DEVELOPMENT OF A GENETIC ALGORITHM BASED SEARCH STRATEGY SUITED FOR DESIGN OPTIMISATION OF INTERNAL COMBUSTION ENGINES

Nalitolela, N.G. and Mshana, J. S.
University of Dar es Salaam, Faculty of Mechanical and Chemical Engineering,
P.O. Box 35131 Dar es Salaam
Tel: 255 22 2410367 Email: noel@uccmail.co.tz

ABSTRACT

Engine design optimisation is a multi-objective, multi-domain problem in a discontinuous design space. The state of the art of optimisation techniques shows that only methods of direct and adaptive search are appropriate for this type of problem. These include, adaptive random search, simulated annealing, evolution strategies and genetic algorithms. Of these methods, the genetic algorithms have been shown to be the most suited for the optimisation of multi-modal response functions in a discontinuous design space. This paper considers the important characteristics of genetic algorithms and their adaptation for use in parametric design optimisation of internal combustion engines. In order to verify the basic functionality of the proposed optimisation strategy, a genetic algorithm based, optimisation software was developed and tested on a number of analytical functions, selected from optimisation literature, with satisfactory results.

Keywords: *Optimisation, engine noise, multi-objective, multi-domain, genetic algorithm.*

1.0 INTRODUCTION

The search for optimal design modifications of engines to meet increasingly tighter performance specifications with respect to emissions is becoming increasingly important. In order to reduce the time and cost associated with empirical testing, predictive techniques of varying degree of sophistication have been advanced. Using these techniques, engine structures can be conceptualized into mathematical models, analysed and subsequently modified after examination of the analysis results. The modified structure is then re-analysed, the analysis examined, and the structure modified again, and so on, until a satisfactory response is achieved. In order to perform the modifications in one step, finite element based optimisation strategies that make use of gradient based iterative optimisation algorithms have been attempted. To locate global optima, these methods rely on the smoothness of the objective function and the existence of derivatives. However, engine noise is a multimodal and noisy function, with many sharp discontinuities, and gradient based methods suffer from the difficulty to distinguish between a local minimum and global optimum values (Zhang, 1992; McCulloch, 1996). Furthermore, for dynamically oriented functions like engine noise, the gradient information is not easy to evaluate (Zhang, 1992). Moreover, most of the optimisation schemes reported in the literature have dealt with the problem of minimisation of radiated noise while meeting engine weight constraint. This has been achieved by employing single objective constrained optimisation algorithms. However, experience has shown that reducing noise levels tends to increase weight, thus creating a conflict. Moreover, strategies to meet other requirements are usually in conflict with the low noise objective. Therefore, the minimisation of radiated noise cannot be considered in isolation, and a suitable optimisation tool must be able to take into account all the conflicting product objectives simultaneously. This calls for the development of suitable constrained multi-objective optimisation algorithms. The traditional gradient based optimisation techniques, limited by function smoothness and continuity requirement, are not suitable for this class of problems.

2.0 OVERVIEW OF CANDIDATE SEARCH STRATEGIES

The limitations of the traditional gradient based search strategies for design optimisation of internal combustion engines have been presented in the introduction. An alternative to gradient based methods is the enumerative schemes. Given a finite search space, the algorithm starts to search on the objective function values at every point in the space, one at a time. However, many practical spaces are simply too large to search one point at a time. Even the best known enumerative scheme 'dynamic programming' breaks down on problems of moderate size and complexity (Bellman, 1961). In the current application where the necessary number of evaluations of the objective function to reach the optima poses a major constraint in view of CPU resource requirement and job completion time, enumerative schemes are certainly inefficient and therefore not suitable.

Random search algorithms such as 'simulated annealing' have achieved increasing popularity, (see for example, Kirkpatrick et al, 1983; Cermy, 1985; Ali and Stoney 1996; DeVicente et al, 2003). Ali and Stoney (1996) reports of modifications to the basic algorithm to incorporate strategic decision rules based on knowledge of the global structure of the problem, and a learning procedure to make effective use of information gained in the previous iterations. Various hybrid schemes incorporating gradient based search and simulated annealing have also been suggested (Semenkin and Semenkina, 1996; Das and Chakrabati, 2005). However random schemes that search and save the best they can be expected to do is no better than enumerative schemes in the long run. They are therefore discarded for this task on grounds of inefficiency.

Developments in computational models of evolutionary processes have led to the realisation of powerful, robust and general search and adaptive schemes collectively known as evolutionary algorithms. These include the genetic algorithms (Goldberg, 1989; Mitchell, 1996; Schmitt, 2001; Goldberg, 2002; Bies et al, 2006), evolutionary programming (Fogel, 1994, 1998, 2006; Eiben and Smith, 2003), evolution strategies (Schwefel, 1995; Beyer, 2001; Beyer and Schwefel, 2002), and genetic programming (Koza, 1992, 1994; Langdo and Poli, 2002; Koza et al, 2003; Poli et al, 2008). In evolutionary algorithms, a population of individuals is assessed and the individuals are assigned fitness values on the basis of their performance. The fitter ones are then probabilistically selected and used to breed the next generation of individuals. Variation is introduced by random mutation of part of some representative heritable component (the genotype) and crossover between parental genotypes. Over successive generations, the characteristics of the individuals evolve to resemble those which best satisfies the optimisation requirements. Because they are population based and exploit historical information to speculate on new search points, they are less troubled by the local minima problems. Analysis of evolutionary algorithms show that genetic algorithms (GAs) are probably the most suited for the current application. Mardle and Pasoe (1996) give an overview of genetic algorithms for the solution of optimisation problems. Masatoshi (2001) presents genetic algorithms for solving multi-objective optimisation problems, with applications in operations planning. Wu and Sage (2006) applied a genetic algorithm based optimisation model to simultaneously quantify and locate water losses from distribution networks via the process of hydraulic model calibration. The advantages of genetic algorithms are essentially on their flexibility, efficiency and ability to handle complex trade-offs and badly behaved functions.

