Micro-Genetic Algorithms For Stationary And Non-Stationary Function Optimization

Kalmanje Krishnakumar

Kalmanje Krishnakumar, "Micro-Genetic Algorithms For Stationary And Non-Stationary Function Optimization," Proc. SPIE 1196, Intelligent Control and Adaptive Systems, (1 February 1990); doi: 10.1117/12.969927

Event: 1989 Symposium on Visual Communications, Image Processing, and Intelligent Robotics Systems, 1989, Philadelphia, PA, United States

MICRO-GENETIC ALGORITHMS FOR STATIONARY AND NON-STATIONARY FUNCTION OPTIMIZATION

Kalmanje Krishnakumar

Department of Aerospace Engineering
The University of Alabama, Tuscaloosa, Alabama 35486

ABSTRACT

Simple Genetic Algorithms (SGA) have been shown to be useful tools for many function optimization problems. One present drawback of SGA is the time penalty involved in evaluating the fitness functions (performance indices) for large populations, generation after generation. This paper explores a small population approach (coined as Micro-Genetic Algorithms--μGA) with some very simple genetic parameters. It is shown that μGA implementation reaches the near-optimal region much earlier than the SGA implementation. The superior performance of the μGA in the presence of multimodality and their merits in solving non-stationary function optimization problems are demonstrated.

1. INTRODUCTION

Genetic Algorithm (GA) has found a permanent place as an optimization algorithm for solving many difficult function optimization problems and as a discovery heuristic in several machine learning systems. The simplest form of a GA, namely SGA (Goldberg, 1988b), uses a binary coded genetic algorithm with the three basic GA operators (reproduction, crossover, and mutation) in a serial implementation. There are many optimization problems in science and engineering which can be solved using the direct implementation of the present structure of the SGA. These function optimization problems are the ones in which the function to be optimized are well defined and do not change faster than the time it takes for the SGA to reach the optimum. For all practical SGA implementation purposes, these problems can be classified as stationary function optimization problems. There are also many a number of problems where the function to be optimized are themselves evolving at a rate faster than the SGA can find an optimum, making the application of SGA meaningless. Problems of these types can be found in most of the real-world situations, including problems in aerospace, such as pursuit and evasion (function to be optimized is redefined depending on what the evader does), on-line aircraft trajectory optimization, and optimal control of an aircraft in wind shear (a problem of interest in this paper). Problems such as these can be classified as non-stationary function optimization problems.

In this paper, an investigation is conducted to implement a genetic algorithm with a very small population, and with some simple genetic parameters, in order to achieve fast turn around time from generation to generation evolution and associated function evaluation. Currently, many SGA users use population ranging in size from 30 to 200. The usual choice is based on earlier studies by De Jong (1975) and Grefenstette (1986), in which suggestions for optimal population choices based on parametric studies are presented. A recent investigation by Goldberg (1988a) showed that for serial implementation of binary coded GAs the optimal population choice is small. This result was obtained from optimizing for effective real-time schema processing in a given population. Goldberg also points out that simply taking a small population size and letting them converge is certainly not very useful, and proceeds to outline a scheme by which small population GA can be implemented.

This paper explores this small population (coined as Micro-Genetic Algorithms--μGA) approach for solving function optimization problems. First, the difference between the SGA and the μGA are established. Then an outline is presented as to the actual implementation of a suggested μGA structure for both stationary and non-stationary optimization problems. The μGA approach is then tested on three separate function optimization problems. The first two test functions represent stationary unimodal and multimodal functions, respectively. The third function represents a real-world non-stationary problem. The results show that the μGA is quicker than the big population approach in finding the near-optimal region, in spite of it's simplicity. Also, the implementation of the μGA for non-stationary optimization look highly feasible and rewarding.

