Genetic Algorithms: A Fundamental Component of an Optimization Toolkit for Improved Engineering Designs

Siu Tong[1] and David J. Powell[2]

[1] Engineous Software, Cary, North Carolina
tong@engineous.com
[2] Elon University, Campus Box 2101, Elon, North Carolina 27244
dpowell2@elon.edu

Abstract. Optimization is being increasing applied to engineering design problems throughout the world. iSIGHT is a generic engineering design environment that provides engineers with an optimization toolkit of leading optimization algorithms and an optimization advisor to solve their optimization needs. This paper focuses on the key role played by the toolkit's genetic algorithm in providing a robust, general purpose solution to nonlinear continuous, mixed integer nonlinear and integer combinatorial problems. The robustness of the genetic algorithm is demonstrated on successful application to 30 engineering benchmark problems and the following three real world problems: a marine naval propeller, a heart pacemaker and a jet engine turbine airfoil.

1 Introduction

This paper describes a generic engineering design environment, iSIGHT, used by the Automotive, Aerospace, Industrial Manufacturing and Electronic industries in the United States, Japan, China, Korea and Europe [1]. Within these industries, the designer's knowledge of optimization varies from that of novice to expert. To meet the needs and challenges of a wide variety of industries and the wide disparity in designer's optimization expertise, iSIGHT provides an Optimization Toolkit. This paper focuses on the role of the Genetic Algorithm in iSIGHT's Optimization Toolkit and its application to three design problems.

The paper starts with a description of the optimization problem in engineering and presents a generic problem formulation. Thirteen well known packages that represent the major numerical and exploratory algorithms are benchmarked against a suite of thirty engineering problems that represent nonlinear continuous, mixed integer and integer combinatorial problems. The benchmark results are analyzed to demonstrate the strengths and weaknesses of each package and to demonstrate the critical role that genetic algorithms serve to the toolkit by providing a single algorithm to solve any nonlinear, constrained continuous or mixed integer problem. The benchmark conclusions are supported by describing the successful application of genetic algorithms to three customer applications

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 2347–2359, 2003.

for the design of a marine naval propeller, a heart pacemaker and a jet engine turbine airfoil. The paper concludes with lessons learned in the development and application of iSIGHT.

2 The Optimization Problem

During the past twenty years, product design and analysis has been routinely done with computer simulation programs developed either in house or by a commercial vendor. The designer supplies the design parameters for the product as input into one or more computer simulation programs, runs the program and then analyzes the results. If the results do not meet the design goals then the designer changes the design parameters and repeats the process. This process is typically called the design, evaluate and redesign process. The challenge to the designer is to find the best design in as short a time period as possible.

These design optimization problems are commonly found in manufacturing industries and can be represented by the following mathematically formulation.

Objective function:	Minimize $y = f(X)$ where $X = \{x1, x2, ..., x_n\}$
	Subject to:
Inequality constraints:	$g_{kb}(X) <= g_k(X) <= g_{ku}(X)$
	where $k = 0, 1, \cdots, K$
	b is lower constraint boundary
	u is upper constraint boundary
Equality constraints:	$g_{le}(X) = h_l(X)$
	where $l = 0, 1, ..., L$
	e is the constraint boundary
Side constraints:	$x_{ib} <= x_i <= x_{iu}$ where x_i is real or integer
	or
	x_i is a member of a discrete set of values

This formulation supports the specification of unconstrained and constrained problems with a single objective. (Note: Multiple objectives are automatically converted into a single weighted objective.) The constrained problems can have either or both inequality and equality constraints. The design parameters can be combinations of type real, integer or discrete. Depending on the mixture, the problem formulation can be considered a continuous nonlinear problem, a pure integer combinatorial problem or a mixed integer nonlinear problem (MINLP).

Non-linear optimization technologies are a natural fit to aid the designer but their successful application has been limited for many reasons to include:

1. The need to modify the simulation source code to interact with the optimization algorithm. If the simulation code is from a third party vendor then the source code may not be available.
2. Most simulation codes were not written with optimization in mind and would require significant reprogramming [2].

