Isight Design Optimization Methodologies

Dr. Alex Van der Velden, Director, SIMULIA
Dr. Pat Koch, Manager, SIMULIA

SIMULIA

CONTENTS

Introduction

Optimization finds application in every branch of engineering and science. The process of optimization involves choosing the best solution from a pool of potential candidate solutions such that the chosen solution is better than the rest in certain aspects. Design optimization is the process whereby a selected set of input *design variables* is varied automatically by an *algorithm* in order to achieve more desired outputs. These outputs typically represent the variation from a target, minimal cost, and/or maximal performance.

In order for design variables to be varied automatically, we need algorithms that search the design domain. For this purpose, we need to be able to compute the outputs of interest automatically. Even though the word “optimization” is used, only trying out all relevant combinations can guarantee that we, indeed, have found “the best” design parameters for an arbitrary complex space. In practice, this takes too much time. If we consider, for instance, a simple problem with ten possible discrete values for five parameters and a five-minute analysis time, we would need a year to analyze all combinations.

As such, the practical value of a particular optimization algorithm is its ability to find a better solution—within a given “clock time”—than a solution obtained by a manual search method or another algorithm. This “clock time” includes the effort it takes to set up the simulation process and to configure the optimization methods. To minimize the set-up time, commercial software like Isight can be used (Ref. 1). The set-up of the simulation process varies from problem to problem; here, we will focus on the optimization methodologies.

The “no-free-lunch” theorem of Wolpert and Macready (Ref. 2) states: “…for any [optimization] algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class.”

This concept is shown in Figure 1. On the diagonal, we plot a set of arbitrary problems f_i ordered by the minimum number of iterations n_{min} and required by a set of methods A, B, … ,Z to solve this particular problem.

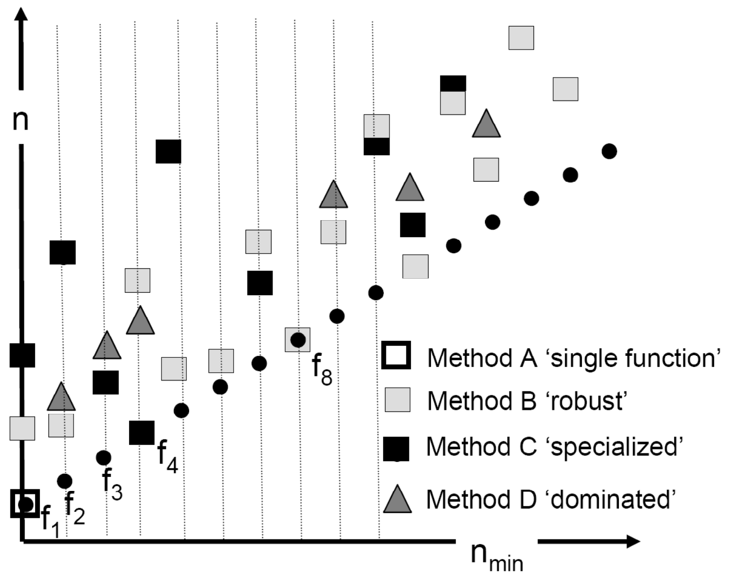


Figure 1. An illustration of the “no-free-lunch” theorem for different types of optimizers.

For instance, consider the problem of finding x_i for Min[f_1 (x_i)]= constant. There exists a method A that finds the minimum value of f_1 (x_i) with a single-function evaluation. Method A, called the “lucky guess” method, simply tries a random number with a fixed seed. Its first guess happens to be the optimal value of this problem. Obviously, this method is not very efficient for any other problem. The efficient performance for Problem 1 goes at the expense of the efficiency to solve all other problems 2, ... ,n.

The minimum value of f_1 is also found by Method B, but this method is not as efficient as our “lucky guess” method. Method B is a genetic algorithm. In its first iteration, Method B first computes a number of random samples of x_i with respect to f_1 before deciding the next set of samples (second iteration) based on proximity to the lowest function values in the first iteration. Since Method B already requires several samples before the first iteration, Method B is not as efficient as Method A for Problem 1. However, it does pretty well on a variety of problems including 1, 2, 4, 5, 7, and 8. It is the most efficient method for Problem 8.

Method C is a gradient method and may need to evaluate the gradients of x_i with respect to f_1 before completing the first iteration step. Because of that, it is obviously not as efficient as the “lucky guess” method for Problem 1. Even though it is the most efficient method for Problem 4 (which happens to be a linear function), for most problems, Method B is more robust. The gradient method often gets stuck in local minima. Method C is not as robust as Method B because it only gets the best answer two times versus Method B’s nine times for the set of methods and problems we are considering,

We also tried Method D. Method D samples the space with a design of experiments technique and shrinks the search space around the best point in the array for each iteration. It is able to solve quite a few problems, but it is inefficient and would therefore be considered dominated by other methods over the set of problems f1, f2… fn.

This meant that we needed to develop an environment that allowed the introduction of many algorithms specifically suited to solve certain classes of customer problems. The open component architecture of Isight (Refs. 1, 9) allows the development of these design drivers independently from the product release cycle. However, in many cases, customers do not have such specialized algorithms available and are looking for a commercial product to improve their designs.

For that purpose, we and our partners provide a set of best-of-class general-purpose and specialized-purpose algorithms that work out of the box. Our optimizers solve both deterministic and non-deterministic single- and multiple-objective functions. A deterministic function always returns the same result when called with a specific set of input values, while a non-deterministic (stochastic) function may return different results when they are called with a specific set of input values. In the following sections we will give a description of all of these classes of problems and the optimization methods that solve them.

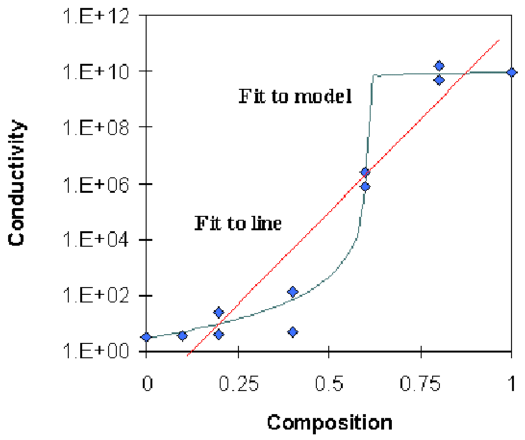
The Deterministic Single Objective Problem

In the case of a single objective problem, we are maximizing or minimizing a single output and/or constraining a set of outputs to stay within a certain range.

Minimize	$f(\mathbf{x})$	
	$f(\mathbf{x})=w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x})\ldots + w_m f_m(\mathbf{x})$	
Subject to	$g_j(\mathbf{x}) \le 0,$	$j = 1, 2, \ldots, J$
	$h_k(\mathbf{x}) = 0,$	$k = 1, 2, \ldots, K$
	$x_i^{(L)} \le x_i \le x_i^{(U)},$	$i = 1, 2, \ldots, N$

A good example of a single objective material-processing application is data matching (a/k/a model fitting, parameter estimation), shown in Figure 2. Here, the objective is to minimize an error function between a parametric model and experimental data.

The selection of the right objective is the most critical aspect of optimization. In the case of Figure 2, the objective is a straightforward error minimization between model and experiment. The only question here is whether the selected parametric form does not "overfit" the data. To make a convincing argument that the model is valid in general, the same model should be fit to several sets of experimental data. The single objective error function could be averaged over the set.



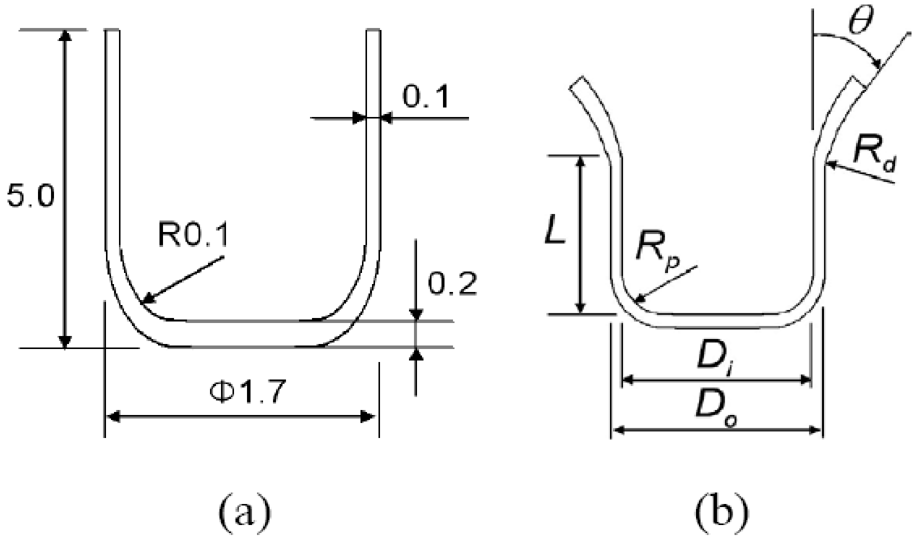
$$(v_i)\frac{\sigma_i^{1/s}-\sigma_m^{1/s}}{\sigma_i^{1/s}+\left(\frac{1}{\phi_c}-1\right)\sigma_m^{1/s}}+(v_c)\frac{\sigma_c^{1/t}-\sigma_m^{1/t}}{\sigma_c^{1/t}+\left(\frac{1}{\phi_c}-1\right)\sigma_m^{1/t}}=0$$

Figure 2. The data-matching application. Composite conductivity should behave according to the McLachlan Equation. Fitting the parameters (design variables) of this equation gives a better fit than a linear function. (Ref. 3)

Apart from the selection of objectives, the second most important thing the user can do to improve the optimization process is to select an appropriate set of design variables. Often, variables can be coupled in such a way that the volume fraction of good solutions in the design space is maximized. This can be done according to the Buckingham PI theorem (Ref. 25) or other scaling methods.

