A Direct-Search Optimization Algorithm for Complex Design Problems

Md. Khalilur Rahman[a]

Abstract

This paper presents a direct-search algorithm for design optimization of engineering problems having mixed variables (continuous, discrete and integer); nonlinear, non-differential and discontinuous design constraints and conflicting multiobjective functions. The intelligent movement of objects (vertices and compounds) is simulated in the algorithm based on Nelder-Mead simplex with added features to handle variable types, bound and design constraints, local optima, search initiation from an infeasible region and numerical instability, which are common requirements for large-scale, complex optimization problems. The algorithm is called INTEMOB (INTElligent Moving OBjects) and tested for a wide range of algebraic problems, simple engineering design problems and large-scale, complex engineering problems. Validation results for several examples, which are manageable within the scope of this paper, are presented herein, and references are provided for large-scale problems that were solved by INTEMOB.

Introduction

Optimization usually involves maximization or minimization of a function $f(x_1, x_2, ..., x_N)$ of several variables $x_1, x_2, ..., x_N$ subject to a set of constraints. To solve linear optimization problems, the simplex algorithm is a powerful tool and its primal and dual applications are widely documented. Most industrial design problems, however, involve nonlinear objective and constraint functions. The Sequential Linear Programming (SLP) has been a good tool for solving nonlinear problems (Mistree, et al., 1981) through a series of linear approximations.

Quadratic Programming (QP) was also introduced to solve nonlinear problems. Various QP algorithms include the primal method (Goldfarb, 1972), the dual method (Goldfarb and Idnani, 1983), the principal pivoting method (Cottle and Dantzig, 1968), the parametric method (Grigoriadis and Ritter, 1969), the primal-dual method (Goncalves, 1972) and the subproblem optimization method (Theil and Van-De-Oanne, 1960). Like SLP, the sequential applications of QP to enhance its solution capability for higher-order nonlinear problems led to the development of Sequential Quadratic Programming (SQP) techniques (Gurwitz and Overton, 1989; Murray and Prieto, 1995; Spellucci, 1998). Other classical methods for continuous variable optimization include mathematical programming (Rao, 1984), geometric programming (Peterson, 1976), optimality criteria (Berke and Venkayya, 1974) and Augmented Lagrangian (Pierre and Lowe, 1975) methods.

Most of the methods mentioned above are based on classical differential calculus, and require continuous differentiability, availability of gradient vectors and existence of second derivatives. For problems with differentiable smooth functions, these derivative based methods are usually reliable and computationally efficient. Many real-world engineering design problems, however, involve discontinuous and non-differentiable functions, design variables requiring a combination of continuous, integer and discrete values and conflicting multiple design objectives. The difficulty of non-differentiability in using the methods is attempted to overcome by various numerical differentiation techniques with various degrees of accuracy; however, they often suffer from numerical instability. On the other hand, manipulations of the original problem by SLP, SQP or special techniques incorporated in classical methods provide a solution to an approximate problem, not the exact problem.

[a] School of Oil & Gas Engineering, UWA, Perth, Australia, (Email: krahman@cyllene.uwa.edu.au)

To overcome many of the above difficulties with gradient-based methods, various direct-search methods evolved over the time. Some examples include normal-boundary intersection algorithm (NBI) (Das and Dennis, 1998), genetic algorithms (GA) (Ndiritu and Daniell, 1999; Wu and Chow, 1995), entropy algorithms (Das et al., 1999) and evolutionary operation (EVOP) (Ghani, 1989). Direct search methods, such as GA, PA and EVOP are generally slow in convergence but are successful to find reliable optimum solutions of problems having high degree of various noises including discontinuity and non-differentiability in functions. Because of these features, many direct search methods are increasingly becoming popular with the advent of ultra-fast computing facilities, and therefore there is ever growing needs to improve various features of these methods, as has been attempted in this work.

The main objectives of this paper are (1) to present the basic principle of an optimization algorithm capable to handle conveniently the above-mentioned complex natures of real-world engineering problems, (2) to present validation results for pure algebraic as well as engineering design problems that are manageable within the scope of this paper, (3) to provide further references to large-scale complex design problems that have been solved by the algorithm.

Proposed Algorithm: INTEMOB

The simplex algorithm of Nelder and Mead (1965) is one widely used contribution to direct-search optimization algorithms. It has also been the basis of the proposed algorithm because of its inherent capability to move a point to a minimum vertex of a function without requiring any derivative information; thus, conveniently handling non-differentiable and even discontinuous functions. The original simplex, however, did not incorporate any features to consider nonlinear design constraints, non-continuous variables (discrete, integer) and the presence of local minima/maxima. It was a strict requirement to initiate search with vertices within the feasible space. Bounds on variables were possible to consider by complex logarithmic scale transformation. Also it did not have any scheme to mitigate non-convergence numerical instability. Due to the lack of such features, the original simplex was not applicable to most real-world engineering problems, and was unreliable even in some fairly simple situations (McKinnon, 1998). Various attempts have been made to overcome some of the short comings (Ghani, 1972; Lagarias et al., 1998). This author incorporated and coded the following additional features: (1) starting of the optimization process with only one initial point which may also be in the feasible or infeasible space (very important, because finding even one feasible point manually may be a very exhaustive task for large and complex design problems), (2) direct handling of bounds on design variables, (3) handling of design constraints of any nature, (4) handling of discrete/integer variables, (5) mitigating potential numerical instabilities, and (6) a global optimization loop to overcome local minima/maxima.

The modified algorithm with the above-mentioned additional features is called in this paper, INTEMOB (an INTElligent Moving OBject), because it finds optimum solution by generating and moving an object (called 'compound') using 'intelligence' more than mathematics.

The general framework of INTEMOB can be expressed as:

Find

$$\underline{\mathbf{x}}^{\mathrm{T}} = \{x_1, x_2, .., x_i, ..., x_N\} \tag{1}$$

subject to bound constraints

$$\left.\begin{array}{l} l_1 \le x_1 \le u_1 \\ . \quad . \quad . \\ . \quad . \quad . \\ l_N \le x_N \le u_N \end{array}\right\} \tag{2}$$

and design constraints

$$\left.\begin{array}{l} C_{l1} \le C_1(\underline{x}) \le C_{u1} \\ . \quad . \quad . \\ . \quad . \quad . \\ C_{lM} \le C_M(\underline{x}) \le C_{uM} \end{array}\right\} \tag{3}$$

to minimize

$$Z = f(\underline{x}) \quad (4)$$

where, $\underline{x}$ represents the vector of free design variables (the superscript 'T' for transpose); l_i's and u_i's are constants or functions of $\underline{x}$ (in the latter case the bound constraints constitute moving boundaries) representing the ranges of x_i's; C_{li}'s and C_{ui}'s are constants or functions of $\underline{x}$ representing the ranges of design constraints, $C_i(\underline{x})$'s, which must not be violated by the optimum design; N is the total number of free design variables and M is the total number of design constraints; $f(\underline{x})$ is the final function to be minimized, representing either a single design objective or a formulated multiobjective function.