2. SGA Vs. μGA

A simple genetic algorithm (SGA) works with a serially implemented binary coded population of size N, with a generation to generation evolution based on reproduction, with crossover (rate of crossover, C) and mutation (rate of mutation, M). The reproduction selection is conducted depending on the fitness of the individual strings compared to the population fitness average. The most popular selection strategy (S) currently in use are the roulette wheel selection and the stochastic remainder selection. In conjunction with the above mentioned selection procedures, the good strings can either be allowed to change (pure selection) or retained in to the next evolution (elite selection). Crossover rate controls the rate at which new structures are introduced, due to crossover, into the population. Mutation operator helps in preventing premature convergence and a low level of mutation rate serves to prevent a particular bit position from converging to a single value. SGA's performance can be measured in terms of on-line and off-line performance measures. The on-line performance measure is the average performance measure of all the tested structures over the course of the search. The off-line performance is the average of all the generation-based best performance structures.

The general choice of SGA parameters (N, C, M, S) are usually based on studies by De Jong (1975) and Grefenstette (1986). The usual choice of population size is based on the conception that bigger population relates to better schema processing, lesser chance of premature convergence, and better optimal results. The current choice of population size, based on the above mentioned studies, range from 30 to 200.

Just as in the SGA, the μGA works with binary coded populations and are implemented serially. The major difference between SGA and the μGA comes in the population choice. In the μGA structure proposed in this paper, the population size is fixed at N=5. It is a known fact that GAs generally do poor with very small population due to insufficient information processing and early convergence to non-optimal results. The key to success with small population was outlined by Goldberg (1988a) as follows:

1. Randomly generate a small population.

2. Perform genetic operations until nominal convergence (as measured by bit wise convergence or some other reasonable measure).

3. Generate a new population by transferring the best individuals of the converged population to the new population and then generating the remaining individuals randomly.

4. Go to step 2 and repeat.

Based on this approach a step by step procedure for the μGA implementation, utilized in this study, is presented below.

1. Select a population of size 5 either randomly or 4 randomly and 1 good string from any previous search.

2. Evaluate fitness and determine the best string. Label it as string 5 and carry it to the next generation (elitist strategy). In this way there is a guarantee that the information about good schema are not lost.

3. Choose the remaining four strings for reproduction (the best string also competes for a copy in the reproduction) based on a deterministic tournament selection strategy (Goldberg, 1989) . Since the population is so small, the law of averages do not hold good and the selection strategy is kept purely deterministic. In the tournament selection strategy, the

strings are grouped randomly and adjacent pairs are made to compete for the final four (Care should be taken to avoid two copies of the same string mating for the next generation).

4. Apply crossover with C=1. This is done to facilitate high order of schema processing. The mutation rate is kept to zero as it is clear that enough diversity is introduced after every convergence through new population of strings.

5. Check for nominal convergence (reasonable measure based on either genotype convergence or phenotype convergence). If converged go to step 1.

6. Go to step 2.

The assumption that maximum real-time schema mixing yields maximum performance is supported by empirical results and this maximum mixing is achieved in the μGA implementation by constant infusion of new schema at regular intervals. SGAs are known to reach premature convergence forcing the search process to rely entirely on the mutation operator to find the optimum. In the case of μGA the "start and restart" procedure helps in avoiding the premature convergence and the μGA is always looking for better strings. In implementing μGA, our interest is purely to find the optimum as quickly as possible and not in the average behavior of the population. In other words, our performance measure for μGA should be based on the best-so-far string, rather than on any average performance.

3. μGA IN NON-STATIONARY OPTIMIZATION

As mentioned earlier, there is a class of engineering problems in which the function to be optimized evolves faster than a SGA can find an optimum solution. The μGA implementation presented earlier can be suitably used for optimizing such non-stationary functions. The general outline is similar to the one presented above with some minor modifications:

1. Every non-stationary optimization problem will have an average solution which could be determined a priori. This is not a requirement for the implementation of the μGA, though it helps the search procedure to adapt quickly and the problem at hand will have, at the least, a suboptimal point to start with.

2. Function evaluations are conducted on-line using the latest available definitions of the function. The best string is used as the optimal point and the best string always evolves into the next generation intact. The problem which requires the optimal solution will always be updated with the best string.

