5-1-2009

Optimization of Strongly Nonlinear Dynamical Systems Using a Modified Genetic Algorithm With Micro-Movement (MGAM)

Xing Wei
Utah State University

OPTIMIZATION OF STRONGLY NONLINEAR DYNAMICAL SYSTEMS USING A MODIFIED GENETIC ALGORITHM WITH MICRO-MOVEMENT (MGAM)

by

Xing Wei

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

Dr. Edmund A. Spencer Major Professor	Dr. Bedri A. Cetiner Committee Member
Dr. Aravind Dasu Committee Member	Dr. Byron R. Burnham Dean of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2009

Abstract

Optimization of Strongly Nonlinear Dynamical Systems Using a Modified Genetic Algorithm with Micro-movement (MGAM)

by

Xing Wei, Master of Science

Utah State University, 2009

Major Professor: Dr. Edmund A. Spencer
Department: Electrical and Computer Engineering

The genetic algorithm (GA) is a popular random search and optimization method inspired by the concepts of crossover, random mutation, and natural selection from evolutionary biology. The real-valued genetic algorithm (RGA) is an improved version of the genetic algorithm designed for direct operation on real-valued variables. In this work, a modified version of a genetic algorithm is introduced, which is called a modified genetic algorithm with micro-movement (MGAM). It implements a particle swarm optimization (PSO)-inspired micro-movement phase that helps to improve the convergence rate, while employing the efficient GA mechanism for maintaining population diversity. In order to test the capability of the MGAM, we first implement it on five generally used test functions. Then we test the MGAM on two typical nonlinear dynamical systems. The performance of the MGAM is compared to a basic RGA on all these applications. Finally, we implement the MGAM on the most important application, which is the plasma physics-based model of the solar wind-driven magnetosphere-ionosphere system (WINDMI). In order to use this model for real-time prediction of geomagnetic activity, the model parameters require updating every 6-8 hours. We use the MGAM to train the parameters of the model in order to achieve the lowest mean square error (MSE) against the measured auroral electrojet (AL)

and Dst indices. The performance of the MGAM is compared to the RGA on historical geomagnetic storm datasets. While the MGAM performs substantially better than the RGA when evaluating standard test functions, the improvement is about 6-12 percent when used on the 20D nonlinear dynamical WINDMI model.

(57 pages)

Contents

List of Tables

List of Figures

Chapter 1

General Search and Optimization Algorithms

1.1 Introduction

Random search and optimization methods, also called global search heuristics, are a category of algorithms that include methods like the genetic algorithm (GA) [1] and particle swarm optimization (PSO) [2]. The genetic algorithm is a widely implemented optimization method for tuning system parameters or finding solutions of complex equations. These two methods are capable of solving complicated nonlinear equations with a large number of variables (dimensions), especially when the mathematics is intractable. The goal for these methods is to locate a good approximation to the global minimum of a given cost function in a search space, with all variables in the cost function tuning the system.

1.2 Original Genetic Algorithm

The genetic algorithm is one of the most popular random search and optimization methods. It is inspired by the concepts of chromosome crossover, random mutation, and natural selection from evolutionary biology. It is widely used in engineering and physics as a general optimization method. Computer simulations of evolution started as early as in 1954 with the work of Nils Aall Barricelli in Princeton, New Jersey. The genetic algorithm was later made popular through the work of John Holland in the early 1970s.

In the GA, a population of some size is constructed. Every individual within the population becomes a candidate solution for a particular application. Simulating a gene chromosome, an individual is a string representing the application variables cluster. The population will evolve to survive, following the evolutionary rules of biology. Individuals will survive or be discarded by a selection process, and new offspring data will be produced.

Ability to solve an application will be improved by the reproduction process. The final solution will be produced at the end of a certain number of generations.

In general, a normal genetic algorithm can be divided into the following sequence of steps. This is shown in fig. 1.1.

1. Initialization
2. Evaluation of fitness
3. Crossover
4. Mutation
5. Repeat step 2 to 4 until stopping criteria is achieved
6. Ending the algorithm

1.2.1 Initialization

The initialization step is to generate the first generation of individuals for starting the algorithm. To initialize the algorithm, every variable of an individual will be randomly generated within their defined range. The range of the initial population will have to cover the entire space of possible solutions. In addition, depending on the nature of the problems, the population size can be from several to hundreds. Typically, because of the convenience of coding, the population number usually is in the form of 2^n (n is a positive integer).

To produce the first generation G_1, we first define the minimum and maximum values that each variable s^{ij} is allowed to take. In this work, $i = 1...N$ is the index over the number of parameter sets in G_1; and $j = 1...M$ is the index number of variables. The minimum and maximum values are denoted, respectively, as s^{ij}_{min} and s^{ij}_{max}. We choose a resolution value n, which is a necessary setting parameter because of the conversion between real values and binary values. The formula for each initial s^{ij} as

$$s^{ij} = s^{ij}_{min} + \left(\frac{s^{ij}_{max} - s^{ij}_{min}}{2^n} \right) m^{ij}, \tag{1.1}$$

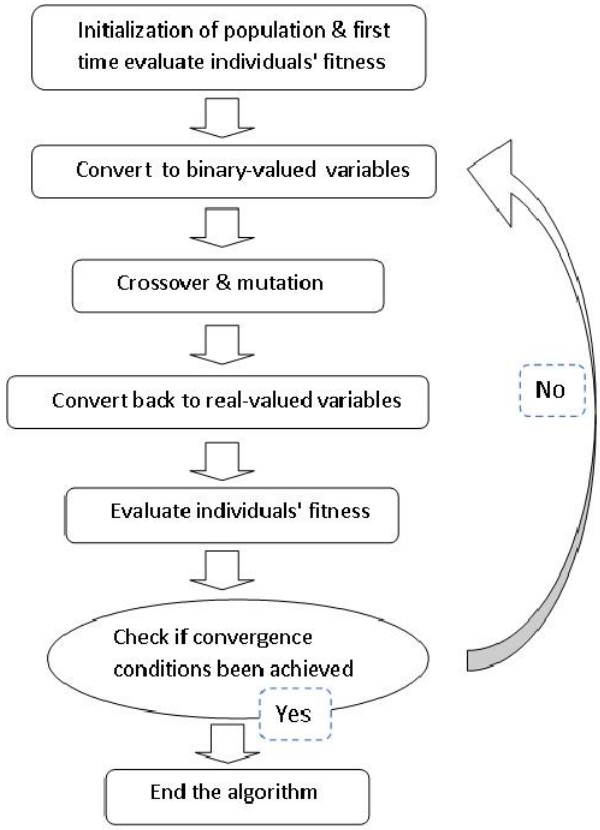

Fig. 1.1: Flow chart of genetic algorithm.

where m^{ij} is an integer that can take values from 0 through $2^n - 1$. Each variable is then set by randomly choosing m^{ij} for all j to construct each s^i and then for all i to construct G_1.

1.2.2 Evaluation of Fitness

This is the step which mainly focuses on the application demands. Since the genetic algorithm is a generally implemented method, it should be able to deal with different targets of optimization applications without changing the basic structure of the algorithm. A fitness function (or called cost function) will be defined here, which depends on particular application aim. Typically, the fitness function could be a lowest mean square error (MSE) or a normalized L2 norm of the simulated value against the target value. A carefully defined

fitness function helps the genetic algorithm to better accomplish its mission of optimization.

1.2.3 Selection

As one of the evolution progress steps, the proportion of the existing population is selected to breed a new generation during each successive generation. Usually, individual solutions are selected through a fitness-based process, which means fitter solutions as measured by a fitness function are more likely to be selected. However, there are other selection methods like a weighted random sample of the population. Different selection methods are the results of different reproduction policies.

It is also very important for most selection methods to be patially stochastic so that a small proportion of less fit solutions are selected. This is because of the importance to keep the diversity of the population large, preventing premature convergence on poor solutions. Roulette wheel selection and tournament selection are two of the most popular and well-studied selection methods.

In this work, we keep half of the existing population and discard the other half in every generation. The selection we implement is also based on a fitness-based process by ranking the fitness of the individuals.

1.2.4 Crossover and Mutation (Reproduction)

The reproduction step consists of crossover and mutation process. It will produce new born 'children' solutions which share the characteristics of their 'parents' solutions. The 'parents' solutions will come from those individuals selected to survive from last generation. The 'children' solutions will be first generated by crossover process.

In the crossover process, all the variables of an individual solution will be clustered and converted into a binary form with ones and zeros, which become the 'parents' genetic sequence solutions. One or more crossover point on both 'parents' organism strings is randomly selected. All data beyond that point in either organism string is swapped between the two 'parents' organisms (fig. 1.2). The 'children' will be the resulting organisms.

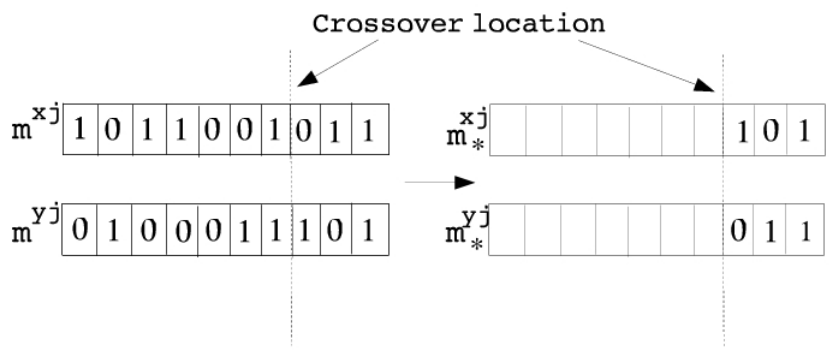

Fig. 1.2: One-point crossover process.

Different crossover methods, such as one-point, two-point, and uniform crossover, have different rules on how the 'children' solutions inherit the characteristics from their 'parents'.

The purpose of mutation is to prevent the premature convergence on poor solutions. In the classic genetic algorithm, the mutation operator involves an arbitrary bit in a genetic sequence having a probability to be changed from its original state. That is to flip some random part of the genetic sequence from '0' to '1,' or from '1' to '0' described in fig. 1.3. A parameter of mutation rate will be defined so that the higher the rate is, the more likely the 'children' will mutate.

The newborn individual data will be evaluated by the fitness values after the reproduction process.

1.2.5 Termination

The reproduction process will keep repeating until one of the conditions to end the algorithm has been achieved. Usually, the ending criteria will be one of the following:

1. A solution is found that satisfies the minimum criteria,

2. A fixed number of generations reached,

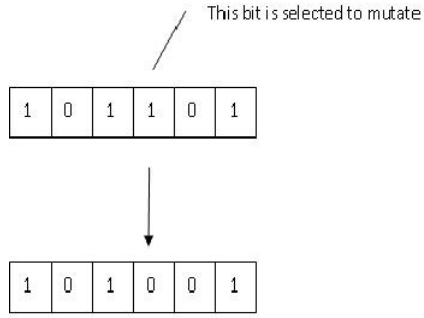

Fig. 1.3: Mutation process.

3. Allocated budget (computation time/money) reached,
4. The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results,
5. Manual inspection,
6. Combinations of the above.

The fixed number of generations will usually be an easy ending criteria, which is what we implement in this work. That is because, for a complicated case with a large amount of variables and equations, it is very difficult to achieve a final optimum. Also, it is very difficult to estimate how much time to take to converge for many applications. A fixed generation number will not only help to end the simulation within some definite time, but it also could be easier to compare convergence results and efficiency between different algorithm.

