A NSGA-II, Web-Enabled, Parallel Optimization Framework for NLP and MINLP

David Powell
Elon University
Campus Box 2101
Elon, North Carolina 27244
dpowell2@elon.edu

Joel Hollingsworth
Elon University
Campus Box 2116
Elon, North Carolina 27244
jkh@elon.edu

ABSTRACT

Engineering design increasingly uses computer simulation models coupled with optimization algorithms to find the best design that meets the customer constraints within a time constrained deadline. The continued application of Moore's law combined with linear speedups of coarse grained parallelization will allow more designs to be evaluated in shorter periods of time. This paper presents a scalable, standards based framework that uses web services and grid services with a multiple objective genetic algorithm to solve continuous, mixed integer, single objective or multiple objective nonlinear, constrained design problems. Test data is provided to validate a linear speedup based on the number of processors and to show the robustness of the genetic algorithm on a set of 10 design problems.

Categories and Subject Descriptors

D.3.3 [**Programming Languages**]: Language Constructs and Features—*Frameworks, concurrent programming structures*; G.4 [**Mathematical Software**]: User Interfaces; I.6 [**Simulation and Modeling**]: Applications and Simulation Support Systems

General Terms

Algorithms, Performance, Design, Economics, Standardization

Keywords

Web services, Service Oriented Architecture, Design Patterns, Grid Engine, Multiple Objective Genetic Algorithm, Nonlinear Constrained Optimization

1. INTRODUCTION

Over the past decade, software development best practices has driven the establishment of world wide standards bodies and has strongly influenced software developers to program to published standards. Free reference implementations of standards are continually provided by both open source organizations and software vendors (e.g., web services by www.apache.org and the Sun Grid Engine by Sun Microsystems). Software developers throughout the world are using these free implementations and creating software packages that are free, reusable and easily distributed with Internet downloads.

Engineering design is a field that can significantly benefit from a standards-based approach. Engineers typically have a computer simulation program that closely models the design artifact that they are designing. The goal of the engineer is to use the model to develop the best possible design within the scheduled amount of time. Since the early sixties, numerous nonlinear optimization algorithms have been created that offer an attractive promise to efficiently and effectively search the large design space of the computer simulation programs to achieve a better design in a shorter period of time. The continued increase in computer power and the continued growth of inexpensive, distributed, parallel processing computer power provides an ideal opportunity for a standards-based approach to optimization.

The authors combined the standard mathematical optimization problem formulation with standards-based web services and standards-based distributed resource management to provide a free optimization environment to support the needs of end users, optimization experts and software application developers. This paper discusses the need for a single framework, the standards selected, the rationale for selection and initial results from the framework's prototype implementation.

2. A SINGLE FRAMEWORK

During the past two years, the authors have taught linear and nonlinear constrained programming to graduate students, undergraduate seniors and to practicing engineers. Though the optimization community is in close agreement on the standard specification of the optimization formulation abstraction shown in Figure 1, they provide little or no guidance on how to select the correct software algorithm, the correct software algorithm implementation and the model changes needed to use the selected implementation.

To get an appreciation for the difficulty in getting started in using optimization, the reader is encouraged to visit the NEOS web site provided by Argonne Labs and Northwestern University [1]. The site provides an Optimization Guide that displays a tree with over 16 nodes to categorize different types of optimization problems. The user can select a

$$\begin{array}{rcl} \text{minimize/maximize} & f_m(x), & m = 1, 2, \cdots, m \\ \text{subject to:} & g_{jb} \leq g_j(x) \leq g_{ju} & j = 1, 2, \cdots, j \\ & h_{le} = h_l(x) & l = 1, 2, \cdots, l \\ & x_{ib} \leq x_i \leq x_{iu} & \text{where } x_i \text{ is} \\ & & \text{real or integer} \end{array}$$

$$\begin{array}{r} b \text{ is lower constraint boundary} \\ u \text{ is upper constraint boundary} \\ e \text{ is the constraint boundary} \end{array}$$