3. New strings are introduced after the population converges to a prescribed measure, or after a fixed number of evolutions. The best string from the previous converged population is carried into the next generation.

A schematic description of the above procedure is presented in Figure 1.

4. EXPERIMENTAL SETUP

The remainder of this paper describes three test function optimization problems and the results obtained. The three test functions are chosen to encompass unimodal and multimodal stationary functions and non-stationary functions. The results of using the μGA approach is compared with the traditional SGA approach.

4.1 Test Function 1--A Stationary Unimodal Function

The first test function chosen is an unimodal 3-dimensional parabola with a minimum of zero at the origin:

$$J1 = X^2 + Y^2 + Z^2$$

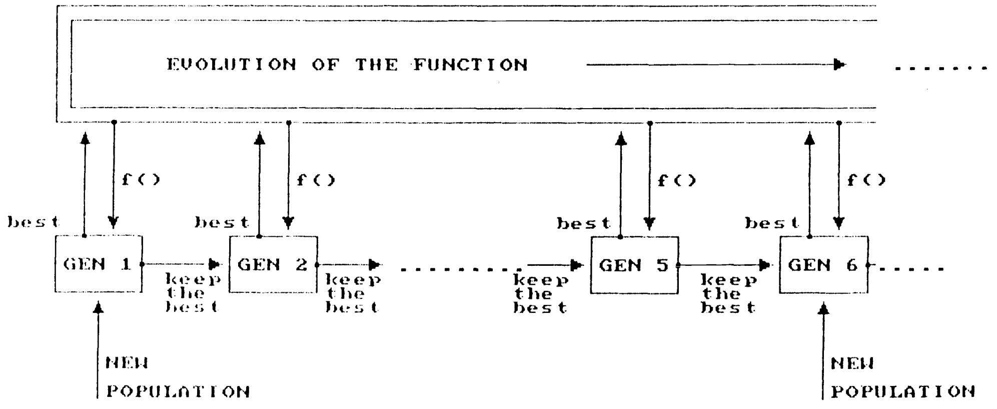

Fig 1. Schematic Representation of μGA-based Non-stationary Optimization.

restricted to the space $-5.12 \leq X,Y,Z \leq 5.12$, with a resolution factor $\delta X = \delta Y = \delta Z = 0.01$. This search space consists of approximately 10^9 alternate solutions.

Here the objective is to minimize J1 with the above given constraint. A string of length L=30 is chosen to map the variables X, Y, and Z for the above given range and resolution. The μGA optimization is conducted for 25 different random starts and an ensemble average of the best string so far is calculated. A similar optimization is conducted using the SGA with population sizes of 50 and 200. The selection strategy for the SGA is chosen to be the same as the one for the μGA, namely, the deterministic tournament selection strategy. Figure 2 shows the evolution of the ensemble average of the best-so-far fitness. It is clearly seen that the μGA reaches the near-optimal region quicker than the other two SGA approaches. The SGA with population size of 50, chosen as the optimal population size for this problem by both De Jong and Grefenstette, performed better than the 200-population SGA but did worse than the μGA.

