Review of Optimization Techniques

G. Venter

Department of Mechanical and Mechatronic Engineering
Stellenbosch University
Private Bag X1 Matieland 7602
South Africa

ABSTRACT

A basic overview of optimization techniques is provided. The standard form of the general non-linear, constrained optimization problem is presented, and various techniques for solving the resulting optimization problem are discussed. The techniques are classified as either local (typically gradient-based) or global (typically non-gradient based or evolutionary) algorithms. A great many optimization techniques exist and it is not possible to provide a complete review in the limited space available here. Instead, an effort is made to concentrate on techniques that are commonly used in engineering optimization applications. The review is kept general in nature, without considering special cases like linear programming, convex problems, multi-objective optimization, multi-disciplinary optimization, etc. The advantages and disadvantages of the different techniques are highlighted, and suggestions are made to aid the designer in selecting an appropriate technique for a specific problem at hand. Where possible, a short overview of a representative method is presented to aid the discussion of that particular class of algorithms.

KEY WORDS: Local algorithms; Gradient-based algorithms; Global algorithms; Non-gradient based algorithms; Evolutionary algorithms; Global optimization

1. INTRODUCTION

Before starting a review of the available optimization techniques, it is useful to present the standard form of the general optimization problem that will be solved by these techniques. The standard form for a single-objective, non-linear, constrained optimization problem is provided in Eq. 1 below.

$$\begin{array}{lll} \text{Minimize:} & f(\boldsymbol{x}) & \\ \text{Subject to:} & g_j(\boldsymbol{x}) \leq 0 & j = 1, m \\ & h_k(\boldsymbol{x}) = 0 & k = 1, p \\ & x_{iL} \leq x_i \leq x_{iU} & i = 1, n \end{array} \tag{1}$$

In Eq. 1, $f(\boldsymbol{x})$ represents the objective (or goal) function, $g_j(\boldsymbol{x})$ an inequality constraint and $h_k(\boldsymbol{x})$ an equality constraint function. The $\boldsymbol{x}$ vector represents the n design variables that are modified to obtain the optimum. The searchable design space is defined by the upper and lower bounds, x_{iL} and x_{iU}, of the design variables, referred to as the side constraints. In the general case, the objective and constraint functions can be linear or non-linear and can be explicit or implicit functions. Implicit functions

Encyclopedia of Aerospace Engineering. Edited by Richard Blockley and Wei Shyy.

commonly appear when, for example, a numerical simulation (e.g., a finite element simulation) is used to evaluate a response function (e.g., a stress value). Also, the design variables need not be continuous. Optimization problems having some or all of the design variables restricted to integer or discrete values are not uncommon and are referred to as integer or discrete optimization problems. Generally, the local algorithms have difficulty in solving optimization problems with integer and/or discrete variables, while several global algorithms are well adapted to this class of problems.

Most optimization algorithms consider the side constraints separately from the equality and inequality constraints. The side constraints can be handled efficiently by direct implemention in the algorithm. A good algorithm will never violate any of the side constraints. An unconstrained optimization problem thus has no equality or inequality constraints, but may have side constraints. A constrained optimization problem has one or more equality and/or inequality constraints, with or without side constraints. For a constrained optimization problem, an equality constraint can either be violated or satisfied, while an inequality constraint can be violated, active or satisfied. An active inequality constraint is one for which $g_j(\boldsymbol{x}) = 0$.

Optimization techniques, or algorithms, are used to find the solution to the problem specified in Eq. 1. The procedure consists of finding the combination of design variable values that results in the best objective function value, while satisfying all the equality, inequality and side constraints. Note that for many problems, more than one optimum (referred to as local or relative optima) may exist. There are many options for classifying the available optimization techniques. A short overview of the available algorithms, using a broad classification as either local or global algorithms, is presented. Where appropriate, a representative algorithm is discussed briefly in more detail to aid the discussion of that particular class of algorithms.

2. LOCAL OPTIMIZATION ALGORITHMS

Most local optimization algorithms are gradient-based. As indicated by the name, gradient-based optimization techniques make use of gradient information to find the optimum solution of Eq. 1. Gradient-based algorithms are widely used for solving a variety of optimization problems in engineering. These techniques are popular because they are efficient (in terms of the number of function evaluations required to find the optimum), they can solve problems with large numbers of design variables, and they typically require little problem-specific parameter tuning. These algorithms, however, also have several drawbacks which include that they can only locate a local optimum, they have difficulty solving discrete optimization problems, they are complex algorithms that are difficult to implement efficiently, and they may be susceptible to numerical noise. In general, the algorithms are well established and are thoroughly covered in a number of textbooks, a small selection of which include the work by Haftka and Gürdal (1993), Arora (2004), Snyman (2005) and Vanderplaats (2007). The interested reader is referred to these textbooks for additional references.