Figure 1: Standard Problem Formulation

node describing their type of formulation to get a list of appropriate algorithms. For example, after selecting the node, Optimization - Continuous - Constrained - Nonlinearly Constrained, the Optimization Guide advises the user to select any of the following four algorithms: sequential quadratic programming, augmented Lagrangian methods, reduced gradient methods or feasible sequential quadratic methods. After the user selects an algorithm then a list of implementations are displayed. For example, if the user chose reduced gradient methods then the user would get a list of at least seven different implementations. This presents the end user who originally had one problem to solve with an additional optimization problem. The user now needs to correctly decide on the correct categorization, correct algorithm and correct implementation to apply to their original problem. The new approach eliminates this additional problem by providing the user with only one choice of an algorithm with a single implementation. The algorithm, NSGA-II [10], is a multiple objective genetic algorithm implemented by Deb and supports the full standard mathematical problem formulation abstraction. Before discussing the abstraction, and implementation, let's first negate the classical complaint of the large evaluation expense required by a genetic algorithm by considering Moore's law.

3. EVALUATION COST IS NOT AN ISSUE

In the past, the major complaint against using genetic algorithms has been its large computational cost. However, the emergence of grid computing for parallel evaluation of the genetic algorithm population combined with the continued application of Moore's law has made and will continually make computational cost less of an issue.

The evaluation of a population of designs by a genetic algorithm is ideal for parallelization. When the cost for an individual processor to evaluate an individual population member is large compared to the network cost to distribute the member then the speed up is linear based on the number of processors [14]. Created in 1965, Moore's law stated that the number of transistors on a chip would double every two years. This law has held up for the past 40 years and appears to be valid for the upcoming decade [7]. Moore's law has been demonstrated to hold for software applications with the execution time of the application reduced by a factor of two every two years [13]. The exponential speed up of an individual design evaluation every two years combined with a linear speedup of parallel design evaluation greatly reduces the wall clock time to optimize a design. For example, today the evaluation of five generations of a population of 100 designs on a single processor takes the same amount of time that it took to evaluate a single design in 1990. When 10 processors are used then the linear speed up allows 50 generations of a population of 100 designs to be evaluated in the time that it took to evaluate a single design in 1990. With the emergence of grid engines combined with faster processors and faster networks, the cost of a genetic algorithm for a population of 100 on 10 machines for 400 generations will take the same time in 2012 that it took to evaluate a single design in 1990.

The new framework, which is discussed in the next section, leverages the emergence of grid computing for distributed evaluation by using the Sun Grid Engine for automatic distribution and load balancing of course grained evaluation within a campus grid. The framework will continually achieve significant reduction in optimization time through the continued application of Moore's Law as the grid machines are upgraded.

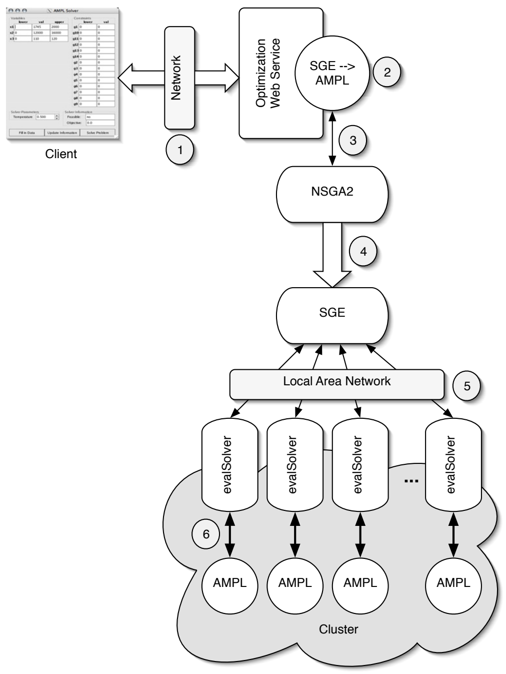

Figure 2: Optimization Framework

4. THE FRAMEWORK

The framework shown in Figure 2 is freely available and contains many pieces that can be individually or collectively used in a variety of ways to best meet the needs of end users, optimization experts and application programmers. This section discusses the pieces and how they can be used.

