Numerical experiences with parallel clusters for generating Pareto surfaces: application in structural topology optimization

S. Wuppalapati[1], A. D. Belegundu[2], A. Aziz[3] & V. Agarwala[3]
[1]*Xerox Corporation, USA*
[2]*Department of Mechanical Engineering,*
The Pennsylvania State University, USA
[3]*High Performance Computing and Visualization Group,*
The Pennsylvania State University, USA

Abstract

Generating Pareto surfaces is a well-accepted technique in multi-attribute decision making. For computationally intensive applications like finite element based optimization, it can become very expensive to generate the complete Pareto surface. Hence, using parallel computer clusters in these scenarios becomes very attractive. Pareto surfaces are generated using two different clusters with a structural topology problem as a test problem and the performance gains realized are analyzed. Near linear speed-ups and high efficiencies are observed on both the clusters. It is possible to integrate this methodology into commercial software applications, leading to less turn around times to make critical decisions in various applications.
Keywords: topology optimization, multi-attribute optimization, finite elements, Pareto surfaces, parallel clusters, MPI, speed-up.

1 Introduction

Multi-attribute or multi-criteria optimization typically involves the problem of 'simultaneously' minimizing (or maximizing) different objectives within a given domain. Mathematically it involves solving the optimization problem:

$$\begin{aligned} &\min && [f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}), \ldots, f_m(\mathbf{x})] \\ &\text{subject to} && \mathbf{x} \in \Omega \end{aligned} \tag{1}$$

WIT Transactions on The Built Environment, Vol 91, © 2007 WIT Press
www.witpress.com, ISSN 1743-3509 (on-line)
doi:10.2495/OP070011

Table 1: Nomenclature used.

Symbol	Description
f_i	Objective function
$\mathbf{x}$	Design variable vector
Ω	Design space
Δ_i	Displacement at a node
ρ	Density of the material used
x_i	i^{th} element of the design variable vector $\mathbf{x}$
NDV	Number of design variables.
NDC	Number of displacement constraints
$\mathbf{E}$	Young's Modulus
υ	Poisson's ratio
w	Weight of the structure at a given design variable.
w_0	Maximum weight of the structure.
np	Number of Processors.
T_{np}	Time taken to solve the problem on np number of processors.
S_{np}	Speed-up when np number of processors are used.
E_{np}	Efficiency when np number of processors are used.
FE	Finite element

Though a utility function can be defined and optimized, leading to a single best compromise among the competing objectives, generation of the entire Pareto surface is equally attractive among decision makers. Both these methodologies of solving multi-attribute optimization problems are detailed by Belegundu and Chandrupatla [1]. Pareto surfaces, apart from not being sensitive to weights used in the utility function, allow visualization and give a *feel* for the entire design space. Pareto surfaces can be generated using various methods like weighted approach, constraint approach, genetic algorithms etc. Irrespective of the method adopted, generation of these surfaces usually is computationally expensive, involving a high number of function evaluations and a need to solve the optimization problem repeatedly. Moreover, when computationally expensive function evaluations like finite element (FE) analysis are involved in the multi-attribute optimization, it can become prohibitively expensive to generate the entire Pareto surface on a single computer.

In this paper, a constraint approach [1] is adopted to generate the complete Pareto surface using computer clusters with an application in topology optimization of structural systems. In this approach, eqn. (1) is reformulated as

$$\begin{aligned} &\min && f_1(\mathbf{x}) \\ &\text{subject to} && f_j(\mathbf{x}) \le c_j \qquad j = 2,\ldots,m \\ & && \mathbf{x} \in \Omega \end{aligned} \tag{2}$$

The entire Pareto surface can be generated by solving eqn. (2) repeatedly for different values of c_j. Since each of the optimization problems solving eqn. (2) is independent of the others, it is possible to solve each instance of it simultaneously on different computing nodes of a computer cluster. This approach is implemented in this paper to generate Pareto surfaces on two different computer clusters, LION-XO and LION-XM, available with the High Performance Computing Group, The Pennsylvania State University [2]. Numerical experiences gained using these two clusters are presented and analyzed here. By using adequately large number of computing nodes in a cluster it would be possible to reduce the time taken to generate the entire Pareto surface to a time comparable to that of a single optimization run. This will lead to near real-time decision making involving trade-off analysis.