2.1. Background

Gradient-based algorithms typically make use of a two-step process to reach the optimum. This two-step process can be explained using the following picture (Vanderplaats, 2007) of a blindfolded boy on a hill. The boy has to reach the highest point on the hill (the objective function), while staying inside the fences (the constraints). The design variables are the x and y coordinates of the boy. Note that the boy can actually start outside the fences, illustrating an important use of optimization techniques,

Encyclopedia of Aerospace Engineering. Edited by Richard Blockley and Wei Shyy.

which is to find a feasible design.

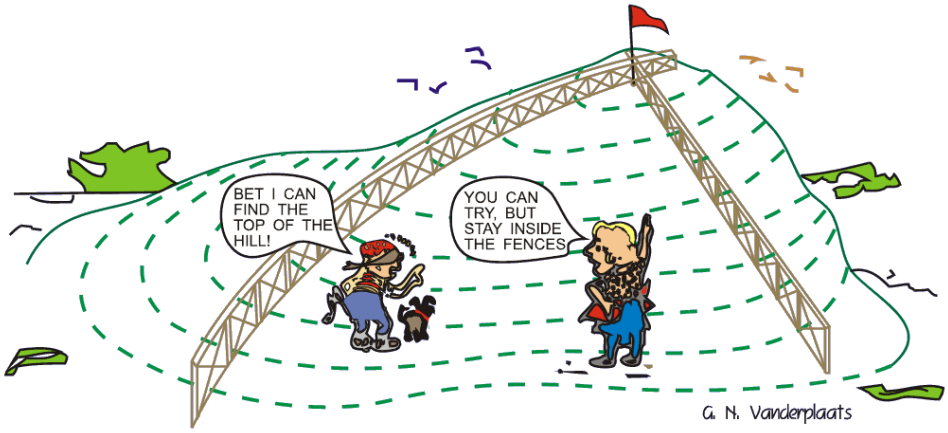

Figure 1. Gradient-based optimization (published with permission)

In an analogy to gradient-based optimization, the blindfolded boy can reach the top of the hill by taking a single step in the x direction and another step in the y direction. Based on the information gained from these two steps, he can estimate a direction that would take him uphill. The boy can then start walking in this direction until no more progress is made, which may include reaching a fence. At this point the boy can again take two small steps to determine a new direction that will take him uphill, while staying inside the fences, and continue the process until he reaches the top of the hill.

This two-step iterative process of finding the optimum can be summarized mathematically as:

$$\boldsymbol{x}^q = \boldsymbol{x}^{q-1} + \alpha^\star \boldsymbol{S}^q \tag{2}$$

where the first step is to use gradient information for finding a search direction $\boldsymbol{S}$ in which to move. The second step is to move in this direction until no more progress can be made. The second step is known as the one-dimensional or line search, and provides the optimum step size, $\alpha^\star$. Note that there are also gradient-based algorithms that do not rely on a one-dimensional search.

For most optimization problems, the gradient information is not readily available and is obtained using finite difference gradient calculations. Finite difference gradients provide a flexible means of estimating the gradient information. However, when used, they typically dominate the total computing time required to complete an optimization study. When the designer has access to the source code, automatic differentiation (e.g., Griewank and Walther (2008)) can be used to obtain the required gradient information. Automatic differentiation has the benefit of providing gradient information that is accurate to working precision. In contrast, finite difference calculations provide only an approximation to the gradient, with the accuracy depending on the selected step size. Finally, some numerical simulations can provide analytic or semi-analytic gradient information directly. For example, analytic and semi-analytic gradient calculations can be implemented for linear finite element codes to provide computationally inexpensive gradient information for structural optimization applications. Similarly, semi-analytic gradient calculations can be implemented efficiently in computational fluid dynamics codes. The general rule of thumb is to use the gradient provided by the analysis program, if it is

Encyclopedia of Aerospace Engineering. Edited by Richard Blockley and Wei Shyy.

available. The gradient information obtained from the analysis code is typically obtained at a significant reduction in computation cost and is often more accurate than performing finite difference gradient calculations.