4.1 AMPL and NSGA-II Coupling

AMPL [11] is a mathematical programming language developed at Bell Labs, designed to fully support and mimic as close as possible the standard mathematical problem formulation shown in Figure 1. A full featured, student edition is free and allows specification of a problem with up to 300 variables and 300 constraints. This allowable problem size allows the majority of real world, non linear constrained optimization problems to be solved. In fact, a large problem is typically categorized as one with 100 variables. AMPL is available for Windows, Mac OS, UNIX and Linux platforms and can be downloaded over the Internet from the company web site. The language is interpretive and extendible. The language does not have any built in optimization technique but provides the application programming interfaces to couple an optimizer, pass the mathematical formulation to the optimizer and perform the design evaluation for the optimizer. The company web site lists over 35 numerical optimizers that have been coupled and are either freely available or commercially available. However, none of these optimizers supports mixed integer, multiple objective optimization. The new framework has coupled the multiple objective, mixed integer, NSGA-II to AMPL (see item 2 in Figure 2). The coupling provides the user with the following advantages:

1. The user specifies the problem formulation in a standard mathematical language that is well documented in the AMPL textbook. The language is interpretive, ASCII and non-architecture specific. The language can be treated as strings and easily generated by the user with a text based editor or another program and passed over the network.

2. NSGA-II provides an algorithm that supports multiple objective, mixed integer, pareto based, nonlinear constrained optimization. NSGA-II makes no assumptions on unimodality, convexity or continuity. NSGA-II directly supports the full mathematical problem formulation of Figure 1 except equality constraints. The NSGA-II author, Deb, recommends converting equality constraints into two inequality constraints.

3. The NSGA-II coupling automatically and transparently translates the standard formulation into a formulation needed by NSGA-II with all objectives minimized, all inequality constraints converted to ≥ 0 format and all equality constraints converted to two ≥ 0 inequality constraints. If the user provided an initial starting design point then the initial population contains this design point with the rest randomly determined.

4. The NSGA-II coupling fully supports the setting of all NSGA-II options in AMPL (e.g. population size, number of generations, crossover and mutation rates).

5. The NSGA-II coupling automatically calls AMPL for each design evaluation. The coupling can evaluate the population sequentially or in parallel effectively using all available machines under the control of the Sun Grid Engine.

6. AMPL provides the interface to evaluate simulation models written in the AMPL language or to extend the AMPL language with user defined C functions. The extension of the AMPL language with C functions allows the user to call simulation codes, library functions or code developed in another language (e.g. FORTRAN, C++, and Java).

7. The NSGA-II coupling automatically returns the entire feasible pareto optimal set to the user. If no feasible design is found then the coupling returns the population member that came the closest to satisfying constraints.

8. The coupling uses the AMPL standard interface which allows other AMPL coupled optimizers to be used or added to solve the problem either independently or in a hybrid fashion with NSGA-II.

4.2 Wrapping AMPL with a Facade Design Pattern as a Web Service

AMPL is designed to be run either from the command line or in an interactive command driven application from a console window. When run in command line mode, AMPL is typically invoked with three arguments representing a model file, a data file and a command file. The Facade design patter [12] is used to wrap the invocation of AMPL within a Java object. The wrapping of AMPL within a Java object allows us to support the following capabilities:

1. The wrapped AMPL object has been made into a web service (see item 1 in Figure 2). A key advantage of using a web service is that the client application is platform and language independent. For example, Figure 2 shows a client GUI written in Visual C# running on a windows operating system connecting to the web service. The new framework is currently using the freely available Tomcat servlet engine [2] and Axis Web Services Project [3] for Windows and Linux servers. The Axis web services supports JAX RPC 1.1 and maintains session state.