1.3 Real-Valued GA

In this originally conceived form, the genetic algorithm involves binary genetic sequences that are converted from real-valued variables. [1, 3, 4]. This results in extra effort on conversion between binary and real values, as well as unavoidably facing a problem of comparison with other algorithms, for example, the PSO [2]. The resolution of accuracy

of every individual solution will be difficult to decide. Therefore, a more robust version of the GA has been widely implemented. It can handle real-valued variables directly while processing both the crossover and mutation process. It is named the real-valued genetic algorithm (RGA). By using an RGA, every individual solution retains a machine specific bit precision. The flow chart of an RGA is demonstrated in fig.1.4.

1.3.1 Initialization of RGA

Since the RGA operates on real-valued numbers directly, there will be no resolution requirement when we set up the initial generation. A initial individual solution now is a dataset of some random values chosen within the range of the variables, as eq. (1.2). The resolution is automatically set by machine precision.

$$s^{ij} = s^{ij}_{min} + \left(s^{ij}_{max} - s^{ij}_{min}\right) m^{ij}, \tag{1.2}$$

where m^{ij} is an random number that can take values from 0 through 1. Each variable is then set by randomly choosing m^{ij} for all j to construct each s^i, and then for all i to construct the first generation.

1.3.2 Crossover and Mutation of RGA

In the RGA, we cannot cluster all variables to form some data strings and randomly exchange some part of them like the binary GA. This is because different parameters could have entirely different length of decimal bits and precisions. Therefore, different crossover methods for the RGA have been published [5, 6].

For real data, particular variables have to crossover among themselves. There are three main rules for any kind of RGA crossover to follow. First, the offspring must retain the good properties (fitness-based good characteristics) from the 'parents'. Second, the 'children' must be different from their 'parents', as well as from each other. Third, the 'children' should be produced randomly; their distribution of them will have to cover a large enough range. These are all very important rules to simulate evolution, while trying

to avoid premature convergence on poor solutions.

In this work, we will use the method below to process the real-valued crossover. Let the 'parents' be s_1 and s_2, then s_{1j} and s_{2j} are the jth element of s_1 and s_2, respectively. We implement mean value to produce new generations.

$$S_{1j} = (1 + r_1)(k_1 s_{1j} + (1 - k_1)s_{2j}), \tag{1.3}$$

$$S_{2j} = (1 + r_2)(k_2 s_{1j} + (1 - k_2)s_{2j}), \tag{1.4}$$

where k_1 and k_2 are random numbers between 0 and 1. Then S_{1j} and S_{2j} are the j_{th} element of 'children' of s_1 and s_2. r_1 and r_2 are two random range factors between 0 to 1 for randomly enlarging the offspring data range. In this method, the offspring of the individuals with best fitness from last generation will inherit good information from their 'parents.' The 'children' data will randomly combine the genetic information from their 'parents' by k_1 and k_2 and will be different from each other. The range of the 'children' data will cover a larger space by using $(1 + r_1)$ and $(1 + r_2)$, instead of only covering the range between 'parents' data. However, this might bring up an offspring value which is born outside of the range. If this happens, we will reproduce the child within the range. For example, in the j_{th} generation, if S_{1j}s or S_{2j}s are born out of the data range, we use the following two equations to substitute the 'children' data to fix the data inside of their appropriate range.

$$S_{1j} = k_1 s_{1j} + (1 - k_1)s_{2j} \tag{1.5}$$

$$S_{2j} = k_2 s_{1j} + (1 - k_2)s_{2j} \tag{1.6}$$

By using this method of crossover, the RGA can follow the same spirit of the original binary GA, as well as meeting the required rules above. It still utilizes the Darwinian principle of survival of the fittest. The offspring data will retain the higher normalized fitness of genes from the previous generations, which is more likely to achieve a better

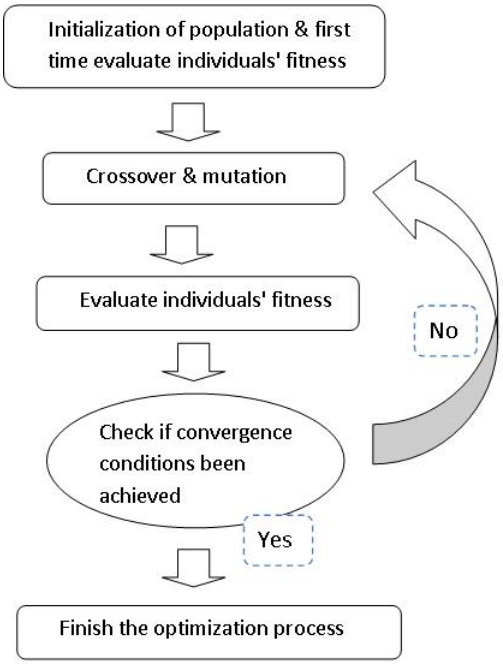

Fig. 1.4: Flow chart of RGA.

solution.

The mutation process of RGA has to produce a random new individual in its particular variable range, the same way as in the first generation individual in eq. (1.2). Because of the mechanism of the mutation process, the mutation rate values of the RGA has different scales from the binary GA.

1.4 Sinusoidally Changing Rate Mutation

With a fixed minor portion of mutation probability parameter μ, the RGA code will process the mutation on selected 'children' solutions. However, the fixed small mutation rate will not be sufficient to help the population to escape local minimums, especially when the code has already been running for a very long period of generations, but the best fitness in the population still has not improved. A relatively higher mutation rate will help the individuals to have a higher possibility to jump out of the local minimums. However, to converge to a better fitness, the mutation rate cannot be persistently high.

Therefore, we implement a fixed-to-sinusoidally changing mutation rate for the RGA. The mutation rate will stay at a small value like $\mu = 0.15$ at the beginning. A counter of cost function value counts the number of generations to keep track of how long the best fitness value has not improved. If the counter has reached a threshold value of generation number, a sinusoidal format of mutation rate will be activated. The mutation rate μ will then be changed into the following sinusoidal format as eq. (1.7).

$$\mu = 0.15 + \alpha * sin(m/2\pi M), \tag{1.7}$$

where the α is a scale factor which could be customized depending on different applications. Here we use $\alpha = 0.30$, giving the mutation rate maximum as 0.45. The M is the total period of the sinusoid changing rate, set as 80 generations in our tests. The m is a counter value, which is counting the generation numbers of reproduction after the sinusoidal mutation rate has been triggered. It will be reset as 0 after reaching the maximum value of M and the count begins again.

1.5 Introduction of Particle Swarm Optimization

1.5.1 Overview of PSO

The particle swarm optimization (PSO) [2, 7] is another popular random search algorithm. It simulates a kind of social optimization. As an optimization algorithm, if a problem is given, there will be some way to evaluate a proposed solution if it is given in the form of a fitness function. A communication structure or social network is defined in PSO, which share common information among members of the society. Then a population of individuals defined as random guesses of the problem solution is initialized. These individuals are candidate solutions, or called particles in a swarm. They will first start at random positions in the search space and move in a random direction with a random velocity in order to search for an optimum. Most importantly, an iterative process to improve these candidate solutions is set in motion. The particles iteratively evaluate the fitness of the candidate so-

lutions and remember the location where they had their highest success. The best solution is called the local best or particle best. Each particle makes this information available to the society and a temporary global best position will be conveyed to everyone. Movement through the search space are affected by these successes. Every particle has a trend to move towards the local best and global best position. But not necessarily all individuals go at the correct speed and in the right direction, because of a random component added to every individual's velocity. With a trend towards the best position, the population will converge, by the end of a trial, on an optimum solution for a particular application.

1.5.2 Searching Processing of PSO

In a PSO problem, the 'best' value simply means the position with the smallest objective function value (or the smallest fitness value). The swarm is typically modeled by particles in multidimensional space that have a position and a velocity. These particles move all over the search space and have two essential reasoning capabilities: memory of their best position and knowledge of the global best position. Therefore, members of a swarm communicate good positions to each other and adjust their own positions and velocities based on the information. So a particle in PSO has the following information to change its position and velocity:

1. A global best that is known to all particles and immediately updated for every iteration, when a new best position is found by any particle in the swarm,

2. The local best, which is the best solution that the particle has seen.

Below, we can use the simple equations about particle position and velocity update equations to show the basic iteration loop contents of PSO. However, to prevent the premature convergence, every particle will have a third component with a random velocity. By this random part, every particle possesses their own scale of velocity and direction, trying to keep the diversity of the swarm, just as mutation in GA.

$$v_{i,j} = c_0 v_i + c_1 v_1 \left(gb_j - x_j\right) + c_2 v_2 \left(lb_{i,j} - x_{i,j}\right), \tag{1.8}$$

$$x_{i,j} = \left(x_{i,j} + v_{i,j}\right), \tag{1.9}$$

where gb, lb, and nb are global best, local best, and neighborhood best, respectively. c_0, c_1, and c_2 are velocity scale which can be customized for the particular application. v_i is the random part added to the total velocity. As the swarm iterates, the fitness of the global best solution improves (decreases for minimization problem), and finally converges.

Chapter 2

A Modified Genetic Algorithm with Micro-Movement (MGAM)

2.1 Introduction of MGAM

The genetic algorithm is capable of searching for the global optimum. However, when the search space is huge and the number of variables is great, it will still have problems with premature convergence on poor solutions and the speed of convergence to a good approximation. This is a common problem that most other general search and optimization algorithms, etc., the PSO and the simulated annealing, have. As a result, new ideas on modifications of these algorithms and hybrid approaches have been published.

Some have proposed hybrid approaches combining GA and PSO inspired by the mechanisms of these two algorithms [6, 8, 9]. In order to take advantage of both, these hybrid algorithms basically use two ways to hybridize the GA and PSO. One is to take the result solutions of the GA as the initial solutions of the PSO. Individual solutions will experience both evolution from GA and particle movement from PSO in every generation. The other way is to divide the population into a GA function part and a PSO function part by a certainty proportion, and evaluate the fitness values of individuals by the end of every generation.

Some improved optimization effects have been achieved by these approaches. However, such methods do not fully integrate the advantages of the two methods, because they are a simple combination or stack of the two algorithms. Furthermore, such a combination of methods will further complicate the algorithm, as well as sacrificing the computing speed and consuming more resources.

To better exploit the real-valued genetic algorithm, we further analyzed its mechanism.

The most powerful concept within the GA to prevent premature convergence on poor solutions is the mutation process inspired from chromosome mutation in biology. It has an advantage over many other general searching optimization algorithms like PSO. The mutation process helps the population to keep diversity, preventing a temporary best solution misleading the whole population. However, we found out that there is one unsatisfying feature of RGA. As soon as the newborn 'children' are produced from crossover and mutation, they will be ranked with their 'parents' to form the new population. It is possible that a mutated 'child' solution is produced around an even better solution than the current temporary best. But since the application could be nonlinear, if the newborn individual is not extremely close to the exact point of the new minimum location, it might be ranked with less fitness, and later could possibly be eliminated by the selection process. Actually, not only 'child' solutions have this problem, an even better solution might also be close to a temporary 'parents' solution without being discovered.