2 Pareto surfaces in structural topology optimization

2.1 Structural topology optimization

Structural topology optimization in its most general form refers to the problem of describing optimal material distribution within a given domain for a specific objective function subject to various constraints. It is a common practice in literature to obtain optimal material distribution using a stiffness based approach, [3–5], where the stiffest structure for a given constraint on mass fraction is solved for. However, a structural engineer would ideally like to maximize stiffness and also minimize weight without having to restrict to a given mass-fraction. This would lead to a multi-attribute optimization problem with weight and stiffness as the two competing objectives. To solve this optimization problem completely and aid the engineer in a better decision making process, the entire Pareto *curve* (as there are only *two* competing objectives here) has to be obtained.

In this paper, to obtain the Pareto curve a minimum weight based formulation for topology optimization as follows is used:

$$\begin{aligned} &\min && \int \rho d\Omega \\ &\text{subject to} && \left|\Delta_i\right| \le \Delta\max_i \quad i = 1,\ldots,n \\ & && 0 \le \rho \le 1 \end{aligned} \tag{3}$$

In this formulation, stiffness is implicitly incorporated into the optimization problem by constraining deflection at the points where load is applied, Δ_i, to be less than a limit, $\Delta\max_i$. By varying the value of $\Delta\max_i$ and solving the optimization problem repeatedly on a computer cluster, the entire Pareto curve could be obtained. A plot of the displacement limit $\Delta\max_i$ and optimum weight w gives the Pareto curve.

It should be noted that though the problem is posed as a continuum model with the density of material at a given point, ρ, being allowed to take any value between 0 and 1, it is desired that ρ be either close to the limits, 0 and 1, at the optimum. This will aid in a clear description of the topology of the final structure.

The topology optimization problem is solved using widely popular micro-structure approach [3], where the entire domain is divided into various finite elements. Each element (or a group of elements) is characterized by a design variable, x_i, the density of the material. Hence the objective function and constraints are continuous functions of the design variable, **x**. Since intermediate densities appear in the formulation in eqn. (3), it becomes necessary to determine various elastic properties, $E(\mathbf{x})$, $G_{12}(\mathbf{x})$, etc. as functions of intermediate densities. This procedure, known as *parameterization of material properties,* is carried out using a novel procedure called Virtual Testing Methodology proposed by Belegundu *et al* [6]. Though not taken advantage of here, this methodology allows parameterization of not just elastic properties but also strength properties of intermediate densities [6]. The actual problem formulation employed in the solution of topology optimization is as follows:

$$\begin{aligned} \min \quad & \left(\mathrm{W}/\mathrm{W}_{max} + r * p(\mathbf{x})\right) \\ \text{subject to} \quad & xl_i \le x_i \le xu_i \qquad i = 1,\ldots,NDV \\ & |\Delta_i| \le \Delta \max_i \qquad i = 1,\ldots,NDC \end{aligned} \tag{4}$$

The term $p(\mathbf{x})$ is added as a penalty function to the objective function so that the densities at the optimum are driven to the limits xl_i and xu_i giving a clear material distribution. The objective function is normalized with respect to maximum weight w_{max}. The penalty function used, strategies used with the penalty function etc. are described in detail by Wuppalapati [7]. The optimizer used to solve eqn (4) employs Method of Feasible Directions (MFD) algorithm [8], a gradient based optimization algorithm. Entire Pareto surface can be obtained by changing the value of $\Delta \max_i$ in steps between two limits and solving eqn (4) repeatedly.

2.2 Test problem chosen

To obtain Pareto curves in topology optimization the L-Bracket shown in fig. 1 is chosen. Material properties used for the analysis are **E**=200 e9 and υ=0.3. A load of 9.0e6N is applied as shown in fig. 1. To obtain the Pareto curve, maximum allowable distance at the point where load is applied, $\Delta \max$, is varied from a value of 1.55e–03m to 5.0e–03m in equal steps. The structure is discretized using 1250 4-noded QUAD elements. The density of each element is used as a design variable (without any grouping) and hence the optimization problem has 1250 design variables. The optimization problem is solved at each value of $\Delta \max$ on each computing node of the cluster obtain the Pareto Curve.