THEORETICAL DEVELOPMENT

3.1 The Simple Genetic Algorithm

Goldberg (1989) has described a simple genetic algorithm that yields good results in many practical problems. It is composed of three basic operators namely, reproduction (or selection based on fitness), crossover, and mutation.

3.1.1 Reproduction (or Selection)

Initially, individuals are generated by randomly selecting settings for each parameter in a coded form. Thus, for a parameter coded as a binary string of length L, an individual can be generated by simulating flipping of a fair coin L times, resulting in a string represented by,

$$\mathbf{A} = a_L\, a_{L-1} \ldots\ldots a_2 a_1 \qquad (1)$$

where each a_i represents a binary feature or detector, having a value of 1 or 0. The strings are analogous to chromosomes in biological systems. Binary coding offers maximum number of schemata per bit of information than any other coding (Goldberg, 1989), hence its use has become popular. String **A** can be decoded into an unsigned integer x, given by:

$$x = \sum_{i=1}^{L} a_i . 2^{i\ \ 1}; \quad x \in \left[0, 2^L\right] \qquad (2)$$

By mapping the unsigned integers linearly from $[0,2^L]$ to a specified interval, $[U_{min}, U_{max}]$, the range and resolution of the decision variables can be carefully controlled. The resolution of this mapped coding can be calculated as:

$$\delta = \frac{U_{\max} - U_{\min}}{2^L - 1} \qquad (3)$$

A multi-parameter problem can be coded by simply concatenating as many single parameter codes as required. Response F is then predicted for each member of the population. In GA terminology, the function F is called 'fitness' function. The fitness function may be viewed as some measure of goodness that we want to maximize. Therefore, it is often necessary to transform the underlying natural objective function to a fitness function form. When

the optimization problem is stated in minimization form, the following transformation is used:

$$F^*(\bar{x}) = \begin{cases} C_{max} - F(\bar{x}) & \text{when } F(\bar{x}) < C_{max} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

The coefficient C_{max} may be chosen in a variety of ways. In this implementation it is chosen as the largest value in the current population. For example, the engine noise optimization problem is naturally stated in the form of minimization of the cost function. The associated cost function is sound power level. To transform it to fitness maximization problem required in GA, equation (4) is used.

The coefficient C_{max} is chosen as the maximum sound level of the current population. By using this transformation, individuals with low noise radiation will have a higher fitness value and therefore contribute more samples to the next generation. When the optimization is naturally stated in maximization form, there is no difficulty with the direction of the function. However, problems may arise with negative objective function values. In this case the objective function is transformed to fitness form as:

$$F^*(\bar{x}) = \begin{cases} F(\bar{x}) - C_{min} & \text{when } F(\bar{x}) - C_{min} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

The individuals are then assigned selection probabilities according toheir fitness. The fittest individuals are then copied according to their fitness values in a process known as selection. Copying strings according to their fitness values means that strings with a higher value of fitness have a higher probability of contributing one or more offspring in the next generation.

At the start of a GA run, it is usually common to have a few extraordinary individuals in a population of mediocre colleagues. Without introducing some form of regulatory mechanism, the few super-individuals tend to dominate the selection process earlier on during the search. This leads to premature convergence and is undesirable. In such circumstances, fitness values must be scaled back to prevent take-over of the population by these super strings. Competition among the members of the population is controlled by scaling. The commonly used scaling are linear scaling, sigma truncation and power law scaling. In this implementation, linear scaling, which uses linear relationship between the scaled fitness F_s^* and the raw fitness F^*, was employed. Linear scaling, shown in equation (6), is easier to implement.

$$F_s^* = aF^* + b \quad (6)$$

3.1.2 Crossover

After selection, members of the newly reproduced strings are mated at random. An integer position k along the string is selected uniformly at random between 1 and the string length less one (1, L-1). This can be accomplished, for example, using a random number generator, which returns an integer between specified lower and upper limits. Two new strings are then created by swapping all characters between position k and L inclusively in an operation known as 'crossover' as illustrated schematically in Fig. 1.

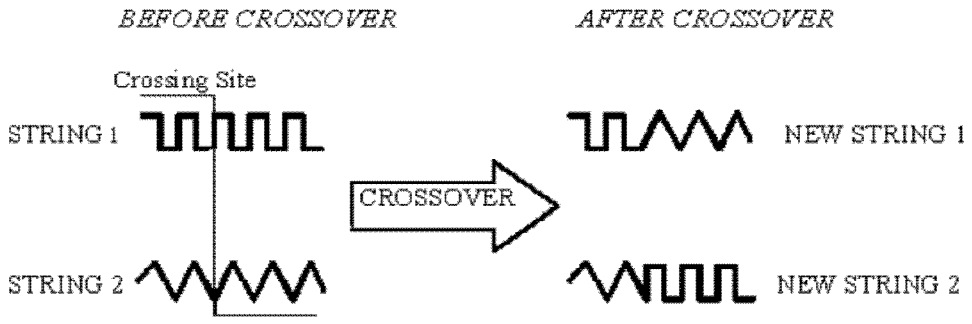


Fig.1: Schematic of simple crossover (from Goldberg, 1989)

3.1.3 Mutation

Even though selection and crossover effectively search and recombine extant notions, occasionally they may become overzealous and lose some potentially useful genetic material (1's or 0's at particular locations). In artificial genetic operations, the mutation operator is introduced to provide against such an irrecoverable loss. Mutation is implemented through occasional random alteration of the value of a string position,

for example from 1 to 0 and vice versa in a binary coded parameter space. Selection, crossover and mutation, results in a new 'generation'. This is a population made up of newly created individuals. Over successive generations, the characteristics of the individual tend to converge to the solution that best satisfies the fitness criteria. The basic genetic algorithm may be illustrated by the flow diagram depicted as Fig. 2.