Depending on the scenario, different search directions are required in Eq. 2. For unconstrained optimization problems, or constrained optimization problems with no active or violated constraints, a search direction that will improve the objective function is desired. Any search direction that will improve the objective function is referred to as a usable direction. For constrained optimization problems with one or more violated constraints, a search direction that will overcome the constraint violation is desired. For constrained optimization problems with one or more active constraints and no violated constraints, a search direction that is both usable and feasible is required. A feasible direction is any direction that will not violate the constraint bounds.

The different gradient-based algorithms that exist, differ mostly in the logic used to determine the search direction. For the one-dimensional search, there are many algorithms that will find the best step size, and generally any of these techniques can be combined with a particular gradient-based algorithm to perform the required one-dimensional search. Some of the popular one-dimensional search algorithms include the Golden Section search, the Fibonacci search, and many variations of polynomial approximations.

When gradient information is available, the Karush-Kuhn-Tucker (KKT) conditions can be used to determine if a constrained local optimum has been found. The Karush-Kuhn-Tucker conditions provide the necessary conditions for a local optimum and can be summarized as:

1. The optimum design point $\boldsymbol{x}^\star$ must be feasible.
2. At the optimum design point, the gradient of the Lagranian must vanish

$$\nabla f(\boldsymbol{x}^\star) + \sum_{j=1}^{m} \lambda_j \nabla g_j(\boldsymbol{x}^\star) + \sum_{k=1}^{p} \lambda_{m+k} \nabla h_k(\boldsymbol{x}^\star) = \mathbf{0} \tag{3}$$

   where the Lagrange multipliers $\lambda_j \geq 0$ and λ_{m+k} are unrestricted in sign.
3. For each inequality constraint $\lambda_j g_j(\boldsymbol{X}) = 0$, where $j = 1, m$.

Note that, for unconstrained problems, the Karush-Kuhn-Tucker conditions only require the gradient of the objective function to vanish at the optimum design point. The Karush-Kuhn-Tucker conditions are useful for identifying a local optimum, but cannot indicate whether a global optimum has been found.

2.2. Newton's Method

One of the classical gradient-based optimization algorithms is Newton's algorithm. Newton's algorithm is an unconstrained algorithm that is derived from a second-order Taylor series expansion of the objective function about an initial design point $\boldsymbol{x}^0$

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^0) + \nabla f(\boldsymbol{x}^0)^T (\boldsymbol{x} - \boldsymbol{x}^0) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^0)^T H(\boldsymbol{x}^0) (\boldsymbol{x} - \boldsymbol{x}^0) \tag{4}$$

where $H(\boldsymbol{x}^0)$ is the Hessian matrix that contains the second-order gradient information of the objective function. Differentiating Eq. 4 with respect to $\boldsymbol{x}$ and setting the result equal to zero according to the Karush-Kuhn-Tucker conditions, results in the following update formula for the current design point:

$$\boldsymbol{x} = \boldsymbol{x}^0 - H(\boldsymbol{x}^0)^{-1} \nabla f(\boldsymbol{x}^\mathbf{0}) \tag{5}$$

Encyclopedia of Aerospace Engineering. Edited by Richard Blockley and Wei Shyy.

Note that, in this classic form, Newton's method makes use of a fixed step size of 1 (no one-dimensional search is required) and the search direction is provided by $-H(\boldsymbol{x}^0)^{-1}\,\nabla f(\boldsymbol{x}^{\mathbf{0}})$. Newton's method has a quadratic rate of convergence, requiring only a single step (with step size equal to 1) to obtain the optimum for any positive definite quadratic function. In practice the method is modified to include a one-dimensional search that improve both the efficiency and robustness of the method.

Although the method has a quadratic convergence rate which is highly desirable, the computational cost associated with obtaining the second-order gradient information in the Hessian matrix makes the method impractical in most cases. As a result, most gradient-based methods makes use of first order gradient information only.

2.3. Unconstrained Optimization

For unconstrained problems, two very popular methods are the Fletcher-Reeves and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) methods. The Fletcher-Reeves method makes use of conjugate search directions (the method is referred to as a conjugate gradient method) to reach the optimum. The conjugate search directions are created using information gained from the previous design iteration. This method works very well and will theoretically minimize a quadratic function in n or fewer iterations. It has the added advantage of having a small computer memory requirement.

