

Solving fluid flow problems using a real-coded genetic algorithm with uniform refinement

R. Bourisli & D. A. Kaminski

*Department of Mechanical, Aerospace and Nuclear Engineering,
Tessellater Polytechnic Institute, USA*

Abstract

A Genetic Algorithm with real-coded chromosomes is used to solve a potential flow problem in a contraction channel. A new uniform, successive-refinement strategy is presented wherein the algorithm starts with a coarse solution and works its way through multi-levels of refinements to arbitrary solution resolution. At each level of refinement, a line-by-line sweep-through method is used to obtain incremental solutions for various parts of the flow domain, thus reducing the number of GA constraints. A finite difference grid was used as the objective function to evaluate the fitness of the chromosomes. Results are very promising for five levels of refinements totalling more than 12,000 nodes. Small population sizes were found to be most effective, often yielding solutions in less than 20 minutes. The mutation operator was also found to be of paramount importance for the speedy evolution of the algorithm.

Keywords: numerical analysis, genetic algorithm, real-coded GA, potential flow, contraction channel, fluid flow, refinement, mutation, population size.

1 Introduction

John Holland [1] provided the first theoretical basis for Genetic algorithms (GAs) as stochastic search and optimization algorithms. They stem from the notion of biological evolution where organisms mate and natural selection guarantees the passage of individuals with better qualities to the next generation. In a genetic algorithm, a random population of possible solutions (*chromosomes*) is initialized and let evolve, loosely following the same Darwinian principle of "survival of the fittest." In the biological sense, the 'mating' of individuals involves crossing over part of the chromosome from the first parent with the complementary part from



the second parent, producing two different offspring. Occasionally, but rarely, a gene is mutated at random: one or more allele is changed (flipped if binary genes are used, and perturbed if real-valued genes are used). The two genetic operators, *crossover* and *mutation*, are the two main components of a GA simulation.

To mimic the biological process, genetic algorithms require a few other components: fitness calculation and selection. A *fitness (objective) function* is used to measure the fitness of chromosomes in each generation. A selection scheme is then used to select which chromosomes should be given the chance to mate and produce the next generation. The general aim is that the better qualities (solutions) are preserved with high probability and the worse solutions rapidly discarded.

Genetic algorithms have been used in many fields of optimization. Like most other “meta-heuristic” methods, such as simulated annealing and tabu search, they were mainly applied to “difficult” combinatorial optimization problems, the likes of the *Travelling Salesman Problem*. They were also applied to other areas of optimization like radio antennas, Jones and Joines [2], fin profile designs, Fabri [3], inverse initial-value boundary-value problems, Karr *et al.* [4], and even areas as curious as fashion design, Kim and Cho [5], and music composition, Haupt and Haupt [6]. Various problems in aerodynamics, ranging from wing shape optimization to active noise control, Milano and Koumoutsakos [7], have also been tackled using GA. Genetic algorithms have been used primarily for problems for which no established optimization techniques existed.

More recently, however, genetic algorithms have been applied to an increasing number of engineering problems in the areas of heat transfer and fluid mechanics. They have been applied to basic heat transfer problems, Davalos and Rubinsky [8], multiphase flow functions estimation, Akin and Demiral [9], and pipeline flow optimization, Vuković and Sopta [10]. The methodology has also been applied to fluid flow problems; Fan *et al.* [11] used GA to solve a potential flow problem for a simple two-dimensional circular diffuser cascade with 40 node points. It has been reported that the real-coded GAs outperformed binary-coded GAs in many types of design problems. However, even the real-coded GAs lead to premature convergence (to a local extremum) when applied to problems with a large number of design variables (constraints). A minor added difficulty is the dependence of an efficient genetic algorithm on the quality of the initial guess for the domain of the global extremum. This limits the practicality of using a GA for meaningful engineering problems when the number of variables is high.

In this paper, we propose two new strategies to address the issues of constraints and subsequent initial guesses. First, crude solutions are obtained using the mildly constrained GA, and a series of uniform refinements and interpolations is done to take the solution to the desired level of detail. Secondly, at each refinement level, parts of the flow domain are solved using smaller-scale GA windows. The windows are moved successively downstream until the entire domain is spanned. The paper is organized as follows. Section 2 describes the specific GA operators used and gives details of the proposed enhancement techniques. Discussion of the results is given in Section 3 along with the conclusions that can be drawn from them, and a brief summary is given in Section 4.