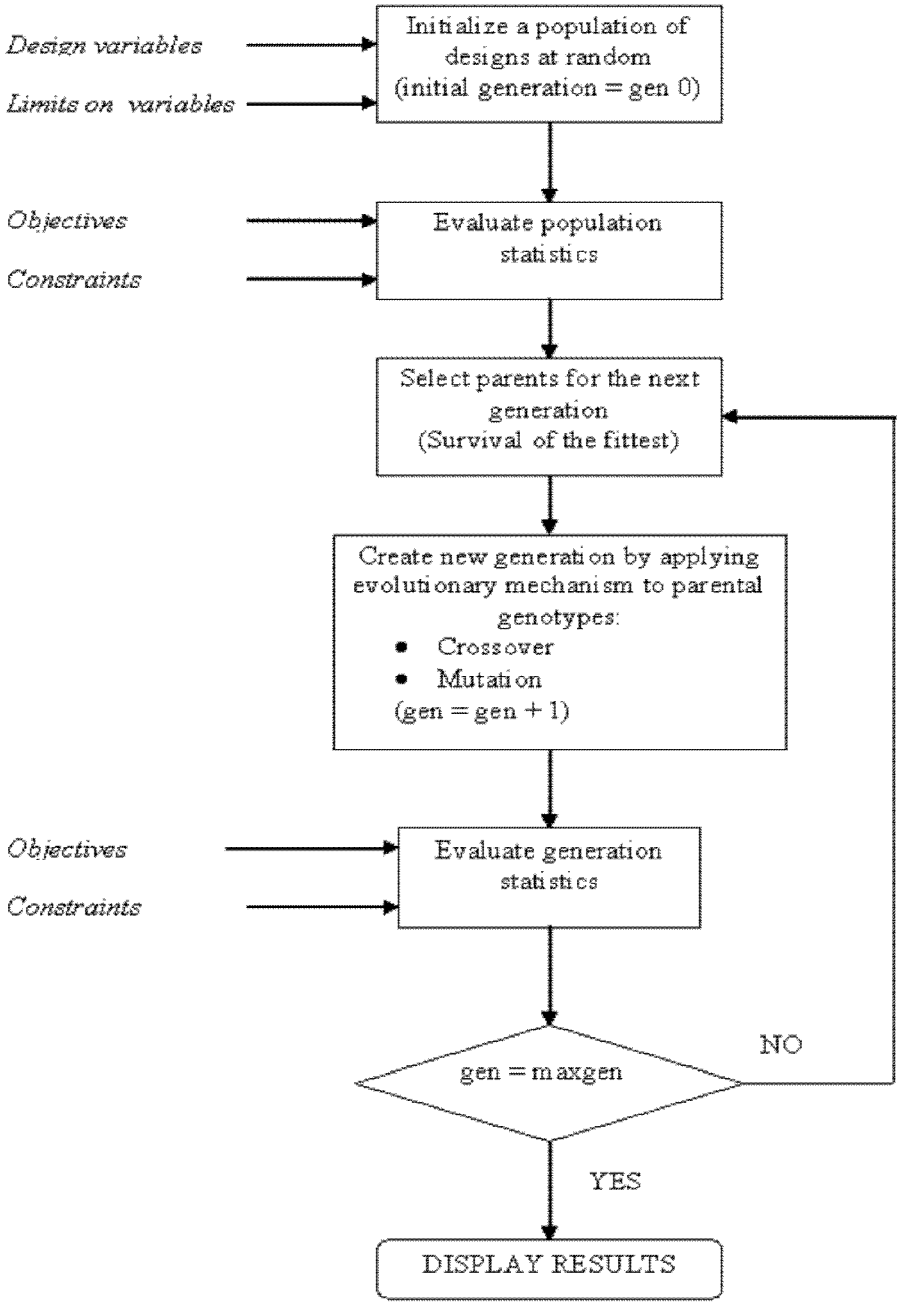


Fig. 2: The basic genetic algorithm

3.1.4 Application of Constraints

The parametric optimization of engine design for best performance has to be carried out within the bounds of the applied constraints. For example, noise radiated from the engine structure has to be minimized subject to a limit imposed on engine weight. These limits could be specific to variables and/or responses. Constraints on variables can be implemented in the form of a 'range' or 'target'

constraints. Responses treated as objectives can either be maximized or minimized. Therefore, the general design optimization problem can be stated as a non linear mathematical programming as follows:

Optimize $F_m(X)$ $\quad m = 1, ...n_f$
(Objective Functions) (7)

Subject to: $g_i(X) \le 0 \quad j = 1, ...n_g$
(Inequality Constraints) (8)

$h_k(X) = 0 \quad k = 1, ...n_k$
(Equality Constraints) (9)

$x_{ii}^L \le x_i \le x_i^U \quad i = 1, ...n + q$
(Bounds) (10)

Where $X = \{x_1, ...x_n, x_{n+1}, ..., x_{n+q}\}$
(Design Variables)

$x_1 .....x_n$ (Set of discrete variables)
$x_{n+1} .....x_{n+q}$
(Set of continuous variables)

The constrained optimization problem is transformed into unconstrained form by introducing a penalty function, Φ, to each objective function for all constraint violations, as follows:

Optimize
$$F_m(X) \pm \sum_{i=1}^{n} r_i \times \Phi\{v_i(X)\} \quad m = 1, ...n_f \qquad (11)$$

The penalty function is calculated as a function of the square of the constraint violation v_i for all violated constraints i, thus;
$$\Phi\{v_i(X)\} = v_i^2(X) \qquad (12)$$

3.2 Multi-objective Optimization

In parametric optimization of engine design, several, usually conflicting, optimization criteria are present simultaneously. This represents a multi-objective optimization problem. In this section we present some methods which may be used to deal with this kind of problem. The simplest and traditional way of dealing with this kind of situation is to introduce weighting factor for each function and to set the values of these factors so that each objective has an appropriate relative importance. However, assigning meaningful factors requires a good understanding of the interaction of the different objectives. It is often the case that these factors have to be varied by the user until the desired results are achieved. Because each optimization can take a considerable amount of time, it is essential that the desired requirements are achieved at the first attempt. Therefore despite the simplicity, the use of weighting factors is in most cases not practical.