3 Solution methodology on computer cluster

A master-slave paradigm is employed here to solve the problem on a computer cluster [9]. In this methodology, the optimization algorithm is started simultaneously on all compute nodes used in the cluster. One of the computing nodes is chosen as a master node (here the node with rank 0) [9]. The master node handles all I/O interactions needed for the execution of the FE based optimization like input data, optima calculated on all compute nodes etc. It broadcasts all data needed for successful execution of the optimization algorithm to the other computing nodes and also synchronizes their execution. The master node apart from controlling the computation on the slave nodes, also computes optima for certain values of $\Delta \max_i$.

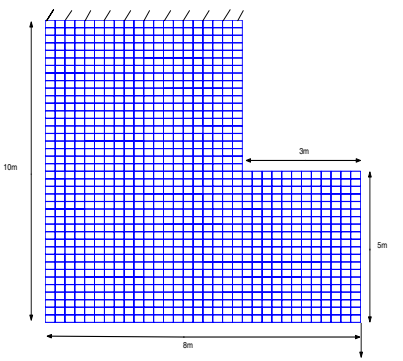

Figure 1: Test problem used for topology optimization: L-bracket.

To obtain the entire Pareto curve, the number of points required on the Pareto curve is fixed. The interval between which $\Delta \max_i$ is to be varied is divided in to equal steps and each optimization is carried out on different computing nodes. If the number of points needed is more than the number of compute nodes used, the same procedure is carried out in a loop till the optimization is carried out at all the values of $\Delta \max_i$ and the complete Pareto curve is obtained. It should be noted that the time of execution of the optimization algorithm is not equal at all values of $\Delta \max_i$ and hence different compute nodes may finish a particular instance of the algorithm at different times. Hence synchronization of the execution is needed. This is carried out at the master node.

By varying the number of points required on the Pareto curve, it is possible to obtain a relatively fine Pareto curve. By using a larger number of points, the computational resources of a massive cluster can be used to obtain a fine Pareto curve in a far less time.

4 Cluster description

The two clusters used in this study are LION-XO and LION-XM cluster [2]. LION-XO is a heterogeneous cluster with one login node and 132 compute

nodes. Of the 132 nodes, 80 compute nodes are dual 2.4GHz AMD Opteron Processors with 8 GB of ECC RAM connected through a Force 10 E600 Series Gigabit Ethernet switch and a high speed Silverstorm Infiniband Network. Only these 80 nodes of the cluster are used to select the compute nodes for the optimization. LION-XM is a homogeneous cluster with 1 login node and 168 compute nodes with dual 3.2GHz Intel Xeon Processors with 4GB of ECC RAM connected through a high speed Myrinet Network.

The parallel implementation of the code is carried out using MPI-2.0 [9] standard in Fortran 90. Intel FORTRAN 90 compilers are used for compiling the codes. MPIGM, a version of Argonne National Laboratory's MPICH available on the clusters is used for the tests [9]. It should be noted from the description of solution methodology that communication between master and nodes is relatively minimal and hence near linear speed-ups are to be expected. Also the speed of interconnects between different processors and the nature of communication between different nodes is to have very little effect on the performance of the algorithm.

5 Results and conclusions

For the test problem described in Section II, test runs were carried out on both the clusters LION-XO and LION-XM. The number of points on the Pareto curve needed is fixed at 24 and runs were carried with different number of compute nodes starting from 1 through 12, using one processor per node. Hence the number of processors used in the computation, *np,* is equal to the number of compute nodes used. The total wall-clock time to completely solve the problem is measured in each run. It should be noted here that, for consistency of comparison, in this implementation the number of processors on which the test is run is set to be a multiple of the total number of points needed on the Pareto Curve. If it were not the case, a different strategy would have to be implemented in the algorithm which takes advantage that not all instances of the optimization algorithm take same amount of time to reach the optimum.

As expected the Pareto optimal curve obtained is the same using different number of processors and on both the clusters. It is shown in fig. 2. Representative optimal topologies of the structure in fig 1 at different $\Delta \max_i$ are shown in fig. 3. As $\Delta \max_i$ is increased, there is a possibility for material to be removed at the optimum, thus making the identifiable members of the optimum topology thinner and thinner. Appearance of grey-areas and checker-boarding patterns are common issues in topology optimization [3, 7].

The wall-clock time expended to obtain the entire Pareto Curve using different number of processors on both clusters is given in table 2. It can be observed that the problem is solved considerably faster on LION-XO, which has less processing speed but far more RAM availability than on LION-XM. FE based optimization is a data intensive application requiring the storage of huge matrices both in FE-analysis and in optimization. Though the actual run-time for

any application is dependent on various factors, it can be said that computer clusters with high-end RAM capability can potentially be more efficient for this kind of application.