The BFGS method is part of family of methods, known as the variable metric methods, that makes use of information gained from the previous n iterations to find a new search direction. Numerical experiments indicate that the BFGS method is the most efficient variable metric method. The BFGS method creates an approximation to the inverse of the Hessian matrix, $H(\boldsymbol{X}^0)^{-1}$, in Eq. 5. This approximation is updated after each design iteration with the first-order gradient information from that iteration. Since only an approximation to the inverse of the Hessian matrix is used, the method is known as a quasi-Newton method. Mathematically, the BFGS method is considered superior to the Fletcher-Reeves method, but has the drawback of requiring substantially more computer memory to store the approximate inverse of the Hessian matrix.

2.4. Constrained Optimization

For constrained optimization problems, two approaches are considered here. The first is known as the Sequential Unconstrained Minimization Techniques (SUMT) approach (a comprehensive overview of the approach is provided in Fiacco and McCormick (1968)), and the second as direct (or constrained) methods. The SUMT approach solves the general constrained optimization problem by first converting it to an equivalent unconstrained problem. This equivalent unconstrained problem is then solved using any one of the unconstrained algorithms already discussed. The SUMT approach obtains an equivalent unconstrained problem by penalizing the original objective function for any constraint violations. The penalized objective function is obtained from

$$f_p(\boldsymbol{x}, r_p) = f(\boldsymbol{x}) + r_p p(\boldsymbol{x}) \tag{6}$$

where $f_p(\boldsymbol{x})$ refers to the penalized objective function, r_p to the penalty parameter and $p(\boldsymbol{x})$ to the penalty function. The classical approach is to keep r_p constant for a complete unconstrained minimization cycle of f_p. Once the unconstrained optimum solution is found, the value of the penalty parameter r_p is increased and the unconstrained optimization cycle is repeated. The result is sequential progress towards the constrained optimum. The process is terminated when the objective function value converges between successive unconstrained optimization cycles. There are many different penalty functions (see the already mentioned text books for more detail). One popular function is the exterior

quadratic penalty function, defined as:

$$p(\boldsymbol{x}) = \sum_{j=1}^{m} (max[0, g_j(\boldsymbol{x})])^2 + \sum_{k=1}^{l} h_k(\boldsymbol{x})^2 \tag{7}$$

The exterior penalty function approaches the constrained optimum from the infeasible region of the design space. There are also interior and extended interior penalty functions available that approach the constrained optimum from the feasible region of the design space. The biggest drawback of these methods is related to the value of the penalty parameter. The penalty parameter has a significant influence on the performance of the algorithm, but is problem dependent and often must be set extremely large to obtain satisfactory results, which may lead to numerical ill-conditioning. The Augmented Lagrange multiplier method is one method that overcomes this limitation by creating a penalty function based on the Lagrangian and defines penalty parameters based on estimates of the Lagrange multipliers. The Augmented Lagrange multiplier method has the advantage of achieving exact constraint satisfaction for a finite value of r_p and is less sensitive to the selected r_p value.

The SUMT methods have become less popular as the direct (or constrained) methods became more mature and efficient. Today, the gradient-based technique of choice for constrained optimization problems is the direct methods, which will be discussed next. There is one interesting area where the SUMT methods are making a comeback, and that is for very large-scale (in terms of numbers of design variables) optimization problems. An example is the commercially available BigDOT algorithm from Vanderplaats Research and Development, Inc. (Vanderplaats, 2004) which can be used to solve constrained optimization problems with hundreds of thousands of design variables.

The direct methods directly solve the non-linear constrained optimization problem of Eq. 1. Many different constrained optimization algorithms are available. In engineering, three algorithms that are commonly encountered are the Sequential Linear Programming (SLP) algorithm, the Modified Method of Feasible Directions (MMFD) algorithm, and the Sequential Quadratic Programming (SQP) algorithm.

The SLP algorithm simplifies the general non-linear constrained optimization problem to an equivalent linear problem by creating linear approximations to the objective and constraint functions about the current design point. Using an appropriate algorithm, the optimum solution for these linear approximations is found and the resulting design point is evaluated. A new set of linear approximations is constructed about this newly evaluated design point, and the process is repeated until convergence. The method is very sensitive to move limits and may not always find a feasible solution. As a result, it is generally considered inferior to the MMFD and SQP algorithms.