Reducing the design space complexity will make it easier for the algorithm to find improvements. However, to find an improvement, we need at least as many active degrees of freedom to improve the design as there are active constraints. If not enough degrees of freedom are available, the design is effectively frozen in its current state.

A good example of systematic selection of variables was given by Kim et al (Ref. 30) for the multi-stage deep-drawing process of molybdenum sheet (Figure 3.). Since molybdenum has a low drawing ratio, it requires many drawing stages to be transformed into a cup shape. For each part of the drawing process, the authors investigated the proper set of design parameters considering process continuity (clearance, die corner radius, intake angle, etc). The purpose of this study is to find out the proper eight-stage drawing process that minimizes the maximum strain in the resulting cup shape.



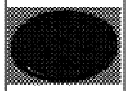
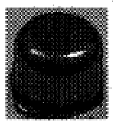
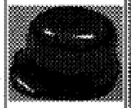
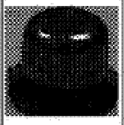
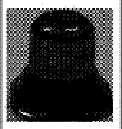
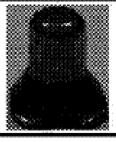
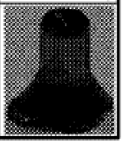
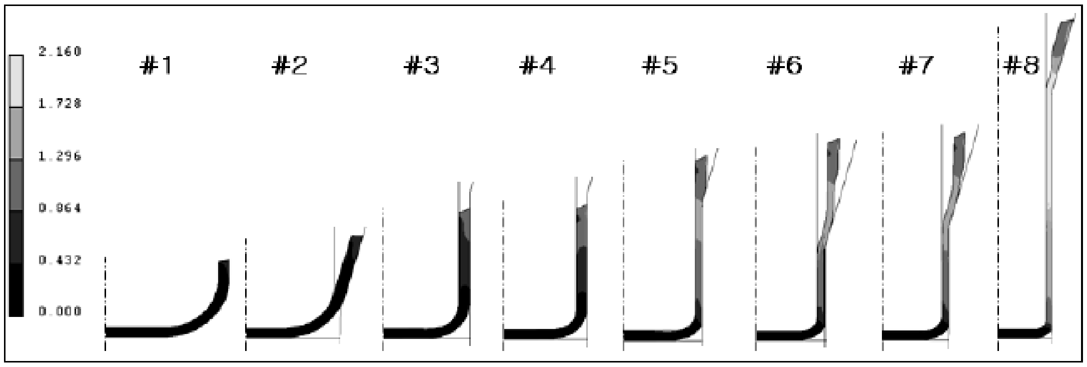
Stage	Blank	#1	#2	#3	#4	#5	#6
Picture							

Figure 3a. The simulated drawing process to produce a molybdenum cup (Ref. 30). The top drawings show the design variables for the (a) final target shape and (b) the intermediate stages. The bottom pictures show a representative presentation of the draw process for the initial design.

Prior to executing the optimization, the authors did a thorough study of how the design variables interacted with each other. For instance, the authors discovered that the intake angle *θ* and the drawing radius R_d had an impact on the maximum stroke *L* for every stage. The maximum stroke is defined as the value of *L* at which the maximum strain in the cup exceeds the limiting material strain value. The reason for the importance of the intake angle was the fact that the frictional force between the flange and the die hindered the material flow into the punch-die gap, and the flange-die contact area decreases as the intake angle increases. Therefore, if the intake angle were not a design variable, it would significantly influence the outcome of any optimization effort.



Predicted maximum effective strains, $\bar{\varepsilon}_{\max}$, after each stage for the initial and the optimum design			
Stage	Initial design (A)	Optimum design (B)	Strain ratio (B/A)
#2	1.11	0.77	0.69
#3	1.59	1.32	0.83
#4	1.86	1.43	0.77
#5	2.12	1.54	0.73
#6	2.56	1.53	0.60
#7	2.66	1.55	0.58
#8	2.78	2.16	0.78

Figure 3b. The top graph shows the cross-sections of the optimal design for each of the drawing stages. The bottom table shows the reduction in maximum effective strain in the cup during the optimization process.

Even after the critical design variable interactions have been found, it is non-trivial to find the optimal combination of the 24-design-variable problem. Most of the effects are coupled due to the nonlinearity of the geometry and the material response. The authors chose the adaptive simulated annealing optimizer (Ref. 31) to find a cup design with a 22% lower strain as compared to the initial process.

Single Objective Optimization Methodologies

In this section, we will describe optimization algorithms that provide a good set of complementary approaches to solve a wide variety of single-objective mechanical engineering applications.

For differentiable functions, gradient methods can be used. The constraints are handled directly without being converted into penalty functions. The gradient methods are very suitable for parallel execution because the gradients can be computed independently from each other. The process of gradient optimization can be easily illustrated with a Rosenbrock function with a local and a global minimum (Fig. 4).

$Z = 100*(Y-X^2)^2 + (1-X)^2$
Local Minimum [.71, .51]; Z = 0.0876, Global Minimum: [1,1] ; Z = 0

For this purpose we used the LSGRG (Ref. 3) gradient optimizer that we will describe later in this section. In our first attempt, we start from the left part of the design space [-1,1] and the optimizer finds a solution z=0.033 close to the local optimum following the path of steepest descend [0.81, 0.66]. If the optimizer is started from the top right-hand corner of the design space [2,3], it does find a value of the objective z=1.2e-7 extremely close to the global minimum [0.99, 0.99], even though it is initially "side-tracked" in its search. Gradient methods, including very good ones like LSGRG, only find minima in the path of the steepest descend. The optimization is stopped when a convergence condition, such as the Kuhn-Tucker criterion, is satisfied:

- The vector of first derivatives of the objective function *f* (projected toward the feasible region if the problem is constrained) at the point x^* is zero.
- The matrix of second derivatives of the objective function *f* (projected toward the feasible region *G* in the constrained case) at the point x^* is positive definite.

For most engineering problems (with multiple minima), the Kuhn-Tucker criterion can be satisfied without having found the global minimum. The implication is that the solution found the gradient optimizer is now dependent upon the starting point, and this is not very desirable. It is for this reason that we also have to consider other techniques which may be less efficient, but more reliable.

Attempt 1 starting point
X = -1.0; Y = 1.0
Best design point:
X = 0.81 ; Y = 0.66
Objective = 0.033
Close to local optimum

Attempt 2 starting point
X = 2.0; Y = 3.0
Best design point:
X = 0.99; Y = 0.99
Objective = 1.2E-7
Close to Global Optimum

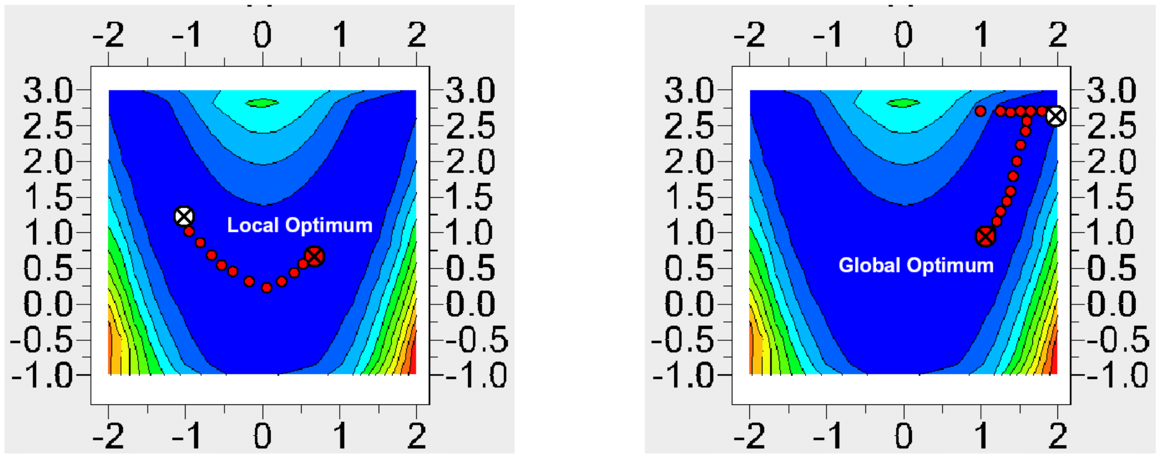


Figure 4. Gradient optimization exercise with the Rosenbrock function from two starting points.