3. The difficulty in identifying which optimization algorithm(s) is appropriate for the product. For example, the domain modeled by the simulation code may be discontinuous, non convex, highly nonlinear, poorly scaled, have limited precision in output values, have parameters with different orders of magnitude, and have a mixture of discrete, integer and real parameter design variables.

To overcome these limitations and still solve real world problems, the authors decided to create a toolkit consisting of best-of-breed optimization algorithms that would work together in a coordinated manner. Accordingly, the first step was to identify the best algorithms for each class of industrial problems.

2.1 Optimization Algorithms

In order to identify the most suitable algorithms to be included in the optimization toolkit, a number of analytical and real world problems were set up as benchmarks. Each package was benchmarked to determine how often it achieved the known optimum (i.e. robustness) and the number of function evaluations needed to achieve an optimal result. Table 1 lists the packages, their underlying algorithms, whether they are gradient based or non gradient based, whether they support mixed integer, and their number of basic and advanced tuning parameters. A value of No for Mixed Integer indicates that the algorithm applies only for continuous problems. A value of Yes followed by (BB) indicates that the algorithm is based on a branch and bound algorithm and the underlying code must be able to support both real and integer values for integer parameters. A value of Yes not followed by (BB) for Mixed Integer indicates that the algorithm can work directly with integer values. The Tuning Parameters for each algorithm are set to default values recommended by either the implementer or expert users. The Basic Tuning Parameters are those most likely to be tuned by a user. The Advanced Tuning Parameters are those usually manipulated by only an expert in optimization.

Table 1 lists both numerical optimization algorithms and exploratory genetic algorithms used in this study. For certain algorithms (e.g., Genetic Algorithms and Sequential Quadratic Programming), multiple packages are listed. The rationale for testing multiple packages for an algorithm is due to the importance of the implementation on performance and robustness.

2.2 Benchmarks and Analysis

In 1977, Sandgren established a test set of single objective, nonlinear, continuous engineering problems with different numbers of design variables, inequality constraints and equality constraints [17]. Each problem had a given starting point and a known solution. Twenty eight of his thirty problems and two mixed integer nonlinear problems were used to benchmark the iSIGHT optimization algorithms. The two MINLP problems are a coil compression spring, San 31 [18], and a cantilevered beam, Van 32 [22]. Table 2 lists the benchmark data

Table 1. The optimization packages and their associated characteristics, technique and tuning parameters.

Package	Algorithm	Gradient Based	Mixed Integer	Basic Tuning Parameters	Advanced Tuning Parameters
1. GENEsYs [7]	Genetic Algorithm	No	Yes	2	11
2. Ga2k [8]	Genetic Algorithm	No	Yes	3	7
3. NSGA-II [6]	Genetic Algorithm	No	Yes	2	4
4. ASA [9]	Simulated Annealing	No	Yes	8	7
5. Hooke [11]	Hooke Jeeves	No	Yes	2	2
6. ADS-EP [10]	Exterior Penalty	Yes	No	6	22
7. ADS-SLP [10]	Sequential Linear Programming	Yes	No	6	25
8. Donlp [12]	Sequential Quadratic Programming	Yes	No	3	6
9. NLPQL [13]	Sequential QuadraticProgramming	Yes	No	4	3
10. MOST [14]	Sequential Quadratic Programming	Yes	Yes (BB)	3	5
11. LSGRG [15]	Generalized Reduced Gradient	Yes	No	3	5
12. ADS-MFD [10]	Method of Feasible Directions	Yes	No	5	11
13. ADS-MMFD [10]	Modified Method of Feasible Directions	Yes	No	5	25

for these thirty test cases. The relative error for the starting point, the optimization algorithm that achieved the published optimum in the fewest function evaluations, the relative error achieved by the ga2k genetic algorithm and the percentage of relative error reduced by the genetic algorithm from the starting point are shown. The calculation of relative error is shown in equation 1. The penalty is the sum of constraint violations.