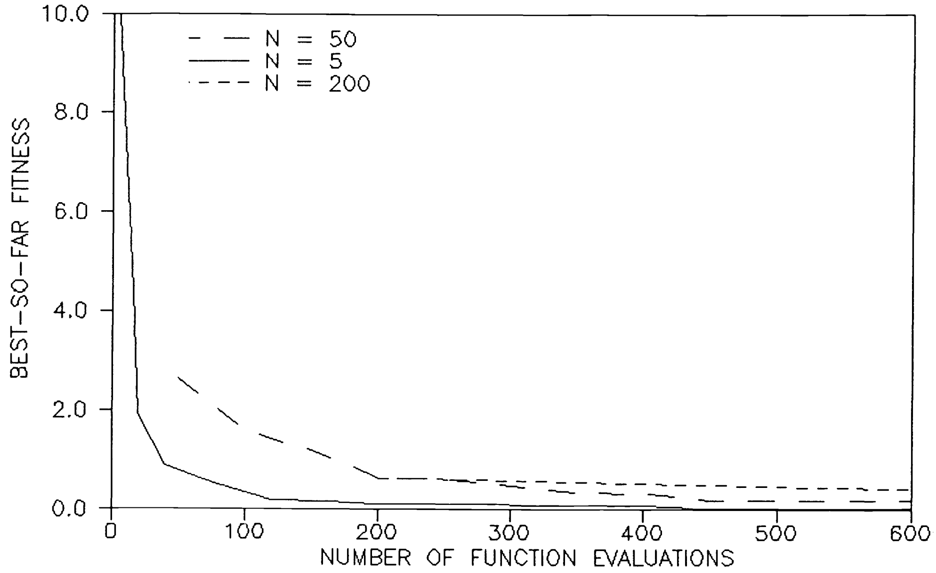

Fig. 2. Best-so-far Fitness Evolution for Test Function 1.

4.2 Test Function 2--A Stationary Multimodal Function

The second test function chosen is a continuous, non-convex, non-quadratic, two dimensional function with 25 local minima (De Jong, 1975):

$$J2 = (\ 0.002 + (\sum_{j=1}^{25} (f_j(x))^{-1}))^{-1}$$

where

$$f_j(x) = j + \sum_{i=1}^{2} (x_i - a_{ij})^6$$

restricted to the space $-65.536 \leq x_i \leq 65.536$, with a resolution factor $\delta X_i = 0.001$. This search space consists of approximately $1.6 * 10^{10}$ alternate solutions. The cost surface J2 is a flat surface with 25 deep "foxholes" located at $x_i = a_{ij}$.

The comparison of the performances of μGA and SGA is carried out with the help of the off-line performance measure evaluated by De Jong (1975). Figure 3 shows the evolution of the ensemble average of the best-so-far fitness for μGA and off-line performance of SGA. It is clearly seen that the μGA reaches the optimal region quicker than the SGA approach.

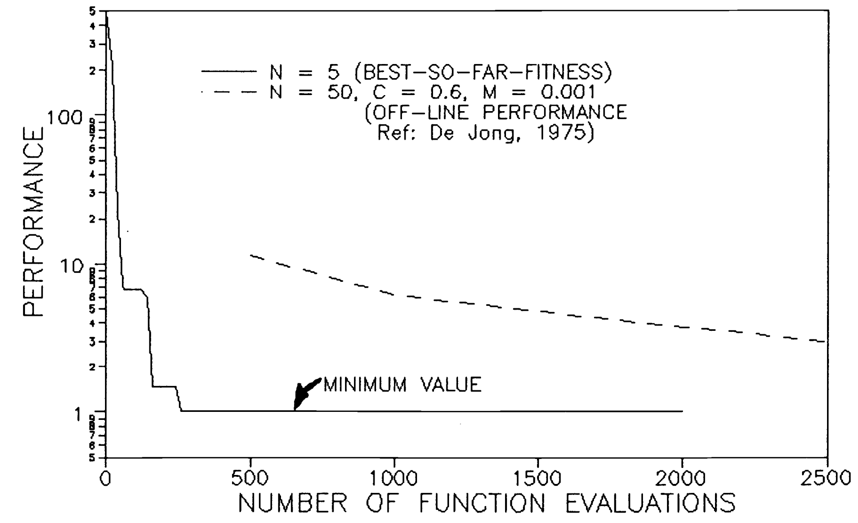

Fig. 3. Performance Comparison for Test Function 2.

4.3 Test Function 3--A Non-stationary Function

A real-life non-stationary function optimization problem is chosen as the third test function. This optimization problem is related to the wind shear optimal guidance problem and is of keen interest to airframe manufacturers regarding aircraft safety. Simply stated, the function to be minimized is the maximum flight path angle deviation of a transport airplane flying in a take-off mode through a severe wind shear, as shown in Figure 4, with several constraints (a detailed description of the wind shear phenomenon, the wind shear and the aircraft models used in this study, and other pertinent information can be found in Krishnakumar and Bailey, 1989 and Zhu and Etkin, 1983).