INTEMOB Solution Procedure

The solution procedure of INTEMOB includes 4 major steps: (1) generation of 'vertices' and formation of a 'compound'; (2) moving a compound (3) identifying and mitigating numerical instability; and (4) terminating the process by convergence tests.

Generating Vertexes and Compounds

The optimization procedure starts with an initial vertex (also called point, design) in the N-dimensional space bounded by the ranges of design variables, as shown in Fig. 1 in a two-dimensional space for the convenience of description. Straight lines (l_1, u_1 and l_2, u_2) parallel to the co-ordinate axes represent the lower and upper bounds on variables, x_1 and x_2, respectively. Curved lines C_{l1} and C_{u1} represents the lower and upper bounds, respectively, on design constraint 1, $C_1(\underline{x})$, and C_{l2} and C_{u2} on design constraint 2, $C_2(\underline{x})$. Certainly, there could be more than these two design constraints and their lower and upper bounds. The hatched area is the two-dimensional feasible search space. The initial vertex must be within the variable bounds, and may or may not satisfy the design constraints. If the initial vertex does not satisfy any of the design constraints (i.e. not within the hatched area), a random vertex is generated as follows:

$$x_i = l_i + r_i(u_i - l_i); \quad i = 1,.....N \quad (5)$$

A value of r_i is generated between 0 and 1 for the i-th coordinate by a random number generator. Any random number generator routine can be used for this purpose. From this point onward, whenever the generation of a vertex (point) or a compound is mentioned that basically means the random number generator subroutine in INTEMOB is called as required.

If the generated random vertex is still in the infeasible region, it is moved stepwise halftimes the distance between the initial vertex and the generated vertex along the straight line with these two vertices. After each movement, the design constraints are checked and the process is continued until the vertex satisfies all the design constraints. If the positive step-length moves the vertex away from the unsatisfied design constraint bound(s), a negative step-length is used. The half-step movement process will be further obvious from Eq. (6). The vertex 'a' in Fig.1 is either an initial feasible vertex, or a randomly generated feasible vertex, or feasible vertex through the movement. This overcomes the strict requirement for an initial feasible vertex.

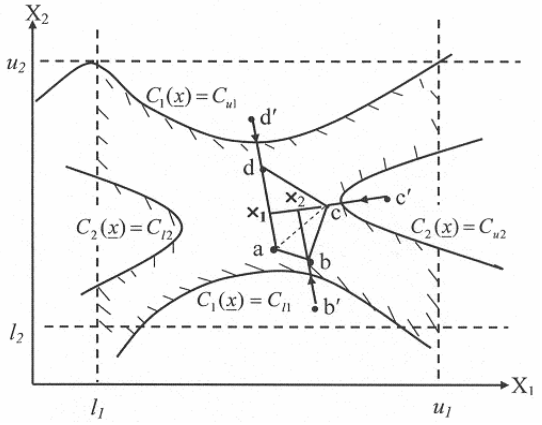

Fig. 1: Generation of initial compound.

Once a feasible vertex is established, Eq. (5) is executed further K-1 times to generate further K-1 different random vertices, where $K = 2N$ for $N \leq 5$ and $K = N + 1$ for $N > 5$. Note that the Nelder-Mead simplex requires N+1 vertices for an N-dimensional space, whereas this author has used $2N$ vertices for low-dimensional spaces based on experience of better convergence to a minimum. Equation

(5) itself ensures that the randomly generated vertices remain in the space bounded by the ranges of variables defined by Eq. (2). However, any of the generated points may initially violate any of the design constraints defined by Eq. (3) and therefore a technique is required to move such points towards satisfying Eq. (3). The four vertices for the two-dimensional space (2*N*) a, b', c' and d' are also shown in Fig. 1. Obviously, vertices b', c' and d' violate Eq. (3). These vertices are modified in the order of d', c' and b' by moving successively towards the centriod, $\underline{c}$ by:

$$\underline{x} = \frac{1}{2}(\underline{c} + \underline{x'}) \tag{6}$$

until the new point, $\underline{x}$ satisfies Eq. (3). The coordinates of the cenrtriod, $\underline{c}$ are calculated using vertices that have already satisfied Eq. (3) as follows:

$$c_i = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{7}$$

where *n* is the number of vertices which have already satisfied Eq. (3).

The modified points are a, b, c and d which satisfy both Eqs. (2) and (3). These four feasible vertices comprise an object called 'compound' abcd, as shown in Fig. 1. The values of the objective function, *f*(a), *f*(b), *f*(c) and *f*(d) at these four vertices are calculated and assumed to be in the order of *f*(a)<*f*(b)<*f*(c)<*f*(d). If the initial notations of vertices do not satisfy this order, vertices are re-denoted according to this order.

For a convex feasible parameter space the above method for moving an infeasible vertex to the feasible space would always succeed in generating a compound with *K* vertices. However, if the parameter space is nonconvex, and the centroid happens to lie in the infeasible area, there is every chance that a compound can not be generated. Fig. 2 shows such a possibility. Three vertices a, b and c in the feasible space have already been generated. The fourth vertex, e.g. a trial point T_1, satisfies the variable bounds, but violates a design constraint. To move T_1 in the feasible space it is continually moved halfway towards the centroid, X. Since the centroid itself is infeasible no amount of such moves would make T_1 feasible, and a compound with four vertices can never be generated. Safeguard against such a possibility is never to allow an infeasible centroid. If a new feasible vertex results in the new centroid to lie in the infeasible area, that new vertex is discarded, and another generated until a feasible centroid is obtained.

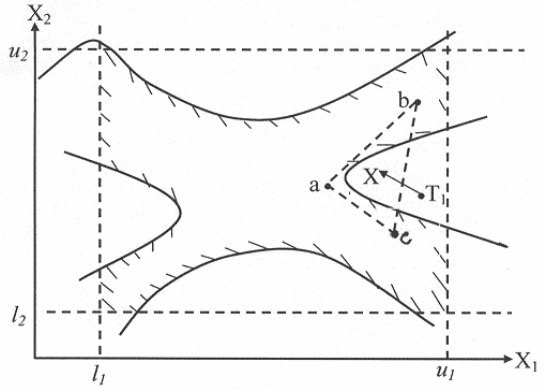

Fig.2: Centroid in the infeasible space can not generate a compound.