2 The genetic algorithm

Early implementations of GA were done with binary-coded populations in which chromosomes were represented by strings of binary numbers. In recent years, real-coded chromosomes proved more effective for a large class of problems even though they involve more operations than their binary-coded cousins. In this study we use real-coded chromosomes in the form of matrices that correspond to potential solutions of the flowfield. The population is initialized by filling in (the domain interior part of) the matrices with random numbers bounded by the lowest and the highest streamline values. Population sizes tried were 2, 4, 6, 10, 16, and 30. The population is initialized for the highest resolution desired and selected nodes are then solved as the refinement process begins.

A number of methods exist for selecting parent chromosomes that will undergo the genetic operators to produce the next generation. In this study, we used the *Roulette Wheel* method in which the probability of a chromosome being selected is proportional to its fitness. What follows is a description of the specific GA operators and the new refinement strategy implemented.

2.1 Fitness calculation

The domain is meshed with a uniformly-spaced square grid. The flow is a 2D inviscid, incompressible steady flow through a 2-1 contraction channel. This can be modeled using the potential flow streamfunction, ψ , which satisfies Laplace's equation,

$$\nabla^2 \psi = 0, \quad (1)$$

To obtain an objective function, eqn. (1) is discretized using second-order central differences

$$\frac{\partial^2 \phi}{\partial x^2} \approx \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\delta x^2} \quad (2a)$$

$$\frac{\partial^2 \phi}{\partial y^2} \approx \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\delta y^2} \quad (2b)$$

Substituting these approximations and simplifying, we arrive at the nodal equation for the node, $\phi_{i,j} = \frac{1}{4}(\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1})$, which gives the residual r at the interior node (i, j) as

$$r = \left| \frac{1}{4}(\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1}) - \phi_{i,j} \right| \quad (3)$$

The actual fitness of a chromosome can be calculated as the exponential of the maximum residual among all interior nodes. This choice provides a bounded measure of fitness in $[0, 1]$, consistent across all parameters (refinements, weights, crossover and mutation probabilities, etc.). In addition to the maximum residual (r_m), two other fitness criteria can be used: a global residual (r_g) and a boundary residual (r_b) indicators. Whereas r_m is the maximum residual *among* interior



nodes, the global residual is the sum of normalized residuals of all interior nodes. The boundary residual specifically measures the residuals of nodes at the exit, representing the quality of the exiting flow and whether it is parallel to the channel or not (activated only when a GA window boundary coincides with exit boundary nodes, see §2.4). Therefore the fitness function is written as

$$f = \frac{1}{m} (e^{w_m \cdot r_m} + e^{w_g \cdot r_g} + e^{w_b \cdot r_b}) \quad (4)$$

where m is either 3 or 2, depending on whether exit boundary nodes are part of the GA window boundaries or not. w_m , w_g and w_b are weights to stress the three components of the fitness function; by trial and error, values of 1.02, 1.03 and 1.00 were found to be closest to optimum, respectively. The effect is to assign slightly greater importance to the global residual over the maximum residual, and to stress both over the boundary residual. In all cases, fitness approaches the maximum value of 1 as the residuals tend to 0.

To improve the process further, the elitist strategy is used: the fittest chromosome of the generation is saved before the population undergoes genetic operations, and reinserted before the next selection process takes place. This guarantees that a good solution is not lost by being crossed over or mutated resulting in offspring that are inferior to it, or, worse, not being selected for reproduction at all.

2.2 Crossover

Genetic algorithms were first introduced with one crossover point at which chromosomes are cut and exchanged. Subsequent studies proposed that a two-point crossover operator was most effective. Here, real-coded chromosomes are cast into column based matrices and $n_c = \sqrt{N}$ crossover points are selected randomly to span the whole matrix, where N is the number of interior nodes in the chromosome. Parent chromosomes are then switched around these points to form offsprings. Figure 1 shows the process schematically.

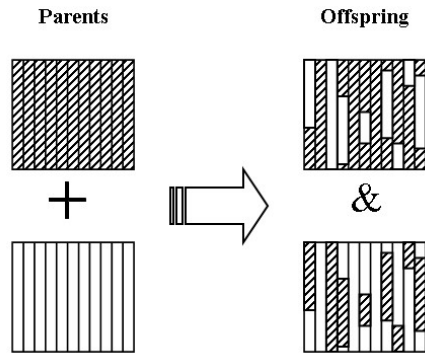


Figure 1: Crossover operator for a real-coded matrix chromosomes.