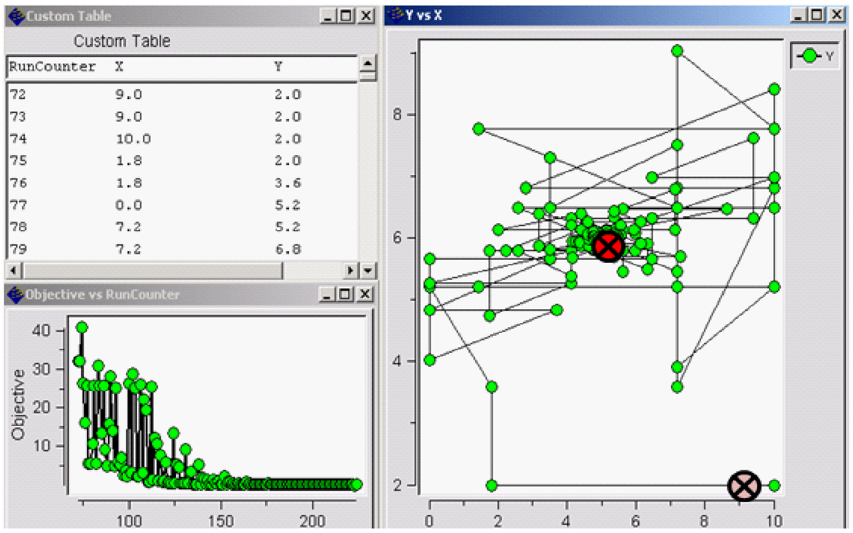
Nonlinear Programming by Quadratic Lagrangian (NLPQL) - Sequential Quadratic Programming (SQP) (Ref. 11) This method builds a quadratic approximation to the Lagrange function and linear approximations to all output constraints at each iteration, starting with the identity matrix for the Hessian of the Lagrangian, and gradually updating it using the BFGS (Broydon-Fletcher-Goldfarb-Shanno) method. On each iteration, a quadratic programming problem is solved to find an improved design until the final convergence to the optimum design.

Modified Method of Feasible Directions (MMFD) (Ref. 12) This method exploits the local area around the initial design point, handles inequality and equality constraints directly, and rapidly obtains a local optimum design. MMFD is used best when starting from a feasible design point. It usually requires multiple iterations consisting of a search direction calculation (using gradients of each variable) and a one-dimensional search. MMFD follows the active constraints during the search until no further improvement can be made. It is well-suited for highly nonlinear design spaces, but not suited for discontinuous design spaces. The method operates on reals and the gradient evaluation can be executed in parallel.

Large-Scale Generalized Reduced Gradient (LSGRG) (Ref. 13) This method uses the generalized reduced gradient algorithm for solving constrained nonlinear optimization problems. The algorithm uses a search direction such that any active constraints remain precisely active for some small move in that direction. The generalized reduced gradient method is an extension of an earlier reduced gradient method that solved equality-constrained problems only.

The next group of optimization methods does not require gradient information, and can be used on non-differentiable functions. The search direction relies on the information obtained by sampling the design space. Constraints violations are added as penalties to the objectives.

Hooke-Jeeves Direct Search (Ref. 15) The Hooke-Jeeves algorithm examines points near the current point by perturbing design variables—one axis at a time—until an improved point is found. It then follows the favorable direction until no more design improvement is possible. The size of variable perturbations is determined by the Relative Step Size. It is gradually reduced by applying the Step Size Reduction Factor until convergence is detected. It is not easily possible to parallelize this method, but it can be very efficient on moderately coupled problems. Hooke-Jeeves is a pattern-search method and not a gradient method. During the search it covers a wide range of the design space. The idea behind this is that the nature of the function is not known a priori so you have to do a wide exploration and not just go down the path of steepest descend. This means that if Hooke-Jeeves is used on a quadratic function like the one shown in Fig. 5, it is obviously less efficient than gradient methods. However, with some tweaking of the tuning parameters like step sizes and number of iterations, it does find the optimum and does so even if multiple local minima are present. The Hooke-Jeeves method does not have a convergence criterion and stops whenever a preset maximum number of runs is reached.



Custom Table

RunCounter	X	Y
72	9.0	2.0
73	9.0	2.0
74	10.0	2.0
75	1.8	2.0
76	1.8	3.6
77	0.0	5.2
78	7.2	5.2
79	7.2	6.8

Figure 5. Hooke-Jeeves optimization exercise with a quadratic function y= (x1-5) ^{2} +(x1-6) ^{2} starting from point [9, 2]

Nealder & Mead Downhill Simplex (Ref. 14) This method samples the space across a sub-region and moves from the worst point in the direction of the opposite face of the simplex toward better solutions. The simplex is a geometrical body with N+1 vertices represented by a triangle in two dimensions and a tetrahedron in three dimensions. The method calculates and compares the objective function at the vertices of a simplex in the variable space, selects the worst one, and moves this point through the opposite face of the simplex to a lower point. If this new vertex is better, the old one is deleted. If there is no improvement after a number of steps, the method "shrinks" the simplex by reducing the length of each side, thus trapping the optimum solution. It is not easily possible to parallelize this method, but it can be very efficient on moderately coupled problems.

Adaptive Simulated Annealing (ASA) (Ref. 31) This algorithm is very well-suited for solving highly nonlinear problems with short-running analysis codes, when finding the global optimum is more important than a quick improvement of the design. Each iteration in the SA perturbates the current solution by a random step size that is chosen based on a probability that depends upon the difference between corresponding function values and a global parameter T. The algorithm is inspired by the annealing process and T (temperature) starts out large and is reduced to very small values as the process advances. The parameter T is automatically adjusted.

Multi-Island Genetic Algorithm (MIGA) (Ref. 33) Genetic algorithms work well because they incorporate randomness in their search. It gives the algorithm the ability to correct deterministic search bottlenecks that are caused by the reasoning in the previous two "space sampling" methods and the gradient methods. The MIGA algorithm divides the population into several islands, performs traditional genetic operations on each island separately, and then migrates individuals between the islands. It searches many designs and multiple locations of the design space. Genetic algorithms like MIGA tend to be less efficient than the two previous methods in this class, but they have the advantage that function evaluations can be executed in parallel.

Hybrid algorithms combine the benefits of several algorithms, usually at some computational expense.

Multifunction Optimization System Tool (MOST) (Ref 43.) can be efficiently used both for continuous optimization problems and for integer or discrete design space optimization, where one or more design variables are restricted to an integer domain. MOST initially executes an SQP algorithm to obtain a continuous solution to the problem. At this stage, all integer variables are treated as continuous variables with a minimum step size of 1.0. If there are any integer (or discrete) variables, MOST will use the continuous solution as the starting point for its modified branch-and-bound algorithm. During this stage, integer variables are dropped one at a time. The reduced continuous optimization problem is solved for each of the dropped variables, fixing their values at integer levels above and below their previously found optimum values. Again, all remaining integer variables are treated as continuous variables with a minimum step size of 1.0.

POINTER - Pointer Automatic Optimizer (Ref. 26) Pointer is an automatic optimization engine that controls a set of standard optimization techniques. It currently controls four optimization methods: an evolutionary algorithm (Ref. 28), the Nelder and Mead downhill simplex method, sequential quadratic programming (NPQL), a linear solver, and a TABU method (Ref. 27). This complementary set of algorithms was selected because each succeeds and fails for different topography features. It has been found that a hybrid combination of these methods solves a broad range of design optimization problems. The Pointer Automatic Optimizer can control one algorithm at a time or all four at once. As the optimization is proceeding, the technique determines which algorithms are most successful as well as optimal internal control parameter settings (step sizes, numbers of iterations, number of restarts, etc.). This procedure is hidden from the user. The goal is to enable the non-optimization expert to successfully utilize these methods.

The Pointer algorithm performance can be seen as an example of a robust method as illustrated in Figure 6. We compared all these optimizers against a standard benchmark test created by Dr. Sandgren (Ref. 6) to present a wide variety of single-objective optimization problems in many fields of mathematics and engineering. No attempt was made to use parallelization of hardware.

Figure 4 shows the results of the test. In each case there was no expert intervention, and all algorithms were used in one setting considered suitable for the benchmark problems considered (ASA, NLPQL, MOST, LSGRG, IOSO). Although in a few cases Pointer was also the most efficient method, on average, Pointer was more than two times as expensive as the most efficient algorithm for the individual benchmark problem. In the case of Problem 13, Pointer was the only algorithm to find a solution. This shows, again, that there is no free lunch.