$$\text{RelativeError} = \frac{|\text{current objective} - \text{published optimum}|}{|\text{published optimum}|} + \text{penalty} \quad (1)$$

The key results revealed by the benchmarks are:

1. No single algorithm worked the best in all test cases. In fact, 7 different algorithms proved to be the best for selective test cases.
2. If one had to pick a single algorithm to try for any optimization problem then the genetic algorithm would be a very robust and sound choice. In the benchmarks, the relative error of the optimum point found by ga2k averaged a reduction in relative error from the starting relative error by over 85%. This is a significant result. Keep in mind that the goal of the designer is to find the best design that meets customer constraints within the design deadline. The genetic algorithm found a feasible design on 29 of the thirty test cases. On only Sandgren 27 did the genetic algorithm fail to achieve a better design.
3. Genetic algorithms require one to two orders of magnitude more function evaluations than numerical algorithms.
4. Genetic algorithms are clearly superior to numerical optimization techniques on mixed integer problems when the simulation program does not accept non

Table 2. Benchmark results on test suite

Problem	Real Design Variables	Integer Design Variables	Inequality Constraints	Equality Constraints	Relative Error Starting Point	Feasible Starting Point	Name Best Algorithm	Function Evaluations Best Algorithm	Relative Error ga2k	Error Reduction ga2k
Sandgren 1	5	0	10	0	1.6192	YES	NLPQL	32	0.03096	0.98
Sandgren 2	3	0	2	0	0.697	YES	GRG	25	0.0003	0.99
Sandgren 3	5	0	6	0	0.0119	YES	DONLP	32	0.00799	0.32
Sandgren 4	4	0	0	0	19192	YES	ADS-EP	184	0.378	0.99
Sandgren 5	2	0	0	0	24.2	YES	GRG	166	0.026	0.99
Sandgren 6	6	0	0	4	1961.4	NO	DONLP	96	0.00528	0.99
Sandgren 7	2	0	1	0	53.554	NO	NLPQL	36	0.00031	0.99
Sandgren 8	3	0	2	0	0.6755	NO	NLPQL	21	0.00083	0.99
Sandgren 9	3	0	9	0	0.7892	YES	GRG	1692	0.27248	0.65
Sandgren 10	2	0	0	0	1468.7	YES	MMFD	32	5.7E-05	1.0
Sandgren 11	2	0	2	0	0.4236	YES	MMFD	18	0.00419	0.99
Sandgren 12	4	0	0	0	0.024	YES	NLPQL	61	0.00112	0.95
Sandgren 13	4	0	3	0	0.0456	YES	GA2K	10000	0.0	1.0
Sandgren 14	15	0	5	0	73.192	YES	DONLP	412	7.75308	0.89
Sandgren 15	16	0	0	8	18.212	NO	NLPQL	141	7.80919	0.57
Sandgren 16	3	0	14	0	0.253	YES	NLPQL	85	3.1E-05	0.99
Sandgren 17	12	0	3	0	2.4394	NO	GRG	132	0.29052	0.88
Sandgren 18	7	0	14	0	0.7315	YES	MOST	97	0.4954	0.32
Sandgren 19	8	0	4	0	0.6672	NO	MOST	136	0.17183	0.74
Sandgren 20	8	0	6	0	1.1829	NO	MOST	155	0.17048	0.85
Sandgren 21	13	0	13	0	4.0012	NO	MOST	1196	1.02186	0.74
Sandgren 22	7	0	4	0	647.91	NO	NLPQL	179	0.25652	0.99
Sandgren 24	4	0	5	0	5.6398	YES	NLPQL	56	0.79119	0.85
Sandgren 25	6	0	4	0	36.954	YES	NLPQL	180	0.71947	0.98
Sandgren 26	3	0	0	1	211.5	NO	GRG	121	0.47662	0.99
Sandgren 27	48	0	0	2	1.1545	YES	MOST	4708	1.1545	0.0
Sandgren 29	10	0	14	1	2.8149	YES	MOST	442	1.32457	0.52
Sandgren 30	19	0	1	11	6809.3	NO	NLPQL	132	5157.12	0.24
Sandgren 31	1	2	8	0	90.069	NO	GA2K	10000	0.48947	0.99
Van 32	4	6	11	0	2.5544	NO	GA2K	10000	0.11022	0.95

integer values for integer parameters (e.g., problems Sandgren 31 and Van 32).

