Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, CSREA Press, Las Vegas 1999, pp. 2523-2529.

Blackford, L., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D. Whaley, R., *ScaLAPACK Users' Guide*, Society for Industrial and Applied Mathematics, 1997.

Barth, T., Flender, G., Freisleben, B., and Thilo, F., "Load Distribution in a CORBA Environment", in: *Proc. of Int'l Symposium on Distributed Object and Application 99*, IEEE Press, Edinburgh 1999, pp. 158–166.

Table 2. Comparison of the reference and the optimal solution for the water engineering problem using the Copmplex Box method and the distributed Polytope method with different degree of parallelism.

Pumps	Values of the decision variables [m^3/day]								
	Reference solution	Complex Box	Distributed Polytope method						
			1 Host	4 Hosts	7 Hosts	9 Hosts	12 Hosts	14 Hosts	17 Hosts
1	400	282.2	274.317	274.317	274.317	399.64	394.396	400.125	
	400	479.6	417.572	417.572	417.572	198.834	198.544	257.554	
3	400	323.6	234.611	234.611	234.611	372.51	370.306	288.688	
4	400	109.1	279.964	279.964	279.964	242.131	249.502	264.19	
Value of the objective function	1600 100%	1194.5 74.66%	1206.46 75.4%	1206.46 75.4%	1206.46 75.4%	1213.11 75.82%	1212.75 75.8%	1210.56 75.66%	
Elapsed time [s]		23399	37263	8572	5917	5284	6195	3851	
Relative speedup	-	-	-	4.35	6.3	7.05	6.02	9.68	9.46
Efficiency	-	-	1	1.09	0.9	0.78	0.5	0.69	0.56

Barth, T., Freisleben, B., Grauer, M., Thilo, F., "A Scalable Algorithm for the Solution of Simulation-based Optimization Problems", *Proc. Int. Conf. On Par. and Dist. Programming Techniques and Applications (PDPTA '2000)*, Las Vegas 2000, pp. 469-475.

Becker, D., Sterling, T., Savarese, D., Dorband, J., Ranawake, U., Packer, C. "BEOWULF: A Parallel Workstation for Scientific Computation", In: *Proceedings of the 1995 International Conference on Parallel Processing (ICPP)*, 1995, pp. 11–14.

Boden, H., Gehne, R., Grauer, M., "Parallel Nonlinear Optimization on a Multiprocessor System with Distributed Memory", in: Grauer, M., Pressmar, D. (eds.), *Parallel Computing and Mathematical Optimization*, Springer, 1991, pp. 65–78.

Boden, H., *Multidisciplinary Optimization and Cluster Computing*, (in German), Springer/Physica Verlag, Heidelberg 1996.

Boden, H., Grauer, M., "OpTiX-II: a Software Environment for the Parallel Solution of Nonlinear Optimization Problems", *Annals of OR*, J.C. Baltzer Science Publisher, 1995, pp. 129-140.

Box, M., "A New Method of Constrained Optimization and a Comparison with Other Methods", *Computer Journal*, 1965, Vol. 8, pp. 42–52.

Brüggemann, F., *Object-Oriented and Distributed Solution of Optimization Problems*, (in German), Springer/Physica Verlag, Heidelberg 1997.

Brüggemann, F., Grauer, M., "Optix – an Object-Oriented Environment for Parallel Optimization", in: Grauer, M., Pressmar, D. (eds.), *Parallel Computing and Mathematical Optimization*, Springer, 1991, pp. 133–153.

Bertsekas, D., Tsitsiklis J., *Parallel and Distributed Computation*, Prentice–Hall, 1989.

Choi, J., Dongarra, J., Ostrouchov, S., Petitet, A., Walker, D., Whaley, R., *LAPACK Working Note 100 – A Proposal for a Set of Parallel Basic Linear Algebra Subprograms*, `http://www.netlib.org/utk/papers/pblas/pblas.html`, 1995.

Cifuentes, A. O., *Using MSC/NASTRAN: Statics and Dynamics*, Springer-Verlag, 1989.

Dandy, G. C., Simpson, A. R., Murphy, L. J., "An Improved Genetic Algorithm for Pipe Network Optimization", *Water Resources Research*, Vol.32, No.2, 1996, pp. 449-458.

