Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

Exploiting parallelism in general purpose optimization

G. Venter & B. Watson
Vanderplaats Research and Development, Inc.
Colorado Springs, CO, USA

Abstract

VisualDOC, a general purpose design optimization code that is commercially available from Vanderplaats Research and Development, Inc. (VR&D), is used as a test-bed for evaluating the efficiency of parallelizing current optimization algorithms. The finite difference gradient calculations of VisualDOC are implemented in parallel form and these changes are evaluated using a typical aircraft wing example problem. VisualDOC provides three algorithms for performing constrained nonlinear optimization and the investigation focuses on the influence of the optimization algorithm and the number of design variables on the efficiency of performing the optimization in parallel. The need to additionally parallelize the one-dimensional search calculations is also investigated. The Local Area Multiprocessor (LAM) system, originally developed at Ohio State University, has been configured on VR&D workstations to allow them to be used as a parallel processing computer, referred to as a virtual parallel machine. This group of workstations consists of a combination of UNIX and Windows NT workstations. The LAM system contains an implementation of the MPI standard for message passing that allows for dynamic load balancing.

1 Introduction

Despite many years of research, resulting in the availability of several general purpose optimization programs, optimization has only realized limited success in the industrial environment. There are many reasons for this lack of acceptance, including (a) lack of user familiarity with optimization concepts, and (b) immense computational resource requirements for general purpose optimization.

The first issue is due to the fact that optimization is rarely taught at the undergraduate level, creating the need for user training, that companies are often unwilling to invest in. The VisualDOC program (e.g., [1]), from VR&D, was created to address this issue. It provides an intuitive graphical user interface, guiding the user through the steps required to set up an optimization problem. This environment allows engineers to apply optimization to real problems with minimal training requirements. The high computational resource requirement of general purpose optimization stems from the "general purpose" nature of the problem and is the focus of this paper.

Recognizing that many robust and efficient general purpose optimization techniques are gradient-based, this paper addresses the computational resource requirement of existing gradient-based optimization algorithms. In general, gradient-based optimization algorithms reach an optimum design point by moving from one design point to the next. This process of moving from one design point to the next typically consists of calculating the gradient values of the objective function and active constraint set to obtain a search direction, followed by a one-dimensional search in that search direction. The one-dimensional search determines how far to move in the search direction and identifies the next design point where gradient calculations will be performed. Because typical analysis packages do not generally provide gradients, most general purpose optimizers employ finite difference gradient calculations to obtain the gradients, and VisualDOC is no exception.

Engineers typically use tools that were developed for performing only a single analysis. To perform optimization using these tools, a large number of analyses are required, either to provide gradient information via finite difference calculations, or to provide data for response surface or other non-gradient based optimization methods. A typical industrial analysis can require many hours of computer time. Given the time constraints that are placed on design engineers, this makes many potential optimization problems impractical.

Parallel processing has the potential to reduce the time requirements such that general purpose optimization becomes practical for a wide range of industrial applications. The objective of this study is to develop techniques to use existing optimization algorithms to gain maximum efficiency from parallelization. Several previous studies have focused on the parallelization of the gradient calculations, with moderate success (e.g., Rogers [2], Sikiotis [3], El-Sayed [4] and Watson [5]).

When the number of design variables increases, the bulk of the computational time required to complete the optimization is consumed by the finite difference gradient calculations. A set of gradients is calculated at each design iteration to determine a search direction that is used by the optimization algorithm during the one-dimensional search. In VisualDOC, the default is to use forward finite difference calculations, which requires as many analyses as there are independent design variables for each set of finite difference calculations. Since the analyses required during the finite difference calculations are independent of each other, these calculations may be easily performed in parallel. When the number of parallel processors is greater or equal to the

number of independent design variables, each set of finite difference calculations may be performed in the same time it takes to complete a single analysis.

An aspect of optimization that, to the authors' knowledge, has never been fully investigated or exploited for parallel processing is the one-dimensional search calculations. Most researchers have ignored parallelizing the one-dimensional search calculations since it is more challenging than parallelizing the finite difference gradient calculations, because the one-dimensional search is inherently a sequential process. The present paper will investigate the need to parallelize the one-dimensional search calculations in current optimization algorithms.