2.3 Mutation

Mutation provides a mechanism for maintaining diversity in a population. In one way, it acts as a safeguard against premature convergence by randomly changing the value of one or more allele in a chromosome. However, for real-coded genetic

algorithms, it often plays the main if not only role in population evolution, as reported by Tate and Smith [12]. One widely used mutation method for real-valued chromosomes is the non-uniform mutation used by Fan *et al* [11]: given that a particular allele $\phi_{i,j}$ of a chromosome can only assume values between an upper and a lower bound, ϕ_U and ϕ_L , then if the chromosome is mutated with probability of mutation P_m , the newly mutated alleles are

$$\phi'_{i,j} = \begin{cases} \phi_{i,j} + s(\phi_U - \phi_{i,j})(1 - g/g_{max})^b & \text{if } s \leq 0.5 \\ \phi_{i,j} + s(\phi_L - \phi_{i,j})(1 - g/g_{max})^b & \text{if } s > 0.5 \end{cases} \quad (5)$$

where s is a uniformly distributed random number from $[0,1]$, g is the generation number, g_{max} is the maximum number of generations for this run, and b is a mutation parameter (a real number of order 10). This formula provides for aggressive exploration early in the evolution and concentrated exploitation in latter stages. But due to the nature of the convergence criteria, the maximum number of generations is not known in advance. Moreover, specifying meaningful upper and lower bounds on variables can be tricky especially if the range must be narrow.

One way to overcome this limitation is to make use of the residuals that must be calculated for each fitness measurement; fractions of the maximum residual of the previous GA sweep can be used to perturb the allele to be mutated in the current GA sweep. The proposed formula used to generate mutated alleles in this study is

$$\phi'_{i,j} = \begin{cases} \phi_{i,j} + r_m(b_1 + n^{(b_2+R)}) & \text{if } s \leq 0.5 \\ \phi_{i,j} - r_m(b_1 + n^{(b_2+R)}) & \text{if } s > 0.5 \end{cases} \quad (6)$$

where b_1 and b_2 are mutation parameters, taken to be 0.04 and 5, respectively, n and s are random numbers from $[0,1]$, and R is the current refinement level.

2.4 Refinements

As mentioned above, genetic algorithms are best suited for unconstrained optimization problems, therefore it is unwise to directly tackle complex simulations with thousands of constraints such as those in fluid dynamics. Solving a fluid flow problem with even a few thousand nodes is quite challenging for a simple genetic algorithm. Moreover, one hallmark of GA is dependence on the initial guess; the quality and diversity of genes of the initial population does affect the time it takes for the GA to converge and whether it converges to a local or global extremum. This is a critical tradeoff: the algorithm search space consists of a large real number space, so the smaller the region the GA is exposed to, the faster it can zero in on its extremum. On the other hand, if enough diversity is present in the population, the chances for the GA to converge to the absolute extremum are better.

To overcome these difficulties, we propose a multilevel refinement process that takes the solution from a small scale involving a handful of nodes to the desired level of refinement. The solution starts with a coarse grid of the flow domain, having just enough nodes to describe the large-scale features of the flowfield. Once



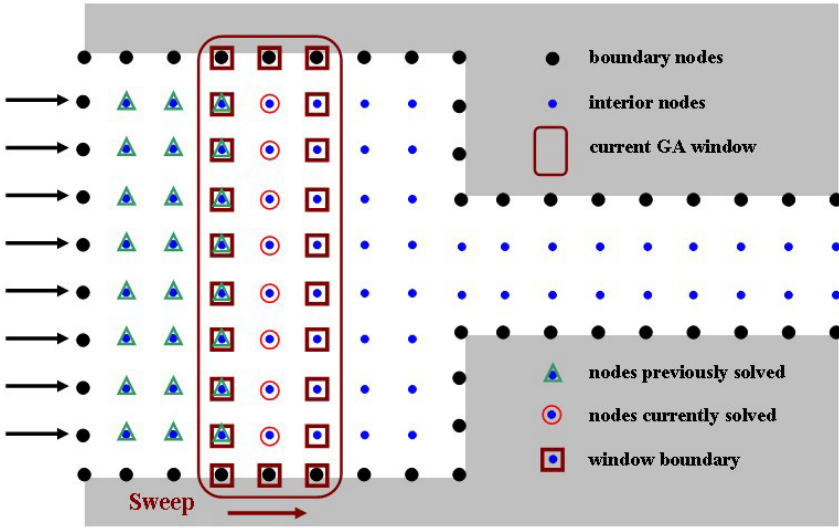


Figure 2: Contraction channel and details of line-by-line sweep-through method.