2. The wrapped AMPL web service can submit the AMPL job to an underlying grid engine instead of being executed using Java Runtime. The submission to a grid engine allows AMPL to be run on any of the cluster machines based on a load balancing scheduling policy. The submission uses the Java API of the Distributed Resource Management Application API (DRMAA) 1.0 [4]. This specification currently is supported by the Sun Grid Engine, Condor and Globus services. The framework is currently using the Sun Grid Engine.

4.3 Sun Grid Engine

The Sun Grid Engine [9] is a freely available software application that controls a cluster of machines. Jobs are submitted to a job manager and the job manager schedules and oversees the execution of the job on any of the cluster machines. The machines can be of heterogeneous operating systems. Sun currently supports Solaris, Linux, Windows, Mac OS, HP-UX and IRIX. Sun designed the grid engine to be highly scalable and a job manager can support up to 10,000 execution hosts.

The Distributed Resource Management Application API (DRMAA) 1.0 specification was published in June 2004. This specification defines a standard set of Java and C API calls for submitting a job to a grid engine. The Sun Grid Engine has implemented the DRMAA standard for both Java and C. The framework uses DRMAA for its standards based interface to its Sun Grid Engine cluster and will run without modification with other grid engines such as LSF and Condor that support the DRMAA interface.

The framework uses the Sun Grid Engine in two ways. First, as shown in item 1 of Figure 2, AMPL jobs which use the NSGA-II optimizer are submitted to it using the previous described Java DRMAA API. Second, as shown in item 4 of Figure 2, the NSGA-II optimizer uses the C based DRMAA API to submit an entire population of designs to the job manager to schedule and execute across the cluster. The AMPL extensible interface has made this an easy task. Each member of the population runs the same evaluation function by using a C program called evalSolver that invokes the AMPL "nl" stub file on a set of inputs representing the individual design. The AMPL stub file is ASCII as well as the set of inputs which makes the invocation across heterogeneous operating systems an easy task.

5. END USER, OPTIMIZATION EXPERT AND APPLICATION DEVELOPER

For a company to successfully use optimization, three levels of support are needed. These are end user support, optimization expert support and application developer support. The new framework is ideal for each support level.

End users are domain experts. They understand the simulation models and are responsible for developing a design to meet the customer specifications. The framework provides a web service that supports the client side development of customized end user interfaces that present and display results in a format expected by the domain expert.

An optimization expert needs the capability to easily express and validate an optimization model with multiple objectives, multiple bounds and multiple constraints. The framework supports AMPL which allows the optimization expert complete control over the formulation. The optimization expert has many resources for learning AMPL. There is a text book. There is an active news group. There are hundred of AMPL test cases available for nonlinear constrained optimization [8], mixed integer nonlinear constrained optimization [6] and multiple objective constrained optimizations [5]. In addition, there is a NEOS Solver web site that allows users to submit AMPL models through a web browser to 26 different AMPL coupled optimizers.

The application developer needs to support the development of client side end user graphical user interfaces, the customization of AMPL with user defined C functions and installation of software. The framework supports the application developer with the use of a Service Oriented Architecture (SOA) and widely used commercial and open source software. Web services are widely used and documented. The use of web services supports client side development in Java or another language. The Apache Tomcat and Axis tools are widely used at thousands of sites and extremely stable. The Sun Grid Engine is distributed by the developer of Java and the most stable of the freely available grid engines. The AMPL language is widely used.