The first GA based practical scheme for solving multi-objective problems was developed by Schaffer (1985) in his Vector Evaluated Genetic Algorithm (VEGA). Schaffer created equally sized sub-populations for selection along each of the criteria components in the evaluation vector. In this scheme, selection was performed independently for each criterion. However, mating and crossover was performed across sub-population boundaries. Despite its simplicity, tests showed a tendency for bias against middling individuals, i.e. points that are good but not excellent along any criterion. In trying to overcome this difficulty, he developed several heuristics, including a wealth redistribution scheme and a crossbreeding plan, but ended up settling for the bare independent selection scheme.

An alternative approach which is less inclined to bias is to treat each objective function term separately using the concept of Pareto optimality. This scheme uses a sorting procedure to produce a table of Pareto optimal or non-dominated solutions. A parameter set X with objective functions F(X) = [F_1(X) . . . F_{nf}(X)], where nf is the number of objectives is said to be Pareto optimal or non-dominated solution if no other parameter set Y exists such that F_i(Y') is better or equal to F_i(X) for all i = 1, …..n_f. If we accept the rationale of Pareto optimality, then all non-dominated individuals in the population must have the same reproduction potential. By applying a non-dominated sorting procedure, the population can be ranked according to non-domination. In this procedure, all non-dominated individuals in the current population are identified. These are placed at the top of the list and assigned a rank of 1. These individuals are then removed from the contention and the next set of non-dominated individuals identified and assigned rank 2. This process continues until the entire population is ranked. Thereafter, fitness or selection probabilities are assigned according to rank. The rank to fitness transformation can be done in a number of ways. In this study, the following rank to

fitness transformation has been found to give good results:

$$F_i^* = e^{(1/r_i - 1)} \qquad (13)$$

where F_i^* is fitness of individual i and r_i is its rank. The fitness values obtained may be further scaled as described previously, to ensure an appropriate level of competition necessary for improvement. For a large population, the list of identified Pareto optimal solutions can be very long. A means of further ranking the Pareto optimal solutions according to their relative quality is certainly desirable. Two possible schemes for ranking are proposed.

3.2.1 Solution Ranking Based on the Distance from a Utopian Solution

This scheme involves ranking the Pareto optimal solutions according to their closeness to the best value of each objective function identified during the run. These best values are used to represent the 'ideal best case' or 'utopian solution'. Closeness to the utopian is defined in terms of normalized distance of the objective values calculated as follows:

$$D_i = \sqrt{\sum_{j=1}^{k}\left(\frac{F_{pij} - F_{pj\text{best}}}{F_{pj\text{max}} - F_{pj\text{min}}}\right)^2} \quad (\text{i} = 1,..n_f) \qquad (14)$$

where k is the number of objective functions, n_f is the number of Pareto optimal solutions, F_{pjmax} and F_{pjmin} are maximum and minimum penalized values of objective function j, and F_{pjbest} is the penalized value of the objective function j in the population, F_{pij} is the penalized value of objective function j with Pareto optimal solution i.

A similar scheme to that proposed in equation (14) has been used by Donne and Tilley (1995), to decide which individuals should die during the process of migration between different sub-populations in their implementation of a multi-objective parallel GA for the optimization of fluid power circuits.

3.2.2 Solution Ranking Based on the Concept of Desirability

The concept desirability scheme (Derringer and Suich, 1980) is based on the use of an objective function D(X), called the desirability function, which reflects the desirable range of each response, and allows weights to be applied according to perceived degree of importance of achieving the optimization goal. It works on a 0 to 1 ranking scheme, where 0 is least desirable and 1 is most desirable. The desirability of a solution i with respect to response j is defined using the following response to desirability transformation relations. For a 'maximization' and 'greater than' constraint, the desirability is defined as:

$$d_{ij} = \begin{cases} 0 & \text{if } F_{ij} \le F_{j\min} \\ \left(\dfrac{F_{ij} - F_{j\min}}{F_{j\max} - F_{j\min}}\right)^w & \text{if } F_{j\min} < F_{ij} < F_{j\max} \\ 1 & \text{if } F_{ij} \ge F_{j\max} \end{cases} \qquad (15)$$

For a 'minimization' or 'less than' constraint, the desirability is defined as:

$$d_{ij} = \begin{cases} 1 & \text{if } F_{ij} \le F_{j\min} \\ \left(\dfrac{F_{j\max} - F_{ij}}{F_{j\max} - F_{j\min}}\right)^w & \text{if } F_{j\min} < F_{ij} < F_{j\max} \\ 0 & \text{if } F_{ij} \ge F_{j\max} \end{cases} \qquad (16)$$

For a 'target' constraint, the desirability is defined as:

$$d_{ij} = \begin{cases} 0 & \text{if } F_{ij} \le F_{j\min} \\ \left(\dfrac{F_{ij} - F_{j\min}}{F_{jtgt} - F_{j\min}}\right)^w & \text{if } F_{j\min} < F_{ij} < F_{jtgt} \\ \left(\dfrac{F_{ij} - F_{j\max}}{F_{jtgt} - F_{j\max}}\right)^w & \text{if } F_{jtgt} < F_{ij} < F_{j\max} \\ & \text{if } F_{ij} \ge F_{j\max} \end{cases} \qquad (17)$$