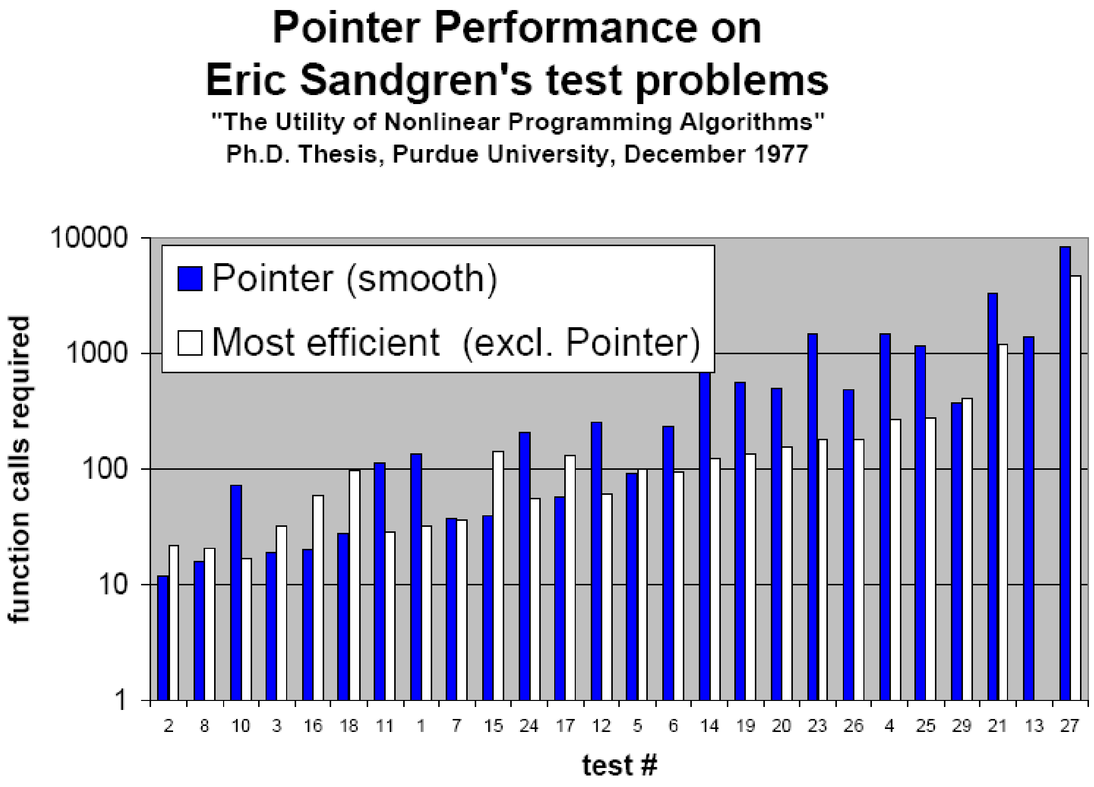


Figure 6. The Pointer broadband optimizer using the 'smooth topology setting' on differentiable functions is compared to the most efficient method. Test problems are sorted by the minimum number of function call required by any method used the benchmark set.

The Deterministic Multi-Objective Problem

While the single-objective design formulation with an efficient algorithm is computationally the least expensive solution for a particular problem, most real-world problems often involve multiple conflicting objectives. Therefore, a significant amount of research has been performed towards the design of multi-objective optimization algorithms. The multi-objective optimization problem that such algorithms attempt to solve is formally stated as:

$$\begin{array}{lll} \text{Minimize} & (f_1(\mathbf{x}),\ f_2(\mathbf{x}),\ \ldots,\ f_M(\mathbf{x})) & \\ \text{Subject to} & g_j(\mathbf{x}) \leq 0, & j = 1, 2, \ldots, J \\ & h_k(\mathbf{x}) = 0, & k = 1, 2, \ldots, K \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \ldots, N \end{array}$$

In most scenarios, the outcome of a multi-objective optimization process is a set of non-dominated *Pareto* solutions (Ref. 16). The usual definition of Pareto-domination that is used in the present context is as follows:

A feasible solution a dominates another feasible solution b for an M-objective minimization problem, if following conditions are met:

1. $f_i^a \leq f_i^b$ for all $i = 1, 2, \ldots, M$
2. $f_i^a < f_i^b$ for at least one $i \in \{1, M\}$.

The identified Pareto solutions define a *Pareto front* or *M*-dimensional *Pareto surface*. Plotting and visualizing the Pareto front is key to understanding the solution space and evaluating tradeoffs between the *M* objectives. The Pareto front can be a simple smooth curve, or a complex discontinuous set of curves/surfaces. Two such examples of Pareto fronts are shown in Figure 7.

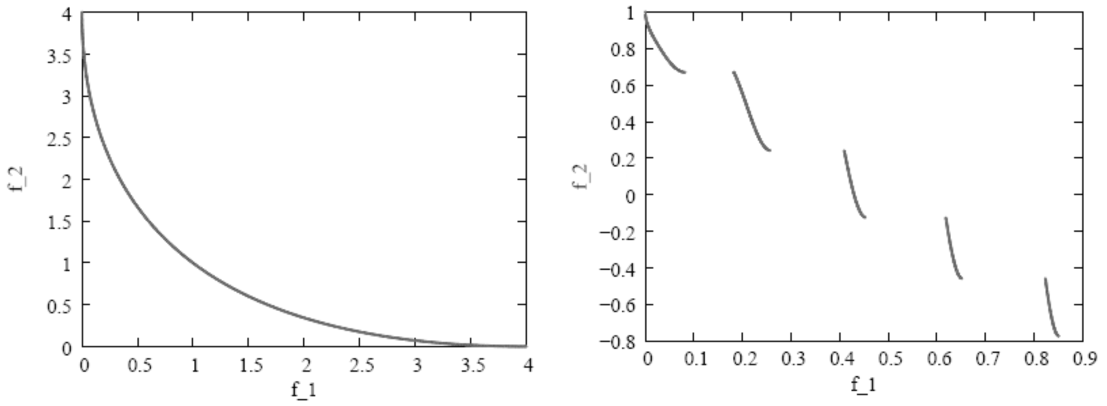


Figure 7. Two example Pareto fronts depicting the tradeoff between objectives f_1 and f_2.

Multi-objective problems can be solved by single-objective methods using a single weighted-sum type objective. The minimum summed objective for each set of weights ($w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) \ldots + w_m f_m(\mathbf{x})$) represents one particular optimal solution on the Pareto front. Even though this is the most efficient way of finding a multi-objective tradeoff, there are a number of drawbacks to this approach. First of all, the weighted sum creates a convex combination of objectives and optimal solutions in non-convex regions are not detected (Figure 8). Second, the proper weighting between objectives and constraints is not always clear up front.

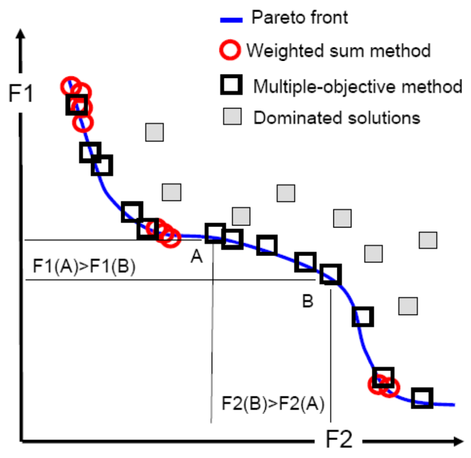


Figure 8. Using the weighted sum approach and true multi-objective algorithms to find the Pareto front.

The method by Kim and de Weck (Ref. 5) addresses the problem of finding solutions in non-convex regions using weighted sums, but other methods are more widespread. These true multi-objective methods are easy to use and do well at capturing the Pareto front at the expense of more function calls.

Multi-Objective Optimization Methodologies

Multi-objective optimization has become mainstream in recent years, and many algorithms to solve multi-objective optimization problems have been suggested. The use of multi-objective optimization in industry has been accelerated by the availability of faster processing units and the computational analysis models for various engineering problems and disciplines. Multi-objective optimization algorithms, especially those based on evolutionary principles, have seen wide acceptability because for most engineering problems, a quick computation of approximate solutions is often desirable. Evolutionary algorithms (EAs) are adaptive search techniques inspired by nature and their working principle is based on Darwin's theory of survival-of-the-fittest (Refs. 17, 18). The adaptive nature of EAs can be exploited to design optimization algorithms by designing suitable variation operators and an appropriate fitness function. The Genetic algorithm (GA) (Ref. 19) is one of the evolutionary techniques that have been successfully used as an optimization tool. Typically, a GA works with a population (a set of solutions) instead of a single solution (individual). This property of a GA makes it an ideal candidate for solving multi-objective optimization problems where the outcome (in most cases) is a set of solutions, rather than a single solution. The population approach of a GA also makes it resilient to premature convergence, thereby making it a powerful tool for handling highly nonlinear and multi-modal functions.

We found that the following genetic algorithms provide a good set of complimentary approaches to solve multi-objective problems in mechanical engineering applications.

Non-dominated Sorting Algorithm (NSGA-II) (Ref. 20) In the Non-dominated Sorting Genetic Algorithm (NSGA-II), each objective parameter is treated separately. Standard genetic operation of mutation and crossover are performed on the designs. The selection process is based on two main mechanisms, "non-dominated sorting", and "crowding distance sorting." By the end of the optimization run, a Pareto set is constructed where each design has the "best" combination of objective values and improving one objective is impossible without sacrificing one or more of the other objectives. NSGA-II is widely used and has become a de facto standard against which the performance of other algorithms is compared.