The freely available LAM system, developed by Ohio State University, is a set of programs and libraries that allows a cluster of workstations connected with a local area network to be used as a parallel processing computer. LAM contains an implementation of the MPI standard for message passing that allows for dynamic load balancing. The LAM system was used to develop and test a parallel version of VisualDOC using existing UNIX and Windows NT workstations available at VR&D.

2 Parallelism in the optimization process

The computational time of a gradient-based general purpose optimization algorithm may be divided into three main parts:

- Analyses required for gradients
- Analyses required for one-dimensional search
- Other optimization computations

For problems with moderate numbers of design variables, the time to complete the analyses, both for gradient and one-dimensional search calculations, dominates the total solution time. Additionally, the number of one-dimensional search calculations is fairly independent of the number of design variables. However, the number of analyses required to perform the finite difference gradient calculations increases with the number of design variables and will dominate the total solution time for any problem with more than just a few design variables.

A typical commercial analysis can take many hours and it is currently impractical to attempt optimization of this class of problems using more than a small number of design variables. Parallel processing provides an opportunity to reduce the total solution time, and make general purpose optimization practical for large numbers of design variables.

3 Parallel implementation of VisualDOC

The present work will investigate the efficiency of parallelizing only the finite difference gradient calculations and will make recommendations with respect to

Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

parallelizing the one-dimensional search calculations. The finite difference gradient calculations are independent of each other and are easily parallelized.

A function, F that depends on a single variable, x, may be used to introduce finite difference gradient calculations. The gradient of F with respect to x at the point x_0 is defined as

$$\left.\frac{dF(x)}{dx}\right|_{x=x_0} = \lim_{x \to x_0} \frac{F(x) - F(x_0)}{x - x_0} \tag{1}$$

The gradient of F with respect to x may be approximated at the point x_0, by perturbing x with a small value, h, as follows:

$$\left.\frac{dF(x)}{dx}\right|_{x=x_0} \approx \frac{F(x_0 + h) - F(x_0)}{h} . \tag{2}$$

The right hand side of Eqn. (2) is the forward finite difference formula for the gradient of F with respect to x at $x=x_0$. In the case where F is a function of many variables, the right hand side of Eqn. (2) is repeated for each variable, resulting in a number of function evaluations that are independent of each other. It is thus fairly easy to parallelize the finite difference gradient calculations, in which case a number of processors equal to the number of design variables may be utilized.

VisualDOC performs a set of gradient calculations after each design cycle where the optimization algorithm made progress towards the optimum solution during the one-dimensional search. To perform the gradient calculations, VisualDOC perturbs the design variables one at a time, and calls the analysis module to perform the required analysis. This process was parallelized by changing the analysis module within VisualDOC to accept a set of perturbed design variables and perform the required analyses in parallel by distributing the analyses to the available processors. The analysis module was parallelized using a master-slave paradigm where the master process allocates the tasks to all available slave processors (see e.g., Smith [6]). When a slave finishes its task, it becomes available again, and can be allocated another task. This paradigm is ideally suited to a heterogeneous parallel environment, such as a local area network of workstations, because it is intrinsically dynamically load balanced. That is, faster processors will be allocated more tasks. Also, this scheme requires only minimal inter-processor communication. The design variable values are sent to the slaves, and the response values are sent back to the master. A single slave process running on the same processor as the master process performs all the remaining analyses required during the optimization.

4 Example problem

To test the effectiveness of performing the finite difference gradient calculations in parallel, structural optimization of a typical aircraft wing was considered as an example problem. The wing structure considered is constructed of aluminum and has a length of 70 *ft*. A finite element model of the wing was constructed to evaluate the required stresses, displacements and frequency constraints. The finite element analyses required during the optimization process were performed

Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

using GENESIS (e.g., [7]). The finite element model consisted of 2,400 two-dimensional shell elements (CQUAD4), 600 one-dimensional truss elements (CROD) and has a total of 1,917 nodes. This finite element model is shown graphically in Fig. 1.