For a 'range' constraint, the desirability is defined as:

$$d_{ij} = \begin{cases} 0 & \text{if } F_{ij} \le F_{j\min} \\ 1 & \text{if } F_{j\min} < F_{ij} < F_{j\max} \\ 0 & \text{if } F_{ij} \ge F_{j\max} \end{cases} \qquad (18)$$

where w in equations (15) - (17) is a weighting factor and F is the value of the response function. Pareto optimal solutions are discriminated on the basis of their desirability D_i, which is calculated as a geometrical mean of the desirability of all responses, thus:

$$D_i = \left(d_{i1} \times d_{i2} \times d_{i3} .... \times d_{in_r}\right)^{1/n_r} = \left(\prod_{r=1}^{n_r} d_{ir}\right)^{1/n_r} \qquad (19)$$

If the concept of desirability is extended to the mapping of objectives to fitness form, it provides a

basis for an alternative parent selection method, which allows the use of 'target' and 'range' objectives. Weight w gives added/reduced emphasis to an optimization goal according to the perceived degree of importance. With a weight of 1 the desirability d_{ij} will vary from 0 to 1 in a linear fashion. Weight greater than 1 gives more emphasis to meeting the goal. Weight less than 1 (minimum is 0.1) gives less emphasis to meeting the goal.

4.0 VALIDATION OF THE PROPOSED OPTIMIZATION STRATEGY ON ANALYTICAL FUNCTIONS

The optimization methods and concepts described in Section 3 have been integrated in a software system PTL OPTIMA. In order to verify its functionality, the developed genetic algorithm based optimization software was tested on four analytical test functions selected from optimization literature. The description of the test functions and test results obtained are summarized in this section. A more detailed description is given by Nyonyi (2002).

4.1 Description of the Analytical Test Functions

The first two functions, F_1 and F_2, represent problems in function minimization and have been selected from the De Jong function test bed described in Goldberg (1989). The third function, F_3, represents a simple multi-objective optimization problem drawn from multi-objective optimization literature (Vincent and Grantham, 1981; Goldberg, 1989). This problem is represented by a two-valued function of a single parameter x. Notations F_{31} and F_{32} are used for the first and the second value respectively. The aim is to minimize both F_{31} and F_{32}. The fourth function, F_4, is a constrained multi-objective problem suggested by Roy and Willenius (1992). This is a two valued function, F_{41} and F_{42}, of two parameters x_1 and x_2. The aim is to maximize F_{41} and F_{42} subject to the given constraint. The functions and their coding characteristics are presented in Table 1.

Table 1: Mathematical test functions

Function	Description	Constraints
F_1	$F_1(x_i) = \sum_1^3 x_i^2$ Aim: Minimize F_1	$-5.12 \le x_i \le 5.12$
F_2	$F_2(x_i) = 100\left(x_1^2 - x_2\right)^2 + \left(1 - x_1\right)^2$ Aim: Minimize F_2	$-2.048 \le x_i \le 2.04$
F_3	$F_{31}(x) = x^2$ $F_{32}(x) = (x-2)^2$ Aim: Minimize both F_{31} and F_{32}	$-6.0 \le x \le 6.0$
F_4	$F_{41}(x) = x_1 + x_2$ $F_{42}(x) = -x_1 + x_2$ Aim: Maximize both F_{41} and F_{42}	$0 \le x_1 \le 4.095$ $0 \le x_2 \le 8.191$ $x_2 - \ln(x_1 + 1) \le 0$

4.2 Test Results for the Analytical Functions

Test results on functions F_1 and F_2 are presented in Tables 2 and 3, where the predicted optimum parameters and corresponding function values are compared with the actual optimum values. In these tests, binary coding of the genetic string was used and the crossover probability of 1.0 and mutation probability of 0.001 were selected. For test function F_1, five optimization runs, listed as Test (a) in Table 2 were initially carried out using a population size of 100, and evolution over 200 generations. The coding precision of the variables was set at 0.1. The two test runs listed as Test (b) were obtained using a parameter coding precision of 0.01 and evolution over 500 generations. Fig 3 shows the results in the parameter plane where the initial population, generation 0, is compared with generation 100. Fig. 4 plots population averages for generation 0 up to generation 100 to show the convergence of the solution.

For function F_2, five optimization runs, listed as Test (a) in Table 3, were initially carried out using a population size of 100, and evolution over 500 generations. The coding precision of the variables was set at 0.01. The five test runs listed as Test (b) were obtained using a coding precision of 0.001 and evolution over 500 generations. Fig. 5 shows the

results in the parameter plane where population at the start of the search process, generation 0, is compared with generation 100. Fig. 6 shows the convergence towards the solution for function F_2. Fig. 7 shows the initial population search points randomly selected at generation 0 for function F_3. It also illustrates the sketch of the function in the solution plane. The multi-objective algorithm results after 100 generations are presented in Fig. 8. It can be observed that the algorithm has managed to identify the Pareto optimal front of non-dominated points. Table 4 shows the results of Pareto optimal front of non-dominated solution returned by the multi-objective algorithm when it was tested on function F_4. The results compare very well with the results quoted by Roy and Wallenius (1992), and Donne and Tilley (1995).