Diersch, H.–J., *Interactive, Graphics-Based Finite-Element Simulation System FEFLOW for Modeling Groundwater Flow Contaminant Mass and Heat Transport, Users Manual*, WASY GmbH, Berlin, 1998.

Dennis, J, Torczon, V., "Direct Search Methods on Parallel Machines", *SIAM Journal on Optimization*, 1 (1991), pp. 448-474.

Eschenauer, H., Grauer, M., "Decomposition and Parallelization Strategies for Solving Large-Scale MDO problems", in: *Design Optimization*, Vol. 1, No. 1, MCB University Press 1999, pp. 24-43.

El–Rewini, H., Lewis, T., *Distributed and Parallel Computing*, Manning Publications, 1998.

Geist, A., *PVM: Parallel Virtual Machine – A Users Guide and Tutorial for Network Parallel Computing*, MIT Press, 1994.

Giesing, J. P., Barthelemy, J. F., "A Summary of Industry MDO Applications and Needs", 7-th *AIAA/USAF/NASA/ISSMO*, St. Louis, 1998.

Göpfert, A, Bittner, L., et. al., *Lexikon der Optimierung*, (in german), Akademie–Verlag Berlin, 1986.

Gropp, W., Lusk, E., Skjellum, A., *Using MPI: Portable Parallel Programming with the Message–Passing Interface*, MIT Press, 1994.

Grauer, M., "Optimierung verfahrenstechnischer Systeme", in: Wei"s, S., Berghoff, W. u.a. (Hrsg.), *Verfahrenstechnische Berechnungsmethoden*, (in German), Part 6, Section 4, pp.127–129, Verlag Chemie, Weinheim 1987.

Grauer, M., Barth, T., "Multidisciplinary Optimization and Cluster Computing using the OpTiX-Workbench", *Proceedings Conference on Optimization in Industry*, Palm Coast, Florida 1997, pp.73-80.

Grauer, M., Barth, T., "Cluster Computing for Treating MDO–Problems by OpTiX", in: **Mistree, F.**, Belegundu, A. (eds.), to appear in: *Proc. Conference on Optimization in Industry* II, Banff/Canada, June 1999.

Grauer, M., Barth, T., Kaden, S., Michels, I., "Decision Support and Distributed Computing in Groundwater Management", in: Savic, D., Walters, G. (eds.): *Water Industry Systems: Modelling and Optimization Applications*, Research Studies Press, 1999, pp. 23-38.

Grauer, M., Gruhn, G., Richter, C., "Design and Application of the GRG-Algorithm for Process Optimization", 11th *Conference on Computers and Chemical Engineering* (CACE'78), Paris 1978.

Gründler, R., *Interface Manager – Extensions and Programming Interface for FEFLOW*, Technical Paper, Wasy, 1997.

Hönlinger, H.G., Hutin, P.-M., Pendleton, E.W., "The Benefits of the Passive and Active Aeroelastic Design of Aircraft Structures", *Proc. of Optimization in Industry* I, Belegundu, A., Mistree, F. (eds.), ASME 1998.

Hönlinger, H.; Voß, R., "Dynamics of Flexible Aircraft (DYNAFLEX) An Innovative Cooperation between the Industry, Research Establishments, and Universities". *CEAS/AIAA/ICASE/NASA Langley: Int. Forum on Aeroelasticity and Structural Dynamics* 1999, Williamsburg, VA / USA, June 22-25, 1999

Hörnlein, H.R.E.M., Stettner, M., "Structural Design Process: Projects - Programs - Prospects", *Proc. of Optimization in Industry* I, Belegundu, A., Mistree, F. (eds.), ASME 1998.

Jonoski, A., Zhou, Y., Nonner, J., Meijer, S., "Model-Aided Design and Optimization of Artificial Recharge-Pumping Systems", *Hydrogeological Science Journal*, 42(6), 1997.

Kearsley, A. T., Tapia, R. A., Torczon, V., "On the Use of Parallel Direct Search Methods for Nonlinear Programming Problems", *Technical Report 93-33*, Rice University, Houston 1993.

Kumar, V., Grama, A., Gupta, A., Karypis, G., *Introduction to Parallel Computing – Design and Analysis of Algorithms*, Benjamin/Cummings Publishing, 1994.