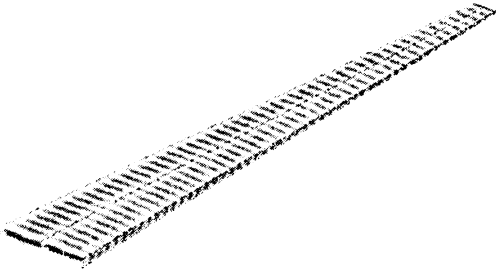

Figure 1: Finite element model for the wing example problem.

Three load conditions, typical of an aircraft wing, are considered during the optimization as follows:

- **Load case 1 (Static):** Normal lift and engine weight
- **Load case 2 (Static):** Landing, half lift and engine weight
- **Load case 3 (Frequency):** Fundamental frequency

The optimization problem is then defined as minimizing the mass of the wing with the three load cases applied and subject to stress, displacement and natural frequency constraints, resulting in a total of 21,648 constraints. The problem has a total of twenty four design variables, with each design variable representing the thickness value of a group of shell elements. All design variables have a lower bound of 0.02 *in*, an upper bound of 1.00 *in* and an initial value of 0.2 *in*.

The GENESIS software was used to perform the finite element analyses (i.e., function evaluations) required during the VisualDOC optimization. However, although GENESIS provides finite element analysis capabilities, it is primarily a powerful *structural optimization* tool. To obtain a baseline optimum design for validating our VisualDOC results, the wing structure was optimized using GENESIS. GENESIS found an optimum design that satisfied all the constraints with a mass of 9,553.76 *lb*.

5 Results

The VisualDOC wing optimization was performed on a heterogeneous cluster of six workstations consisting of three SUN workstations, two SGI workstations and a Windows NT workstation. These machines were linked into a virtual parallel machine using the MPI message passing protocol implemented in the LAM software. All the workstations, except for two of the SUN workstations, had different configurations and the parallel runs were distributed using dynamic load balancing. Two cases were considered, with the first case having twelve

design variables and the second case twenty four design variables. For each case, the three optimization algorithms of VisualDOC were considered resulting in a total of twelve optimizations. The three optimization algorithms considered were (1) the modified method of feasible directions (MMFD), (2) sequential linear programming (SLP), and (3) sequential quadratic programming (SQP).

For each of the twelve optimization runs the total number of analyses, the total number of analyses performed in parallel and the total time to complete the optimization were recorded. Additionally, the total time to complete each optimization if performed in a serial manner on the slowest and fastest workstations respectively were estimated. These times were estimated by multiplying the average time to complete five analyses by the total number of analyses required to complete each optimization. This information was used to evaluate the efficiency of performing the optimization in parallel.

5.1 Case 1: Twelve design variables

As mentioned in the example problem description, the original example problem has a total of twenty four design variables. For the twelve design variable case considered here, the first twelve design variables were set to their initial values, while the remaining design variables were kept at their optimum values as obtained from the baseline optimum obtained from GENESIS (see Section 4). Changing only twelve of the original twenty four design variables from their baseline optimum values ensured that the optimum of the VisualDOC twelve design variable case would correspond to the GENESIS twenty four design variable case. By comparing the optimum results we ensured that all three of the VisualDOC optimization algorithms did converge. The optimum results obtained from the three VisualDOC optimization algorithms are summarized in Table 1 and the timing information in Table 2.

Table 1: Twelve design variable case results.

	MMFD	SLP	SQP
Mass [*lb*]	9,541.41 (-0.13%)	9,552.60 (-0.01%)	9,552.99 (-0.01%)
Design Cycles	8	18	11
Total Analyses	141	239	164
Parallel Analyses	84	216	132

(Values in parentheses are the present difference with respect to the optimum mass found by GENESIS)

Table 1 demonstrates that all three VisualDOC optimization algorithms resulted in well converged optimum solutions that correlates well with the optimum found by GENESIS. Comparing the results, it is clear that the MMFD algorithm required a larger percentage of one-dimensional search calculations with respect to the total number of analyses (40%) compared to 10% for the SLP and 20% for the SQP algorithms. Since the one-dimensional search calculations

Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

were not parallelized, the SLP and SQP algorithms thus resulted in more efficient parallel algorithms as compared to the MMFD algorithm.