Moving a Compound

A compound is moved in this algorithm by shifting its worst vertex towards an optimum location in the feasible space. To initiate the compound movement, the compound vertices having highest, second highest and the lowest objective function values are identified and numbered by integers 'nh', 'nm' and 'nl', respectively. For our two-dimensional example, these correspond to *f*(d), *f*(c) and *f*(a), respectively. The movement process begins by over-reflecting the worst vertex 'nh' of a compound on the feasible centroid, $\underline{c}$, of the remaining vertices. This over-reflection generates a new trial point $\underline{x}_t$:

$$\underline{x}_t = (1.+\alpha)\underline{c} - \alpha\underline{x}_{nh} \tag{8}$$

where α is the reflection coefficient in the range of 1.0 to 2.0 and the vector $\underline{x}_{nh}$ contains the coordinate values of vertex 'nh'. If a coordinate value of the trial point violates its any of the bounds, that coordinate is just shifted inside the violated bound by a small distance, i.e. $x_a = x_b \pm \delta$, where x_a is the adjusted coordinate, x_b is the bound value that was violated by that coordinate, and δ is

a very small value, 10^{-12} is used for double precision computation. If any design constraint is violated, the trial point is repeatedly moved halfway towards the centroid of the rest of the vertices (i.e. excluding the vertex 'nh' and the trial point $\underline{x}_t$) in the compound until the constraint is satisfied. The new trial point that satisfies all the variable bounds and design constraints is called feasible trial point, $\underline{x}_{ft}$.

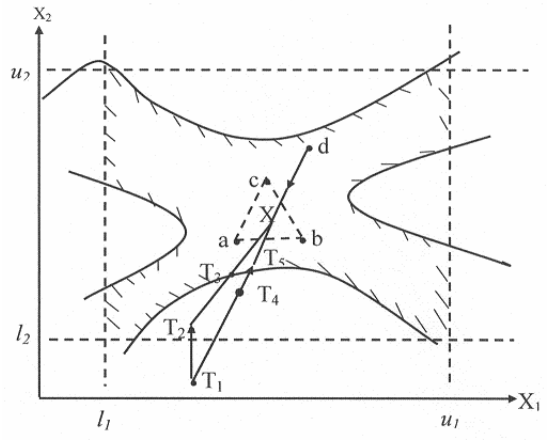

Fig. 3: Unsuccessful over-reflection and first type of contraction.

The function value at the feasible trial point, $\underline{x}_{ft}$ is next evaluated. The reflection step is considered successful if the objective function value at this new trial point is lower than that at vertex 'nm'. The current vertex 'nh' is then deleted, the vertex 'nm' is redefined as vertex 'nh' and vertices 'nm' and 'nl' are redefined according to new objective function values to form a new and superior compound. If, however, the objective function value at the trial point is greater than that at vertex 'nm' of the current compound, the trial point would still be the worst vertex in the new compound. The reflection step is, therefore, considered not successful enough and a contraction step is applied. Depending on the outcome of current over-reflection, any of the following three contraction schemes is applied.

If $f(\underline{x}_{nm}) \le f(\underline{x}_{ft}) < f(\underline{x}_{nh})$, the trial point suffers from excessive over-reflection, and the coordinates of a new trial point $\underline{x}$ is generated as:

$$\underline{x} = (1.+\beta)\underline{c} - \beta \underline{x}_{nh} \tag{9}$$

If $f(\underline{x}_{ft}) \ge f(\underline{x}_{nh})$, the coordinates of the new trial point are estimated as:

$$\underline{x} = \beta \underline{x}_{nh} + (1.-\beta)\underline{c} \tag{10}$$

In Eqs. (9) and (10), the range of β is 0.0 to 1.0. Note that Eq. (9) is an additional feature to enforce accelerated improvements in the objective function.

The third type of contraction is generation of a small compound using vertex 'nl' as the starting point. This type of contraction is applied only after first and second types of contraction (Eqs. 9 and 10, respectively) have been previously applied consecutively for more than 2*K* times.

Fig. 3 explains the first type of contraction for the two-dimensional example. The worst vertex, d of the current compound 'abcd' is over-reflected on the feasible centroid, X of 'abc'. The trial point, T_1 so obtained is moved just inside the variable bound to T_2 as $x_2 = l_2 + \delta$, which still violates a design constraint. T_2 is moved towards the centroid, X according to half-step procedure, as described in the previous section, along the line joining T_2 and X resulting in a completely feasible trial point, T_3. Function value at T_3 is calculated, and found to be intermediate between the second highest function value at vertex, c, and the highest function value at vertex, d. If the vertex, d is replaced by the feasible trial point, T_3 to form a new compound 'abcT_3', then T_3 would be the worst vertex in the new compound. Although it is an improved compound, this reflection step is considered not successful enough to change the compound configuration by changing the order of superiority among vertices. To enforce this, point T_3 is rejected, and the first type of contraction (Eq. 9) is applied. Due to this contraction, the worst vertex, d is now under-reflected on the feasible centroid, X to T_4. Since the trial point, T_4 violates a design constraint it is moved halfway towards the centroid to T_5. The trial point, T_5 is feasible, and replaces the worst vertex, d to form a new complex 'abcT_5'. The trial point, T_4 would have been on the other side of X, i.e. closer

to vertex, d in case the second type of contraction was necessary to apply. Then the movement of this point, if it were in the infeasible region, to the feasible space would have been similar.

If on over-reflection the trial point has not violated any constraints (i.e. $\underline{x}_t$ and $\underline{x}_{ft}$ are identical without any movement), and has an objective function value lower than the lowest function value at vertex 'nl' of the current compound (i.e. $f(\underline{x}_t) < f(\underline{x}_{nl})$), and the previous action was not contraction by any of the three schemes above, this over-reflection is considered over-successful. An expansion attempt is then taken to generate a new trial point further away from the feasible centroid along the same straight line used for over-reflection. The co-ordinates of this expanded trial point is given by:

$$\underline{x}_e = \gamma \underline{x}_t + (1. - \gamma)\underline{c} \qquad (11)$$

The usual range of γ is 1.0-3.0. The feasibility of this expanded trial point, $\underline{x}_e$ is checked. If any of variable bounds or design constraints is violated, the expansion attempt is considered unsuccessful, and a new compound is formed with the over-reflected feasible trial point $\underline{x}_t$ that replaces the worst vertex 'nh' of the current compound. Otherwise, the objective function value at $\underline{x}_e$ is evaluated. If $f(\underline{x}_e) < f(\underline{x}_t)$, the expansion attempt is considered successful. The expanded vertex, $\underline{x}_e$ then replaces the worst vertex 'nh' to form the new compound. If $f(\underline{x}_e) \geq f(\underline{x}_t)$, the expansion attempt is also considered unsuccessful. The expanded vertex, $\underline{x}_e$ is rejected and the trial point, $\underline{x}_t$ is used to form the new compound.