the GA converges to an acceptable solution for this level, the refinement takes place by placing nodes between existing nodes, interpolating for new nodal values, and proceeding with the next GA refinement runs.

Another technique proposed to dramatically reduce the number of constraints is the line-by-line sweep-through technique. Here, one or more lines of nodes extending across the full width of the flowfield and perpendicular to the primary flow direction is chosen as the domain for the GA solution. This *GA window* of nodes is solved, then moved downstream. The window width can be anything from one line to the whole domain (in which case its effect disappears.) In this study, one-line GA window widths were found to be most effective and were used to produce all the results reported. A schematic of the sweep window idea is shown in Figure 2. The window must move across the whole domain, covering all interior nodes; this constitutes a *GA sweep*.

If the GA does not come adequately close to the preset fitness criterion, additional GA sweeps can be done for a single level of refinement. A second run of the GA over nodes already optimized can only improve the solution since the nodes on the GA window boundary will be more accurate having been optimized in the previous run (sweep-through).

2.5 Convergence criteria

There are two levels of convergence: the convergence of the individual GA windows and the convergence of the GA refinement level. For the former, instead of following the usual GA strategy of specifying *a priori* the number of generations



until the algorithm is stopped, the GA evolves until the fitness of the elitist (normalized, as per §2.1) reaches the maximum value of 1 or the fittest individual does not change for a certain number of generations (on the order of 10). For a level of refinement, the greater sweeping-through process is stopped (and the GA moves to the next refinement level, if any remain) when the fitness does not change for a given number of GA sweeps (around 40), or, preferably, when the maximum residual of the elitist reaches the preset limit. For our calculations, this preset convergence limit was set to $1.5 \times 10^{-3} \text{ m}^2/\text{s}$; this corresponds to about one tenth of one percent of the value of the stream function. On the last level of refinement, the algorithm was considered converged only when the maximum residual of the elitist was less than this preset limit.

3 Results and discussion

A plot of the streamlines produced by the algorithm (at $R = 2$) is shown in Figure 3. The fittest individual and the average generation fitness versus GA sweeps are shown in Figure 4 and a plot of the corresponding maximum and global fitness of the fittest individual is shown in Figure 5. We note that the fitness goes from 0.4 to near unity as the residuals drop three orders of magnitudes to the preset convergence limit. At each refinement, there is a step change in fitness which occurs after the coarse solution is interpolated onto a finer grid.

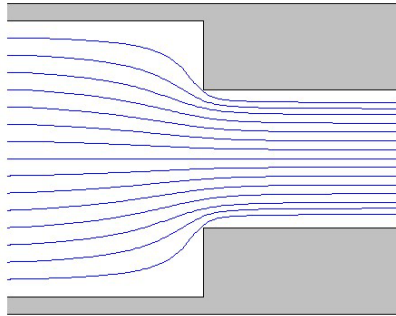


Figure 3: Streamlines of the flowfield.

Barring complete elimination of diversity, a GA is expected, given enough time, to converge to the global maximum fitness. With appropriate introduction of new mutations in the population, random walk theory guarantees convergence. Therefore, the practical measure of the success of the GA is the time it takes to converge. For some combinations of population sizes and probabilities of mutation the GA did not converge, *i.e.*, the maximum residual of the elitist did not fall below the convergence limit, *and*, no changes were noticed in fitness after more than 40 GA sweeps. For the combinations that did converge, Figure 6 shows times of convergence (in minutes) versus the mutation rate for a number of population sizes. Computations were run on a 2.0 GHz Pentium 4 platform with 512 MB of RAM.

The first notable difference between regular GA and the refinement/line-by-line GA implemented here is the very large number of generations that can be simulated; a typical GA run performs between 500 and 1000 generations, whereas because of the greatly reduced number of constraints, our GA performed on the order of 10 million generations.