The MMFD algorithm is based on a modification of the original method of the feasible directions algorithm. The modification provides a search direction that follows the active constraint bounds (rather than moving inside the feasible region) towards the optimum. The MMFD algorithm is extremely robust, has the property of quickly finding the feasible design space, and is widely used in structural optimization.

The SQP algorithm is probably the most popular direct algorithm for engineering optimization applications. This algorithm finds a search direction by solving an approximate problem based on a quadratic approximation of the objective function and linear approximations of the constraint functions. Once the new search direction is determined, a penalty function is typically used to determine the step size in the obtained search direction. The new design point obtained from combining the search direction and the optimum step size (using Eq. 2) is then evaluated as the new design point, and the process is repeated until convergence. Although a penalty function is typically used to perform the one-

Encyclopedia of Aerospace Engineering. Edited by Richard Blockley and Wei Shyy.

dimensional search, a new algorithm by Fletcher and Leyffer (2002) provides an interesting alternative, where a bi-objective formulation is used to replace the one-dimensional search with a filter.

Other direct methods that may be encountered, but that will not be discussed here, include gradient projection and the generalized reduced gradient methods. There are many other gradient-based algorithms, but too many to mention here. An interesting collection of relatively new gradient-based algorithms by Snyman and his co-workers is contained in Snyman (2005). These algorithms were developed specifically to deal with real-world problems, where expensive function evaluations, numerical noise, discontinuities, local minima, areas in the design space where the functions are not defined and large numbers of design variables are encountered. The collection consists of algorithms that require only first-order gradient information and that do not rely on a one-dimensional search. Algorithms for constrained and unconstrained optimization, approximation methods and methods for global unconstrained optimization are presented.

2.5. Non-gradient based Methods

The discussion of local search techniques would not be complete without also mentioning the existence of non-gradient based local search algorithms. Popular examples are Powell's method and the Nelder-Mead simplex algorithm. Both algorithms are capable of solving non-linear, unconstrained optimization problems. Powell's method is based on the concept of conjugate directions, while the Nelder-Mead algorithm makes use of a simplex and a set of simple rules that reflects the worst vertex through the centroid of the simplex.

3. GLOBAL OPTIMIZATION ALGORITHMS

The issue of local versus global optimization has already been discussed briefly in previous sections. Many problems have multiple optima, with a simple one variable function shown in Fig. 2 below.

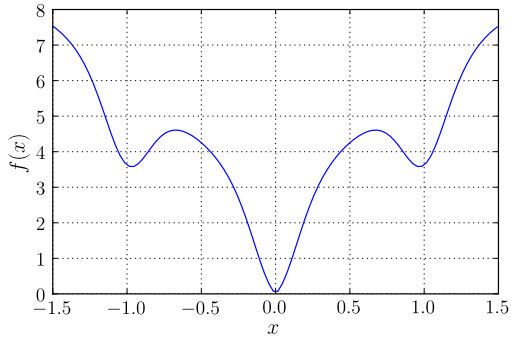

Figure 2. One dimensional multi-modal function

In Fig. 2, the minima at $x \approx \pm 1$ represent local (or relative) minima, while the minimum at $x = 0$ represents the global (or absolute) minimum. All three these points satisfy the Karush-Kuhn-Tucker conditions. The local algorithms discussed so far will converge on any of these three points, depending on which one is encountered first. When using local optimization algorithms, a simple approach for

dealing with multiple local minima in the design space is to use a multistart approach (e.g., Haim *et al.* (1999) and Cox *et al.* (2001)). In a multi-start approach, multiple local searches are performed, each starting from a different starting point. Often, a design of experiments (DOE) approach is used to generate the set of starting points.

Global optimization algorithms provide a much better chance of finding the global or near global optimum than the local algorithms discussed so far. It is important to note that no algorithm can guarantee convergence on a global optimum in the general sense, and it may be more accurate to refer to these algorithms as having global properties. Global optimization algorithms may be classified as either evolutionary algorithms or deterministic algorithms.

3.1. Evolutionary Algorithms

Evolutionary optimization algorithms have become very popular in the last decade or two. Unlike the local techniques, where a single design point is updated (typically using gradient information) from one iteration to the next, these algorithms do not require any gradient information and typically make use of a set of design points (generally referred to as a population) to find the optimum design. These methods are typically inspired by some phenomena from nature and have the advantage of being extremely robust, having an increased chance of finding a global or near global optimum, being easy to implement and being well suited for discrete optimization problems. The big drawbacks associated with these algorithms are high computational cost, poor constraint-handling abilities, problem-specific parameter tuning and limited problem size.