The next cycle of compound movement is started with the current compound improved through the above-described procedure.

Mitigating Numerical Instability

Two types of numerical instability are experienced during repeated compound movement: (1) collapse of a compound on an infeasible centroid, and (2) collapse of a compound on a straight line.

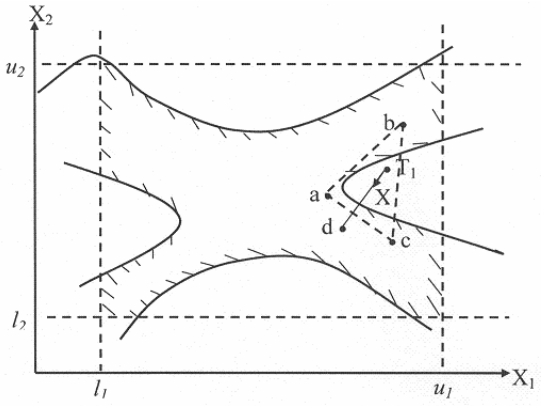

Fig. 4: Collapse on infeasible centroid.

Compound Centroid in Infeasible Space

If the search space is non-convex, there is a possibility that the compound would collapse due to shifting by over-reflection. For example, referring to Fig. 4, the worst point, d is reflected over the centroid, X to create a trial point, T_1, which violates a design constraint. Since the centroid itself is in the infeasible region, repeated movement of point, T_1 halfway towards the centroid would result in collapse of the point on the centroid. Collapsing of a point on the centroid means their corresponding coordinate values are equal within a pre-specified resolution. The new compound has now three vertices a, b and c. One more such collapse would result in complete collapse of the compound, because an object with two vertices can not span in a two dimensional space. Note that a compound not to collapse in an N-dimensional space, $K \geq (N+1)$ must be satisfied.

This collapse problem can be overcome as follows:

1. The centroid of all vertices except 'nh' is calculated, and its feasibility is verified.
2. If the centroid is feasible, the steps below here are not necessary and the process then passes on to testing for collapse on a straight line. Otherwise, continue from (3) below.
3. Vertex 'nm' is excluded, and the centroid of all other vertices is calculated (i.e. excluding vertices 'nh' and 'nm').
4. If the centroid is in the feasible space then continue from (5) below. Otherwise, if the centroid is infeasible a new compound of

normal size is generated using vertex 'nl' as the starting point, and steps from (1) are repeated. A check is made to ascertain whether the compound is the initial one. If so, 'nl' is set to *K* to ensure a high probability that the starting point for the new compound is well inside the feasible space. However, if the new centroid is feasible then steps from (5) below are continued.

5. Vertex 'nm' is replaced by a randomly generated feasible point.

6. The new centroid of all vertices except 'nh' is once again calculated, and its feasibility is checked.

7. If the centroid is feasible the function value at the newly generated vertex 'nm' is calculated, and steps from (1) are repeated. Otherwise, steps from (3) are repeated.

Compound Collapse on a Straight Line

A compound is said to have collapsed on a straight line if the absolute difference between the *i*-th coordinate of the compound centroid and that of all *K* vertices becomes less than a specified value, making effectively a straight line. A resolution factor, φ_{cpx} is used to detect such a collapse. For example, if the value of i-*th* coordinate of the compound centroid is v_1 and the value of that coordinate for the farthest vertex of that compound is v_2, the compound will be considered to collapse if v_1 and $(v_1 + \varphi_{cpx} \times (v_2 - v_1))$ are identical within the resolution of φ_{cpx}. If φ_{cpx} is set to 10^{-1}, the compound is considered collapsed if v_1 and v_2 differ at the most by the least significant digit. If φ_{cpx} is set to 10^{-2}, the compound is considered collapsed if v_1 and v_2 differ by not more than the last two significant digits. Its value is, however, used much finer, typically 10^{-11} for double precision computation.

Fig. 5 shows a compound with vertices a, b, c and d and centroid, X, which has collapsed to virtually on a straight line. In terms of X_2 coordinate, the farthest vertex from the centroid X is 'a'. If X_{2a} and X_{2x} are identical within the resolution of φ_{cpx}, then the X_2 coordinates of all vertices and the centroid are also identical within the resolution of φ_{cpx} and the compound 'abcd' is said to have collapsed.

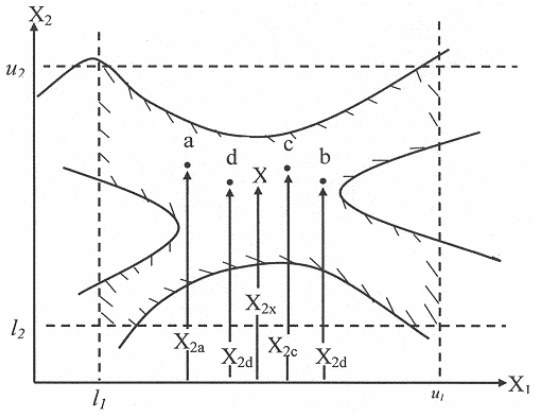

Fig. 5: Compound collapse on a straight line.

On detecting such a collapse of a compound, the following actions are taken:

1. If the compound has collapsed for the first time the coordinates of the centroid are stored.

2. Another compound is generated with vertex 'nl' as the starting point.

3. If the compound has collapsed consecutively more than once, the centroid of the currently collapsed compound is adjusted as $\underline{c} = \underline{c1} + 10\varphi_{cpx}(\underline{cc} - \underline{c1})$ in which $\underline{c1}$ is the centroid of the first collapsed compound and $\underline{cc}$ is the centroid of the currently collapsed compound. The distance between the centroid of the first collapsed compound and this adjusted centroid is calculated.

4. If this distance is zero, a compound of normal size is generated with vertex 'nl' as the starting point, and no further testing for collapse of compound is carried out.

5. Otherwise, whether the compound has collapsed consecutively for the second time is determined. If so, the objective function value f_o at the centroid of the first collapsed compound is calculated.