In order to prevent this from occurring, we require a method to better explore the neighboring area of individual solutions. Here we propose a new modified genetic algorithm specifically focusing on more efficiently searching the variables space. This approach will follow the basic structure of the real-valued GA described in the previous chapter, but with a new feature to make the algorithm more robust and powerful. After the reproduction and the re-evaluation step of the genetic algorithm, every individual solution will experience an extra random position movement. These movements send individuals to search around their neighboring area and explore possible better solutions.

This extra movement is inspired from the particle movements of PSO, with actions similar to the micro-scale movement of a little particle. Therefore, we name this algorithm a modified genetic algorithm with micro-movement (MGAM). The flow chart of the MGAM is shown in fig. 2.1.

2.2 Specification of MGAM

The following equation will demonstrate the mechanism of the micro-movement.

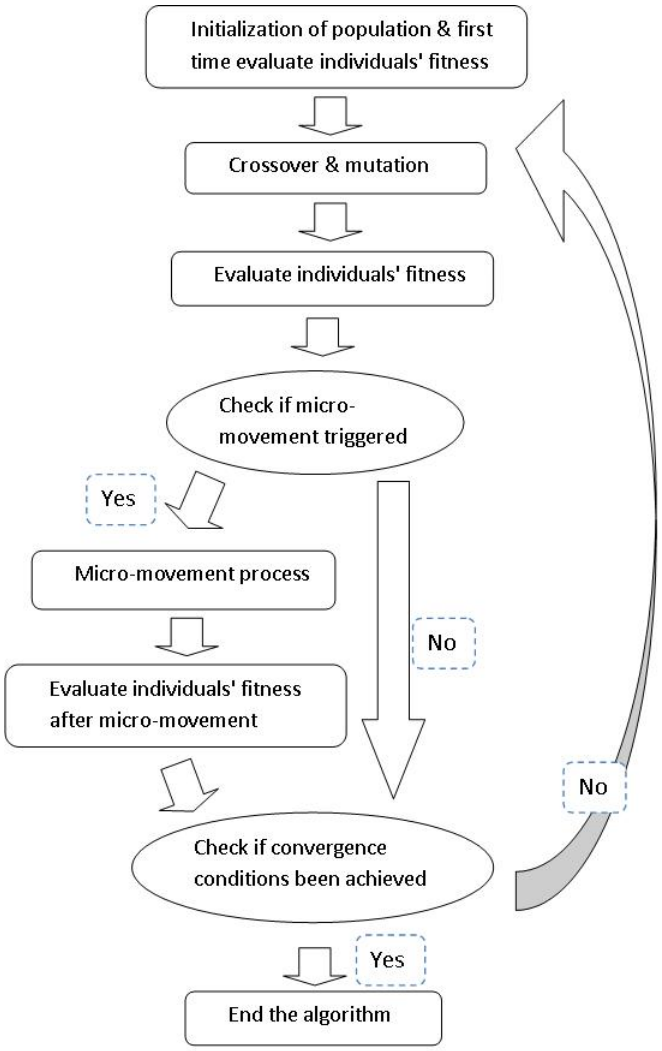

Fig. 2.1: Flow chart of MGAM.

$$Gen_{1j} = gen_{1j} + ms * signv * r_1, \tag{2.1}$$

where gen_{1j} is the newborn offspring data in jth generation, while Gen_{1j} is the offspring data after micro-movement happened. The ms is a factor with value 0.0001 or even smaller, controlling the micro-movement scale. The $signv$ is a random sign number, which randomly becomes 1 or -1. The r_1 is a random number taking values between 0 and 1. In other words, the second term of the right-hand side of this equation adds an extra small value (which could either make it greater or smaller) to the offspring data, in order to execute the movement around the immediate area.

There are several parameters here in the micro-movement equation controlling the specifics. Therefore, with careful control on its behavior, this extra movement could be more efficient and powerful.

2.3 Scale of Micro-Movement

The extra movement at a 'micro' scale is playing a great role in its original inspiration. Actually, if the extra movement is executed under a large scale (letting ms in eq. (2.1) be as large as 0.1 or 0.01), all individual solutions will move around within a very large part of the entire search space. These will make the movement have the similar effects of random mutation process, which is unnecessary and pointless since a higher mutation rate could be more efficient to accomplish this job. Thus, we usually set the mr in the equation as small as 0.0001 or 0.00001.

2.4 Iteration of Micro-Movement

The micro-movement process is a random search process for the individual solutions. It repeats several times during every generations in order to possibly obtain a better solution position in the search space. The iteration numbers of the micro-movement in every generation depends on different requirements of the applications, with different convergence features.

It is very important to notice that the micro-movement process also consumes the same computing resources as a reproduction process. That is because every time a micro-movement has occurred, we need to re-evaluate the fitness. A tradeoff between the repeats of micro-movement and its exploration thoroughness is then crucial for this algorithm. Thus, the micro-movement process will occur only during particular generations. Usually it is reasonable for it to take place when the best fitness of the population has not improved for a long time. Besides, in order to save computing time, the micro-movement will not continuously be activated for many generations.

2.5 Selection of Micro-Movement Results

After the micro-movement process, we still have to decide whether to keep the moved individuals or discard them. We have different choices for particular strategies. Considering the importance of keeping diversity of the population, we will discard all the moved individuals unless any improvement of the best fitness occurred.

2.6 Flow of Micro-Movement

As explained in previous sections, the micro-movement process will have the following steps.

1. Micro-movement exploration
2. Fitness re-evaluation
3. Comparison with former fitness
4. Keep or discard the moved individuals
5. Repeat steps above
6. Ending the micro-movement process for this generation after several repeats.

A flow chart of the micro-movement process is demonstrated in fig. 2.2.

With a condition controller in the simulation code, MGAM is able to select the particular iteration numbers (or also called generations) to perform the micro-movement phase, aiming to improve efficiency in exploring the search space, while retaining computing speed.

A trigger is needed here to decide during which generation the micro-movement take place. This trigger of the micro-movement utilizes a counter monitering the generations during which the fitness value has not improved. Once a better fitness has been found, the counter will be reset and start counting again. The micro-movement will not happen until the next time the counter has triggered the process.

Since we are not activating the micro-movement continuously, we set another counter to count how often we activate the micro-movement. Once it reaches a threshold number, which means the particles are trapped in local minimums for a certain period, the micro-movement will be paused for several generations. Meanwhile, the same counter changes to count the number of generations that the micro-movements have paused. Later, this counter will activate the micro-movement again by achieving a certain value.

If any improvement of best fitness is obtained by the micro-movement process, the two counters above will both be reset and start the count again. In other words, every time the reset happens, a better solution has come out through the micro-movement process.

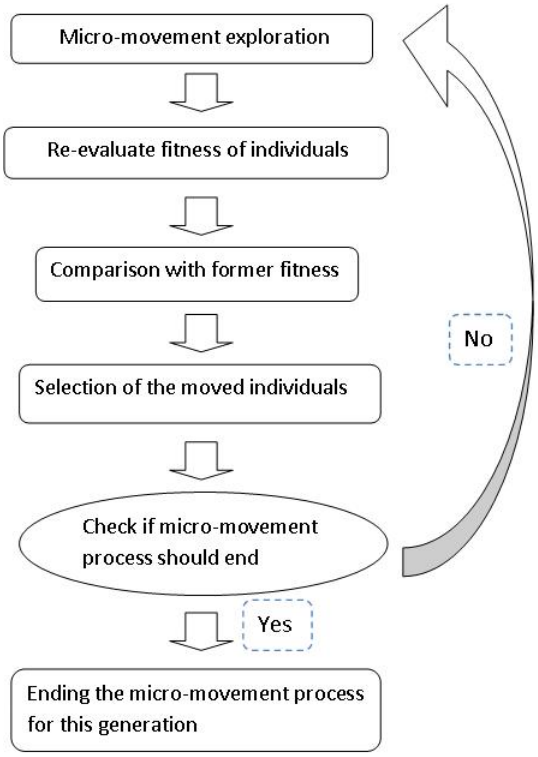

Fig. 2.2: Flow chart of micro-movement.

Chapter 3

MGAM Optimization of Standard Test Functions

The genetic algorithm can be examined by being implemented on different test functions. With various convergence features, standard test functions help to testify the optimization ability of the GAs. In this work, we ran simulations of five test functions [7], which are widely used for testing of random search algorithms, on both MGAM and RGA. Although there is no single algorithm that can promise to have ideal performance on all applications, the MGAM turns out to have an obvious advantage over the usual RGA on these five test functions.

To have a ideal approximate of the optimum, the result of all these test functions are objectively aiming to have a accuracy of 0.00001. The result might be far from this precision target, which depends on different properties of applications. The MGAM here can reasonably have the total number of generations as the same as, or less, than RGA. The total number of generations means the micro-movement iteration numbers for the whole population added up to reproduction generation numbers of the MGAM.

3.1 Test Functions Comparison Result

We uniformly configure both RGA and MGAM to optimize all these test functions. For the RGA, it runs with a fixed-to-sinusoidally changing mutation rate. Is starts with a fixed mutation rate of $\mu = 0.15$. Later, the sinusoidal mutation rate is triggered when 120 generations have been reproduced yet there is still no improvement on the best fitness. The sinusoid period of the mutation rate is 80 generations.

The MGAM has the same mutation rate type as RGA. The micro-movement will be triggered when 800 generations have been evolved, but still no improvement on the best fitness. The micro-movement will repeat four times within one generation, with a movement

scale of 0.0001, and it will activate for seven continuous generations. If there is still no improvement of fitness, the micro-movement will pause for 50 generations and start again. By these settings, the MGAM actually only have a relatively small number of generations with micro-movement process activated, aiming to save computation time.

3.1.1 Application on Tripod (2D)

The Tripod equation has only two variables, given by

$$f(x_1, x_2) = p(x_2)(1 + p(x_1)) + |x_1 + 50p(x_2)(1 - 2p(x_1))| + |x_2 + 50(1 - 2p(x_2))|, \quad (3.1)$$

with $p(u) = 1$ if $u >= 0$,$p(u) = 0$ if $u < 0$.

This function is theoretically easy, but misleads many algorithms, which are easily trapped in one or other of the local minima. Every variable in the test are set to have a range of [-300 300]. Table 3.1 shows the results of best fitness value acheived by both RGA and MGAM, which both simulate the optimization on Tripod five times. The RGA run 80,000 generations of reproductions. The MGAM, targeting to spend less time to converge, was set to run for 50,000 generations with no more than 20,000 micro-movement generations.

We can tell from Table 3.1 that the MGAM has a dominantly better performance over RGA for this test function. This result is the best example that demonstrates the micro-movement capability to better search the neighborhood of individual solutions. The Tripod function has a shape looks like many up-side-down house roofs with several minimums on the very buttom of the deepest 'roof.' The micro-movement of MGAM will send the individuals to move around those roofs to get to a deeper place, while the RGA is not able to search around. This will be the reason why the MGAM is greatly improve the genetic algorithm performance on this test function.

3.1.2 Application on Alpine (3D)

$$f(x_d) = \sum |x_d \sin(x_d) + 0.1x_d|, \quad (3.2)$$

Table 3.1: Results for Tripod (2D) test function.

case of tests	RGA 80000 generations	MGAM 50000 + (20000-) generations
1	0.2130	0.0004
2	0.2930	0.0001
3	1.0022	0.0003
4	0.3723	0.00007
5	1.6011	0.0001
Average fitness	0.6965	0.0003

with $d = 1, 2, 3$.