Currently, two of the most popular evolutionary algorithms are the more established Genetic Algorithm (GA) (Holland, 1975), which was inspired by Darwin's principle of survival of the fittest, and Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995), which is based on a simplified social model. Other algorithms that fall into this category include evolutionary programming (Fogel, Owens and Walsh, 1966), genetic programming (Koza, 1992), differential evolution (Storn and Price, 1997), simulated annealing (Kirkpatrick, Gelatt and Vecchi, 1983), tabu search (Glover, 1977), ant colony optimization (Dorigo, Maniezzo and Colorni, 1996), harmony search (Geem, Kim and Loganathan, 2001), etc.

The basic steps of a GA are illustrated in Fig. 3 below. The first step is to create a random initial population. Typically, the population size does not change during the optimization study. The population is then ranked, based on the fitness (objective function) of each individual, and parents are randomly selected for reproduction. The parent designs are selected in such a way that the higher ranked (fitter) individuals have a higher probability of being selected. The next generation is then filled with children designs, created by randomly mixing the selected parent designs in a process known as cross-over. The new generation is again ranked, and the process is repeated until convergence. Figure 3 illustrates the process for an initial population consisting of only four individuals. By looking at the children designs, it should be clear which parents were selected to create each child design. In Fig. 3 the children are created using one point cross-over, where both parent designs are split at a random position and the child is created by taking one part of the split from one parent and the other part from the other parent.

In contrast, PSO models the behavior of, for example, a swarm of bees searching for a food source. The population, or swarm, converges on the optimum by using information gained from each individual, referred to as a particle, as well as from the information gained by the swarm as a whole. The algorithm starts with an initial population that is randomly distributed throughout the design space. The position of each particle is then updated from one design iteration to the next, using the following

Encyclopedia of Aerospace Engineering. Edited by Richard Blockley and Wei Shyy.

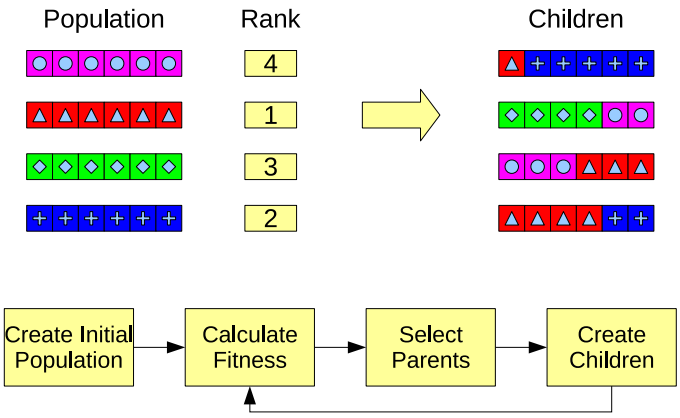

Figure 3. Overview of a basic Genetic Algorithm

update formula,

$$\boldsymbol{x}_i^{q+1} = \boldsymbol{x}_i^q + \boldsymbol{v}_i^q \Delta t \tag{8}$$

where i refers to the i^{th} individual in the swarm, q refers to the q^{th} iteration and $\boldsymbol{v}_i^q$ refers to the velocity vector of the i^{th} individual at the q^{th} iteration. The time increment Δt is typically taken as unity. Initially each particle is assigned a random velocity vector that is updated at each iteration using

$$\boldsymbol{v}_i^{q+1} = w\boldsymbol{v}_i^q + c_1 r_1 \frac{(\boldsymbol{p}_i - \boldsymbol{x}_i^q)}{\Delta t} + c_2 r_2 \frac{(\boldsymbol{p}^g - \boldsymbol{x}_i^q)}{\Delta t} \tag{9}$$

where w is known as the inertia parameter, r_1 and r_2 are random numbers between 0 and 1, and c_1 and c_2 are known as trust parameters. Additionally, $\boldsymbol{p}_i$ is the best point found so far by the i^{th} particle, while $\boldsymbol{p}^g$ is the best point found by the swarm. The inertia parameter w controls the search behavior of the algorithm, with larger values (in the region of 1.4) resulting in a more global search and smaller values (in the region of 0.5) resulting in a more local search. The c_1 trust parameter indicates how much the particle trusts itself (also referred to as the cognitive memory), while c_2 indicates how much the particle trusts the group (also referred to as the social memory). The literature recommends values of $c_1 = c_2 = 2$. Finally, $\boldsymbol{p}^g$ can be selected to represent either a "local" topology, where the best point is obtained from only a small subset of particles, or a "global" topology, where the best point is obtained from the swarm as a whole.