5. Genetic algorithms are least effective on problems with equality constraints (e.g. Sandgren problems 15, 27, 29 and 30).

This section describes the genetic algorithm in more detail as they are not as well known and not as well tested in the engineering community. Three genetic algorithm packages, GENEsYs, ga2k, and NSGA-II were benchmarked. All packages supported the parallel evaluation of each design in the population.

GENEsYs' default settings are for elitism, binary gray encoding, linear rank based selection with a maximum fitness of 1.1 and a minimum fitness of .9, two point crossover with a .6 crossover rate, standard mutation with a .01 mutation rate and seeding. The first population is seeded with the initial design point, 20% small creep, 20% large creep, 20% boundaries and random selection. With

small creep, large creep and boundary each design parameter has a 50 percent chance of selection. If the parameter is selected with small creep, the parameter will be randomly varied by 0-3%. If selected by large creep the parameter will be varied by 0-10% and if selected with boundary, the parameter will be randomly set to either the upper or lower bound. The rationale for the small and large creep operators is to leverage the implicit knowledge provided in the initial user supplied design point. The boundary seeding leverages the heuristic that many design parameters are active at their boundaries at the optimal design point.

ga2k is a distributed genetic algorithm [16]. The package's default settings are for elitism, binary gray encoding, tournament based selection, 10 subpopulations of size 10 with a migration rate of .5 at every 5th generation, two point crossover with a 1.0 crossover rate, standard mutation with a .01 mutation rate and no seeding. This algorithm significantly outperformed GENEsYs in the set of benchmark test cases described in Section 2.2.

NSGA-II is a multiobjective genetic algorithm. The package's default settings are for elitism, real encoding, tournament selection, a population size of 100, 100 generations, SBX crossover at a rate of .9, real mutation crossover at a rate of 1/(number of design variables), a crossover distribution index of 20 and a mutation distribution index of 100. The NSGA-II algorithm was similar in performance to ga2k. In the 30 benchmarked test cases, it was better in 14, worse in 9 and equal in 7.

The benchmark tests are helpful but are not completely consistent with the types of applications encountered in industry. They are lacking in the following areas:

1. Their execution time is short compared to the times required of simulation programs which take anywhere from 1 minute to a few hours to execute a single evaluation. The simulation programs are typically thousands or hundreds of thousands of lines of code.
2. The continuous benchmark codes are relatively smooth, well behaved landscapes. In practice, the simulation programs are not smooth, have many discontinuities introduced by if statements, casts and parameter settings. In addition, the programs frequently provide less precision in their output parameters then desired by finite difference algorithms. For example, results may be provided to two digit decimal accuracy where numerical finite gradient algorithms typically are looking at changes in the 4th or 5th decimal digit.
3. The design parameters are typically not independent variables.

These differences make the results of the benchmark only a guideline rather than an absolute measurement of the usefulness of each algorithm in real world situations.

The iSIGHT design environment [1] was developed to provide an industrial solution to this design optimization problem. It has a process integration toolkit that allows simulation codes to be coupled into the environment without source code or reprogramming. The iSIGHT optimization toolkit, which is discussed in the next section, supplies a variety of algorithms which make different as-

sumptions about the design space. An optimization advisor was developed to recommend the appropriate algorithms to novice users.

3 Optimization Toolkit

The iSIGHT Optimization Toolkit provides a number of industry leading implementations of optimization algorithms and a mechanism to provide an optimization plan from an interdigitation of individual algorithms to solve complex problems.