The Alpine function has many local and global minima of zero value. The surface of space is not completely symmetrical compared to the origin. Here the MGAM have 50000 generations. While we let RGA have 80000 generations to have more mutation chances to get a better result. Variable range here is again [-300 300].

With 30000 more generations of reproduction, the RGA still has a worse performance of fitness searching in Table 3.2. We further consider the micro-movement as a reproduction process, since it also need to evaluate the fitness for every individual solution. With the setting of parameters we previously described, the MGAM has less than 30000 times of micro-movement process occurred in all the cases of tests. This means with a totally less than 80000 generations of reproduction, the MGAM have over 10 times better result of fitness than RGA. The mechanism of micro-movement is playing a great role to help individuals to find best fitness around.

3.1.3 Application on Parabola (5D)

$$f(x_d) = \sum x_d^2, \tag{3.3}$$

with $d = 1, 2, 3, 4, 5$.

The Parabola function has only one minimum globally. The target optimal value for this function will be difficult to achieve because of the very low convexity in the area around the minimum. We increase the size of dimensions as 5D to testify the optimization ability

Table 3.2: Results for Alpine (3D) test function.

case of tests	RGA 80000 generations	MGAM 50000 + (30000-) generations
1	0.5136	0.0230
2	0.3417	0.0263
3	1.3035	0.0326
4	0.3757	0.1867
5	2.2584	0.1637
Average fitness	0.95858	0.08646

Table 3.3: Results for Parabola (5D) test function.

case of tests	RGA 60000 generations	MGAM 50000 + (500-) generations
1	1.4260	0.1712
2	0.0629	0.1604
3	0.2387	0.0042
4	1.0322	0.5843
5	0.2944	0.6662
6	0.7250	0.0260
Average fitness	0.6299	0.2687

of the MGAM on higher dimensional probelms. The variable range here we implement is [-20 20].

From the results in Table 3.3, we can see that this test function will be more challenging for both RGA and MGAM. And we found out that the MGAM with the same parameters setting as previous tests has much less micro-movement activated than the previous tests. Because of the parabola shape of space, it will be easier for the individuals to search a better fitness but with a relatively higher fitness value. Thus the micro-movement will not be activated too often since the best fitness keeps improving. The very low convexity of area that around the global minimum leads to the relatively higher fitness value for both RGA and MGAM. However, the MGAM still succeeds to have a better performance than RGA in this test. The exact number of iterations of activated micro-movement turns out to be so important for the result.

3.1.4 Application on Ackley (3D)

$$f(x_d) = -20\exp(-0.2\sqrt{\frac{\sum x_d^2}{3}}) - \exp(\frac{\sum \cos(2\pi x_d)}{3}) + 20 + e, \tag{3.4}$$

with $d = 1, 2, 3$.

The Ackley function apparently looks like Alpine. But it actually is more difficult, even with the same dimensionality, as other test functions. Here the function has a variable range [-30 30]. The very narrow and focused deep bottom of the space, where the global minimum lies, decreases the effectiveness of random replacement. However, the MGAM with micro-movement will be able to search the small region of the minimum once any one of the individuals have been placed in that region. This will greatly improves its effectiveness on this sort of problem. That is why, with less than 10000 generation of micro-movement process, the MGAM succeed to find an over 100 times better fitness than RGA with 80000 generations of reproduction (Table 3.4).

3.1.5 Application on Griewank (3D)

$$f(x_d) = \frac{\sum (x_d + 100)^2}{4000} - \prod \cos(\frac{x_d - 100}{\sqrt{d}}) + 1, \tag{3.5}$$

with $d = 1, 2, 3$.

The Griewank function (simulation result demonstrated in Table 3.5) is more difficult for these searching algorithms, because the global minimum 0 is almost indistinguishable from many closely packed local minima that surround it. Here it has a variable range [-300 300].

This test function activates micro-movement process more frequently than the Parabola function, because those many close local minmum all over the search space easily traps individual solutions. With less than 15000 generations of micro-movement, the MGAM again has an obvious better fitness searching ability than RGA with 80000 generations

Table 3.4: Results for Ackley (3D) test function.

case of tests	RGA 80000 generations	MGAM 50000 + (16000-) generations
1	2.1278	0.00095
2	1.8164	0.00059
3	1.1887	0.00039
4	2.1467	0.00093
5	1.8101	0.00078
Average fitness	1.8180	0.00073

Table 3.5: Results for Griewank (3D) test function.

case of tests	RGA 70000 generations	MGAM 50000 + (15000-) generations
1	0.1724	0.0117
2	0.1473	0.1600
3	0.1128	0.0897
4	0.1607	0.1008
5	0.1096	0.1095
Average fitness	0.1405	0.0943

reproduction. The mutation process of the genetic algorithm helps the individuals to get rid of the local minimums and reach to a better fitness value position after they are mutated.

3.2 Tests Conclusion

The five test functions are very good demonstrations of the advantageous mechanism of micro-movement. Depending on different properties of these test functions, the MGAM has different levels of improved performance than RGA. We could find out that the micro-movement process has especially impressive performance when the test function of fitness has one very deep minimum among the many local minimums like Ackley, or when there are many separately distributed, but respectively concentrated minimums, such as Tripod and Alpine.

Chapter 4

Optimization of Nonlinear Dynamic Systems with the MGAM

Nonlinear dynamic systems application will be a very suitable application field for MGAM optimization. Because of the complication of variables relationship and intractable property, a complicated nonlinear dynamic system will be tough for most of general search algorithms, especially when the system has a strong nonlinear property, the minimum value of the fitness function could be very difficult to find out. The MGAM could be tested during running optimization on this kind of application.

4.1 Using the MGAM on the Second Order Van der Pol System

Van der Pol equation, which is also called the Van der Pol oscillator equation (named for Dutch physicist Balthasar Van der Pol), is a type of nonconservative oscillator with nonlinear damping [10]. It evolves in time according to the second order differential equation. It is one of the most famous and typical example of nonlinear dynamical system.

$$\frac{d^2x}{d^2t} + \mu(x^2 - 1)\frac{dx}{dt} + x = 0 \tag{4.1}$$

The Van der Pol equation has the position coordinate variable x, time variable of t, and μ is a scalar parameter indicating the strength of the nonlinear damping. We can transform this second order differential equation into two first order equations.

$$\frac{dy_1}{dt} = y_2 \tag{4.2}$$

$$\frac{dy_2}{dt} = \mu(1 - {y_1}^2)y_2 - y_1 \tag{4.3}$$

The Van der Pol equation will not be intractable, but its complication will be a suitable level for a primary test on nonlinear dynamical systems. To implement MGAM for these equations, we select a certain μ, get its corresponding frequency f. Then, we use both MGAM and RGA to optimize the equation for the target frequency f. The result μ parameter from the optimization will be compared with the known μ to check how close the MGAM and RGA can get.

Here in the tests, we use target frequency as $f = 0.0524Hz$, with the already known $\mu = 10$. And we are optimizing the Van der Pol equation for the μ within a range of [0 50].

Results of Table 4.1, from both RGA and MGAM, are close enough for our target. The RGA (fig. 4.1) is able to find an approximate μ, but not as good as MGAM because of getting stuck in a local minimum. In fig. 4.2, MGAM is showing its ability to better search nearby regions and get rid of local minimum traps. As we expected from theory analysis, although the Van der Pol is simple, this test is still informing us that the MGAM is able to solve a nonlinear dynamical system problem as good as or even better than RGA, most importantly, within a less number of generations.

The simulation result shows that both RGA and MGAM can handle the Van der Pol oscillator equation quite well within certain generation of reproductions.

4.2 Using the MGAM on the Third Order Lorenz System

The Lorenz oscillator (named for Edward N. Lorenz), is a 3D dynamical system that exhibits chaotic flow [11]. With three first order differential equations, the Lorenz equation has a much more complicated structure and much stronger nonlinear variable relationship than the Van der Pol oscillator.

$$\frac{dx}{dt} = \sigma(y - x), \tag{4.4}$$

$$\frac{dy}{dt} = x(\rho - z) - y, \tag{4.5}$$

$$\frac{dz}{dt} = xy - \beta z, \tag{4.6}$$

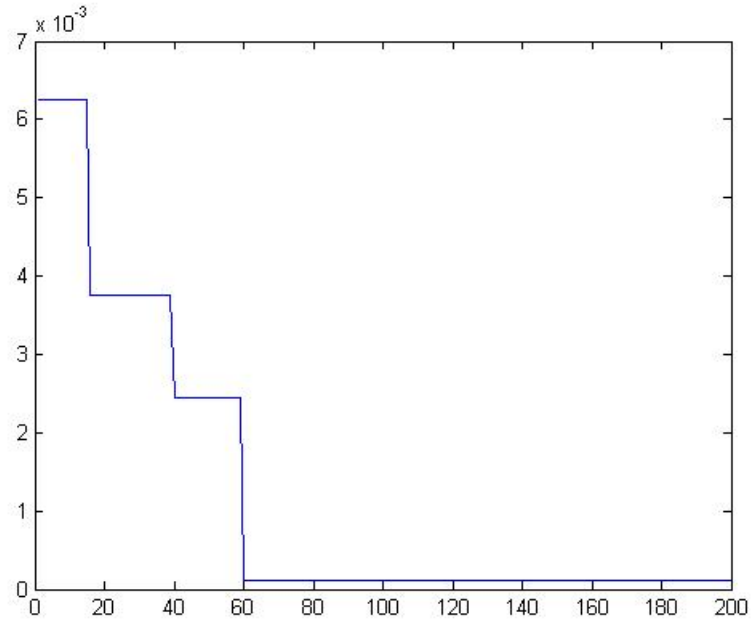

Fig. 4.1: Simulation result of Van der Pol equation for RGA.

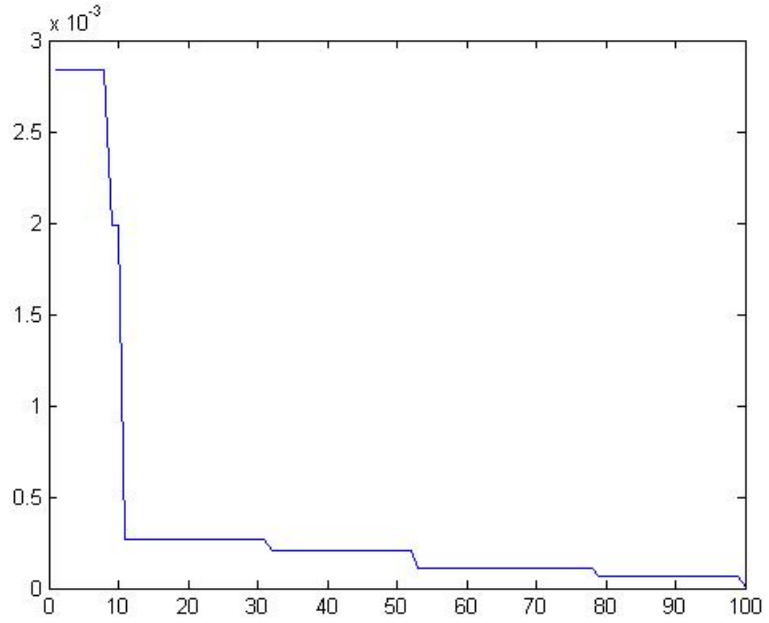

Fig. 4.2: Simulation result of Van der Pol equation for MGAM.

Table 4.1: Results for Van der Pol oscillator equation.