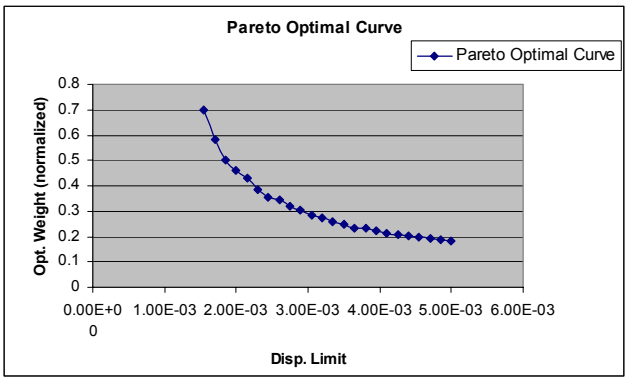

Figure 2: Pareto curve for the test problem in Figure 1.

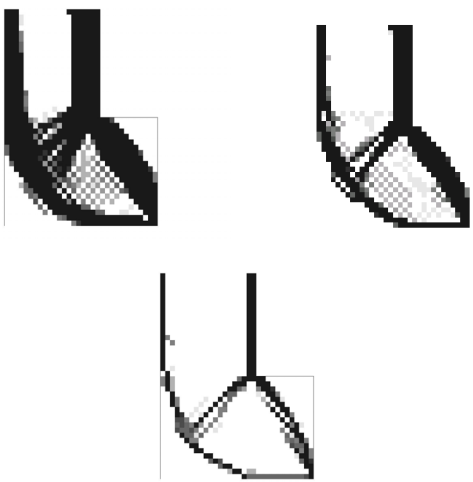

Figure 3: Change in topology of the structure as $\Delta \max_i$ is changed.

Speed-ups achieved using both the clusters are plotted in fig. 4. It is observed that though actual time taken to obtain the entire Pareto curve is different in the two clusters, very similar speed-ups are realized on both of them. Hence the speed-ups obtained for this application are practically independent of the hardware components like the interconnects and the switches used to build the cluster, the processing speed, RAM availability etc. This is to be expected because there is very minimal communication between processors while the optimization problem is being solved.

Table 2: Wall-clock time on the clusters.

# Procs	Time Taken(s)	
	LION-XM	LION-XO
1	177517	104793
2	96423	55868
3	65056	38101
4	51569	29539
6	37182	21322
8	30218	17174
12	21380	12417

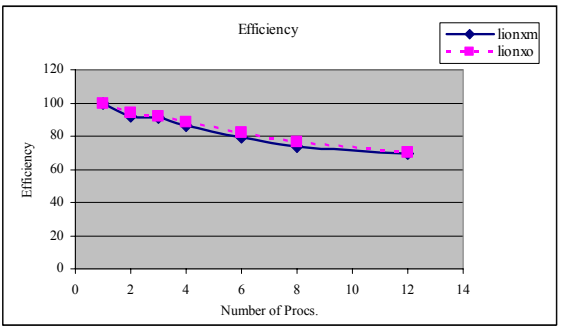

Figure 4: Efficiency of the computer cluster observed.

Linear-speeds up are not observed because, though the computation of obtaining the Pareto Curve is distributed among various clusters, the time taken to solve each optimization problem is not equal. As a result, some processors remain idle till the problem is completely solved on all the other processors. However, the time taken to obtain the entire Pareto Curve decreases from 177517s when one processor is used to 21380s when 12 processors are used, a reduction by a factor of 8.3. This means that it is possible to take more informed decisions at a considerably faster rate than is possible using just one processor. Also, the test problem used here has 1250 design variables. In various FE based optimizations of practical importance, the number of design variables can be considerably larger, sometimes running in to the order of tens of thousands of design variables. In such a scenario, the speed-up observed obtains even more importance in terms of turn around times to take a decision.

Even though tests were carried out on homogeneous computing nodes on each cluster, it is not necessary that the actual performance of the cluster be homogenous. To determine the homogeneous nature of the cluster, time taken to solve a particular instance of the optimization problem in Eq. 4 and the number of function evaluations (NFV) for that particular instance are calculated. From

these, time taken for the computation of one function evaluation is calculated on all processors. The time taken for each function evaluation is normalized with respect to the first processor and the results are plotted in fig. 6.