6. PROTOTYPE STATUS AND RESULTS

The framework has been fully prototyped by the authors. The current status is:

1. The coupling of AMPL and NSGA-II is completed and tested for Linux systems and Windows systems. The coupling has been tested on a selected set of 10 single objective, nonlinear, constrained engineering benchmarks from the 30 problems in the Sandgren test suite [16, 15]. Sandgren categorized his thirty problems in three categories. He classified 14 as simple, 9 as advanced and 7 as difficult. The ten problems that we selected were distributed among these three categories with four simple, four advanced and two difficult. The results shown in Table 1 lists the Sandgren problem number, the number of design variables, the number of inequality constraints, the number of equality constraints, the objective value at the starting point (F), the published optimum (F*), the best value achieved by the default AMPL optimizer, MINOS, which uses the generalized reduced gradient technique and the best, median and worst population objective value after running NSGA-II for 500 generations with a population size of 100. The starting point for problems 6 and 22 was infeasible. The key results of the benchmark are:

   (a) In all 10 cases, NSGA-II achieved an improved feasible design from the starting point.

   (b) In 5 of the 10 cases, NSGA-II beat MINOS (Problems 6, 11, 13, 20 and 22).

   (c) In 2 of the 10 cases, NSGA-II tied MINOS (Problems 3 and 24).

   (d) The test problems are all nonlinear, constrained and continuous. MINOS only works in this domain. If the problem was mixed integer then MINOS would not be appropriate. If the problem had multiple objectives then MINOS would have to have the problem transformed to a single objective using one of the classical techniques such as goal programming or min-max. NSGA-II does not have this problem and provides a single algorithm that the user can select and expect to get improvement.

2. The coupling of AMPL and NSGA-II with distributed evaluation by DRMAA submission to SGE has been successfully completed and benchmarked using a nine processor cluster running Linux. The results are shown in Table 2. The row labeled "Individual Evaluation" shows the times required in seconds to run a single evaluation of various simulation codes. The fastest simulation code runs in .27 seconds and the longest codes takes 17.4 seconds. The row labeled "Sequential Population Evaluation" indicates the required time to run one generation with a population of 100 designs on a single processor. These times are roughly 100 times that of a single evaluation as there is no overhead for networking. The row labeled "Parallel Population Evaluation" indicates the time to evaluate the population of size 100 using 9 processors under the control of the Sun Grid Engine. The row labeled "Speedup" shows the advantage (if any) from using 9 processors. The results clearly show that the benefits of multiple processors can be seen once the simulation code single evaluation time exceeds one second and that for long running code that the speedup will be a linear speedup equal to the number of processors. For example, the long running simulation code that takes 17.4 seconds

for a single design will have a speedup of a factor of 9 when a population is evaluated in parallel across all 9 hyperthreaded processors instead of sequentially on a single machine. For long running codes, an ideal cluster would have the same number of machines as members of the genetic algorithm population.

3. The optimization web service has been deployed and tested on Linux and Windows with client GUIs written in Java, Ruby and Visual C#.

4. The support for client side initiated transfer of data files and architecture specific binaries using the web service has been demonstrated. This feature allows the user to upload customized AMPL libraries along with simulation codes for distributed and parallel evaluation. The feature also allows the user to download server files created during the optimization.

If the prototype was to be enhanced to a production level system then additional work is needed on sever side management to allow users the ability to upload binary files and execute them without endangering site security and to prevent runaway applications from consuming the server.

7. CONCLUSION

In the upcoming decade, computers will continue their trend of doubling in processor and network speed every two years. The increased speed of individual processors combined with the ease of course grained parallelization will allow engineers to continually search larger design spaces in a shorter period of time. In this paper, the authors presented a standards-based framework that utilizes current standards of web services, grid services and DRMAA to provide designers a single approach for the design of nonlinear constrained problems. A genetic algorithm, NSGA-II, was presented that supports a standard optimization problem formulation, works with both continuous and mixed integer variables, supports multiple objectives and makes no limiting assumptions about the continuity or unimodality of the design space. Test data was provided to demonstrate the robustness of the genetic algorithm on a test set of 10 problems and to validate the linear speedup available by parallel evaluation of a population.