This problem can be restated in terms of a feedback control problem as follows: In the absence of any additional thrust, the aircraft states and the wind states are fed back to the elevator

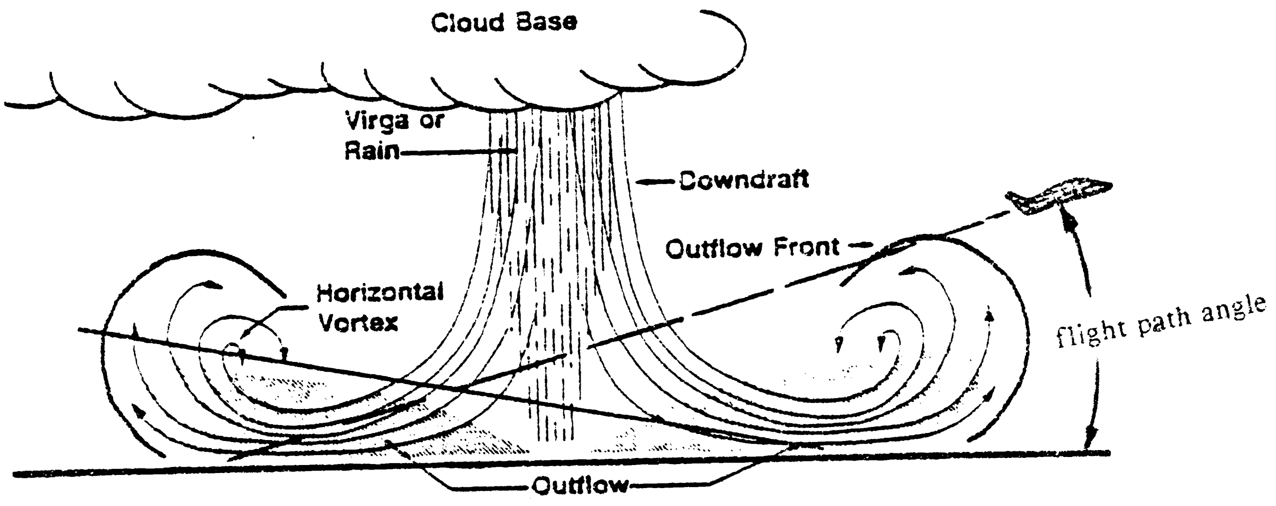

Fig. 4. A Typical Microburst Wind Shear Encounter.

in order to control the airplane to achieve minimum flight path angle deviations; In doing so, it is also desired to keep the airspeed of the airplane within some constraints. This problem can be written as a function minimization problem as:

J3 = Maximum flight path angle deviation of the
aircraft in the next "T" seconds.

where the flight path angle is a function of airspeed, wind components, and elevator control.

The elevator feedback control used in this study (Krishnakumar and Bailey, 1989) is defined as follows:

$$\delta e = K_1 * h_{di} + K_2 * \dot{h}_{di}$$

where

$h_{di} = 2 \lambda h_i - (2 - 2 \lambda)(V_I^2 - V_{I0}^2)/2g$; $\dot{h}_{di} = 2 \lambda \dot{h}_i - (2 - 2 \lambda)(V_I \dot{V}_I)/g$
h_i = Inertial height error. ; V_I = Total inertial velocity.
h_{di} = Specific inertial energy distribution error.

The choice of λ is based on the maximum allowable fluctuations in airspeed (V_{min} and V_{max}). The choice of λ used in this study is presented in Figure 5. K_{1V} and K_{2V} are optimized by augmenting the original function with penalty functions. The optimization problem presented above has four gains (K_1, K_2, K_{1V}, and K_{2V}) to be optimized and these are mapped on to a 28 bit binary string (7 bits for each gain) with a range from 0 to 0.01.