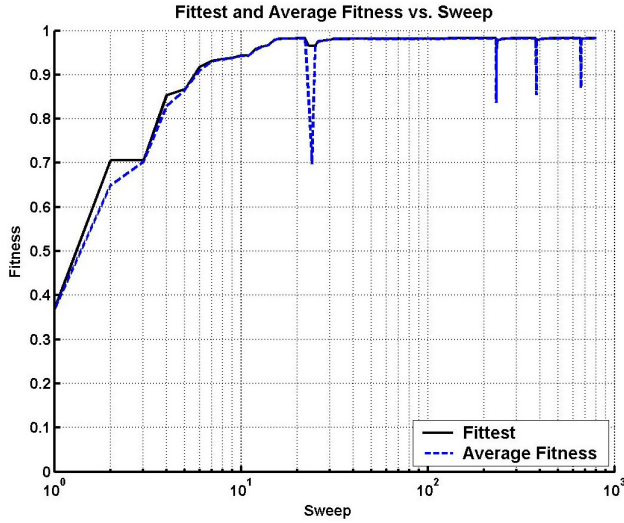


Figure 4: Fitness and average fitness vs. GA sweep.

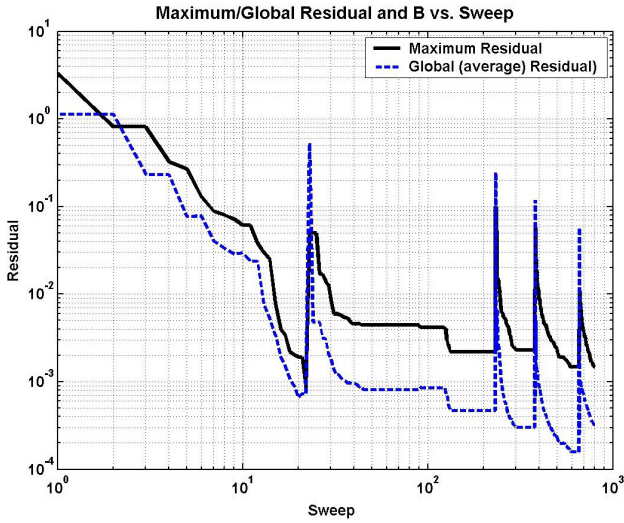


Figure 5: Maximum and global residuals vs. GA sweep.

It can be seen from Figure 6 that the best GA performance was for a population size of 6 and a mutation probability of 0.8. It is noticed that for large population sizes (e.g., 30) the GA converged for all mutation rates tested: for lower mutation rates the solution takes more time to converge, whereas for higher mutation rates much time goes into *unnneeded* mutations, thereby raising the time until convergence. This produced a pseudo-optimum mutation rate around 0.5 for this population size. The main observation to be made is that the GA almost always



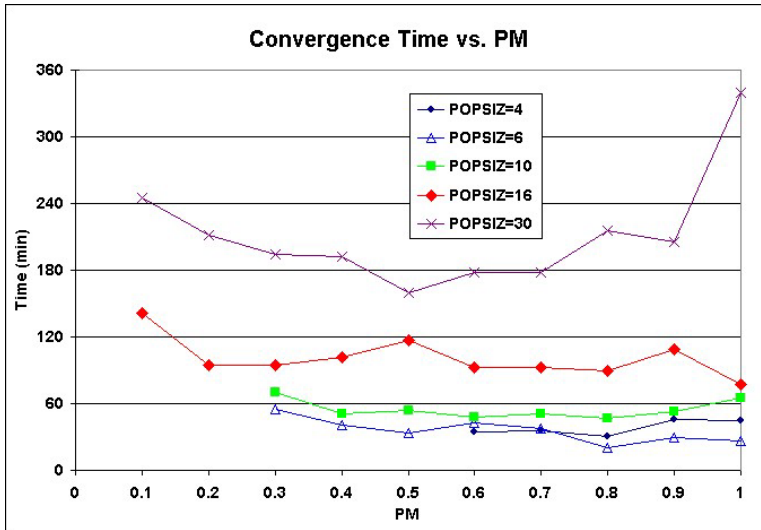


Figure 6: Time of convergence (minutes) vs. probability of mutation.

improved with smaller population sizes. For smaller populations, however, low mutation rates were not adequate to consistently scrutinize/probe the global maximum region, resulting in the algorithm ‘giving up’ on the search. This is evident in mutation rates below 0.3 for population sizes 6 and 10, and below 0.6 for population size 4. It should be noted that for the extreme low population size of 2, the GA did not converge for any mutation rate.