Table 2: Total time to complete the twelve design variable case.

	MMFD	SLP	SQP
Parallel Time [s]	7,479	6,753	5,964
Slowest Serial Time [s]	31,020 (4.15)	52,580 (7.79)	36,080 (6.05)
Fastest Serial Time [s]	12,267 (1.64)	20,793 (3.08)	14,268 (2.39)

(Values in parentheses are the speedup factor between the parallel and serial optimizations)

This higher efficiency is clearly illustrated by the total time to complete the respective optimizations in parallel and the speedup factors as summarized in Table 2. The SQP algorithm required 16% more analyses but took 20% less time to complete as compared to the MMFD algorithm, while the SLP algorithm required 70% more analyses but took 10% less time as compared to the MMFD algorithm. Additionally, the SLP algorithm required 46% more analyses than the SQP algorithm, but was only 13% slower. Finally, the SLP algorithm had the highest speedup factors, while the MMFD algorithm had the lowest speedup factors.

Apart from the percentage of total analyses required to complete the one-dimensional search calculations, one should also consider the average number of one-dimensional search calculations required for each design cycle. If the one-dimensional search calculations could be performed in parallel, this ratio would ideally be equal to 1.0. For the MMFD algorithm this ratio is 7.1, for SLP it is 1.3 and for SQP it is 2.9. It is thus clear that the MMFD algorithm would benefit the most from parallelizing the one-dimensional search calculations while the impact on the SLP algorithm would be minimal.

The higher parallel efficiency of the SLP and SQP algorithms can be explained by the fact that the linear (in the case of SLP) and quadratic (in the case of SQP) sub-problems constructed during the optimization are used in the one-dimensional search. The one-dimensional search calculations are mostly based on linear or quadratic approximations of the actual function values at the current design point and a smaller number of actual function evaluations are required during the one-dimensional search. However, note that when the optimizations are performed in a serial manner, the MMFD algorithm is the most efficient since it requires the smallest number of total analyses.

The number of parallel analyses distributed to each node in the virtual parallel machine is shown graphically in Fig. 2. Figure 2 clearly illustrates the dynamic load balancing property of the present parallel implementation where the fastest processors (nodes 2 and 5) performed the largest number of analyses while the slowest processors (nodes 3 and 6) performed the smallest number of analyses.

Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

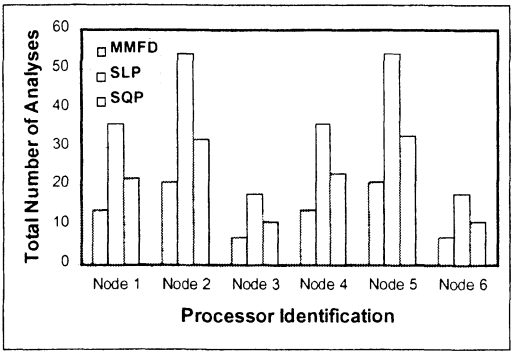

Figure 2: Finite element model for the wing example problem.

5.2 Case 2: Twenty four design variables

For the twenty four design variable case all the design variables of the original example problem were considered and set to their original values to start the different optimizations. Again, the original example problem was used to ensure that each optimization algorithm did indeed converge to the optimum design point. The results obtained for the twenty four design variable case are summarized in Table 3 and the timing information in Table 4.

Table 3: Twenty four design variable case results.

	MMFD	SLP	SQP
Objective [*lb*]	9,548.36 (-0.06)	9,567.28 (0.14%)	9,663.80 (1.15%)
Time [*s*]	12,300	8,754	9,476
Design Cycles	9	13	12
Total Analyses	255	329	321
Parallel Analyses	168	312	288

As for the twelve design variable case, the twenty four design variable case had excellent correlation with the baseline optimum. Again, the MMFD algorithm had the most one-dimensional search calculations (34%), compared to the SLP (5%) and SQP (10%) algorithms. However, the percentage of total analyses required for the one-dimensional search calculations reduced when the number of design variables was increased. Although the percentage of total analyses required for the one-dimensional search calculations reduced, the average number of one-dimensional search calculations per design cycle remained almost constant.