3.1 Strength and Weaknesses of Optimization Packages

More than a dozen packages from both public and private domains are provided in the toolkit and in a few cases multiple implementations of each. There are advantages and disadvantages to having this amount. The advantages are:

1. Each algorithm has one or more strengths that enables it to find a better design than another algorithm for certain design space conditions. For example, Exterior Penalty and Sequential Quadratic Programming work well when started from an infeasible starting point, GRG works well with equality constraints and Genetic Algorithm works well with non convex, multi-modal or mixed integer problems.
2. Some algorithms are extremely efficient for certain design space conditions. For example, gradient based techniques work well in smooth, unimodal, convex design spaces with twenty or fewer design variables.
3. An optimization expert has access to an assortment of algorithms, packages and tuning parameters to best match the characteristics of the algorithm to the characteristics of the product design space.
4. Each package has proven itself on at least one user application and is maintained for backward compatibility. However, certain packages such as GENEsYs have not been routinely updated by the developer and have fallen behind other packages such as ga2k in performance. If a designer selects one of these packages (e.g., GENEsYs) then the designer is notified that the selected package is not recommended and they should consider an alternative package (e.g., ga2k).

The disadvantages are:

1. For a novice user, the choices are confusing and intimidating. In fact, the choice of more than a dozen algorithms with close to 200 tuning parameters could be a larger optimization problem then the problem the designer is trying to solve. In this case, the problem of changing parameters has merely been transferred from the product parameters to the optimization parameters.
2. The diagnostic tools to determine how the package is performing are different for each package. A user experienced in one package cannot easily leverage

his experience to work with another package. As a result, designers will oftentimes use a single package regardless of the characteristics of the design space.

These disadvantages are overcome in two ways. First, iSIGHT has an optimization advisor to automatically pick packages for the user. (The advisor will be discussed in section 3.2). Second, the genetic algorithm is a general purpose algorithm that will work in any design space. A key lesson learned is that although the genetic algorithm cannot compete with numerical techniques in terms of function evaluations for smooth optimization problems, it will always find some improvement and with todays computing environment many of the function evaluations can be done in parallel.

iSIGHT actively supports only a handful of the best performing algorithms listed in Table 1 and deprecates the older or poorer performing packages. However, iSIGHT provides a set of application programming interfaces that allow the designers to couple their own package. In addition, iSIGHT supports Interdigitation which allows the user to create a hybrid technique from one or more existing techniques [21]. Interdigitation can be as simple as a sequential execution of an exploratory genetic algorithm followed by an exploitive numerical optimization or can have complicated scripting with conditional branches and loops among the optimization techniques.

3.2 Optimization Advisor

While the large number of algorithms with direct access to their tuning parameters provides a powerful tool for the expert in optimization, it is overwhelming to the majority of designers. In fact, less than a fraction of 1 percent of users using optimization are Operations Research professionals [19]. To address the needs of this vast majority of novice designers, iSIGHT provides an optimization advisor that automatically selects the best two optimization algorithms based on its analysis of the problem's characteristics. Alternatively, the user can view a priority listing of the recommended optimization packages. For example, on the Van32 benchmark which has a low number of design variables, a low number of design constraints, mixed types of parameters, small variable variance, no equality constraints, a discontinuous design space, non linear simulation codes, an initial infeasible design, low execution time, and no availability of simulation code gradients, the optimization advisor recommended a combination of a genetic algorithm, ga2k, followed by adaptive simulated annealing. For this example, the priority ranking of applicable optimization techniques was: genetic algorithm, adaptive simulated annealing, Hooke Jeeves, and MOST. The continuous numerical optimization techniques are not applicable and are not listed by the advisor for this mixed integer formulation. On the other hand, in a continuous design space with real parameters and high execution time of the simulation code, numerical optimization algorithms will have the highest priority.