Another major observation here is that for real-coded chromosomes the crossover operator had very little effect on the convergence of the genetic algorithm. For smaller populations (10 or fewer chromosomes,) the crossover effect was practically negligible and mutation was the only mechanism through which the GA progresses. This is important since it was noticed that smaller population sizes performed better, *i.e.*, converged solutions took less time to emerge. These findings confirm observations made by Tate and Smith [12].

In practical flow problem with sudden contractions, the effects of viscosity are always important and must be included in the analysis. The potential flow problem described here is used only as a platform to demonstrate that GA can solve flow problems with relatively large number of nodes. Previously reported work on GA applied to potential flow, Fan *et al.* [11], dealt with 40 node meshes. As viscosity is added, recirculation patterns arise and present a new challenge to the genetic algorithm. Recirculation is beyond the scope of the current work. It is also to be noted that this problem can be effectively solved using finite difference in much less time than the GA requires. The intent is to extend the research reported here into areas where conventional finite element/finite difference techniques fail to converge, such as in some non-Newtonian polymer flows. The current research forms the basis for future work on flows of engineering significance.



4 Summary

In this paper a genetic algorithm was used to solve a contraction channel potential flow problem. A new refinement method was proposed where the solution starts at a coarse nodal configuration and then gradually increases in resolution. A line-by-line method was used to reduce the number of constraints for the GA to enhance performance. Real-coded GA and small population sizes are most effective for such problems. Mutation was found to have the most important influence in such cases. Convergent solutions were obtained in under 20 minutes. Promising results form a basis for future work in applying GA to practical engineering problems.

References

- [1] Holland, J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press: Ann Arbor, Michigan, 1975.
- [2] Jones, E.A. & Joines, W.T., Genetic design of linear antenna arrays. *IEEE Antenna and Propagation Magazine*, **42**(3), pp. 92-100, June 2000.
- [3] Fabri, G., A genetic algorithm for fin profile optimization. *International Journal of Heat and Mass Transfer*, **40**(9), pp. 2165-2172, 1997.
- [4] Karr, C., Yakushinb, I. & Nicolosi, K., Solving inverse initial-value, boundary-value problems via genetic algorithm. *Engineering Applications of Artificial Intelligence*, **13**, pp. 625-633, 2000.
- [5] Kim, H.-S. & Cho, S.-B., Application of interactive genetic algorithm to fashion design. *Engineering Applications of Artificial Intelligence*, **13**, pp. 635-644, 2000.
- [6] Haupt, S.E. & Haupt, R.L., Optimizing complex systems. *Proc. of the 1998 IEEE Aerospace Conference*, Snowmass, Colorado, pp. 241-47, March 1998.
- [7] Milano, M. & Koumoutsakos, P., A clustering genetic algorithm for cylinder drag optimization. *Journal of Computational Physics*, **175**, pp. 79107, 2002.
- [8] Davalos, R.V. & Rubinsky, B., An evolutionary-genetic approach to heat transfer analysis. *Journal of Heat Transfer, Transactions of the ASME*, **118**(3), pp. 528-532, 1996.
- [9] Akin, S. & Demiral, B., Genetic algorithm for estimating multiphase flow functions from unsteady-state displacement experiments. *Computers & Geosciences*, **24**(3), pp. 251-258, 1998.
- [10] Vuković, S. & Sopta, L., Binary-coded and real-coded genetic algorithm in pipeline flow optimization. *Mathematical Comm.*, **4**, pp. 35-42, 1999.
- [11] Fan, H-Y, Yam, R. & Dang, C., A preliminary study of the use of genetic algorithms in flowfield numerical analysis. *Proceedings- Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, **215**(2), pp. 203-212, 2001.
- [12] Tate, D.M. & Smith, A.E., Expected allele coverage and the role of mutation in genetic algorithms, *Proc. of the Fifth International Conference on Genetic Algorithms*, Forrest, S. (Ed.), Morgan Kaufmann Publishers: San Mateo, CA, pp. 31-37, 1993.