	RGA	MGAM
best fitness	0.00011	0.0000015
best μ result	10.0449	10.0204
generation numbers	200	100
micro-movement generations	NA	23

where σ is called the Prandtl number, ρ is called the Rayleigh number, and $\rho, \sigma, \beta > 0$.

This system exhibits a chaotic behavior for a certain ratio of these three parameters, which will have two equilibrium points for the limit cycles. For example, $\sigma = 10$, $\beta = 8/3$, and $\rho = 28$.

Our application target here is to find the equilibrium point in 3D space for the Lorenz attractor. For a meaningful test, the range of these three parameters will have to be carefully selected to have very low damping within certain time span. The chaotic status should be avoid since then it will have more than one equilibrium point and become undistinguishable.

By using analysis method, we can confirm the coordinator of equilibrium point for a particular set of the three parameters. Then we will implement the RGA and MGAM targeting the known equilibrium point to search for an approximate set of parameters to have a comparison. To obtain the right combination of parameters we want, the result of fitness have to possess a high precision. That is because multiple combinations of the three parameters could achieve a very close result to our target equilibrium point. From the tests, the objective accuracy is shown to be at least 10^{-4}.

Letting $\sigma = 10$, $\beta = 8/3$, and $\rho = 18$, the Lorenz oscillator will move in a limit cycle with very low damping. By letting all the left hand side of the three first order equations to be zero, we can solve the equations and get the solution for the equation $x = -6.733$, $y = -6.733$, and $z = 17$.

The result of simulation for Lorenz oscillator is more obvious for demonstrating the advantageous performance of MGAM (Table 4.2). MGAM is able to find a more accurate approximate coordinator, with precision of 0.0001, and thus its result is approximately the right equilibrium point we expected. The RGA is not able to achieve a close enough result

Table 4.2: Results for Lorenz oscillator equation.

	RGA	MGAM
best fitness	0.0213	0.00013
generation numbers	40000	30000
micro-movement generations	NA	6784
best ρ obtained	17.9631	17.9998
best β obtained	2.7134	2.6655
best σ obtained	15.0401	10.0047

of the equilibrium point, which leads to a different the combination of the three parameters from what we expected.

The strongly nonlinear property of the Lorenz attractor is displaying a very interesting example of optimization. Although, the RGA provides a small fitness value of 0.0213, the corresponding parameter set is far from our actual target. The only way to get a satisfactory result is to find a fitness with a very high precision such as 0.0001, which is shown by MGAM.

Chapter 5

MGAM Optimization on WINDMI Model

The previous chapters of applications simulation repesent some basic tests checking the functionality and convergence of the MGAM. A high-dimensional nonlinear dynamical system called WINDMI model is introduced in this chapter, and the MGAM is implemented to optimize this model.

5.1 WINDMI Model Description

5.1.1 Differential Equations of the Model

The plasma physics based WINDMI model uses a voltage V_{sw} derived from solar wind parameters and the interplanetary magnetic field (IMF) as the input to drive eight ordinary differential equations describing the transfer of power through the geomagnetic tail, the central plasma sheet, the ionosphere, and the ring current [12,13]. The model gives a predicted D_{st} and predicted AL as output. The geometry of the WINDMI model is demonstrated in fig. 5.1.

The largest energy reservoirs in the magnetosphere-ionosphere system are the plasma ring current W_{rc} and the geotail lobe magnetic energy W_m formed by the two large solenoidal current flows producing the lobe magnetic fields. Both these energy components are of the order of a few peta Joules. These energies are stored as particle kinetic energy in the ring current and a lobe inductance L in the case of W_m.

A second current loop is the I_1 R1 FAC current that is associated with the westward auroral electrojet. This current has an associated magnetic energy $\frac{1}{2}L_1 I_1^2$ where L_1 is the self-inductance of the region 1 current loop. The area enclosed by the loop contains magnetic flux Φ_{MI} through mutual inductance M with the larger ($\sim$ 20 times) geotail

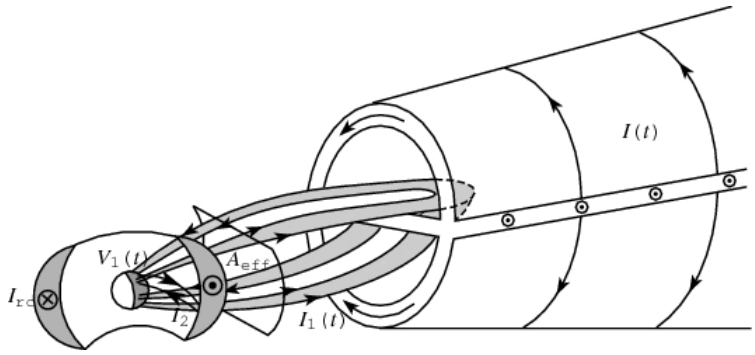

Fig. 5.1: Geometry of the WINDMI model.

cross-field current loop I.

Both current loops have associated voltages V, V_I driven by the solar wind dynamo voltage $V_{\rm sw}(t)$. The resultant electric fields give rise to $\mathbf{E} \times \mathbf{B}$ perpendicular plasma flows whose energies are stored in the capacitances C and C_I. There is parallel kinetic energy $K_{\|}$ due to mass flows along the magnetic field lines.

The energy components V, $K_{\|}$, and p in the central plasma sheet are not shown in fig. 5.1. A_{eff} is an effective aperture for particle injection into the ring current. I_{rc} is the ring current whose energy is $W_{\rm rc}$ given by eq. (5.8). A second current loop is the $I_1(t)$ R1 FAC current associated with the westward auroral electrojet and has the associated voltage V_I. The area enclosed by this loop contains a magnetic flux through mutual inductance with the larger geotail cross-field current loop $I(t)$. The field aligned current at the lower latitude that closes on the partial ring current is designated as I_2.

The high pressure plasma trapped by the reversed lobe magnetic fields gives the thermal energy component $U_p = \frac{3}{2} p \Omega_{cps}$, where $\Omega_{cps} = L_x L_y L_z$ is the volume of the central plasma sheet. The partial ring current I_2 transfers energy along magnetic field lines from the ionosphere to the ring current. The ring current is also energized by particle injection across the effective aperture A_{eff} in the transition region [12]. The result is a set of eight nonlinear ordinary differential equations:

$$
\begin{aligned}
L\frac{dI}{dt} &= V_{\rm sw}(t) - V + M\frac{dI_1}{dt}, && (5.1)\\
C\frac{dV}{dt} &= I - I_1 - I_{\rm ps} - \Sigma V, && (5.2)\\
\frac{3}{2}\frac{dp}{dt} &= \frac{\Sigma V^2}{\Omega_{\rm cps}} - u_0 p K_{\parallel}^{1/2}\Theta(u) - \frac{pVA_{\rm eff}}{\Omega_{\rm cps}B_{\rm tr}L_y} - \frac{3p}{2\tau_E}, && (5.3)\\
\frac{dK_{\parallel}}{dt} &= I_{ps}V - \frac{K_{\parallel}}{\tau_{\parallel}}, && (5.4)\\
L_I\frac{dI_1}{dt} &= V - V_I + M\frac{dI}{dt}, && (5.5)\\
C_I\frac{dV_I}{dt} &= I_1 - I_2 - \Sigma_I V_I, && (5.6)\\
L_2\frac{dI_2}{dt} &= V_I - (R_{\rm prc} + R_{A2})I_2, && (5.7)\\
\frac{dW_{\rm rc}}{dt} &= R_{\rm prc}I_2^2 + \frac{pVA_{\rm eff}}{B_{\rm tr}L_y} - \frac{W_{\rm rc}}{\tau_{\rm rc}}. && (5.8)
\end{aligned}
$$

The nonlinear dynamics of the model traces the flow of the dynamo generated power by electromagnetic and mechanical means through the eight pairs of transfer terms. The remaining terms describe the loss of energy from the magnetosphere-ionosphere system through plasma injection, ionospheric losses, and ring current energy losses. The system of eight ordinary differential equations that make up the model follows the conservation rules of network theory.

In the differential equations, the coefficients are physical parameters of the magnetosphere-ionosphere system. The quantities L, C, Σ, L_1, C_I, and Σ_I are the magnetospheric and ionospheric inductances, capacitances, and conductances, respectively. A_{eff} is an effective aperture for particle injection into the ring current. The resistances in the partial ring current and region2 current I_2 regions are $R_{\rm prc}$ and $R_{\rm A2}$, respectively, and L_2 is the inductance of the region2 current. The coefficient u_0 in eq. (5.3) is a heat flux limiting parameter.

The confinement times for the central plasma sheet, parallel kinetic energy, and ring current are τ_E, $\tau_{\parallel}$ and $\tau_{\rm rc}$. The effective width of the magnetosphere is L_y and the transition region magnetic field is given by $B_{\rm tr}$. The pressure gradient driven current is given by $I_{ps} = L_x(p/\mu_0)^{1/2}$ where L_x is the effective length of the magnetotail.

The pressure unloading function $\Theta(u) = \frac{1}{2}[1 + \tanh u]$, where $u = (I - I_c)/\Delta I$ in eq. (5.3) is specified by a critical current I_c and the interval ΔI for the transition to loss of plasma along newly opened magnetic field lines with a parallel thermal flux $q_{||}$. It changes from zero to unity as a function of I compared to I_c. The unloading function follows from current gradient driven tearing modes or cross-field current instabilities [14].

The parameters are combined appropriately into a vector $\mathbf{P}^{\mathrm{d}}$ where $d = 18$. They can be estimated using semi-analytical techniques, or they can be considered as variables that need to be optimized within physically allowable ranges to fit the data for a given storm. Here we approximated the parameters analytically using the Tsyganenko magnetic field model and then defined a range of allowable values over which each parameter is allowed to vary. In Tables 5.1 and 5.2 we give the calculated estimates and a short description of the major parameters in the WINDMI model (calculations are detailed in [12, 15, 16]). Some parameters listed in Tables 5.1 and 5.2 occur only as combinations, such as the effective aperture A_{eff}, transitional region magnetic field B_{tr}, and the dawn-to-dusk width of the magnetosphere L_y.

5.1.2 Solar Wind Input

The solar wind driving voltage V_{sw} in eq. (5.1) is the input time series for the nonlinear driven-dissipative system. The driving voltage V_{sw} is calculated in two ways. The first is to use the standard rectified vB_s formula, given by $V_{sw} = v_{sw} B_s^{IMF} L_y^{eff}$ where v_{sw} is the x-directed component of the solar wind velocity in GSM coordinates, B_s^{IMF} is the southward IMF component, and L_y^{eff} is an effective cross-tail width over which the dynamo voltage is produced. The second method is to use the formula [18–20] for the coupling of the solar wind to the magnetopause using the solar wind dynamic pressure P_{sw} to determine the standoff distance. The resulting formula for $V_{sw} = V_{sw}(n_{sw}, \vec{v}_{sw}, \vec{B}^{IMF})$ is given by

$$V_{sw}(kV) = 30.0(kV) + 57.6 E_{sw}(mV/m) P_{sw}^{-1/6}(nPa), \tag{5.9}$$

where $E_{sw} = v_{sw}(B_y^2 + B_z^2)^{1/2} \sin(\frac{\theta}{2})$ is the solar wind electric field with respect to the

Table 5.1: WINDMI nominal parameters.