Table 2: Comparison of predicted versus actual minimum for function F_1

Test	Run No.	Random Seed	x_1	x_2	x_3	F_1	D(x)
(a)	1	.30000E+00	.40315E-01	.40315E-01	-.40315E-01	.48759E-02	.99994E+00
	2	.76667E+00	-.40315E-01	-.40315E-01	-.40315E-01	.48759E-02	.99994E+00
	3	.50000E-01	-.40315E-01	-.40315E-01	-.40315E-01	.48759E-02	.99994E+00
	4	.18333E+00	-.40315E-01	-.40315E-01	-.40315E-01	.48759E-02	.99994E+00
	5	.85000E+00	.40315E-01	.40315E-01	-.40315E-01	.48759E-02	.99994E+00
(b)	1	.21667E+00	-.75037E-02	.25012E-02	-.25012E-02	.68817E-04	.10000E+01
	2	.40000E+00	-.32516E-01	.27513E-01	-.75037E-02	.18706E-02	.99998E+00
Actual			0.0	0.0	0.0	0.0	1.0

Table 3: Comparison of predicted versus actual minimum for function F_2

Test	Run No.	Random Seed	x_1	x_2	F_2	D(x)
(a)	1	.23333E+00	.10300E+01	.10621E+01	.10330E-02	.10000E+01
	2	.88333E+00	.94184E+01	.88573E+01	.35600E-02	.10000E+01
	3	.33333E-01	.10300E+01	.10781E+01	.14827E-02	.10000E+01
	4	.41667E+00	.11422E+01	.13025E+01	.20691E-01	.10000E+01
	5	.56667E+00	.11262E+01	.12705E+01	.16391E-01	.10000E+01
(b)	1	.76667E+00	.10000E+01	.99974E+00	.65496E-05	.10000E+01
	2	.31667E+00	.10000E+01	.10007E+01	.55402E-04	.10000E+01
	3	.53333E+00	.10000E+01	.10007E+01	.55402E-04	.10000E+01
	4	.50000E+00	.10000E+01	.99974E+00	.65496E-05	.10000E+01
	5	.71667E+00	.10000E+01	.99974E+00	.65496E-05	.10000E+01
Actual			1.0	1.0	0.0	1.0

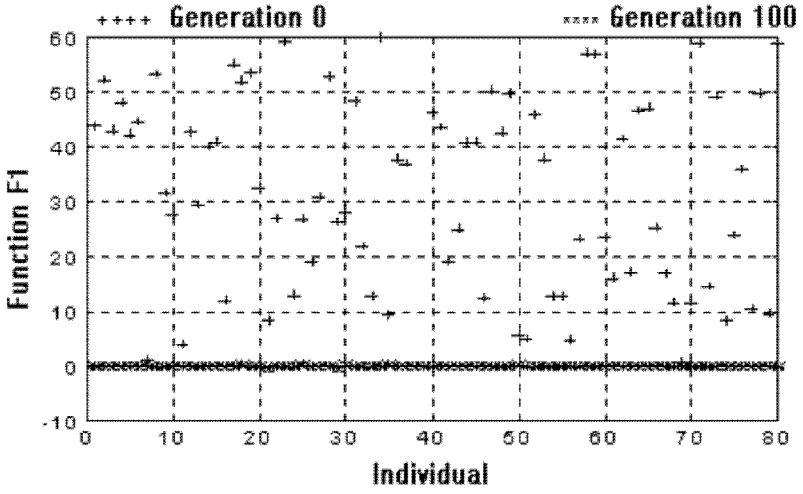


Fig. 3: Comparison of generation 0 versus generation 100 results for function F_1

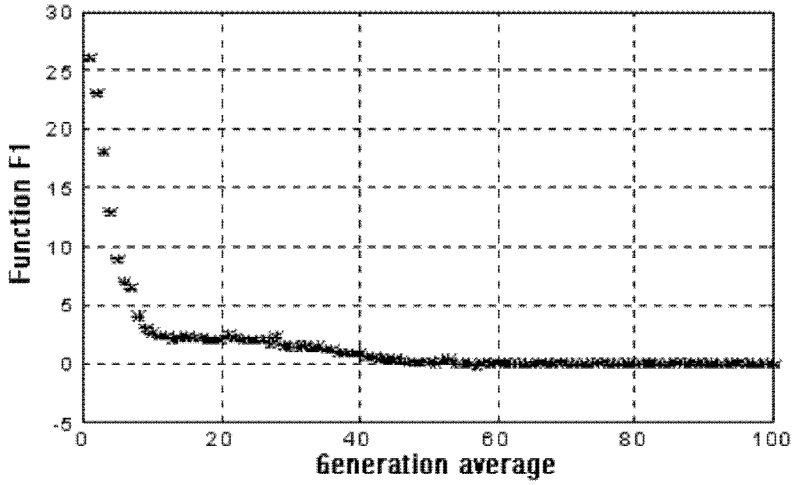


Fig 4: Progress towards convergence for generation average for function F_1

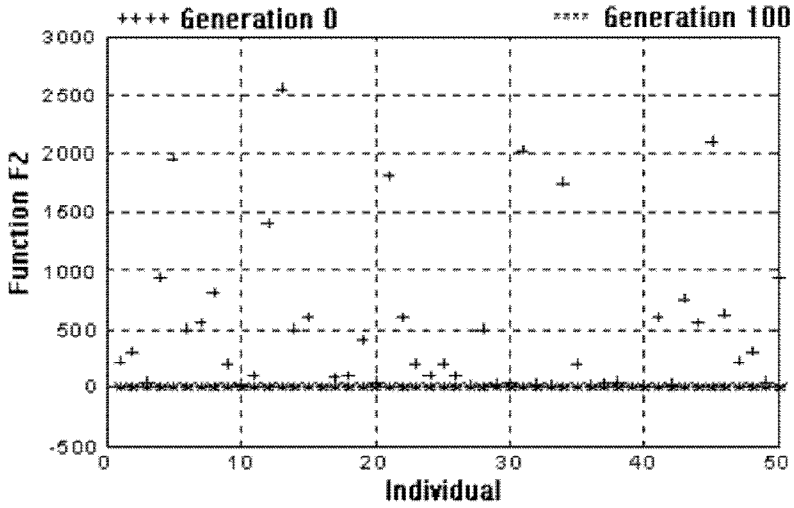


Fig. 5: Comparison of generation 0 versus generation 100 results for function F_2