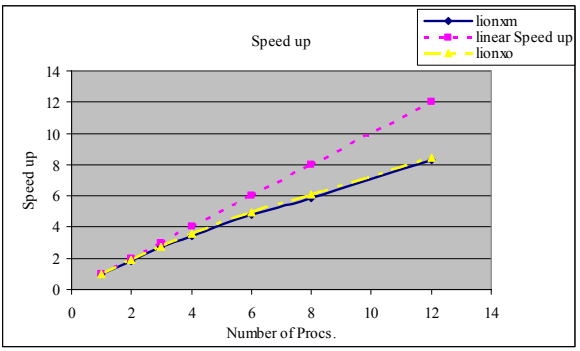

Figure 5: Speed-up realized solving for Pareto surfaces of the L-bracket.

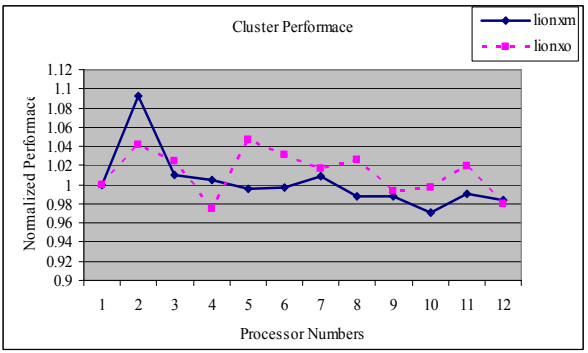

Figure 6: Normalized performance of different processors.

It can be seen that the performance of each computing node in the optimization, normalized with respect to one of the processor remains close to 1. Hence each processor of the cluster shows a near ideal performance in this application. This study would be very important in synchronizing the algorithm, more so on a heterogeneous cluster where processing speeds of different compute nodes are different. While submitting the jobs to computing nodes, taking care of the heterogeneous nature of the cluster can lead to better speed-ups and more efficient use of the computing resources available.

From the studies presented here, it can be concluded that using computer clusters for generating Pareto surfaces in computationally intensive applications like structural topology optimization is very attractive. Near linear speed-ups and high efficiencies obtained make it easier to make critical decisions faster. Also, maximum use of computational resources available is possible because of relatively low communication costs between different processors.

References

[1] Belegundu, A.D. and Chandrupatla, T.R., Pareto Optimality (Chapter 11). *Optimization Concepts and Applications in Engineering,* Prentice-Hall: New Jersey, pp. 373-384, 1999.

[2] Graduate Education and Research Services Group, The Pennsylvania State University. http://gears.aset.psu.edu/.

[3] Eschenauer, H.A. and Olhoff, N., Topology optimization of continuum structures: A review, *Applied Mechanics Reviews*, **54(4)**, pp. 331-390, 2001.

[4] Bendose, M.P., and Sigmund, O., *Topology Optimization: Theory, Methods and Applications,* Springer-Verlag Berlin and Heidelberg GmbH, 2002.

[5] Rozvany, G.I.N., Bendsoe, M.P. and Kirsch, U., Layout optimization of structures, *Applied Mechanics Reviews*, **48,** pp. 41-119, 1995.

[6] Belegundu, A., Rajan, S., Wuppalapati, S *et al.,* Structural Topology Optimization Based on Virtual Testing with General Constraints, *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference,* 2006.

[7] Wuppalapati. S, Virtual Test based Topology Optimization with Stress Constraints, M.S. Dissertation, Dept. of Mechanical and Nuclear Engineering, The Pennsylvania State University, University Park 2005.

[8] Belegundu, A.D., Berke, L. and Patnaik, S.N., An optimization algorithm based on Method of Feasible directions, *Structural Optimization,* **9**, pp. 83-88, 1995.

[9] Gropp, W., *Using MPI: portable parallel with message-passing interface,* MIT Press, 1999.

[10] Aoyama, Y. and Nakano, J., *RS/6000 SP: Practical MPI Programming*, IBM Corporation, 1999.

[11] Belegundu, A.D., Damle, A., Rajan, S.D., Dattaguru, D., St. Ville, J., Parallel Line Search in Method of Feasible Directions, *Optimization and Engineering,* **5(3)**, pp. 379-388, 2004.