During the past three years of iSIGHT application to a variety of real world problems, the genetic algorithm has been the optimization advisor's most fre-

quently recommended technique. In many cases, it is not the highest recommended technique but its frequent recommendation recognizes the robustness of the technique.

4 Applications

This section demonstrates the robustness of the genetic algorithm in its successful application to three different types of real world problem formulations. The first application is a mixed integer design, the second is a pure integer combinatorial problem and the last is a continuous parameter problem in a multimodal design space.

4.1 Marine Propeller Design

Numerous competing design and performance parameters must be considered in propeller design. A properly designed propeller must balance the competing requirements characterized by cavitation, cost, efficiency, noise, strength, thrust, and vibration. The design space contains many local optimal designs.

Several codes are required for the preliminary design. The designers used the process integration toolkit to integrate six simulation codes whose function in the preliminary design process can be described as follows:

- Compute the mean and harmonic wake velocity components
- Calculate propulsive efficiency, tip vortex and cavitation inception speed
- Determine structural stress and weight
- Find surface cavitation inception speeds
- Compute vibratory forces and moments

The mixed integer design variables for this problem include the propeller diameter, speed (RPM), and several spline parameters that characterize the chord distribution, thickness distribution, and loading distribution. The design goal was to minimize the propeller weight, subject to limits on the minimum propulsive efficiency, minimum cavitation inception speed, minimum tip vortex speed and a maximum allowable stress.

Since the problem was mixed integer, the genetic algorithm was used with the default tuning parameter settings. A manual design was used as the baseline for comparison with the genetic algorithm solution. Figure 1 compares the chord distributions and propeller thickness distributions of the baseline and genetic algorithm optimized designs. The genetic algorithm design met all the design constraints and propulsion goal of 0.68 while reducing the weight 181 lbs below the baseline design weight. The net result was an improvement of 17% in the weight.

4.2 Heart Pacemaker Antenna Design

This case study details the design of an implantable patch antenna for a human heart pacemaker. Enabling two-way communication with pacemakers will make

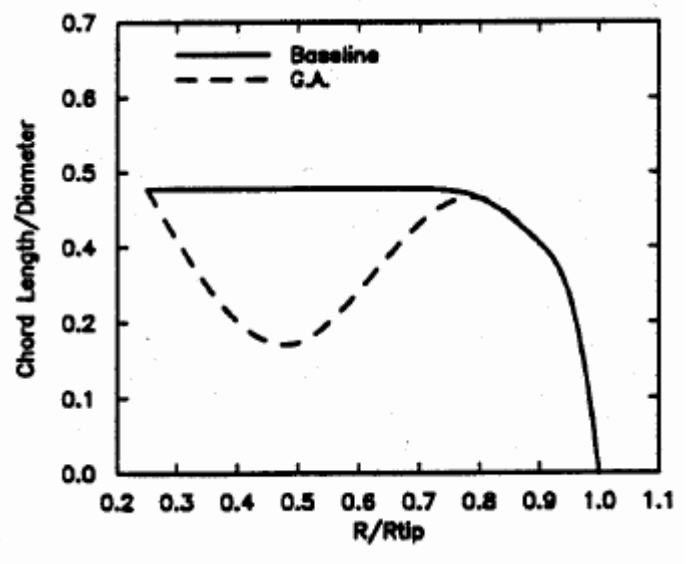

Fig. 1. Chord distribution of Baseline and genetic algorithm designs for marine propeller

it possible to download information about the condition of the pacemaker and the patients heart and upload improved instrument settings. Fitting pacemakers with antennas is challenging, because either the wavelength is too short to penetrate the body or the antenna is so large that it is impractical to implement. Antennas are normally 5 or 6 inches long to be used at the required frequency, long enough that they protrude into the body and risk infection or lung punctures. Researchers at Utah State University solved this problem by developing a two-inch-square 433 MHz patch antenna small enough to fit on a standard pacemaker battery pack [20]. The designers set out to design a 50-ohm antenna with the required performance characteristics that could fit onto the battery pack of the device, which would make it virtually noninvasive