Neighborhood Cultivation Genetic Algorithm (NCGA) (Ref. 21) In NCGA, each objective parameter is treated separately. Standard genetic operation of mutation and crossover are performed on the designs. The crossover process is based on the “neighborhood cultivation” mechanism, where the crossover is performed mostly between individuals with values close to one of the objectives. By the end of the optimization run, a Pareto set is constructed where each design has the “best” combination of objective values, and improving one objective is impossible without sacrificing one or more of the other objectives.

Archive-based Micro Genetic Algorithm (AMGA) (Ref. 22) AMGA is an evolutionary optimization algorithm and relies on genetic variation operators for creating new solutions. The generation scheme deployed in this algorithm can be classified as generational because during a particular iteration (generation), only solutions created before that iteration take part in the selection process. The algorithm, however, generates a small number of new solutions (recommended value is two solutions per iteration) at every iteration and therefore it can also be classified as an almost steady-state genetic algorithm. The algorithm works with a small population size (recommended value is 4) and maintains an external archive of good solutions obtained. At every iteration, a small number of solutions are created using genetic variation operators. The algorithm is referred to as Archive-based Micro Genetic Algorithm (AMGA), owing to the fact that it works with a very small population size and uses an archive to maintain its search history. It is recommended to use a large size for the archive, and the best results are obtained if the size of the archive is the same as the number of function evaluations allowed (i.e., the algorithm stores its complete search history). The size of the archive determines the computational complexity of the algorithm; however, for computationally expensive optimization problems, the actual time taken by the algorithm is negligible. The parent population is updated using the archive and binary tournament selection is performed on the parent population (for creating the mating population).

Multi-Objective Optimization Study

Evaluating and comparing multi-objective optimization algorithms is more involved than evaluating and comparing single objective algorithms. Rather than simply comparing the objective value at a single solution point and the number of system evaluations required to achieve the solution objective value, evaluating the solution set provided by a multi-objective optimization algorithm requires comparison of Pareto sets for accuracy and completeness.

In recent studies, four unary performance indicators are commonly used to compare the ability of multi-objective optimization algorithms to characterize the Pareto front (i.e. they compare a single non-dominated set to a Pareto-optimal frontier). The performance indicators are *Delineation*, *Distance*, *Diversity,* and *Hypervolume* (Ref. 23). A brief description of each performance indicator is as follows:

- Delineation Metric: It measures “how much of the true Pareto-optimal front is represented by the obtained solution set.”
- Distance Metric: It measures the average Euclidean distance between the true Pareto-optimal front and the obtained solution set.

- Diversity Metric: It measures the uniformity of distribution and the spread of obtained solution set.
- Hypervolume Metric: It measures the fraction of search space not dominated by the obtained solution set in comparison to the true Pareto-optimal set.

In order to use these performance indicators, the true Pareto-optimal front must be known. Smaller value for a performance indicator means a better solution set. Ideally, if the original Pareto-optimal front is used as the solution set, all the performance indicators should evaluate to zero. Since a finite number of points (1,000 points for the problems presented here) are used to represent the true Pareto-optimal frontier, a value of 0.01 or less for a performance indicator implies that the obtained solution set is virtually indistinguishable from the Pareto-optimal front. If the value of the performance indicator is 0.5 or more, it implies that an acceptable solution set was not obtained. All the objectives are normalized (the Pareto-optimal set is mapped to the range [0, 1]) before the performance indicators are computed. Only the non-dominated solutions belonging to rank 1 are considered for computing the performance indicators.

Results from NSGA-II, NCGA, and AMGA are presented in Table 1 for five common multi-objective optimization test problems, taken from (Ref. 24): ZDT1, ZDT2, ZDT3, ZDT4, ZDT6. Each problem has two objectives. Minimization for both of the objectives for all of the test problems is assumed. ZDT1-3 have 30 design variables, and ZDT4 and ZDT6 have 10 design variables.

Even though these algorithms are quite robust, a search from a given starting point still produces a slightly different answer because of the randomness in the search procedure. To account for this in our comparison, we executed 15 simulation runs starting with different random seeds for each algorithm-problem pair (a total of 15 × 3 × 5 = 225 simulations are performed). The size of the initial population used for all of the algorithms is 100. The number of generations used is 40 for all except ZDT4, which used 100 generations, giving a total number of function evaluations of 10,000 for ZDT4 and 4000 for all other problems.

Table 1 presents the median value for each performance indicator for each algorithm, across the 15 executions of each algorithm, for each test problem. The best (lowest) metric values for each problem are highlighted in bold. It is evident from the simulation results that AMGA has the overall best performance, obtaining the best metric values for all problems and all metrics except for diversity for ZDT4, for which NSGA-II obtained the best value. AMGA is capable of reporting a large number of non-dominated solutions for the same number of function evaluations. The development of AMGA can also be perceived as an exercise in combining the best features of different algorithms and best practices into a unified optimization framework. Although the computational complexity of the algorithm is more than either NSGA-II or NCGA, the execution time is not affected drastically. Almost the entire execution time is consumed by the analysis routines, and therefore the algorithm can afford to perform expensive operations if such operations can result in reduced number of function evaluations. The two guiding principles that have shaped the design of the AMGA algorithm are focused on reducing the number of function evaluations for the same degree of convergence, and making the algorithm immune to changes in sizing or tuning parameters. Limited success has been achieved by AMGA in fulfilling these goals.

Problem	Algorithm	Delineation	Distance	Diversity	Hypervolume
ZDT1	NSGA-II	0.055573	0.050177	0.130880	0.109433
	NCGA	0.068537	0.063598	0.109231	0.107068
	AMGA	**0.033043**	**0.023070**	**0.071707**	**0.069749**
ZDT2	NSGA-II	0.080988	0.070241	0.326157	0.257817
	NCGA	0.149708	0.136145	0.502602	0.415069
	AMGA	**0.037432**	**0.029006**	**0.095623**	**0.135636**
ZDT3	NSGA-II	0.038556	0.029505	0.143943	0.100310
	NCGA	0.043921	0.028587	0.156448	0.127858
	AMGA	**0.020222**	**0.009450**	**0.088270**	**0.055998**
ZDT4	NSGA-II	0.029920	0.014183	**0.085797**	0.059543
	NCGA	0.958743	1.011684	1.884524	0.993992
	AMGA	**0.026019**	**0.010017**	0.157624	**0.047144**
ZDT6	NSGA-II	0.136153	0.132502	0.388023	0.349802
	NCGA	1.367150	1.249455	3.120534	1.000000
	AMGA	**0.099351**	**0.097869**	**0.279079**	**0.288113**

Table 1. Median value for all multi-objective optimization performance metrics.

The results for problem ZDT3 are shown graphically in Figure 9. The Pareto frontier for ZDT3 is the second, discontinuous front shown in Figure 5. In this problem, AMGA gives performance metric values that are 40 – 80% better than the next best value. Note, however, that NCGA is designed to work with bit strings, whereas NSGA-II and AMGA are designed to work with real variables. All of the problems considered in this study involve real variables that may have impacted the performance of NCGA. For problems involving discrete variables, NCGA can produce better results. Also, the use of bit strings gives NCGA the capability to produce a more uniform distribution across the Pareto-optimal frontier. Again, the no-free-lunch theorem (Ref. 2) applies.

It should also be noted that the number of Pareto points output by AMGA has been deliberately limited to 20 for the purposes of a more fair comparison. If the number of points output by AMGA is not restricted, to exploit one of the advantages of AMGA, the value of performance parameters would be significantly better.

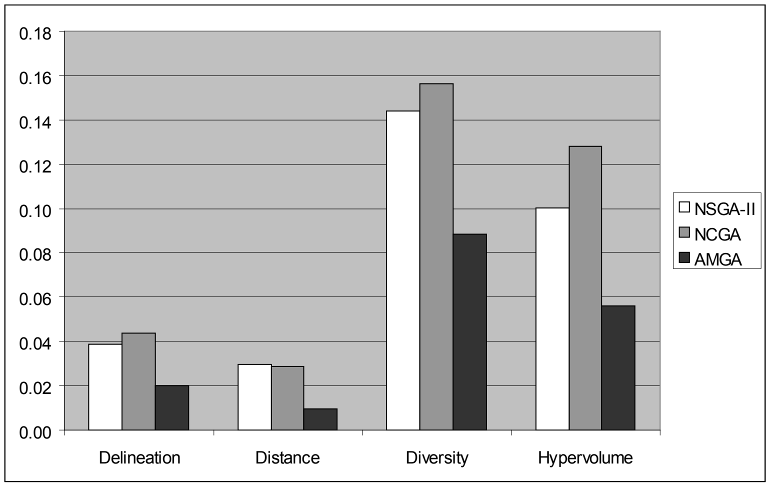


Figure 9. ZDT3 performance results for all three algorithms and all performance metrics.