In terms of parameters, the user needs to tune the values of w, c_1 and c_2 and decide on the number of particles in the swarm, as well as how many iterations to perform. Even for this most basic variation of the algorithm (there are many variations that are much more complicated), there are five problem-dependent parameters that must be tuned by the user. Problem-dependent parameter tuning is common to all evolutionary methods and is one of the major drawbacks associated with these methods.

The PSO algorithm is inherently an unconstrained optimization algorithm. This is a common characteristic of most of the evolutionary algorithms and is another major drawback associated with these methods. Many different constraint-handling techniques have been investigated by researchers to deal with this problem. Constraint-handling techniques for evolutionary algorithms are classified by Koziel and Michalewicz (1999) as: (1) techniques that preserve feasibility, (2) techniques based on penalty functions, (3) techniques making a clear distinction between feasible and infeasible solutions

and (4) other hybrid techniques. More recently, Sienz and Innocente (2008) classified constraint-handling strategies for PSO as: (1) strategies that reject infeasible solutions (also known as a death penalty approach), (2) strategies that penalize infeasible solutions (also known as a penalty function approach), (3) strategies that preserve feasibility, (4) strategies that cut-off at the boundary, (5) strategies based on a bi-section approach and (6) strategies that repair infeasible solutions.

Of these approaches, one of the most popular is to make use of a penalty function approach, where the objective function is penalized for any constraint violation. Penalty functions are popular because they have traditionally been used with gradient-based optimization algorithms, are general in nature and are easy to implement. Static penalty parameters like that of Eq. 6 are typically employed, but more advanced techniques in which the penalty parameters are dynamically tuned during the optimization, have also been developed (e.g., Poon and Joaquim (2007), Hamida and Schoenauer (2000) and Barbosa and Lemonge (2003)). A recent technique for constraint handling is a bi-objective approach similar to that used by Fletcher and Leyffer (2002) for gradient-based optimization. A bi-objective optimization problem that minimizes both a measure of the constraint violation and the objective function is considered (e.g., Surry and Radcliffe (1997) and Venter and Haftka (2008)).

3.2. Deterministic Algorithms

There are also many deterministic algorithms developed specifically to solve global optimization problems. An excellent survey of global optimization algorithms is provided by Neumaier (2004). Many of the global optimization algorithms are specialized to solve only a narrow class of problems. One popular general purpose deterministic global optimization algorithm is the DIRECT algorithm by Jones, Perttunen and Stuckman (1993). The DIRECT algorithm makes use of Lipschitzian optimization to locate promising subregions in the design space. Each of these subregions is then further explored using a local search technique. An interesting comparison of using multistart techniques versus the DIRECT global optimization algorithm is provided by Cox *et al.* (2001).

4. CONCLUDING REMARKS

This chapter provided a short overview of optimization techniques typically encountered in engineering optimization applications. The techniques were classified as either local or global algorithms and both constrained and unconstrained optimization problems were considered.

The designer should be aware that no single optimization algorithm exists that will solve all optimization problems. Some knowledge of the different algorithms available will help to select the appropriate algorithm for the problem at hand. A few guidelines for algorithm selection are:

1. Local algorithms are well suited for optimization problems with many design variables (more than 50), in cases where the analyses are computationally expensive, when numerical noise is not a severe problem, when gradients are readily available, and when local minima is not an issue.
2. Global algorithms are well suited for optimization problems with fewer design variables (less than 50), where the analysis is computationally inexpensive, for discrete and combinatorial optimization problems, when numerical noise is severe, in cases where the gradient does not exists, when the objective and/or constraint functions are discontinuous, and when a global optimum is required.

In general, a rule of thumb is to consider the global methods only in cases where it is not viable to use an efficient local search.

Encyclopedia of Aerospace Engineering. Edited by Richard Blockley and Wei Shyy.