Electromagnetic finite difference time domain software called XFDTD from Remcom Incorporated was used to evaluate the performance of specific designs. The design team worked on the problem and quickly realized that the design space was very sparsely populated with feasible designs that met all the stringent requirements. This made the discovery of an acceptable (much less optimal) patch antenna geometry a very difficult task. After 9 months of manual iteration they finally developed two optimized designs that met the requirements of the project - a u-shaped patch antenna and a spiral shaped antenna. But the design team recognized that the design would require numerous subsequent design iterations to meet additional requirements that arose as the project evolved.

iSIGHT was linked with the XFDTD simulation package. The design parameters included the length of the antenna and the locations of the feed and

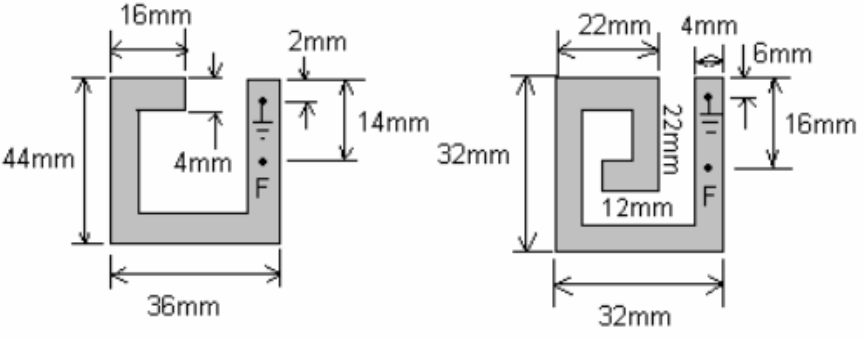

Fig. 2. Comparison of best antenna designs found by iSIGHT and manual design

ground pins. Design constraints on the maximum antenna length, pin locations, and impedance requirements were specified. The design objective was to minimize the antenna length and meet the target frequency of 433 MHz subject to the various design constraints. A genetic algorithm was used as this was a completely integer combinatorial problem. The genetic algorithm found a solution that met all the requirements and was better than the best manual designs. Figure 2 shows a comparison of the iSIGHT and manual trial-and-error best designs. The antenna length of the iSIGHT optimized design, shown on the left, was 128 mm compared with a length of 132 mm found by the 9 month trial-and-error method.

4.3 Jet Engine Airfoil Turbine Design

Researchers at Brigham Young University (BYU) recently completed a complex, multi-discipline design optimization of a cooled jet engine turbine airfoil. The optimization required running multiple software codes to generate a parametric solid model, and execute aerodynamic and structural analyses.

The parametric model was generated in Unigraphics (UG). This model was used as the foundation for the design and subsequent optimization. The structural analysis was performed in ANSYS while the GAMBIT and FLUENT software tools were used to compute the aerodynamic analysis.

The airfoil geometry was created from four sections, each section with seven parameters, for a total of 28 design variables. The design parameters or variables include the chord length, leading and trailing angles, leading and trailing radius, leading and trailing wall thickness and trailing radius thickness. The objectives of the optimization were to minimize the pressure loss and weight and to maximize the safety factor based on the stress in the airfoil subject to stress constraints and geometric limits.

The analysis for this problem was highly complex and required extremely long computational times to solve. The genetic algorithm available in iSIGHT was selected to complete the optimization study. The genetic algorithm was chosen

for its abilities to search for a set of designs, explore the entire multi-modal design space, and run in parallel. A population size of 50 was used and 100 generations were run. The optimization was run automatically in parallel by iSIGHT on a 64 processor SGI and required approximately 20 hours to complete. 65 solutions were selected from the 5000 designs generated to form the pareto set. This set of 65 designs showed a decrease in the blade volume ranging from 2% to 19%, an increase in the safety factor of 2% to 8%, and pressure loss decreases of 2% to 38%.