The Non-deterministic, Stochastic Optimization Problem

Real-world engineered products and processes do not behave in a deterministic manner. Most systems behave *stochastically*—involving chance or probability. Variation is inherent in material characteristics, loading conditions, simulation model accuracy, geometric properties, manufacturing precision, actual product usage, etc. Application of deterministic optimization strategies, without incorporating uncertainty and measuring its effects, leads to designs that cannot be called "optimal," but instead are potentially high-risk solutions that can have a high probability of failing in use. Optimization algorithms tend to push a design towards one or more constraints until the constraints are active. With a design sitting on one or more constraint boundary, even slight uncertainties in the problem formulation or changes in the operating environment could produce failed, unsafe designs, and/or result in substantial performance degradation.

Traditionally, many uncertainties are removed through assumptions, and others are handled through crude safety methods, which often lead to over-designed products and do not offer insight into the effects of individual uncertainties and the actual margin of safety of a design.

More recently, stochastic optimization methods—often called *probabilistic optimization* or *robust optimization*—have been developed to address uncertainty and randomness through statistical modeling and probabilistic analysis (Refs. 10, 34). These probabilistic approaches have been developed to convert deterministic problem formulations into stochastic formulations to model and assess the effects known uncertainties. Until recently, however, the computational expense of stochastic methods, in terms of the number of function evaluations necessary to accurately capture performance variation, has made the application of these methods impractical for all but academic investigations or very critical cases. With the steady increases in computing power, large scale parallel processing capabilities, and availability of probabilistic analysis and optimization tools and systems, however, the combination of these technologies can facilitate effective stochastic analysis and optimization for complex design problems, allowing the identification of designs that qualify as not only feasible, but as consistently feasible in the face of uncertainty.

A stochastic optimization problem can be formally stated as follows:

$$\begin{array}{lll} \text{Minimize} & f_m(\mu_{y_m}(\mathbf{x}),\ \sigma_{y_m}(\mathbf{x})), & m = 1, 2, \dots, M \\ \text{Subject to} & g_j(\mu_{y_j}(\mathbf{x}), \sigma_{y_j}(\mathbf{x})) \le 0, & j = 1, 2, \dots, J \\ & h_k(\mathbf{x}) = 0, & k = 1, 2, \dots, K \\ & x_i^{(L)} + n\sigma_{x_i} \le x_i \le x_i^{(U)} - n\sigma_{x_i}, & i = 1, 2, \dots, N \end{array}$$

The stochastic optimization problem models both nominal, or mean, performance and performance variation through statistics and/or probabilities. For example, the constraints can be modeled as a mean value adjusted by a specified number of standard deviations:

$$\mu_y - n\sigma_y \ge \text{Lower Limit}$$
$$\mu_y + n\sigma_y \le \text{Upper Limit}$$

Or as a probability of violating the specified limit:

$$g(\mathbf{x}) \le 0 \qquad \text{becomes} \qquad P_f = P(g(\mathbf{x}) \le 0) \le P^U$$

The stochastic optimization problem is inherently a multi-objective problem: each performance measure has two objective components corresponding to nominal/mean performance and performance variation. Multi-objective optimization strategies can be used to assess effectively the tradeoffs between the performance measures and between the nominal performance and measured variation of performance. The stochastic optimization problem can also be formulated as a single objective problem, using the weighted sum approach as follows:

$$F = \sum_{m=1}^{M} \left[\frac{w_{1_m}}{s_{1_m}} \left(\mu_{y_m} - T_m \right)^2 + \frac{w_{2_m}}{s_{2_m}} \sigma_{y_m}^2 \right]$$

Where w_{1_m} and w_{2_m} are the weights and s_{1_m} and s_{2_m} are the scale factors for the "mean on target" and "minimize variation" objective components, respectively, for performance response *m*, T_m is the target for performance response *m*, and *M* is the number of performance responses included in the objective. For the case in which the mean performance is to be minimized or maximized, rather than directed toward a target, the objective formulation can be modified as follows, where the first term is positive when the response mean is to be minimized and negative when the response mean is to be maximized:

$$F = \sum_{m=1}^{M} \left[(+/-) \frac{w_{1_m}}{s_{1_m}} \mu_{y_m} + \frac{w_{2_m}}{s_{2_m}} \sigma_{y_m}^2 \right]$$

Calculation of the statistics and probabilities necessary to implement a stochastic optimization strategy requires the identification and characterization of *random variables* and the incorporation of a sampling strategy within the stochastic optimization search. Random variables are inputs to a simulation with known variation. They are described through probability distributions and associated properties. By sampling from these distributions following their prescribed properties, the effects of this input variation on performance can be assessed. Through this sampling, stochastic analysis is used to *measure* design quality (reliability and robustness). Many methods have been developed for stochastic sampling, including *Monte Carlo methods* (Refs. 35, 36), *structural reliability analysis methods* (Refs. 37-39), *sensitivity-based methods, based on Taylor's expansion* (Refs. 40, 41) and *design of experiments* (Ref. 42). Examples of computed quality attributes are: mean, sigma level, defects per million, probability of success, and probability of failure.

Stochastic optimization is then used to *improve* design quality. Any of the authors' multi- and single-objective optimization methods (Ref. 1) mentioned earlier in this paper can be used to optimize the attributes of design quality as measured by the stochastic analysis.

The concept of stochastic, robust optimization is illustrated in Figure 10. If the function in Figure 8 is to be minimized, the solution given by point 1 would be chosen if uncertainty and performance variation are not considered, as with deterministic optimization. Given uncertainty in the design parameter *x*, defined as a variation of $\pm\Delta x$ around the chosen value, the solution at point 1 leads to a large level of variation, Δf_1, of the performance function *f(x)*. To the right of point 1 in the figure, there exists a more "flat" region of *f(x)*, which can be shown to be more robust, or less sensitive to variation in the design parameter *x*. If point 2 is chosen, for the same design parameter variation, $\pm\Delta x$, the variation of the performance function, Δf_2, is significantly less than that at point 1. The sacrifice in choosing point 2 is the increase in the median value of *f(x)*, which is higher at point 2 than at point 1. This is the tradeoff that must be evaluated in searching for a robust solution as opposed to a solution with optimal mean performance. It can be seen in the figure that an even flatter region than that at point 2 exists further to the right of point 2 (direction of increasing *x*). Although the performance variation may be even less in this region, the mean performance may not be acceptable. Both elements of desired mean performance and reduced performance variation must be incorporated in a robust optimization formulation.

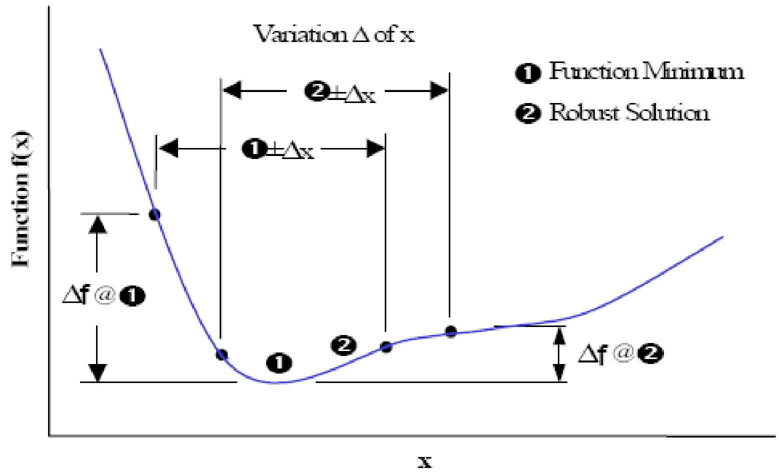
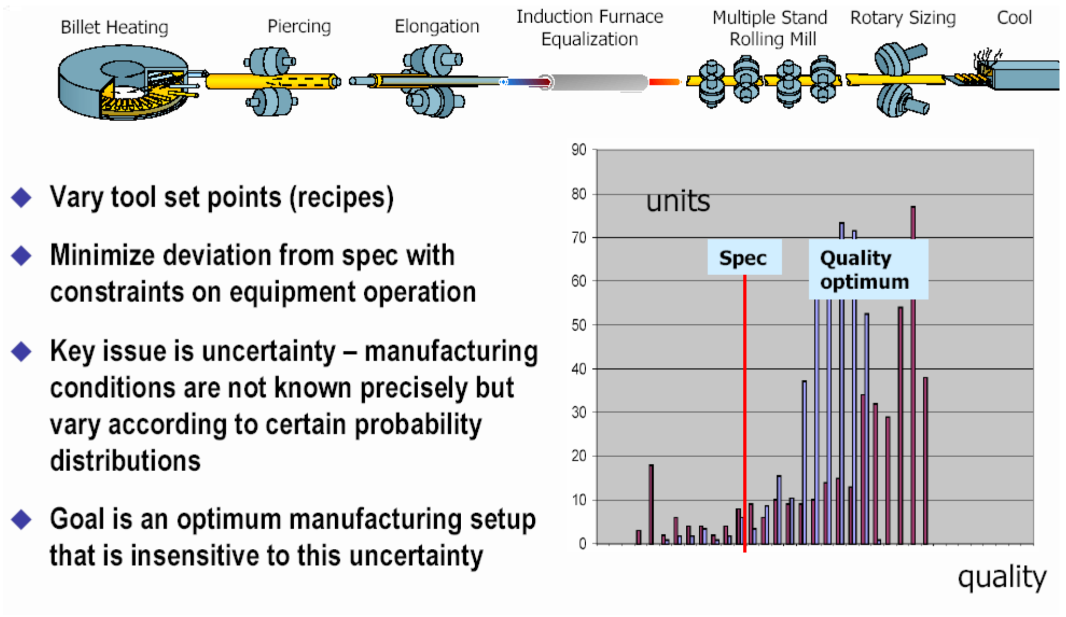


Figure 10. Robust optimization concept

A very common way to express quality of a design is the number of sigmas (standard deviations) a design is away from failing to meet the specifications. Six sigmas correspond to 3 defects per million, a widely quoted (Ref. 44) quality goal in manufacturing processes.