The stationarity of this problem depends on "T", the time period in the future, over which the optimization is conducted. If "T" is greater than the wind shear encounter period then the problem degenerates to a stationary optimization problem. Unfortunately, to determine the maximum flight path angle loss of the airplane for a specified period of time in the future, the wind shear components have to be estimated from the available airplane data and an aircraft model has to be assumed. The wind shear estimates get better as the aircraft penetrates the wind shear. This better estimate of the wind field necessitates "T" to have a smaller value compared to the duration of the wind shear encounter. This means that the optimum of the objective function varies with time making it a non-stationary function optimization problem.

The strategy used here for implementing the μGA is shown in Figure 6. The initial population of five is chosen randomly (to demonstrate the robustness of μGA, an a priori stationary optimal feedback gain set was not used in the initial population) and the μGA is used to evaluate better fitness strings at every generation. It is assumed that the μGA can perform the five population objective function fitness computation in regular two second intervals. This means that there is a new update of controller gains every two seconds (if one exists). The performance of the μGA is compared with a constant optimal set of gains derived in an earlier study (Krishnakumar and Bailey, 1989). The performance is shown in terms of the height response of a Boeing 727 non-linear mathematical model, in a take-off mode, in the presence of a very severe wind shear (Figure 7 shows the wind velocities). As can be seen in Figure 8, the gains K_1 and K_2 change as the aircraft penetrates the wind shear. Figure 9 shows the height response of the simulated wind shear encounter for both the μGA-designed adaptive controller and the constant optimal gain controller. It is apparent from this figure that the adaptive controller performed better than the constant gain controller.

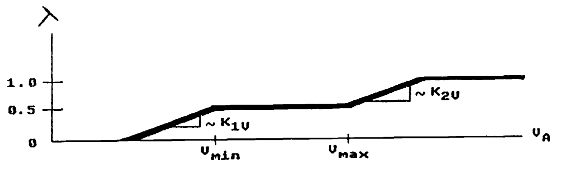

Fig 5. Airspeed-based λ design.

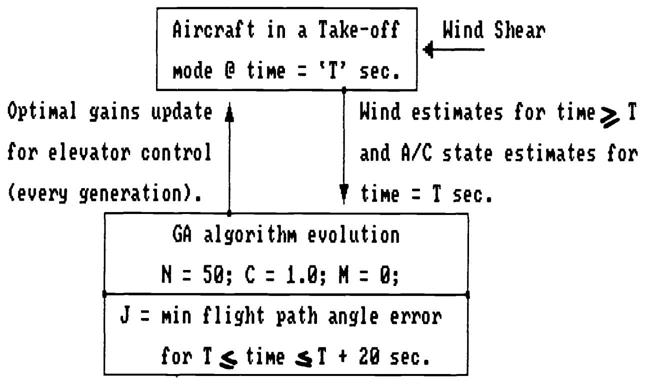
Aircraft in a Take-off mode @ time = 'T' sec.
Wind Shear
Optimal gains update for elevator control (every generation).
Wind estimates for time ≥ T and A/C state estimates for time = T sec.
GA algorithm evolution
N = 50; C = 1.0; M = 0;
J = min flight path angle error for T ≤ time ≤ T + 20 sec.

Fig. 6. μGA Implementation for the Wind Shear Controller Design.

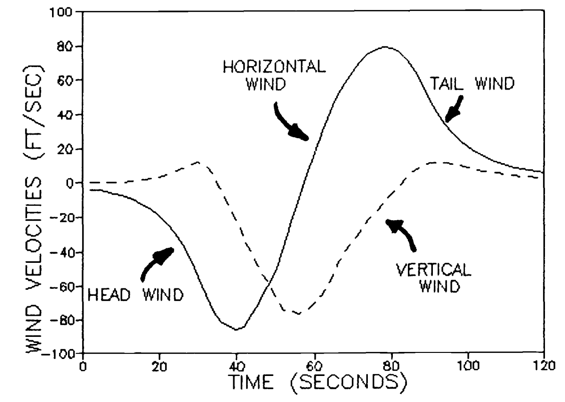

Fig. 7. Wind Velocities of a Severe Microburst.