For the twenty four design variable case the SQP algorithm required 26% more analyses but took 23% less time to complete as compared to the MMFD algorithm, while the SLP algorithm required 29% more analyses but took 29% less time as compared to the MMFD algorithm. The SLP algorithm required 2%

Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

Table 4: Total time to complete the twenty four design variable case.

	MMFD	SLP	SQP
Parallel Time [s]	12,300	8,754	9,476
Slowest Serial Time [s]	56,100 (4.56)	72,380 (8.27)	70,620 (7.45)
Fastest Serial Time [s]	22,185 (1.80)	28,623 (3.27)	27,927 (2.95)

(Values in parentheses are the speedup factor between the parallel and serial optimizations)

more analyses than the SQP algorithm, but was 13% faster. Again, the SLP algorithm had the highest speedup factor, while the MMFD algorithm had the lowest speedup factor. The higher parallel efficiency of the optimization algorithms for a larger number of design variables is illustrated by the higher speedup factors for the twenty four design variable case as compared to those of the twelve design variable case.

6 Conclusions

The goal of the present paper was to investigate the influence of the optimization algorithm and the number of design variables on the efficiency of parallel optimization using existing gradient-based optimization algorithms. In this study only the finite difference gradient calculations were parallelized, but the influence of parallelizing the one-dimensional search calculations was also addressed.

For the present example problem it was found that although the MMFD algorithm was the most efficient algorithm when running the optimizations in series, it resulted in the least efficient parallel algorithm. The MMFD algorithm could be greatly improved by performing the one-dimensional search calculations in parallel. The SLP algorithm were the most efficiently parallelized and parallelizing the one-dimensional search calculations for the SLP algorithm would have a minimal influence on the parallel efficiency. The parallel SQP algorithm, although less efficient than the parallel SLP algorithm, was much more efficient than the parallel MMFD algorithm.

Additionally, it was found that as the number of design variables increased, the percentage of the total number of analyses required to complete the one-dimensional search calculations was reduced and the parallel efficiency was increased. For problems with large numbers of design variables, as is typical of problems that would be solved in parallel, the inefficiency associated with performing the one-dimensional search calculations in series would be minimal for the SLP and SQP algorithms.

References

[1] VisualDOC Design Optimization Software, Version 1.0 Reference Manual, Vanderplaats Research and Development, Inc., Colorado Springs, CO, 1998.

[2] Rogers, J.L., Young, K.C. and Barthelemy, J.M., "Distributed Computer System Enhances Productivity for SRB Joint Optimization", 28th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics and Materials Conference, Monterey, CA, pp. 596-600, April 6-8, 1987.

[3] Sikiotis, E.S. and Saouma, V.E., "Parallel Structural Optimization on a Network of Computer Workstations", *Computers & Structures*, Vol. 29, No. 1 pp. 141-150, 1988.

[4] El-Sayed, M.E.M. and Hsiung, C.K., "Design Optimization with Parallel Sensitivity Analysis on the CRAY X-MP", *Structural Optimization*, Vol. 3, pp. 247-251, 1991.

[5] Watson, B.C. and Noor, A.K., "Sensitivity Analysis for Large-Deflection and Postbuckling Responses on Distributed-Memory Computers", *Computer Methods in Applied Mechanics and Engineering*, Vol. 129, pp. 393-409, 1996.

[6] Smith, S.L., and Schnabel, R.B., "Centralized and Distributed Dynamic Scheduling for Adaptive, Parallel Algorithms", *Unstructured Scientific Computation on Scalable Multiprocessors*, eds. P. Mehrotra, J. Saltz, and R. Voigt, MIT Press, Cambridge, MA, pp.301-322, 1992.

[7] GENESIS Structural Optimization Software, Version 5.0 User Manual, VMA Engineering, Colorado Springs, CO, 1998.