L (90 H)	Inductance of the lobe cavity surrounded by the geotail current $I(t)$. The nominal value is $L = \mu_0 A_\ell / L_x^{\text{eff}}$ in Henries where A_ℓ is the lobe area and L_x^{eff} the effective length of the geotail solenoid. Computation of L as function of the IMF from the Tsyganenko model are given in [16].
M (1 H)	The mutual inductance between the nightside region 1 current loop I_1 and the geotail current loop I.
C (50000 F)	Capacitance of the central plasma sheet in Farads. The nominal value is $C = \rho_m L_x L_z / B^2 L_y$ where ρ_m is the mass density in $kg/m^3, L_x L_z$ is the meridional area of the plasma sheet, L_y the dawn-to-dusk width of the central plasma sheet and B the magnetic field on the equatorial plane. Computations of C are given in [15].
Σ (8 S)	Large gyroradius ρ_i plasma sheet conductance from the quasineutral layer of height $(L_z \rho_i)^{1/2}$ about the equatorial sheet. The nominal value is $\Sigma = 0.1(n_e/B_n)(\rho_i/L_z)^{1/2}$. Computation of Σ is given in [17].
Ω_{cps} (2.6×10^{24} m^3)	Volume of the central plasma sheet that supports mean pressure $p(t)$, initial estimate is $10^4 R_E^3$.
u_0 (4×10^{-9})$/m/kg^{1/2}$	Heat flux limit parameter for parallel thermal flux on open magnetic field lines $q_\parallel = const \times v_\parallel p = u_0 (K_\parallel)^{1/2} p$. The mean parallel flow velocity is $(K_\parallel / (\rho_m \Omega_{cps}))^{1/2}$.
I_c (1.78×10^7 A)	The critical current above which unloading occurs.
$I_{ps}(p)$ and α	The geotail current driven by the plasma pressure p confined in the central plasma sheet. Pressure balance between the lobe and the central plasma sheet gives $B_\ell^2 / 2\mu_0 = p$ with $2 L_x B_\ell = \mu_0 I_{ps}$. This defines the coefficient α in $I_{ps} = \alpha p^{1/2}$ to be approximately $\alpha = 2.8 L_x / \mu_0^{1/2}$.

Table 5.2: WINDMI nominal parameters.

$\tau_{\parallel}$ (10 min)	Confinement time for the parallel flow kinetic energy $K_{\parallel}$ in the central plasma sheet.
τ_E (30 min)	Characteristic time of thermal energy loss through earthward and tailward boundary of plasma sheet.
L_1 (20 H)	The self-inductance of the wedge current or the nightside region 1 current loop $I_1(t)$
C_I (800 F)	The capacitance of the nightside region 1 plasma current loop.
Σ_I (3 mho)	The ionospheric Pedersen conductance of the westward electrojet current closing the I_1 current loop in the auroral (altitude $\sim 100\,\mathrm{km}$, $68°$) zone ionosphere.
R_{prc} (0.1 ohm)	The resistance of the partial ring current.
τ_{rc} (12 hrs)	The decay time for the ring current energy.
L_2 ($8H$)	The inductance of the ring current.
R_{A2} (0.3 ohm)	Resistance of the region 2 footprint in the Auroral Region.
B_{tr} (5×10^{-9}T)	The magnetic field in the transition region.
A_{eff} ($8.14 \times 10^{13}\ m^2$)	The average effective area presented to the geotail plasma for plasma entry into the inner magnetosphere, estimated to be $2R_E^2$.
L_y ($3.2 \times 10^7\ m$)	The effective width of the Alfven layer aperture, estimated to be $5R_E$.
ΔI (1.25×10^5 A)	The rate of turn-on of the unloading function.

magnetosphere and the dynamic solar wind pressure $P_{sw} = n_{sw} m_p v_{sw}^2$. Here m_p is the mass of a proton. The IMF clock angle θ is given by $\tan^{-1}(B_y/B_z)$. The solar wind flow velocity v_{sw} is taken to be approximately v_x. In addition, the position of the ACE satellite introduces a time delay for the solar wind to transit from the L1 point to the nominal coupling region at $X = 10R_E$. This time delay is approximately 1 hour. For this work we use

$$t_d(V, X, Y) = \frac{X - X_0}{V}, \tag{5.10}$$

where t_d is the time delay, $X_0 = 10R_E$, V is the solar wind bulk speed where we get $V = v_x$. The solar wind input voltage is translated to become $V_{sw}(t - t_d)$ at the coupling region.

5.1.3 WINDMI Output

Numerical solution of the eight differential equations gives the state vector $X(t)$ and the associated eight energy components. The Auroral AL index now follows as a magnetic field perturbation ΔB_{AL} from the ambient terrestrial field due to the westward electrojet current I_1 that flows in the E-layer ($\sim 100km$) in the nightside ionosphere. We estimate the relation between I_1 and the AL index by assuming for simplicity that the current I_1 is related linearly to the AL index by a constant of proportionality $\lambda_{AL}[A/nT]$, giving $\Delta B_{AL} = -I_1/\lambda_{AL}$.

To get λ_{AL}, we used the estimated parameters to run the WINDMI model for each of the storms to obtain the ratio between the mean of the AL index and the mean of I_1. The result is shown in Table 5.3. This average value for each storm was then used in all subsequent analysis. The $D_{\rm st}$ signal is given by ring current energy $W_{\rm rc}$ ($\sim 3 - 8 \times 10^{15}$J) through the Dessler-Parker relation, which is

$$D_{\rm st} = -\frac{\mu_0}{2\pi} \frac{W_{\rm rc}(t)}{B_E R_E^3}, \tag{5.11}$$

where $W_{\rm rc}$ is the plasma energy stored in the ring current and B_E is the earth's surface magnetic field along the equator.

Table 5.3: Evaluation of λ_{AL} in A/nT to determine the scaling factor between the westward Auroral Electrojet index AL and the current I_1.

GEM Storm	λ_{AL} with V_{Bs} Input	λ_{AL} with Siscoe Input	Average λ_{AL}
Oct 2000	3193	3358	3275
Apr 2002	2340	2937	2638

5.2 MGAM for WINDMI Model

Because the WINDMI model is a complicated nonlinear dynamic model with large number of dimensions, details about the genetic algorithms (RGA and MGAM) implementation on WINDMI model are described here.

5.2.1 Initialization

The 22 variable coefficients in the WINDMI model are L, β, M, C, Σ, Ω_{cps}, u_0, I_c, A_{eff}, B_{tr}, L_y, τ_E, $\tau_{\|}$, L_I, C_I, Σ_I, L_2, R_{prc}, R_{A2}, τ_{rc}, α, and α_{F}. These parameters are constrained to maximum and minimum physically realizable and allowable values. This results in a 22D search space S over which the optimization is to be performed. Here, a single set of parameters corresponds to a point $s \in S$.

To produce the first generation G_1, we implement the same method described as eq. (1.2).

5.2.2 Evaluation of Fitness and Regeneration

The selection of an appropriate cost function or fitness metric is critical, since the features of an optimized solution depends on the cost function. Different cost functions are used to investigate the quality of solutions returned by the algorithm. Following are the cost functions used in this work:

1. A normalized L2 norm
2. The correlation coefficient
3. The ARV.

The formulas for the norms are:

$$||Y||_{l_2} = \frac{1}{sup|y_i|}\left[\Sigma_i(x_i - y_i)^2\right]^{1/2}. \tag{5.12}$$

The average relative variance (ARV) and correlation coefficient (COR) are calculated here as well. Their formulae, respectively, are

$$ARV = \frac{\Sigma_i(x_i - y_i)^2}{\Sigma_i(\bar{y} - y_i)^2}, \tag{5.13}$$

$$COR = \frac{\Sigma_i(x_i - \bar{x})(y_i - \bar{y})}{\sigma_x \sigma_y}, \tag{5.14}$$

where x_i are model values and y_i are the data values. In order to have a close match between the model output and measured data, ARV should be close to zero. The cost function is calculated for I_1 as x versus the AL Index as y. For the ARV measur, which is less than one, $(1 - ARV) \times 100\%$ of the variation of the data is explained by the model. For a stationary data set, which is not the case for data sets with large storm events, the ARV and COR are related by $COR = 1 - ARV$. Also, a model giving $ARV = 1$ is equivalent to use the average of the data for the prediction.

5.2.3 Crossover

In this work, the process of natural selection is accomplished by simply retaining the best half of a generation forming G_q^{best} based on the fitness metric. To perform crossover at the qth generation, we randomly pair off the best parameter sets in G_q^{best}. Given a pair of parameter sets s^x and s^y, we produce two offspring following the real-valued crossover process described in sec. 1.2.4.

5.2.4 Mutation with Automatic Sinusoidally Changing Mutation Rate

Since we are implementing the RGA and MGAM to optimize the WINDMI model, the automatic sinusoidally changing mutation rate is suitable for the purpose to increase the opportunity for individual solutions to search for an undiscovered area. The simulation will

implement the mutation rate format described as sec. 1.4. A 120 generations long period with no improvement of best fitness, will activate the sinusoidal mutation rate.

5.2.5 Ending the Algorithm

The convergence condition of the optimization for WINDMI application could be achieved in two ways. One is to set a fixed number of reproduction generations to run. The other is to set a fixed best finess as the goal. The simulations in this work are all using the first method as the ending criteria. With an expected numbers of reproduction generations, a clear comparison of performance can be shown between any algorithms.

5.3 Simulation Result with Comparison and Analysis

5.3.1 Optimization Objective

Since the genetic algorithm is a random search and optimization algorithm, the convergence will spend a period of time (many generations of reproductions) to finish. The result of the optimization from less generations might be unstable. The best fitness value will turn out to be different depending on different cases of tests. Our target here is to focus on a good approximate of the minimum within reasonable reproduction generations.

For the WINDMI model, the previous work has tested the cost function minimum after 50,000 generations, which spend over 36 hours to finish [21]. The $ARV_{al} = 0.45$ and $ARV_{dst} = 0.19$ is the best value ever achieved of October 4, 2000 storm case. For April 22, 2002 storm, $ARV_{al} = 0.62$ and $ARV_{dst} = 0.38$ is the best obtained. Spending a long period of time to converge will be meaningless for the space weather system, since the space weather is changing swiftly in hours scale. Usually an ARV_{al} less than 0.52 within 20,000 generations can be consider a good result. Here we compare the MGAM with the RGA for two different storm events, in order to check the performance of the MGAM implemented on the WIDNMI model.

5.3.2 Result Figures Comparison

The RGA has a faster pattern of crossover and mutation, which could be more efficient than binary GA. Our version of RGA could reach a relatively good result of optimization as $ARV_y < 0.50$ and $ARV_{dst} < 0.26$ within $16,000$ to $18,000$ generations. The MGAM is expected to have progress over the RGA by its improved search mechanism.

Considering there are three main items in the list of comparison, ARV_{al}, ARV_{dst}, and generation numbers, it will be clearer and convinient to employ a weighting function here. Equation (5.15) is providing us a weighting index P_{ind}, which is a conclusive fitness value of the measured performance.

$$P_{ind} = \frac{\alpha A_{al} + \beta A_{dst} + \gamma \frac{Gen_{MM} + Gen_{rep}}{Gen_{RGA}}}{\alpha + \beta + \gamma}, \tag{5.15}$$

where α, β, and γ are the weight coefficients for the items of ARV_{al}, ARV_{dst}, and generation numbers. Gen_{MM} is the micro-movement iteration number of the MGAM (also called the generation number of micro-movement). Gen_{rep} is the reproduction generation number of the MGAM. Gen_{RGA} is the largest amount of total generations of RGA. To make a fair comparison, the generation number of micro-movement should also be counted, considering the amount of resources are consumed by the micro-movement phase. This equation produces the final fitness of the result data. A smaller value of P_{ind} indicates a better optimzation overall.