Stochastic Optimization Studies

In Figure 11, the robust optimization of a steel mill is presented. In this case, we are interested in the operation of the mill under varying conditions (times, temperature profiles, cooling air velocities, etc.), so we minimized the deviation from the specification (material characteristics, dimensions, etc.) with constraints on equipment operation. In this particular case, robust optimization reduced the mean specification score by 1%, but the standard deviation of the specification score was reduced by an impressive 95%. This provided significant savings in scrap cost.



- Vary tool set points (recipes)
- Minimize deviation from spec with constraints on equipment operation
- Key issue is uncertainty – manufacturing conditions are not known precisely but vary according to certain probability distributions
- Goal is an optimum manufacturing setup that is insensitive to this uncertainty

Figure 11. Tool set point optimization of a seamless steel tube mill (Ref. 4).
Note: Results are for illustration only.

One quality engineering design application currently of high visibility in the automotive industry is vehicle structural design for crashworthiness. These design problems are not only particularly complex in terms of understanding the problem and defining design requirements and design alternatives, but also involve a very high degree of uncertainty and variability: *velocity of impact, mass of vehicle, angle of impact,* and *mass/stiffness of barrier* are just few of many uncertain parameters. A balance must be struck in designing for crashworthiness between designing the vehicle structure to absorb/manage the crash energy (through structure deformation) and maintaining passenger compartment integrity, all in the face of uncertainty and variability in materials, structural configuration, and crash scenario.

One specific crash scenario is investigated in Ref. 10—side impact—using the stochastic analysis and optimization tools available in Isight. A typical vehicle side impact model is shown in Figure 12. Including a finite element dummy model, the total number of elements in this model is about 90,000, and the total number of nodes is around 96,000. The initial lateral velocity of the side deformable barrier is 31 MPH. The CPU time for a RADIOSS simulation of the model is about 20 hours on a SGI Origin 2000.

For side impact protection, the vehicle design should meet the requirements for the National Highway Traffic Safety Administration (NHTSA) side impact procedure specified in the Federal Motor Vehicle Safety Standards (FMVSS) or the European Enhanced Vehicle-Safety Committee (EEVC) side impact procedure. In this study, the EEVC side impact test configuration was used. The dummy performance is the main concern in side impact, which includes head injury criterion (HIC), abdomen load, pubic symphysis force (pelvic load), V*C's (viscous criterion), and rib deflections (upper, middle, and lower). These dummy responses must at least meet EEVC requirements. Other concerns in side impact design are the velocity of the B-Pillar at middle point, and the velocity of the front door at B-Pillar.

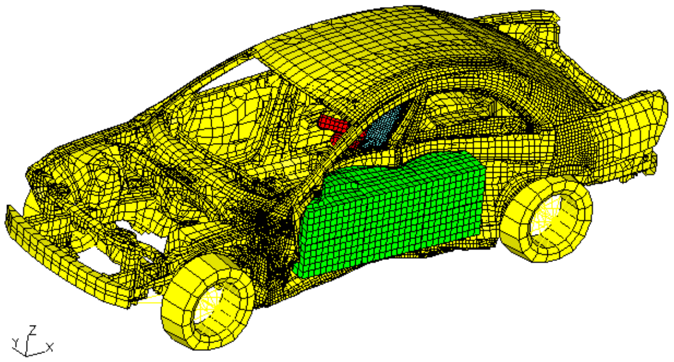


Figure 12. Automotive crashworthiness robust optimization – side impact model (Ref. 10).

For side impact, increase of gauge design variables tends to produce better dummy performance. However, it also increases vehicle weight, which is undesirable. Therefore, a balance must be sought between weight reduction and safety concerns. The objective is to reduce the weight with imposed constraints on the dummy safety. Here, eleven design parameters are used for side impact optimization. Nine design variables include two materials of critical parts and seven thickness parameters. The material design variables are discrete, either mild steel (MS) or high-strength steel (HSS). All thickness and material design variables are also random variables, normally distributed with standard deviation of 3% of the mean for the thickness variables, and 0.6% for the material properties. The final two parameters, barrier height and hitting position, are pure random variables continuously varying from -30mm to 30mm according to the physical test.

A deterministic optimization was initially applied for this problem using the NLPQL sequential quadratic programming (SQP) algorithm. Starting from an infeasible baseline design, this optimization solution results in feasible design with a weight reduction of nearly 20%, but also results in three active constraints associated with rib deflection, pubic symphysis force, and door velocity. When a stochastic analysis is performed at this design solution, using Monte Carlo descriptive sampling with 2,000 points, reliability values of 40% are obtained (60% probability of failure). After applying stochastic optimization, the reliability is increased, but at the expense of the vehicle weight as shown in Figure 13. For this problem, when the number of standard deviations of performance maintained within the required limits (original optimization constraints) reaches around 3σ, the weight is nearly equal to that of the baseline design; no weight savings is achieved, but the quality level is increased. As the quality level is increased further, the weight is increased over the baseline.

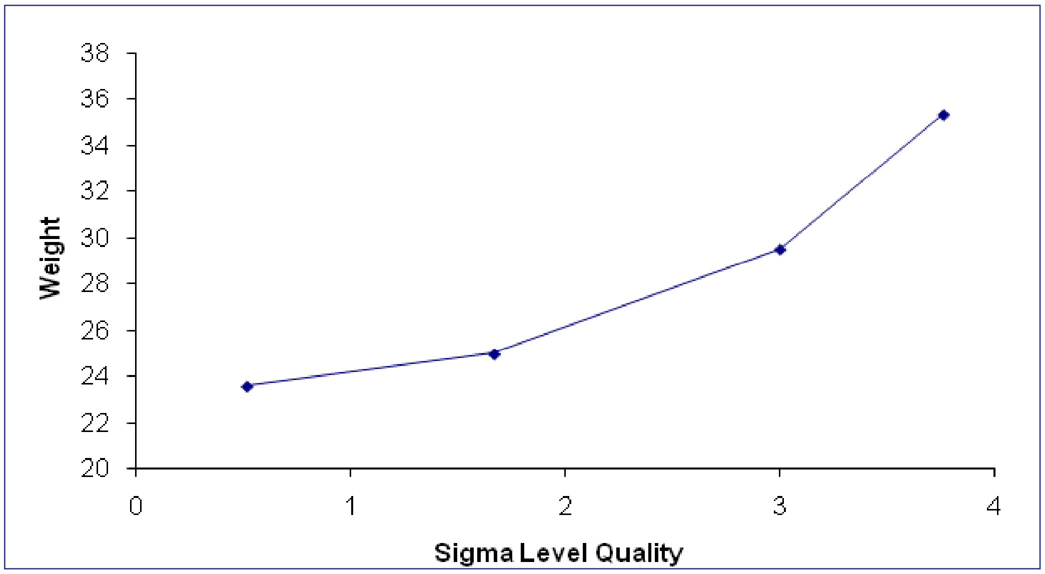


Figure 13. Performance versus Quality Tradeoff –Side Impact Problem (Ref. 10)

Closing

Optimization is a useful and effective simulation-based design tool for identifying one or more designs that best achieves a set of requirements or for improving an existing design. Optimization has been used to solve a wide range of industrial problems, and is being used more and more. One conclusion of these industrial applications of optimization is that no one optimization algorithm—or even class of optimization algorithms—is appropriate or even capable for solving a wide range of problems well. Again, there is no free lunch. Many optimization strategies have been developed to address different types of problems. These optimization strategies can be combined and applied to create new strategies, to allow both global and local search, continuous and discontinuous spaces, continuous and discrete variables, smooth and noisy topologies, etc., to become more robust towards solving a wider range of problems.

There is also always a tradeoff between efficiency and accuracy. Formulating and solving a single-objective problem is usually the most efficient approach, but may not give the best tradeoff solution or allow the problem to be explored sufficiently. Extending a problem to multiple objectives and truly evaluating the tradeoffs between the objectives requires increased computational effort. Incorporating uncertainty and searching for designs of higher quality, requiring additional sampling during a search strategy, adds significantly more computational expense.