8. ACKNOWLEDGMENTS

This work was sponsored by the University of North Carolina, Office of the President for a Consortium to Promote Scientific Computation and High Performance Computing.

9. REFERENCES

[1] Neos guide optimization tree. http://www-fp.mcs.anl.gov/otc/Guide/OptWeb/index.html.

[2] Apache tomcat. `http://tomcat.apache.org`, May 2006.

[3] Apache web services axis project. `http://ws.apache.org/axis`, May 2006.

[4] Drmaa. `http://www.drmaa.org`, May 2006.

[5] Macmoop. `http://www-unix.mcs.anl.gov/~leyffer/MOOP/index.html`, May 2006.

[6] Minlp, ampl collection of mixed integer programs. `http://www-unix.mcs.anl.gov/~leyffer/macminlp`, May 2006.

[7] Moore's law 40th anniversary. `http://www.intel.com/pressroom/kits/events/moores_law_40th/index.htm`, May 2006.

[8] Non linear constrained programming problems. `http://plato.asu.edu/ftp/ampl_files/nlp_ampl`, May 2006.

[9] Sun grid engine. `http://gridengine.sunsource.net`, May 2006.

[10] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Ltd, 2001.

[11] R. Fourer, D. Gay, and B. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Thomson, Pacific Grove, California, 2003.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns:Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[13] C. Kopp. Moore's law and its implications for information warfare. In *Third International AOC Electronic Warfare Conference*, January 2003.

[14] A. Oyama, M. Liou, and S. Obyashi. High fidelity swept and leaned rotor blade design optimization using evolutionary algorithm. In *AIAA Computational Fluid Dynamics Conference*, 2003.

[15] G. Reklaitis, A. Ravindran, and K. Ragsdell. *Engineering Optimization Methods and Applications*. John Wiley & Sons, Ltd, New York, New York, 1983.

[16] E. Sandgren. The utility of nonlinear programming algorithms. West Lafeyette, IN, 1977. Purdue University Ph.D. Thesis.

Problem	#DV	$\#\leq,\geq$ Constraints	#== Constraints	F start	F* Published	F* MINOS	GA Best	GA Median	GA Worst
1	5	10	0	20	-32.348	-32.34	-32.26	-31.93	-29.59
3	5	6	0	-0.303	-0.306	-0.306	-0.306	-0.306	-0.305
6	6	0	4	42.09	8.853 8.927	8.927	8.88	8.96	9.29
11	2	2	0	2.1736	1.149	1.165	1.149	1.149	1.15
13	5	4	0	0.2802	0.2679	0.2749	0.2678	0.2678	0.2678
14	15	5	0	2400.11	32.34	32.34	52.71	148.63	342.09
18	7	14	0	2125.66	1227.2	1227.1	1315.71	1470.6	1866.92
20	8	6	0	15000	7049.24	14166	7089.43	8316.6	10955.2
22	16	19	0	284.74	174.78	336.47	259.93	609.31	1103
24	4	5	0	15.815	2.38	2.38	2.38	3.63	4.5

Table 1: Sandgren Test Case Benchmarks

(7 - Intel Pentium 4 2.8 GHz HyperThreaded, 1024 KB cache, 1 GB RAM
1 - dual processor, Intel Xeon 2.4 GHz, 512 KB cache, 1GB RAM)

Trials	1	2	3	4	5	6	7	8
Individual Evaluation	0.2731	1.0826	2.437	4.3455	6.8103	9.7534	13.3481	17.443
Sequential Population Evaluation	27.315	108.26	243.7	434.55	681.03	975.34	1334.81	1744.3
Parallel Population Evaluation	69.115	71.763	75.32	83.413	108.62	141.34	151.481	192.73
Speedup	0.3952	1.5086	3.236	5.2096	6.2697	6.9005	8.81173	9.0506

Table 2: Speedup Benchmarks with Multiple Processors
(population size = 100, generations = 1, times in seconds)