Usually we treat the three items with the same importance. The final fitness results in Tables 5.4, 5.5, and 5.6 are calculated when all α, β, and γ equal to one.

For the storm event in October 4th of 2000, Tables 5.4 and 5.5 show us that the MGAM has a overall better weight function value than the RGA. Comparing different reproduction generation numbers of the RGA, the MGAM is providing us a relative good and stable performance with less total generation numbers. In fig. 5.2, the whole number of generations is 15377, which is less than 16000 generations of the RGA test in fig. 5.3, while it produces an obvious fitter convergence.

For the storm event in April 17, 2002 (Table 5.6), the time span of that event is much

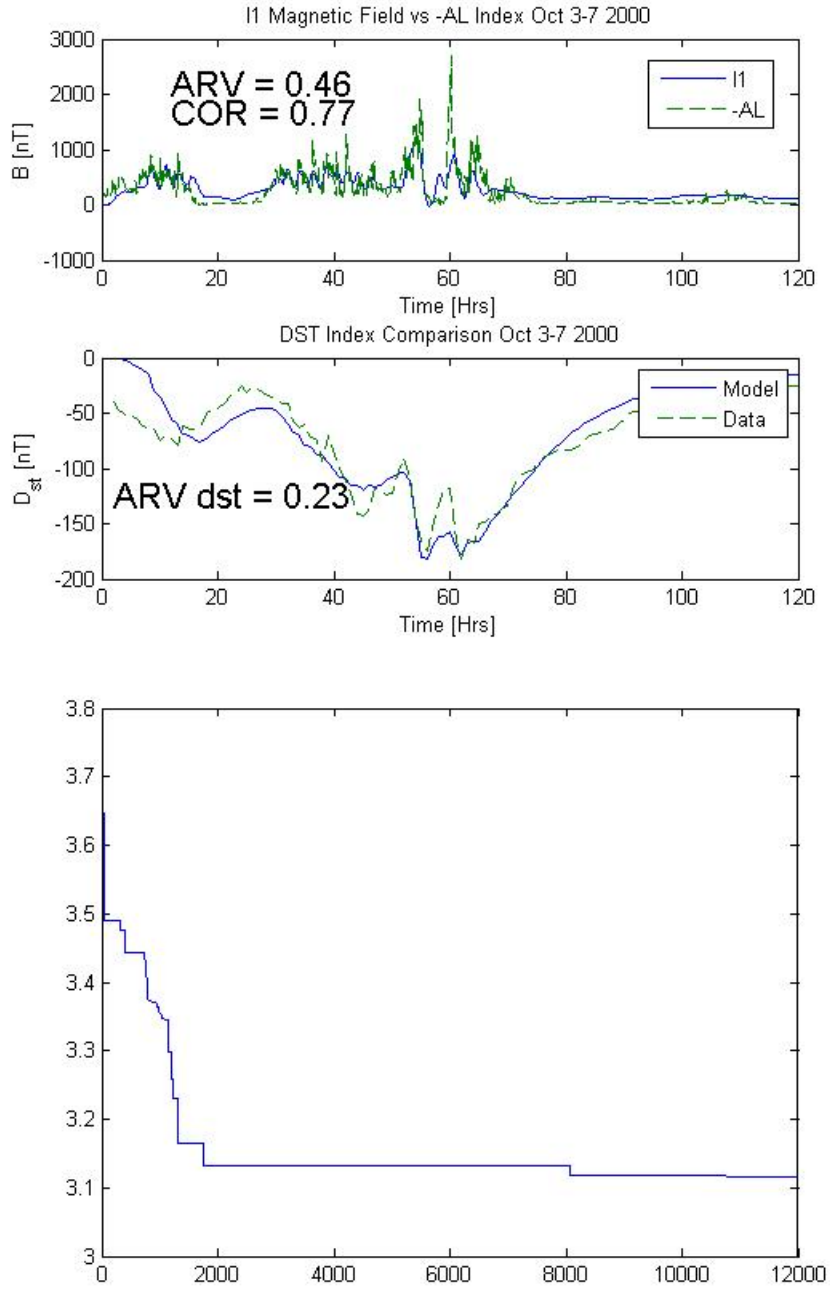

Fig. 5.2: MGAM in 12000 generations for WINDMI storm event in October 2000.

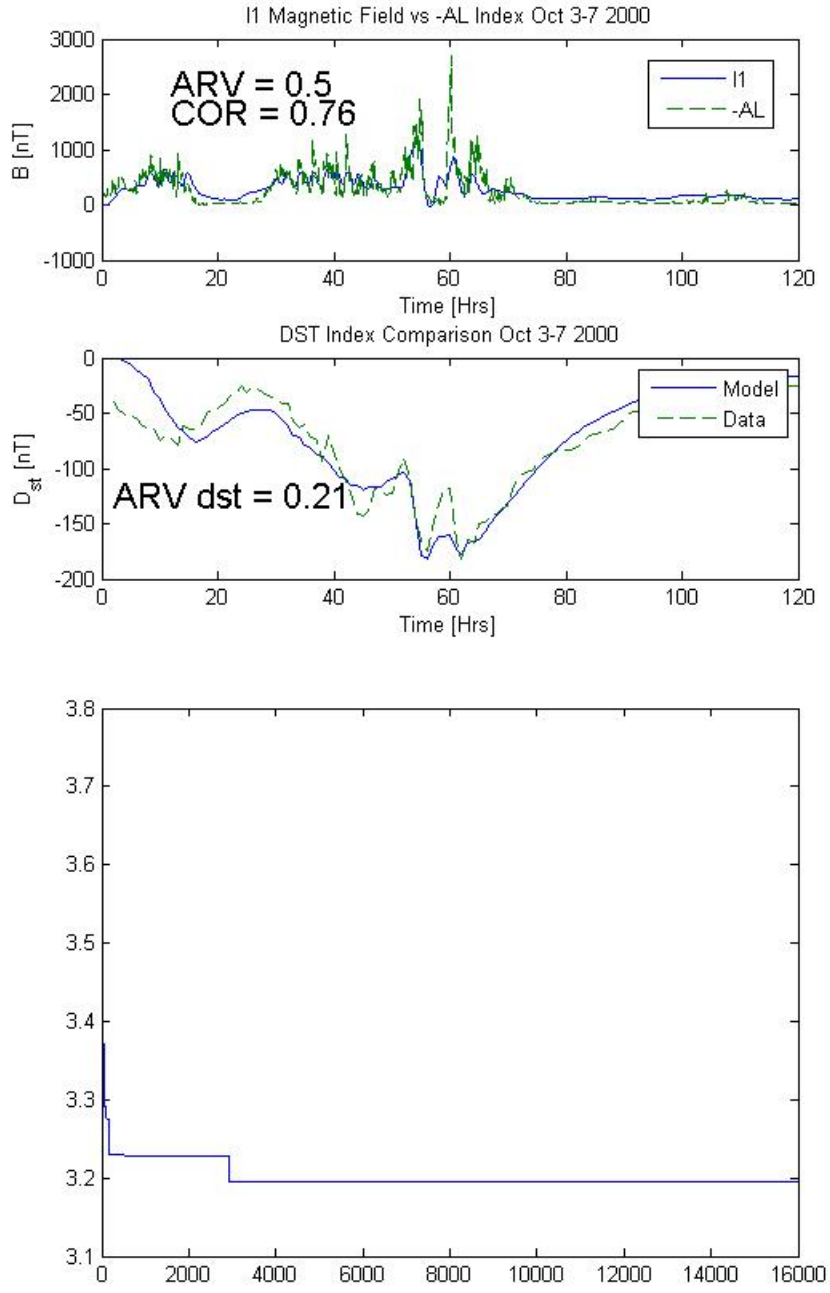

Fig. 5.3: RGA in 16000 generations for WINDMI storm event in October 2000.

Table 5.4: Results for WINDMI storm event in October 2000 with 16000 generations of RGA.

test functions	**RGA**				**MGAM**				
cases	Al	DST	Gen_{RGA}	P_{ind}	AL	DST	Gen_{rep}	Gen_{MM}	P_{ind}
1	0.51	0.25	16000	0.587	0.51	0.22	12000	1951	0.534
2	0.48	0.22	16000	0.567	0.48	0.23	12000	2117	0.531
3	0.48	0.21	16000	0.563	0.51	0.24	12000	2155	0.545

Table 5.5: Results for WINDMI storm event in October 2000 with 18000 generations of RGA.

test functions	**RGA**				**MGAM**				
cases	Al	DST	Gen_{RGA}	P_{ind}	AL	DST	Gen_{rep}	Gen_{MM}	P_{ind}
1	0.48	0.25	18000	0.577	0.50	0.24	12000	2734	0.520
2	0.47	0.22	18000	0.563	0.48	0.23	12000	3017	0.515
3	0.49	0.22	18000	0.570	0.46	0.23	12000	3377	0.515

longer than the October 2000 case. In order to reduce the time consumed, we will restrict the reproduction within 16000 for both RGA and MGAM. The result contrast of this storm event is not as obvious as the October, 2000 case. This is because the optimization for this storm event has already become saturated within less than 20000 generations. However, with around 1000 to 700 less generations, we are still able to get a slightly better result from the MGAM, which is also shown in figs. 5.4 and 5.5.

With the results above, the MGAM is proved to be capable of solving complicated application of nonlinear dynamical system, and its performance is shown to be better than the RGA as expected.

Table 5.6: Simulation results comparison for WINDMI storm event in April 2002.

test functions	**RGA**				**MGAM**				
cases	Al	DST	Gen_{RGA}	P_{ind}	AL	DST	Gen_{rep}	Gen_{MM}	P_{ind}
1	0.72	0.39	16000	0.670	0.63	0.40	12000	3757	0.672
2	0.64	0.40	16000	0.680	0.62	0.40	12000	3135	0.655
3	0.64	0.39	16000	0.677	0.62	0.39	12000	3417	0.658
4	0.64	0.39	16000	0.677	0.63	0.39	12000	2930	0.651
5	0.63	0.40	16000	0.677	0.63	0.40	12000	3580	0.668