6. The function value f_c at the centroid of the currently collapsed compound is calculated.

7. If $f_c < f_o$, the centroid of the first collapsed compound is over-reflected on the centroid of the currently collapsed compound.

8. Otherwise, compounds are continuously regenerated on each consecutive collapse using 'nl' as the starting point, and steps from (6) above are repeated.

Termination by Convergence Tests

While executing the process of compound movement, tests for convergence are carried out periodically after certain preset number of evaluations of the objective function. Three levels of convergence tests are conducted. The first convergence test is considered successful if a predefined number of consecutive values of the objective function are found identical within the resolution of a convergence parameter, φ. The second test for convergence verifies whether the objective function values at all vertices of the current compound are also identical within the convergence resolution. The second test is conducted only if the first test has succeeded. The typical value of φ for double precision computation is 10^{-10}. The value of φ_{cpx} for compound collapse detection is recommended to be a decade lower than φ (i.e. if $\varphi = 10^{-10}$ then $\varphi_{cpx} = 10^{-11}$) to avoid premature convergence.

When both the above convergence tests are successful, the current optimum point is preserved and the whole computation procedure is repeated, this time starting with the current optimum point, to verify if there are any better optimum points (e.g. multi minima). This repetition continues as long as the objective function value can be improved within the resolution of another convergence parameter (the third convergence). The nested three-looped convergence procedure continues as long as the objective function can be improved. The major steps of computation in INTEMOB are summarized in Fig. 6.

Equality Constraints and Integer/Discrete Variables

It is obvious that the procedure does not directly deal with equality constraints in the form of $h_j(\underline{x}) = 0,\ j = 1,2,\ldots\ldots,L$ in which L is the number of equality constraints. These equality constraints can, however, be satisfied by minimizing an augmented objective function as:

$$f(\underline{x}, \underline{\lambda}) = f(\underline{x}) + \sum_{j=1}^{L} \lambda_j h_j(\underline{x}) \qquad (12)$$

where λ_j's are the weighting factors to equality constraints such that all $h_j(\underline{x})$ vanishes at the minimum of $f(\underline{x}, \underline{\lambda})$.

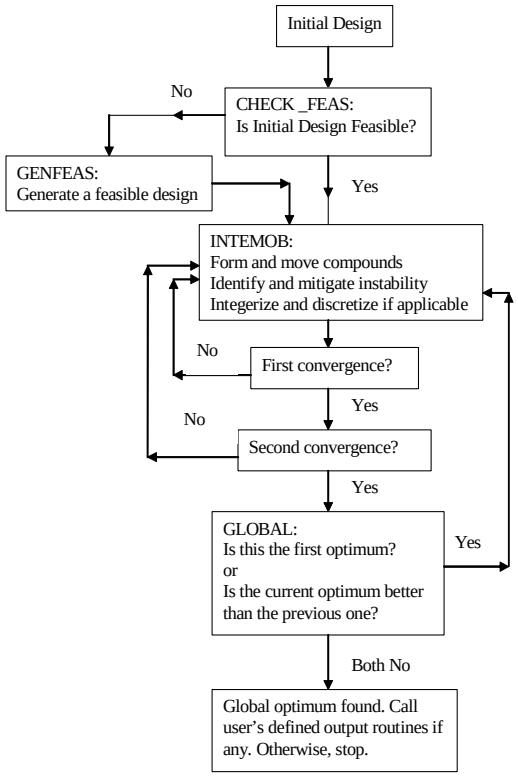

Fig. 6: Flow-chart for INTEMOB.

An integer variable is treated as continuous until a compound is formed with feasible vertices. After successful over-reflection, contraction and expansion of the compound, the variable is integerized as follows:

$$\begin{aligned} &\Delta x = x - x_{\text{int}} \\ &\text{if } \Delta x < 0.5; \quad x_i = x_{\text{int}} \text{ for } x \geq 0 \\ &\text{if } \Delta x \geq 0.5; \quad x_i = x_{\text{int}} - 1 \text{for} x < 0 \\ &\qquad\qquad\quad x_i = x_{\text{int}} + 1 \text{for} x > 0 \end{aligned} \qquad (13)$$

where x is the continuous value of the variable to be integerized, x_{int} is integer portion of x and x_i is the integerized value. If x_i from Eq. (13) violates its bounds, the following adjustments are made:

If $x_i > x_{max}$, integerize x_{max} to x_{imax} with the same rule in Eq. (13) replacing all x's by x_{max}. Then adjust x_i as follows:

If $x_{imax} > x_{max}$; $x_i = x_{imax} - 1$; else $x_i = x_{imax}$ (14)

Similarly, if $x_i < x_{min}$, integerise x_{min} to x_{imin} with same rule in Eq. (13) replacing all x's by x_{min}. Then adjust x_i as follows:

If $x_{imin} < x_{min}$; $x_i = x_{imin} + 1$; else $x_i = x_{imin}$ (15)

A discrete variable is also initially treated as continuous, similar to an integer variable. After over-reflection, contraction and expansion of the compound, the variable is discretized as follows:

$$\text{If } f(\underline{x}_{ud}) < f(\underline{x}_{ld}); x_d = x_{ud}; \text{ else } x_d = x_{ld} \quad (16)$$

where x_d is the discrete value of the variable; x_{ud} is the upper discrete value closest to its continuous value, x; x_{ld} is the lower discrete value closest to x; $f(\underline{x}_{ud})$ is the objective function value with variable vector containing the upper discrete value, x_{ud} and $f(\underline{x}_{ld})$ is the objective function value with variable vector containing the lower discrete value, x_{ld}. If x_d violates any of its bounds and/or design constraints, the closest upper or the lower discrete value will certainly satisfy such constraints and will replace the value of x_d obtained from Eq. (16), because they were previously satisfied by its continuous value, x. It is important to note here that the compound centroid must be calculated using continuous values of variables and the compound collapse test must be bypassed if a problem has integer and/or discrete variables.

It is important to note that a simple rounding strategy is not adopted to treat integer and discrete variables. The discrete variables are properly treated based on function values after every movement of the compound. The integer variables, however, are rounded after every movement, not at the end of optimization. This facilitates the movement along the optimum directions for most practical cases. Although such an integerization scheme is not mathematically correct, it produces an engineering design with acceptable accuracy.

Multiobjective Formulation

In many industrial design problems, the designer is required to optimize several conflicting objectives simultaneously. For such conflicting objective optimization problems, the designer has to be satisfied with a compromise solution called 'Pareto-optimal'.