The optimization algorithms discussed here are generally widely available to the designer. The availability range from open source software like SciLab, Octave and Python, to name only a few, to optimizers included in commonly available software packages like Microsoft Excel and Matlab, to specialized commercial software e.g., DOT and VisualDOC from Vanderplaats Research and Development, iSIGHT from Dassault Systèmes, modeFRONTIER from Esteco and Optimus from Noesis Solutions. In addition, the non-gradient based algorithms are often fairly straight forward to implement and a designer that is comfortable with computer programming can easily create a custom implementation that is tailored to his or her particular problem.

ACKNOWLEDGEMENTS

This work has been supported in part by the National Research Foundation (NRF) of South Africa. Any opinion, findings and conclusions or recommendations expressed in this material are those of the author(s) and therefore the NRF does not accept any liability in regard thereto.

REFERENCES

Arora JS. *Introduction to Optimum Design* (2nd edn). Elsevier Academic Press, 2004.

Barbosa HJC and Lemonge ACC. A new adaptive penalty scheme for genetic algorithms. *Information Sciences* 2003; **156**(3–4):215–251.

Cox SE, Haftka RT, Baker CA, Grossman B, Mason WH and Watson LT. A comparison of global optimization methods for the design of a high-speed civil transport. *Journal of Global Optimization* 2001; **21**(4):415–432.

Dorigo M, Maniezzo V and Colorni A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics, Part B* 1996; **26**(1):29–41.

Fiacco AV and McCormick GP. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley & Sons Inc., 1968.

Fletcher R and Leyffer S. Nonlinear programming without a penalty function. *Mathematical Programming* 2002; **91**(2):239–269.

Fogel LJ, Owens AJ and Walsh MJ. *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons Inc., 1966.

Geem ZW, Kim JH and Loganathan GV. A new heuristic optimization algorithm: Harmony search. *SIMULATION* 2001; **76**(2):60–68.

Glover F. Heuristics for integer programming using surrogate constraints. *Decision Sciences* 1977; **8**(1):156–166.

Griewank A and Walther A. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation* (2nd edn). SIAM, 2008.

Haftka RT and Gürdal Z. *Elements of Structural Optimization* (3rd edn). Kluwer Academic Publishers, 1993.

Haim D, Giunta AA, Holzwarth MM, Mason WH, Watson LT and Haftka RT, 1999. Comparison of optimization software packages for an aircraft multidisciplinary design optimization problem. *Design Optimization* 1999; **1**:9–23.

Hamida BS and Schoenauer M. An adaptive algorithm for constrained optimization problems. In *Proceedings of the 6th Conference on Parallel Problems Solving from Nature*, 2000; 529–539.

Holland JH. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

Encyclopedia of Aerospace Engineering. Edited by Richard Blockley and Wei Shyy.

Neumaier A. Complete search in continuous global optimization and constraint satisfaction. *ACTA NUMERICA*, 2004; **13**:271–370.

Jones DR, Perttunen CD and Stuckman BE. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications* 1993; **79**(1):157–181.

Kennedy J and Eberhart RC. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, Perth, Australia, 1995; 1942–1948.

Kirkpatrick S, Gelatt CD and Vecchi MP. Optimization by simulated annealing. *Science* 1983; **220**(4598):671–680.

Koza JR. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

Koziel S and Michalewicz Z. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation* 1999; **7**(1):19–44.

Poon NMK and Joaquim RRAM. An adaptive approach to constraint aggregation using adjoint sensitivity analysis. *Structural and Multidisciplinary Optimization* 2007; **34**(1):61–73.

Snyman JA. *Practical Mathematical Optimization*. Springer, 2005.

Storn R and Price K. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 1997; **11**(4):341–359.

Surry PD and Radcliffe NJ. The COMOGA method: Constrained optimisation by multi-objective genetic algorithms. *Control and Cybernetics* 1997; **26**(3).

Sienz J and Innocente MS. Particle swarm optimization: Fundamental study and its application to optimization and to jetty scheduling problems. In *Trends in Engineering Computational Technology*, Topping BHV and Papadrakakis M (eds). Saxe-Coburg Publications, 2008; 103–126.

Vanderplaats GN. Very large scale continuous and discrete variable optimization. In *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York, Aug 31–Sep 1, 2004; AIAA-2004-4458.

Vanderplaats GN. *Multidiscipline Design Optimization*. Colorado Springs, Vanderplaats Research and Development, Inc., 2007.

Venter G and Haftka RT. Constrained particle swarm optimisation using a multi-objective formulation. In *Proceedings of the 9th International Conference on Computational Structures Technology*, Athens, Greece, Sep 2–5 2008.