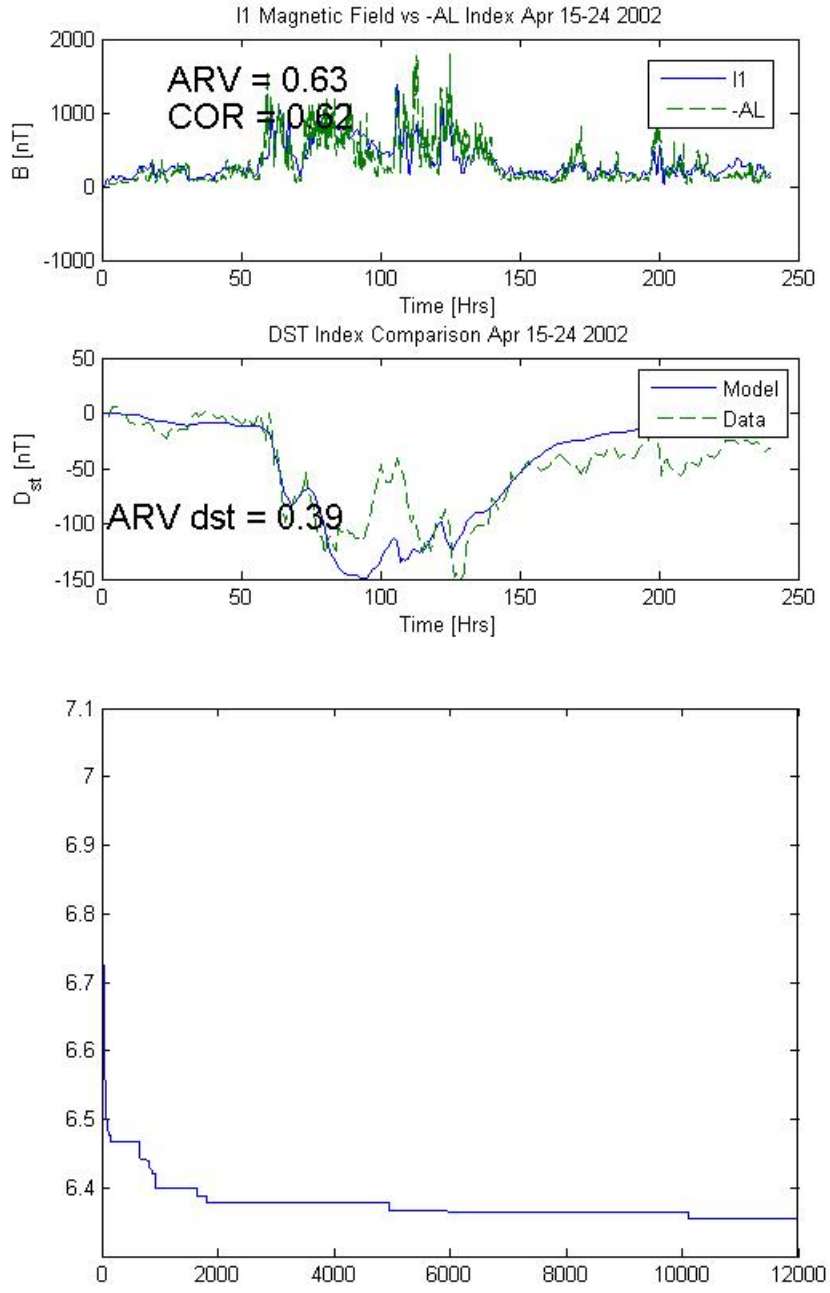

Fig. 5.4: MGAM in 12000 generations for WINDMI storm event in April 2002.

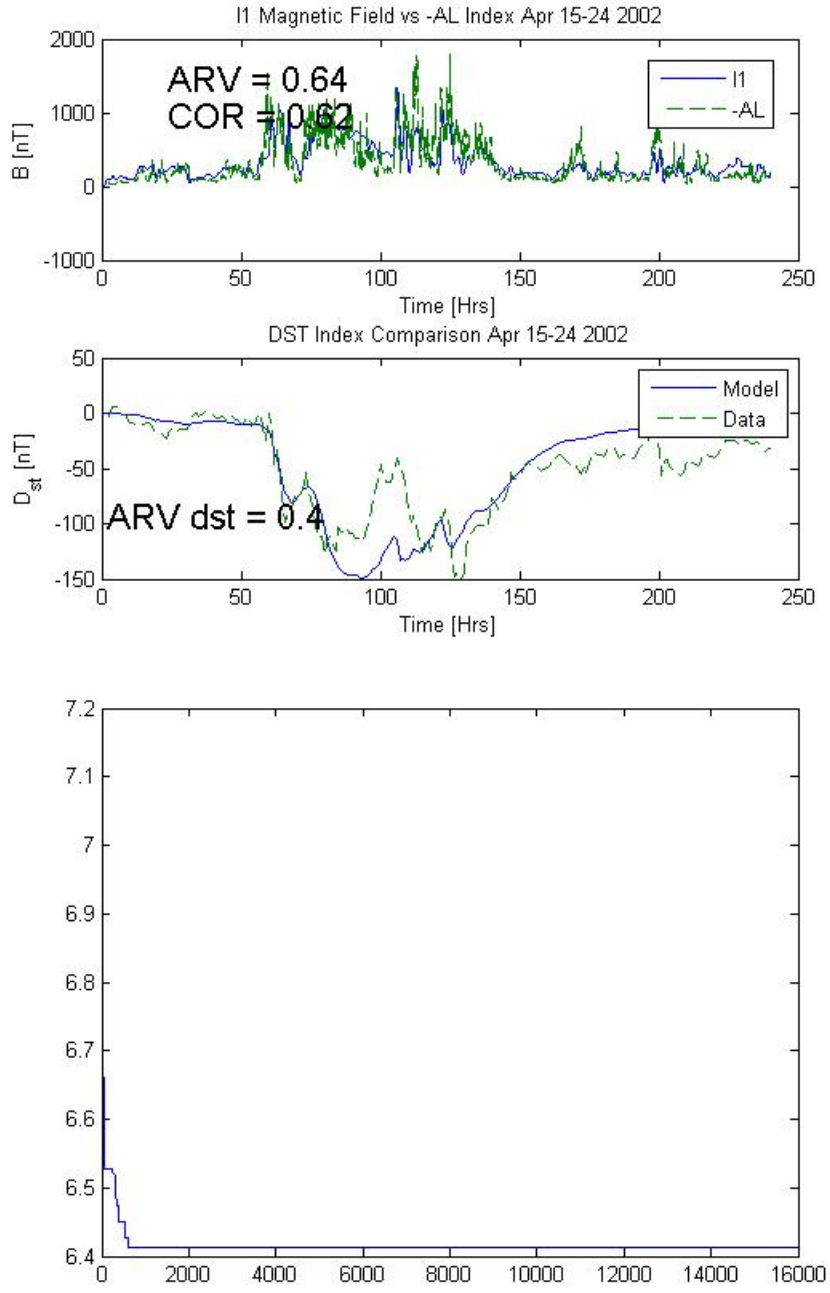

Fig. 5.5: RGA in 16000 generations for WINDMI storm event in April 2002.

Chapter 6

Conclusion and Future Work

In this work, we purpose a modified version of genetic algorithm with micro-movement (MGAM). We incorporate a learning, or experience gained by individuals in a population at each generation. This will increase efficiency by reducing the reliance on improvement only from generation to generation via offspring created through crossover and mutation. We compare the performance of both the real-valued GA and MGAM for five standard test functions (Tripod (3D), Alpine (3D), Parabola (5D), Griewank (3D), and Ackley (3D)). We also have two nonlinear dynamical systems implemented as our test applications, which are Van der Pol oscillator and Lorenz attractor. The most important and interesting application for the MGAM is the WINDMI model of space weather system, which have a great complication of nonlinear related variables and a large size of variable dimensions (22D). The MGAM is shown to be competent for optimizing all the applications above, as well as having an advantageous fitness searching ability over RGA.

Our future work will implement the MGAM on more interesting applications, such as other space weather models and strategic game theory. We are also looking for other methods of movement patterns to optimize our current algorithm.

References

[1] N. Chaiyaratana and A. Zalzala, "Recent developments in evolutionary and genetic algorithms: theory and applications," in *Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA 97 (Conf. Publ. No. 446)*, pp. 270–277, 1997.

[2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks (ICNN95)*, pp. 1942–1948, 1995.

[3] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Boston, MA: Addition-Wesley Longman Publishing Company, 1989.

[4] J. H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: The University of Michigan Press, 1975.

[5] H. Li, "A novel hybrid real-valued genetic algorithm for optimization problems," in *Computational Intelligence and Security International Conference*, pp. 91–95, 2007.

[6] T. Chen, Y. Wang, L. Pang, J. Sun, and J. An, "A new hybrid genetic algorithm and its application to the temperature neural network prediction in transverse flux induction heating (TFIH)," in *Automation Congress*, pp. 1–4, 2008.

[7] M. Clerc, *Particle Swarm Optimization.* Newport Beach, CA: ISTE Ltd., 2006.

[8] Y. Rahmat-Samii, "Genetic algorithm(ga) and particle swarm optimization(pso) in engineering electromagnetics," pp. 1–5. Applied Electromagnetics and Communications, 2003.

[9] E. A. Grimaldi, F. Grimaccia, M. Mussetta, P. Pirinoli, and R. E. Zich, "A new hybrid genetical-swarm algorithm for electromagnetic optimization," in *3rd International Conference on Computational Electromagnetics and Its Application*, p. 157160, 2004.

[10] H. K. Khalil, *Nonlinear Systems.* New York: Macmilan Publishing Company, 2002.

[11] F. C. Moon, *Chaotic and Fractal Dynamics: An Introduction for Applied Scientists and Engineers.* New York: John Wiley & Sons, Inc., 1992.

[12] I. Doxas, W. Horton, W. Lin, S. Seibert, and M. Mithaiwala, "A Dynamical Model for the Coupled Inner Magnetosphere and Tail," *IEEE Transactions on Plasma Science*, vol. 32, no. 4, pp. 1443–1448, Aug. 2004.

[13] W. Horton, M. Mithaiwala, E. Spencer, and I. Doxas, "WINDMI: A Family of Physics Network Models for Storms and Substorms," in *Multi-Scale Coupling of Sun-Earth Processes*, A. Lui, Y. Kamide, and G. Consolini, Eds. Maryland Heights, MO: Elsevier Publish Company, 2005.

[14] P. Yoon, A. Lui, and M. Sitnov, "Generalized lower-hybrid drift instabilities in current sheet equilibrium," *Physics of Plasmas*, vol. 9, no. 5, pp. 1526–1538, 2002.

[15] W. Horton and I. Doxas, "A low-dimensional energy-conserving state space model for substorm dynamics," *Journal of Geophysical Research*, vol. 101, no. A12, pp. 27, 223–27,237, Dec. 1996.

[16] W. Horton and I. Doxas, "A low-dimensional dynamical model for the solar wind driven geotail-ionosphere system," *Journal of Geophysical Research*, vol. 103, no. A3, pp. 4561–4572, Mar. 1998.

[17] W. Horton and T. Tajima, "Collisionless conductivity and stochastic heating of the plasma sheet in the geomagnetic tail," *Journal of Geophysical Research*, vol. 96, no. A9, pp. 15,811–15,829, 1991.

[18] G. L. Siscoe, G. M. Erickson, B. U. O. Sonnerup, N. C. Maynard, J. A. Schoendorf, K. D. Siebert, D. R. Weimer, W. W. White, and G. R. Wilson, "Hill model of transpolar potential saturation: comparisons with MHD simulations," *Journal of Geophysical Research*, vol. 107, no. A6, p. 1075, 2002.

[19] G. L. Siscoe, N. U. Crooker, and K. D. Siebert, "Transpolar potential saturation: roles of region-1 current system and solar wind ram pressure," *Journal of Geophysical Research*, vol. 107, no. A10, p. 1321, 2002.

[20] D. M. Ober, N. C. Maynard, and W. J. Burke, "Testing the Hill model of transpolar potential saturation," *Journal of Geophysical Research*, vol. 108, no. A12, p. 1467, 2003.

[21] E. Spencer and A. Rao, "Evaluation of solar wind- magnetosphere coupling functions during geomagnetic storms with the windmi model," *Journal of Geophysical Research*, vol. 114, no. A02206, 2009.