A recent review Coello and Christiansen (1999) has summarized the continuing development in multiobjective optimization. Bearing in mind that INTEMOB solution algorithm is capable to operate only on a single objective function for minimization, the author proposes a general normalized combined objective function as follows:

$$\text{minimize } Z = \sum_{i=1}^{I} \frac{\left(f_i(\underline{x}) - T_i\right)}{D_i} P_i \quad (17)$$

where $f_i(\underline{x})$ is the objective function for i-th objective; T_i is the target value for the i-th objective; D_i is the dividing factor for i-th objective equation and P_i is the priority to achieve the i-th objective.

The target value, which is desired to achieve for the corresponding objective either may refer to a specified numerical value or can be calculated as a function of design variables. If a particular objective has no target value, that objective requires minimization only. This can be achieved by simply equating the corresponding T_i to zero. If necessary, any of the objectives can be excluded by equating the corresponding priority factor to zero. The maximization of a particular objective can be achieved by using either a very high target value for that objective, or minimizing the negative function for that objective.

Normalization by dividing by D_i is required to ensure contributions of equal magnitude (at least within some reasonable range) by the individual objectives to the overall combined objective function. This normalization factor eliminates the necessity for wide and trial variation of priority factors to obtain proper trade-offs among objectives, as was experienced by Balachandran and Gero (1984) using weighted scalar functions and prioritized goal programming functions. The use of the proposed equation (17), while carefully normalized, provided very sensitive trade-offs among various objectives with very

small changes of priority factors in several large-scale multiobjective optimization problems (Rahman and Caldwell, 1992; Rahman, 1996; Rahman et al., 2001).

Verification Results

The algorithm has been so far applied to numerous algebraic equations and complex design problems, and provided very satisfactory solutions. To shorten presentation within the scope of this paper, solutions of only two algebraic equations and two small engineering design problems will be presented herein. Examples of more complex design optimization works performed by INTEMOB can be found in references (Rahman et al., 2001; Valencia et al., 2003).

An Algebraic Problem with an Exponentially Nonlinear Constraint

The problem is stated as (Klingman and Himmelbalu, 1964):

$$\text{Minimise} f(\underline{x}) = -\exp\left\{-(x_1-1)^2 - \frac{(x_2^2-0.5)^2}{0.132}\right\} \quad 18)$$

subject to

$$\begin{aligned} 0.2 \le x_1 \le 2.0 \\ 0.2 \le x_2 \le 2.0 \end{aligned} \quad (19)$$

and

$$-10000 \le (x_1^2 + x_2^2) \le 4.0\,. \quad (20)$$

The algorithm correctly found its solution at the minimum, x_1 =1.0 and x_2 = 0.707.

An Algebraic Unconstrained Problem

This problem is stated as (Goldstein and Price, 1971):

$$\begin{aligned} &\text{Minimise } f(x) = \exp\left\{\tfrac{1}{2}(x_1^2 + x_2^2 - 25)\right\}^2 \\ &+ \sin^4(4x_1 - 3x_2) + \tfrac{1}{2}(2x_1 + x_2 - 10)^2 \end{aligned} \quad (21)$$

The original problem was unconstrained. The following bound and design constraints were used for compatibility with the algorithm:

$$\begin{aligned} -5.0 \le x_1 \le 5.0 \\ -5.0 \le x_2 \le 5.0 \end{aligned} \quad (22)$$

and

$$-180.2182 \le \left\{\tfrac{1}{2}(x_1^2 + x_2^2 - 25)\right\}^2 \quad (23)$$

The algorithm correctly found coordinates of minimum as x_1 = 3.0 and x_2 = 4.0.

Pressure Vessel Design with mixed Continuous and Discrete Variables

The design objective is to minimize the combined cost of materials, forming and welding and also satisfy the ASME design code requirements (Ndiritu and Daniell, 1999). The four design variables are the cylindrical shell thickness x_1, the spherical head thickness x_2, the radius of the cylindrical shell x_3 and the length of the shell x_4. The shell thickness and the spherical head thickness are required to be discrete multiples of 0.0625 inches according to the available sizes of rolled steel plates while the radius of the cylindrical shell and the length of the shell are continuous variables. Mathematically, the problem can be described as:

Find

$$\underline{x} = \{x_1, x_2, x_3, x_4\}^T \quad (24)$$

to minimize

$$\begin{aligned} &f(\underline{x}) = 0.6224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 \\ &+ 3.1611 x_1^2 x_4 + 19.84 x_1^2 x_3 \end{aligned} \quad 25)$$

subject to bound constraints

$$\begin{aligned} 1.125 \le x_1 \le 65.0625 \\ 0.625 \le x_2 \le 64.5625 \\ 40 \le x_3 \le 80 \\ 20 \le x_4 \le 60 \end{aligned} \quad (26)$$

and design constraints

$$C_1(\underline{x}) = 0.0193x_3 - x_1 \le 0$$
$$C_2(\underline{x}) = 0.00954x_3 - x_2 \le 0 \qquad (27)$$
$$C_3(\underline{x}) = 750 \times 1728 - \pi x_3^2 x_4 - \tfrac{4}{3}\pi_3^3 \le 0$$

Table I: Optimum designs of pressure vessel by INTEMOB and other studies.

Case	x_1	x_2	x_3	x_4	$f(\underline{x})$
Ndiritu and Daniell (1999)	1.125	0.625	58.2209	44.086	7202.517
Wu and Chow (1995)	1.125	0.625	58.1978	44.2930	7207.497
Sandgren (1990)	1.125	0.625	48.97	106.72	7982.5
Fu et al. (1991)	1.125	0.625	48.3807	111.7449	8048.619
INTEMOB	1.125	0.625	58.2367	44.0247	7204.32

The problem was reformulated within the framework of INTEMOB and solved. The INTEMOB design is compared with published designs in Table I. The INTEMOB solution seems to be slightly inferior to that by Ndiritu and Daniell (1999). The author believes that this is because INTEMOB does not accept any percentage of infeasibility, which is acceptable in most of the optimization programs.