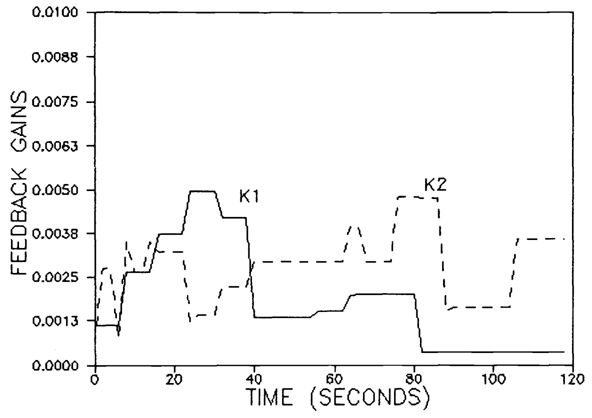

Fig. 8. Optimal Gain Evolution for the Wind Shear Controller.

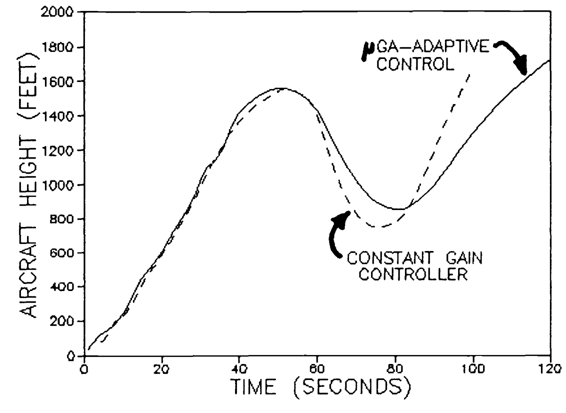

Fig. 9. Optimal Aircraft Height Response in the Presence of Wind Shear.

5. CONCLUSIONS

Stationary and non-stationary function optimization look highly feasible with the help of micro-genetic algorithms, which are serially implemented small population genetic algorithms with simple genetic operators. It is also shown that the μGA reaches the near-optimal region much quicker than the big population genetic algorithms. Behavior of μGA in finding the absolute optimum and its ability in solving optimization problems with deception need to be analyzed. Also, applications of μGA should be further investigated for varying string lengths and varying levels of non-stationarity. The application of μGA for wind shear feedback controller gain tuning looks feasible and needs extensive simulation investigation for further validation. A fact that, with a small population, a "start and restart" procedure performs better than a large population in reaching the near-optimal region is established in this paper.

6. ACKNOWLEDGEMENTS

I would like to thank Dr. David Goldberg for the thought provoking moments he spent with me despite his busy schedule, and for all his suggestions that made this paper a reality.

7. REFERENCES

Goldberg, D. E. (1988a). *Sizing Populations for Serial and Parallel Genetic Algorithms*, (TCGA Report No. 88004). University of Alabama, The Clearinghouse for Genetic Algorithms.

De Jong, K. A. (1975). "An Analysis of the Behavior of Genetic Adaptive Systems," *Dissertation Abstracts International, 41(9)*, 3503B.

Grefenstette, J. J. (1986). "Optimization of Control Parameters for Genetic Algorithms," *IEEE Transactions on Systems, Man, and Cybernetics, SMC-16(1)*, 122-128.

Krishnakumar, K. S., Bailey, J. E. (1989). "Inertial Energy Distribution Control for Optimal Wind Shear Penetration," AIAA Paper 89-0016.

Zhu, S., Etkin, B. (1983). "Fluid-Dynamic Model of a Downburst," University of Toronto UTIAS Report No. 271, CNISSM 0082-5255.

Goldberg, D. E. (1988b). *Genetic Algorithm in Search, Optimization, and Machine Learning*, Addison Wesley Publishing Company.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Michigan.

Goldberg, D. E. (1989). *Personal Conversation.*