5 Lessons Learned

The iSIGHT generic engineering design environment described in this paper has had a genetic algorithm as part of its optimization toolkit since 1990. The genetic algorithm has played a fundamental role in the robustness of the toolkit when applied individually as shown in the three real world applications discussed in this paper or when used with interdigitation to design an aircraft engine turbine [23]. It is the robustness of the genetic algorithm and its ease of use that makes it especially attractive to the majority of users who are not experts in optimization. The concept of the algorithm is easy to understand and there are no limiting design space assumptions of parameter independence or continuity to worry about. The genetic algorithms key advantage is it applies to almost every type of problem. This claim is supported by the genetic algorithm finding improved designs on 29 of 30 benchmarked engineering problems. This high success rate of 96% makes the genetic algorithm a reasonable choice when selecting an initial algorithm for a design when the problem domain is not very well known. The increased number of function evaluations required by the genetic algorithm is mitigated by the use of parallel computing and makes the use of the genetic algorithm feasible in real world problems.

References

1. Engineous Software Incorporated. www.engineous.com
2. Vanderplaats, G.: Numerical Optimization Techniques for Engineering Design. (1999).
3. Belegund, A. and Chandrupatla, T.: Optimization Concepts and Applications in Engineering. Prentice Hall (1999)
4. Onwubiko, C.: Introduction to Engineering Design Optimization. Prentice Hall (2000)
5. Gen, M. and Cheng, R: Genetic Algorithms and Engineering Optimization. John Wiley & Sons (2000).
6. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons (2001).
7. Back, T.: A Users Guide to GENEsYs 1.0. University of Dortumnd. (1992)
8. Hiroyasu, T.: Spec Sheet: Distributed Genetic Algorithms ga2k (ver 1.1). Intelligent Systems Design Lab. Doushisya University (2002)
9. Inger, L.: Adaptive Simulated Annealing. http://www.ingber.com/. (2002)

10. Vanderplaats, G.: ADS – A Fortran Program for Automated Design Synthesis. Santa Barbara, CA: Engineering Design Optimization, Inc. (1988)
11. Johnson, M.: Hooke and Jeeves Algorithm. http://www.netlib.org/opt/hooke.c (1994)
12. Spellucci, P.: Donlp2 User Guide. http://www.netlib.org/opt/donlp2/donlp2doc.ps
13. Schittkowski, K.: NLPQL: A Fortran subroutine for solving constrained non linear programs. Annals of Operations Research, Vol.5, (1985–1986) 4850–500
14. Tseng, C.: MOST 1.1. Applied Optimal Design Laboratory, National Chiao Tung Univeristy, Technical Report AODL-9-01 (1996)
15. Smith, S. and Lasdon, L.: Solving large sparse nonlinear programs using GRG. ORSA J. Comput. 4, (1992) 1–15
16. Tanese, R.: Distributed genetic algorithms. Proceedings of the Third International Conference on Genetic Algorithms, (1989) 434–439.
17. Sandgren, E.: The utility of nonlinear programming algorithms. Purdue University Ph.D. Thesis, West Lafayette, IN (1977).
18. Sandgren, E.: Nonlinear integer and discrete programming in mechanical design optimization. Transactions of the ASME, Journal of Mechanical Design, 112(2), (1990) 223–229
19. Fylstra, D., Lasdon, L., Watson, J. and Waren, A.: Design and Use of the Microsoft Excel Solver. Interfaces 28 (1998) 29–55.
20. Furse, C.: Design an Antenna for Pacemaker Communication. Microwaves & RF, March (2000)
21. Powell, D.: Inter-GEN: A hybrid approach to engineering design optimization. Rensselaer Polytechnic Institute Ph.D. Thesis, Troy, NY (1990).
22. Vanderplaats, G.: Numerical Optimization Techniques for Engineering Design. (1999) 317–320.
23. Powell, D., Skolnick, M., and Tong, S.: Interdigitation: A Hybrid Technique for Engineering Design Optimization. Handbook of Genetic Algorithms. Van Nostrand Reinhold. (1991) 312–331.