For complex simulation models requiring minutes or hours per analysis, optimization—and, especially, stochastic optimization—can quickly become impractical. However, additional simulation-based design *enabling technologies* can be combined to support the implementation of optimization to even highly complex problems. Two such current technologies are parallel processing and approximation methods. Many analyses during optimization and sampling for stochastic analysis are independent analyses known in advance: *gradient runs for gradient-based methods, a population in a genetic algorithm, Monte Carlo or DOE samples, etc.* These analyses can be executed in parallel on multiprocessor machines and/or a network of machines. For the most computationally expensive analyses, requiring days for a single analysis, a set of sampled design points can be executed in parallel and used to construct surrogate models, or approximation models, to replace the high-fidelity code during optimization. Polynomial response surfaces and, more recently, radial basis functions or Kriging surrogate models have been employed for this purpose.

References

1. Van der Velden, “CAD to CAE Process Automation Through iSIGHT-FD”, Engineous Software., GT 2007-27555 Proceedings of the ASME Turbo Expo Montreal, 2007.

2. Wolpert, D.H., et al., 1996. No free lunch theorems for optimization. Evolutionary Computation, IEEE Transactions on Evolutionary Computation.

3. J. Runyan, R.A. Gerhardt, and R. Ruh, “Electrical Properties of BN Matrix Composites – Part I: Analysis of the McLachlan Equation in Modeling the Conductivity of BN-B4C and BN-SiC Composites”, J.Am.Ceram.Soc 84[7], 1490-1496(2001).

4. Robert V. Kolarik and Isaac Chan, US Department of Energy Technology Success Story: “Enhanced Spheroidized Annealing Cycle For Tube and Pipe Manufacturing,” Timken Company, DOE.

5. Il Yong Kim and Olivier de Weck, “Adaptive Weighted Sum Method of Multiobjective Optimization.” 10th AIAA/ISSMO MDAO Conference, AIAA 2004-4322, 2004.

6. Sandgren, E., “The Utility of Nonlinear Programming Algorithms,” Ph.D. Thesis, Purdue University, December 1977.

7. Schittkowski, K., “NLPQL: A Fortran subroutine for solving constrained nonlinear programming problems,” Annals of Operations Research, Vol.5, 485-500, 1985.

8. Nelder, J.A., and Mead, R., “Downhill simplex method in multidimensions,” Computer Journal, 7, pp. 308-313, 1965.

9. Wujek, B., Koch, P. N., McMillan, M., and Chiang, W-S., 2002, “A Distributed, Component-Based Integration Environment for Multidisciplinary Optimal and Quality Design,” 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA, AIAA-2002-5476.

10. Koch, P. N., Yang, R.-J., and Gu, L., 2004, “Design for Six Sigma through Robust Optimization,” Journal of Structural and Multidisciplinary Optimization, Vol. 26, No. 3-4, pp. 235-248.

11. K. Schittkowski, NLPQL: A Fortran subroutine for solving constrained nonlinear programming problems, Annals of Operations Research, Vol. 5, 485-500, 1985.

12. Vanderplaats, G.N., “An Efficient Feasible Directions Algorithm for Design Synthesis,” AIAA J. Vol. 22, No. 11, Oct. 1984, pp. 1633-1640.

13. Lasdon, L.S., Waren, A.D., Jain, A., and Ratner, M., “Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming,” ACM Transactions on Mathematical Software, Vol. 4, No. 1, March 1978, pp. 34-50.

14. Nelder, J.A., and Mead, R., “Downhill simplex method in multidimensions.”, Computer Journal, 7, pp. 308-313, 1965.

15. R. Hooke and T. A. Jeeves, “Direct Search Solution of Numerical and Statistical Problems,” Journal of the ACM, Vol. 8, April 1961, pp. 212-229.

16. Sensor, Y., “Pareto optimality in multi-objective problems,” Applied Mathematics and Optimization, Vol. 4, No. 1, pp. 41–59, March 1977.

17. Dawkins, R., “The Selfish Gene,” Oxford University Press, New York, 1976.

18 Eldredge, N., “Macro-Evolutionary Dynamics: Species, Niches and Adaptive Peaks,” McGraw-Hill, New York, 1989.

19. Goldberg, D. E. “Genetic Algorithms for Search, Optimization, and Machine Learning.” Reading, MA: Addison-Wesley, 1989.

20. Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T., “A fast and elitist multi-objective genetic algorithm: NSGA-II,” IEEE Transactions on Evolutionary Computation, Vol. 6 No. 2, pp. 182-197, 2002.

21. Watanabe, S., Hiroyasu, T., and Miki, M., “NCGA: Neighborhood cultivation genetic algorithm for multi-objective optimization problems,” In Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2002, pp. 458-465, 2002.

22. Tiwari, S., Koch, P., Fadel, G., and Deb, K., “AMGA: An Archive-based Micro Genetic Algorithm for Fast and Reliable Convergence,” Genetic and Evolutionary Computation Conference GECCO 2008, July 12–16, 2008, Atlanta, Georgia, USA.

23. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Da Fonseca, V. G., “Performance assessment of multiobjective optimizers: An analysis and review,” IEEE Transactions on Evolutionary Computation, Vol. 7, No. 2, pp. 117–132, April 2003.

24. Deb, K., “Multi-objective optimization using evolutionary algorithms,” Chichester, UK: Wiley, 2001.

25. Buckingham, E. “The principle of similitude,” Nature 96 (1915).

26. A. Van der Velden, D. Kokan, “The Synaps Pointer Optimization Engine” (with Dr. Kokan), Proceedings of DETC/CIE, ASME Computers and Information in Engineering Conference, Montreal, Canada, 2002.

27. Glover, F. and de Werra, D., *Tabu Search*, Baltzer Science Publishers, 1997.

28. Schwefel, H.P. “Evolutionsstrategie und Numerische Optimierung,” PhD thesis, Verfarenstechnik TU Berlin, 1975.

29. Wooffindin, C., “Corus Smart Weld Optimiser,” Engineous North America user meeting, Orlando, 2007.

30. Heung-Kyu Kim, Seok Kwan Hong, Jong-Kil Lee, Byung-Hee Jeon, Jeong Jin Kang, and Young-moo Heo, “Finite Element Analysis and Optimization for the Multi-stage Deep Drawing of Molybdenum Sheet.” CP778 Volume A, Numisheet, American Institute of Physics, 0-7354-0265-5/05, 2005.

31. Ingber, L., “Simulated Annealing: Practice versus Theory,” Mathematical Computer Modeling, 18, 1993, pp.29-57.

32. Hisao Ishibuchi and Tadashi Yoshida, “Hybrid Evolutionary Multi-Objective Optimization Algorithms,” in A. Abraham, J. Ruiz-del-Solar and M. Köppen (eds), *Soft Computing Systems: Design, Management and Applications (Frontiers in Artificial Intelligence and Applications, Volume 87)*, IOS Press, ISBN 1-58603-297-6,pp. 163--172, 2002.

33. Miki, M.; Hiroyasu, T.; Kaneko, M.; Hatanaka, K., 1999, “**A Parallel Genetic Algorithm with Distributed Environment Scheme**,” IEEE International Conference on Systems, Man, and Cybernetics, Vol. 1, pp. 695-700.

34. Belegundu, A. D., 1988, “Probabilistic Optimal Design Using Second Moment Criteria,” Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 110, No. 3, pp. 324-329.

35. Hammersley, J.M., and Handscomb, D.C., Monte Carlo Methods, Chapman and Hall, London. 1964.

36. Saliby, E., “Descriptive Sampling: A Better Approach to Monte Carlo Simulation,” Journal of the Operational Research Society, Vol. 41, No. 12, pp. 1133-1142, 1990.

37. Hasofer, A.M., and Lind, N.C., “Exact and Invariant Second Moment Code Format,” Journal of Engineering Mechanics, ASCE, Vol. 100, No. EM1, February, pp. 111-121, 1974.

38. Rackwitz, R., and Fiessler, B., “Structural Stability Under Combined Random Load Sequences,” Computers and Structures, Vol. 9, pp. 489-494, 1978.

39. Melchers, R.E., Structural Reliability: Analysis and Prediction, Second Edition, Ellis Horwood Series in Civil Engineering, John Wiley & Sons, New York, 1999.

40. Chen, W., Allen, J. K., Tsui, K-L., and Mistree, F., “A Procedure for Robust Design: Minimizing Variations Caused by Noise Factors and Control Factors,” ASME Journal of Mechanical Design, Vol. 118, No. 4, pp. 478-485, 1996.

41. Hsieh, C-C, and Oh, K. P., “MARS: a computer-based method for achieving robust systems,” FISITA Conference, The Integration of Design and Manufacture, Vol. 1, pp. 115-120, 1992.

42. Montgomery, D. C., Design and Analysis of Experiments, New York: John Wiley & Sons, 1996.

43. Tseng, C.H., Liao, W.C., and Yang, T.C., “Most 1.1 User’s Manual,” Technical Report No AODL-96-01, Dept of ME, National Chiao Tung University, Taiwan ROC, 1996.

44. Harry, M., “Six Sigma,” Currency book, 2000.