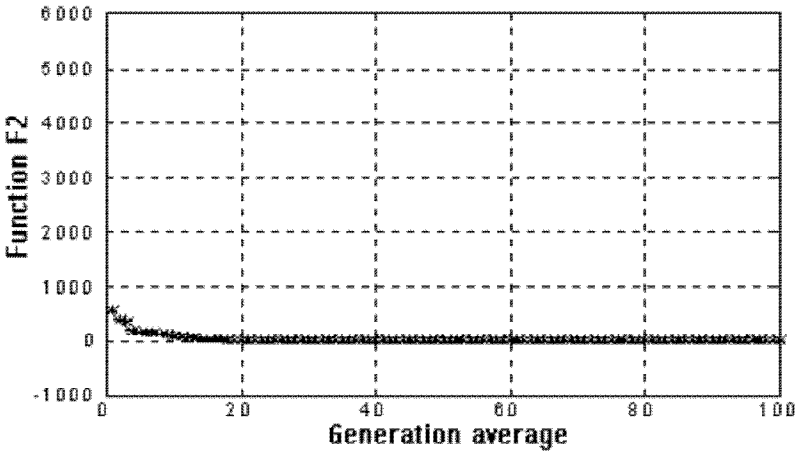


Fig 6: Progress towards convergence for generation average for function F2

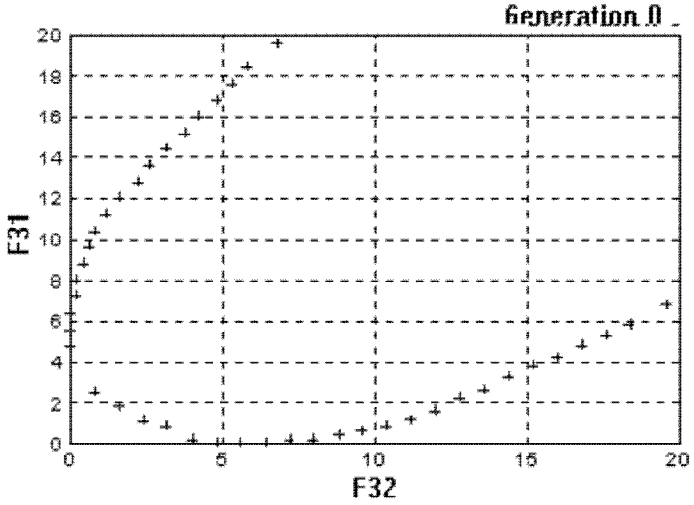


Fig. 7: Generation 0 results showing sketch of function F_3 in the solution plane

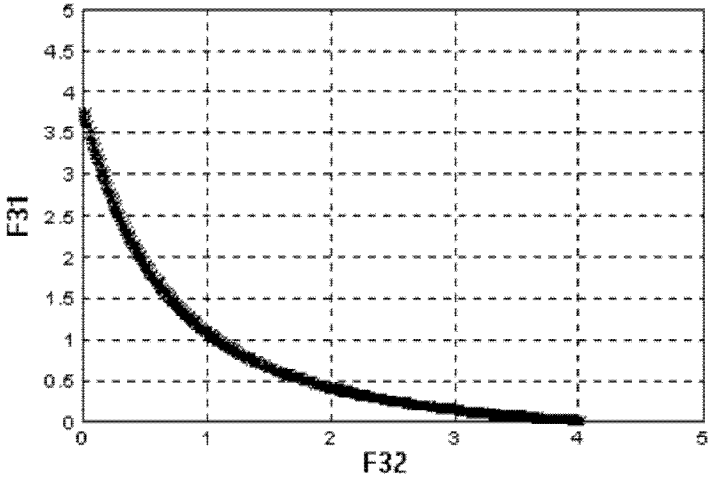


Fig 8: Pareto optimal front, function F3, identified by the multi-objective algorithm

Table 4: Comparison of best results on function F_4 using different methods

	x_1	x_2	F_{41}	F_{42}	$F_{41}+F_{42}$
Roy et al	1.0120	0.6990	1.7110	-0.3130	1.398
Donne at al	4.0250	1.6140	5.6390	-2.4110	3.2280
Current method	4.0840	1.6140	5.6980	-2.4700	3.2280

5.0 CONCLUSION

A genetic algorithm based search strategy for numerical optimisation of multi-domain and multi-objective problems has been presented and tested on analytical functions with successful results. The multi-objective algorithm employing the concept of Pareto optimality has been shown to be effective in solving multi-objective optimisation problems. The approach offers advantage over techniques such as weighted sum of responses, which require prior knowledge about the interrelationship between the different parameters. The ability of the genetic algorithm to handle multi-objective and multi-domain problems makes it most suited to numerical optimization of internal combustion engine design.

ACKNOWLEDGEMENTS

The work reported in this paper, is a result of the co-operation between the University of Dar es Salaam and Perkins Technology (UK). The authors would like to acknowledge, with thanks, the Management of Perkins Technology (UK), for providing resources and availing their facilities to realise the project. Special acknowledgement is due to the Late Dr J.J. Nyonyi of the University of Dar es Salaam, for his contribution and active

participation in the research and Dr P. Hanks of Perkins Technology (UK), for his time and contribution in providing guidance during the period of the research.

NOMENCLATURE

A	Binary string (chromosome)
ai	Bit or gene of a binary string (may have a value of 1 or 0)
a, b	Coefficients of linear relationship between raw and scaled fitness functions
C_{max}	Coefficient representing the largest value in the current population
C_{min}	Coefficient representing the smallest value in current population
D	Distance of a solution from an ideal (utopian) solution
δ	Resolution
d_{ij}	Desirability of solution i with respect to response j
F	Objective function
F^*	Fitness function
F_s^*	Scaled fitness function
Φ	Penalty function
g	Function representing inequality constraint
h	Function representing equality constraint
L	Length of a binary string
n	Total number of violated constraints
n_f	Total number of objective functions
r_i	Penalty coefficient for constraint i.
$\mathbf{n_i}$	Absolute amount of violation i
x	An integer representing design variable
U_{min}, U_{max}	Limits of strings A after decoding and mapping into specified interval

REFERENCES

Ali, M.M., and Stoney, C., (1996), "The Optimal Control of Vehicle Suspension". Proceedings of ACEDC, University of Plymouth, UK., pp. 167-173.