Livny, M., Raman, R., "High-Throughput Resource Management", In: *The GRID: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1998, pp. 311-337.

Mowbray, T. J., Zahavi, R., *The Essential CORBA: Systems Integration Using Distributed Objects*, John Wiley & Sons, 1995.

The Common Object Request Broker: Architecture and Specification - Revision 2.2, Object Management Group, (`ftp://ftp.omg.org/pub/docs/formal/98-07-01.ps`), 1998.

Pruyne, J., Livny, M., "Parallel Processing on Dynamic Resources with CARMI", In: Feitelson, D. G., Rudolph, L. (Hrsg.), *LNCS 949*, Springer, 1995, pp. 259-278.

Rosenbrock, H. H., "An Automatic Method for Finding the Greatest or Least Value of a Function", *The Computer Journal* 3, 1960, pp. 175–184.

Schittkowski, K., *Nonlinear Programming Codes*, Springer, 1980.

Schneider, G., van Dalen, F., Krammer, J., Stettner, M., "Multidisciplinary Wing Design of a Regional Aircraft regarding Aeroelastic Constraints", to appear in: *Proc. of Optimization in Industry* II, Banff/Canada, June 1999.

Schweiger, J., Krammer, J., Hörnlein, H.R.E.M., "Development and Application of the Integrated Structural Design Tool LAGRANGE", *AIAA-96-4169*, 1996.

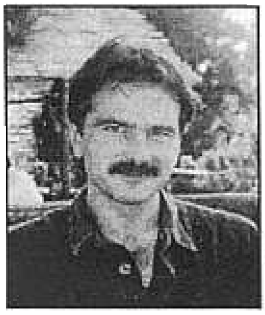
Stander, N., "Crashworthiness Optimization using Response Surface Methodology and Massively Parallel Programming", to appear in: *Proc. of Optimization in Industry* II, Banff/Canada, June 1999.

Veliov, V., "On the Time-Discretization of Control Systems", *SIAM J. on Contr. & Opt.*, Vol.35, No.5, 1997, pp. 1470-1486.

Warren, M., Becker, D., Goda, M., Salmon, J. and Sterling, T., "Parallel Supercomputing with Commodity Components", *Proc. Int. Conf. on Parallel and Distrib. Processing Techniques and Applications*, 1997, p.1372–1381.

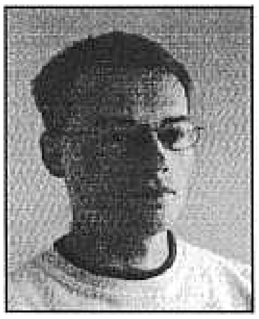
Weinert, M., *Sequential and Parallel Strategies for the Optimum Design of Complex Shells of Revolution*, (in German), TIM-Report T05–05.94, University Press Siegen, Siegen 1994.

***Thomas Barth** received his diploma in computer science from the Darmstadt University of Technology, Germany, in 1994. Since 1995 he is working at the Dept. of Information Systems at the University of Siegen towards his PhD thesis. His research is focused on distributed computing and development of scalable, distributed algorithms for simulation-based optimization.*



***Bernd Freisleben** is associate professor of computer science in the department of electrical engineering and computer science at the University of Siegen, Germany. He received the master's degree in computer science from the Pennsylvania State University, USA, in 1981, and the Ph.D. degree in computer science from Darmstadt University of Technology, Germany, in 1985. His research interests include network computing, parallel programming, and computational engineering.*

***Manfred Grauer** received a diploma degree (Dipl.-Ing.) in engineering/cybernetics in 1970 from MIT (Moscow Institute of Technology, Moscow, Russia), PhD (Dr. Ing) in 1975 and the Habilitation (Dr. sc. techn.) in 1979, both from the Technical University of Merseburg, Germany. Since 1989 he is the head of the Dept. of Information Systems at the Faculty of Economics of the University of Siegen, Germany. His research interests include the design of information systems currently focusing on multimedia systems and cluster computing.*



***Frank Thilo** received his diploma in computer science from the University of Siegen, Germany, in 1998. Currently, he is working towards his Ph.D. thesis. His research interests include cluster computing, resource management, and load distribution.*