Spring Coil Design with mixed Continuous, Discrete and Integer Variables

The objective of this problem is to minimize the volume of wire used to manufacture a coil compression spring. The three design variables include the number of spring coils *N* which is an integer variable, the winding (coil) diameter *D* which is a continuous variable, and the wire diameter *d* which is a discrete variable and has to be chosen from the discrete values listed in Table II. The mathematical formulations of this problem are as follows:

Find $\underline{x} = \{x_1, x_2, x_3\}^T = \{N, D, d\}^T$ (28)

To minimise $f(\underline{x}) = \pi^2 x_2 x_3^2 (x_1 + 2.0)/4.0$ (29)

Subject to bound constraints

$$1 \le x_1 \le 32$$
$$0.01 \le x_2 \le 2.00 \qquad (30)$$
$$0.0090 \le x_3 \le 0.50$$

and design constraints

$$C_1(\underline{x}) = \frac{8 C_f F_{\max} x_2}{\pi x_3^3} - S \le 0$$
$$C_2(\underline{x}) = l_f - l_{\max} \le 0$$
$$C_3(\underline{x}) = d_{\min} - x_3 \le 0 \qquad (31)$$
$$C_4(\underline{x}) = x_2 + x_3 - D_{\max} \le 0$$
$$C_5(\underline{x}) = 3.0 - \frac{x_2}{x_3} \le 0$$
$$C_6(\underline{x}) = \delta_p - \delta_{pm} \le 0$$
$$C_7(\underline{x}) = \delta_p + \frac{F_{\max} - F_p}{K} + 1.05(x_1 + 2)x_3 - l_f \le 0$$
$$C_8(\underline{x}) = \delta_w - \frac{F_{\max} - F_p}{K} \le 0$$

where

$$C_f = \frac{4(x_2/x_3) - 1}{4(x_2/x_3) - 4} - \frac{0.165\, x_3}{x_2}$$
$$K = \frac{G x_3^4}{8 x_1 x_2^3}$$
$$l_f = \frac{F_{\max}}{K} + 1.05(x_1 + 2)x_3$$
$$\delta_p = \frac{F_p}{K}$$

Various parameters used in the design constraints are specified as follows:

(a) the maximum working load, F_{max} = 1000.0 lb
(b) the allowable maximum shear stress, S = 189000.0 psi
(c) the maximum free length, l_{max} = 14.0 inch
(d) the minimum wire diameter, d_{min} = 0.2 inch
(e) the maximum outside diameter of spring, D_{max} = 3.0 inch
(f) the preload compression force, F_p = 300.0 lb
(g) the allowable maximum deflection under preload, δ_{pm} = 6.0 inch
(h) the deflection from preload position to maximum load position, δ_w = 1.25 inch
(i) the shear modulus of the material, G = 11.5×10^6 psi.

The problem was formulated within the framework of INTEMOB and solved. The INTEMOB design is compared with other published designs in Table III. Obviously, the INTEMOB solution is the best one.

Table II: Allowable wire diameters for spring coil.

0.0090	0.0150	0.0280	0.0720	0.1620	0.2830
0.0095	0.0162	0.0320	0.0800	0.1770	0.3070
0.104	0.0173	0.0350	0.920	0.1920	0.3310
0.0118	0.0180	0.0410	0.1050	0.2070	0.3620
0.0128	0.0200	0.0470	0.1200	0.2250	0.3940
0.0132	0.0230	0.0540	0.1350	0.2440	0.4375
0.140	0.0250	0.0630	0.1480	0.2630	0.5000

Table III: Optimum designs of coil spring

Design source	Variables			Objective function
	x_1 (integ.)	x_2 (cont.)	x_3 (disc.)	$f(\underline{x})$
Sandgren (1990)	10	1.180701	0.283	2.7995
Chen & Tsao (1993)	9	1.2287	0.283	2.6709
Wu & Chow (1995)	9	1.227411	0.283	2.6681
INTEMOB	9	1.16	0.265	2.205

Convergence Test

Due to the relatively simple nature of the above example problems, the global optimization loop converged (the third convergence) with a very few iterations. This may however take a reasonable number of global iterations for large-scale, real-world, complex engineering problems. The convergence history of such a large engineering design optimization problem (Rahman et al., 2001) is presented in Figure 7. The problem was to optimize hydraulic fracture treatments for a gas reservoir while maximizing the Net Present Value (NPV) considering gas production over 10 years. The calculation of NPV as the objective function for INTEMOB, and that of 10 highly nonlinear constraint functions was very computation intensive. This involved a large number of subroutines each of which contained numerous internal iterative loops and evaluation of complementary error functions. These made the objective and constraint functions nondifferentiable and discontinuous. The optimization process was started with three very different initial designs, which eventually converged to the same optimum design within 80 redesign iterations (restart for global optimization). As per the inherent mechanism of compound reconfiguration, only the initial design may be infeasible; every subsequent design is feasible and better than its predecessors. As can be seen in Figure 7, the algorithm has not only found the same optimum design through such compound reconfiguration, but it has also overcome a number of local optima terrains in order to reach the final optimum. Running the algorithm on a typical personal computer, the total real time to find a global optimum design took about 40 seconds.

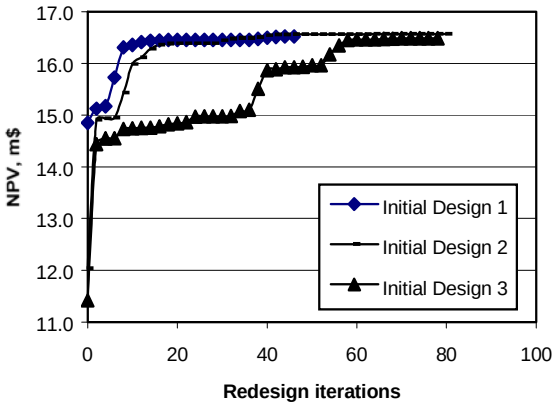

Fig. 7: Convergence to optimum design from different initial designs.

Conclusions

The role of optimization in engineering design is logically established and wide-range capabilities of optimization tools are identified to deal with real-world engineering problems of various natures. Basic principles, capabilities and limitations of various optimization methods in different classes have been briefly reviewed.

For many real-world engineering design problems, benefits of direct search methods are highlighted. The proposed moving object algorithm, INTEMOB is developed and coded by adding a number of features to a classical evolutionary algorithm to enhance its capability to handle engineering design problems of varying natures. Special features of the algorithm, such as, mitigation of

numerical instability, sequential restarting to establish the global optimum and dealing with equality constraints, integer variables, discrete variables and multiple conflicting objectives are presented in details.

The algorithm was verified with a wide range of algebraic problems and real-world engineering design problems. Within the scope of this paper, solutions of two algebraic problems and two relatively simple engineering design problems have been compared with known solutions, and the capability of the algorithm to find the actual optimum solution has been proved. References are made to large-scale optimization works performed by INTEMOB.

Acknowledgements

The author acknowledges the FORTRAN77 source code of the Nelder-Mead simplex contributed by Dr. S.N. Ghani, upon which the current enhanced version is built. The author also welcomes interests from readers to test the current code in their research works, or to develop collaborative research projects in which the code will be used and improved. Finally, the author thanks two anonymous reviewers for their constructive comments that have raised the standard of this paper.