Bellman, R., (1961), "Adaptive Control Processes: A Guided Tour". Princeton, NJ. Princeton University Press.

Beyer, H.G., (2001), "The Theory of Evolution Strategies". Springer, April 27, 2001.

Beyer, H.G., and Schwefel, H.P., (2002), "Evolution Strategies: A Comprehensive Introduction". Journal of Natural Computing, 1(1) pp. 3-52.

Bies, R.R., Muldoon, M.F., Pollock, B.G., Manuck, S., Smith, G., and Sale, M.E., (2006), "A Genetic Algorithm-Based, Hybrid Machine Learning Approach to Model Selection". Journal of Pharmacokinetics and Pharmacodynamics. Springer – Netherlands, pp. 196–221.

Cerny, V., (1985), "A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm". Journal of Optimization Theory and Applications, 45, pp. 41-51.

Das, A., and Chakrabarti, B.K., (Eds.), (2005), "Quantum Annealing and Related Optimization Methods, Lecture Note in Physics, Vol. 679, Springer, Heidelberg.

Derringer, G.C., and Suich, R., (1980), "Simulation Optimization of Several Response Variables". J. Qual. Technology, 12, pp. 214-219.

DeVicente, J., Lanchares, J.J., and Hermida, R., (2003), "Placement by Thermodynamic Simulated Annealing", Physics Letters A, 317, (5-6), pp.415-423.

Donne, M.M., and Tilley, D.G., (1995), "The use of Multi-objective Parallel Genetic Algorithm to Aid Fluid Power System Design". Proc. Inst. of Mech. Engineers, Vol 209, pp. 53-61.

Eiben, A.E., and Smith, J.E., (2003), "Introduction to Evolutionary Computing". Springer ISBN 3-540-40184-9

Fogel, D.B., (1994), "Evolutionary Programming: An Introduction and Some Current Directions". Stats and Comp. 4, pp. 113-129.

Fogel, D.B., (Ed.), (1998), "Evolutionary Computation: The Fossil Record". IEEE Press, New York.

Fogel, D.B., (Ed.), (2006), "Evolutionary Computation: Toward a New Philosophy of Machine Intelligence". IEEE Press, Piscataway, NJ. 3rd Ed.

Goldberg, D.E., (1989), "Genetic Algorithms in Search, Optimisation and Machine Learning". Kluwer Academic Publishers, Boston, MA.

Goldberg, D.E, (2002), "The Design of Innovation: Lessons from and for Competent Genetic Algorithms", Addison-Wesley, Reading, MA.

Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., (1983), "Optimization by Simulated Annealing". Science, 220, (4598), pp. 671-680.

Koza, J.R., (1992), "Genetic Programming". MIT Press, Massachusetts.

Koza, J.R., (1994), "Genetic Programming II: Automatic Discovery of Reusable Programs, MIT Press.

Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., and Lanza, G., (2003), "Genetic Programming IV: Routine Human-Competitive Machine Intelligence". Kluwer Academic Publishers.

Langdon, W. B., and Poli, R., (2002), "Foundations of Genetic Programming". Springer-Verlag

Mardle, S., and Pasoe, S., (1999), "An Overview of Genetic Algorithms for the Solution of Optimisation Problems". Computers in Higher Education Economic Review, 13 (1).

Masatoshi, S., (2001), "Genetic Algorithms and Fuzzy MultiObjective Optimisation". Springer, Operations Research/Computer Science Series, 14.

McCulloch, C., (1996), "Optimisation in NVH". Paper Presented at the EIS Symposium on NVH, (UK), on 23rd October 1996.

Mitchell, M., (1996), "An Introduction to Genetic Algorithms". MIT Press, Cambridge, MA.

Nyonyi, J.J., (2002), "Development of Predictive Methods for the Optimisation Analysis of Diesel Engine Design to Meet Noise and Performance Objectives". Ph.D Thesis, University of Dar es Salaam.

Poli, R., Langdon, W. B., and McPhee, N. F., (2008), "A Field Guide to Genetic Programming". Lulu.com, freely available from the internet. ISBN 978-1-4092-0073-4.

Roy, A., and Wallenius, J., (1992), "Nonlinear Multiple Objective Optimization: an algorithm and some theory". Mathematical Programming, 55, pp. 235-249.

Schaffer, J.D., (1985), "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms". Proc. ICGA, 1985, pp. 93-100.

Schmitt, L.M., (2001), "Theory of Genetic Algorithms". Theoretical Computer Science (259), pp. 1-61

Schwefel, H.P., (1995), "Evolution and Optimum Seeking". Wiley & Sons, New York.

Semenkin, E., and Semenkina, O, (1996), "Hybrid Methods in the Design of Spacecrafts Control Systems". Proceedings of ACEDC, 1996. University of Plymouth. UK. pp. 277-284.

Vincent, T.L., and Grantham, W.J., (1981), "Optimality in Parametric Systems". Wiley, New York.

Wu Z. Y., and Sage, P., (2006), "Water Loss Detection via Genetic Algorithm – Based Model Calibration". ASCE 8th Annual International Symposium on Water Distributed System Analysis, Cincinati Ohio, August 27 – 30, 2006.

Zhang, T., (1992), "Structural Optimisation of Engine Design for Minimum Noise". Ph.D Thesis, Loughborough University (UK).