References

[1] Balachandran, M. and Gero, J.S., (1984), A comparison of three methods for generating the Pareto optimal set, *Eng. Optim*., **7**(4), 319-336.

[2] Berke, L. and Venkayya, V.B., (1974), Review of optimality criteria approaches to structural optimization, *ASME, AMD*, **7**, 23-24.

[3] Chen, J.L. and Tsao, Y.C., (1993), Optimal design of machine elements using genetic algorithms, *J. of the Chinese Soc. of Mech. Engrs.*, **14**(2), 193-199.

[4] Coello, C.A.C. and Christiansen, A.D., (1999), Moses: a multiobjective optimization tool for engineering design, *Eng. Optim*., **31**, 337-368.

[5] Cottle, R.W. and Dantzig, G.B., (1968), Complementary pivot theory of mathematical programming, in: G.B. Dantzig and A.F. Veinott, eds., *Lectures in Applied Mathematics II, Mathematics of the Decision Sciences*, Part I (American Mathematical Society, Providence, RI), 115-136.

[6] Das, I. and Dennis, J.E. JR., (1998), Normal-boundary intersection: a new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM J. Optim.*, **8**, 631-657.

[7] Das, N.C., Mazumder, S.K. and Kajal, D.E., (1999), Constrained non-linear programming: a minimum cross-entropy algorithm, *Eng. Optim.*, **31**, 479-487.

[8] Fu, J., Fenton, R.G. and Cleghorn, W., (1991), A mixed integer-discrete-continuous programming method and its application to engineering design optimization, *Eng. Optim.*, **17**, 263-280.

[9] Ghani, S.N., (1972), An improved complex method of function minimization, Computer-Aided Design, 71-78.

[10] Ghani, S.N., (1989), A versatile procedure for optimization of a nonlinear nondifferentiable constrained objective function, AERE R13714, United Kingdom Atomic Energy Authority, Nuclear Physics and Instrumentation Division, Harwell Laboratory, Oxfordshire, UK, December.

[11] Goldfarb, D., (1972), Extension of Newton's method and simplex methods for solving quadratic programs, in: F.A. Lootsma, ed., *Numerical Methods for Nonlinear Optimization*, Academic Press, London, 239-254.

[12] Goldfarb, D. and Idnani, A., (1983), A numerically stable dual method for solving strictly convex quadratic programs, *Math. Program.*, **27**, 1-33.

[13] Goldstein, A.A. and Price, J.F., (1971), On descent from local minima, *Mathematics of Computation,* **25** (115), 569-574.

[14] Goncalves, A.S., (1972),A primal-dual method for quadratic programming with bounded variables, in: F.A. Lootsma, ed., *Numerical Methods for Nonlinear*

Optimization, Academic Press, London, 255-263.

[15]Grigoriadis, M.D. and Ritter, K., (1969), A parametric method for semidefinite quadratic programs, *SIAM J. Control*, **7**, 559-577.

[16]Gurwitz, C. and Overton, M., (1989), Sequential quadratic programming methods based on approximating a projected Hessian matrix, *SIAM J. Sci. Comput.*, **10**, 631-653.

[17]Lagarias, J.C., Reeds, J.A., Wright, M.H. and Wright, P.E., (1998), *SIAM J. Optim.*, ***9***(1), 112-147.

[18]Klingman, W.R. and Himmelblau, D.M., (1964), Nonlinear programming with the aid of a multi-gradient summation technique, *J. of the Assoc. for Comp. Machinery*, **11**(4), 400-415.

[19]McKinnon, K.I.M., (1998), Convergence of the Nelder-Mead simplex method to a nonstationary point, *SIAM J. Optim.*, **9**(1), 148-158.

[20]Mistree, F., Hughes, O.F. and Phouc, H.B., (1981), An optimization method for the design of large, highly constrained complex systems, *Eng. Optim.*, **5**, 179-197.

[21]Murray, W. and Prieto, J.P., (1995), A sequential quadratic programming algorithm using an incomplete solution of the subproblem, *SIAM J. Optim.*, **5**, 590-640.

[22]Ndiritu, J.G. and Daniell, T.M., (1999), An improved genetic algorithm for continuous and mixed discrete-continuous optimization, *Eng. Optim.*, **31**, 589-614.

[23]Nelder, J.A. and Mead, R., (1965), A simplex method for function minimization, *Computer Journal*, **7**(4), 308-313.

[24]Peterson, E.L., (1976), Geometric Programming, *SIAM Review*, **18** (1), 1-51.

[25]Pierre, D.A. and Lowe, M.J., (1975), *Mathematical Programming via Augmented Lagrangian: An Introduction with Computer Programs*, Addison-Wesley Publishing Co., Inc., Reading, Massachusetts.

[26]Rahman, M.K. and Caldwell, J.B., (1992), Rule-based optimization of midship structures, *Marine Structure*, **5**, 467-490.

[27]Rahman, M.K., (1996), Optimization of panel forms for improvement in ship structures, *Structural Optimization*, **11**, 195-212.

[28]Rahman, M.M., Rahman, M.K. and Rahman, S.S., (2001), An integrated model for multiobjective design optimization of hydraulic fracturing, *Journal of Petroleum Science and Engineering*, **31**, 41-62.

[29]Rao, S.S., (1984), *Optimization: Theory and Applications*, second ed., Halsted Press, New York.

[30]Sandgren, E., (1990), Nonlinear integer and discrete programming in mechanical design optimization, Trans. of the ASME, *J. of Mech. Design*, **112** (2), 223-229.

[31]Spellucci, P., (1998), An SQP method for general nonlinear programs using only equality constrained subprograms, *Math. Program.*, **82**, 413-448.

[32]Theil, H. and Van De Panne, C., (1960), Quadratic programming as an extension of conventional quadratic maximization, *Management Science*, **7**, 1-20.

[33]Valencia, K.L., Chen, Z, Rahman, M.K. and Rahman, S.S., (2003), An integrated model for the design and evaluation of multiwell hydraulic fracture treatments for gas-condensate reservoirs, paper SPE 84860, *International Improved Oil Recovery Conference in Asia Pacific*, Kuala Lumpur, Malaysia, Oct.20-21.

[34]Wu, S. and Chow, P, (1995), Genetic algorithms for nonlinear mixed discrete-integer optimization problems via meta-genetic parameter optimization, *Eng. Optim.*, **24**, 137-159.