IMPLICATION OF FUZZY LOGIC IN GA FOR SOLVING MATHEMATICAL PROBLEMS

A. K. M. Mahbubur Rahman, Md. Bashir Uddin Khan and Md. Al-Amin Bhuiyan
Department of Computer Science and Engineering
Jahangirnagar University, Savar, Dhaka, Bangladesh
E-mail: mahboob263@yahoo.com, bashir_273@yahoo.com, alamin@juniv.edu

Abstract: *This paper presents a robust and novel approach to find the roots of mathematical equations using genetic algorithm. A modified approach of genetic algorithm has been provided where fuzzy logic has been used for chromosomes selection. A GA formula has also been presented to solve equations with multiple iterations in order to justify the error versus iteration graph. Finally, a comparison between the results of solving the same equation by GA with fuzzy logic and those without fuzzy logic has been furnished. Experimental results demonstrate that GA with fuzzy logic is more successful and efficient than that without fuzzy logic for solving equations.*

Keywords: *Genetic Algorithm, Fuzzy Logic, Mutation, Mating, Chromosome, Fitness evaluation function.*

1. Introduction

Mathematical equations are the equations containing the terms with trigonometric, algebraic, exponential, logarithmic and so on expressions. Many analytical/iterative methods are used to solve mathematical equations. Although these methods are capable of solving many mathematical equations they, nevertheless, suffer from many common drawbacks. Usually, mathematical equations have many solutions in a given range, and the analytical methods are not able to find all these roots in a given interval, even when they find several solutions, it is not possible to conclude that the given method has found the complete set of roots/solutions, and has not missed any particular solution. Also, these methods fail in case of discontinuous functions [1]. Hence, though these methods may work very well in some situations, they are not general in nature and need a lot of homework from the analyst.

An analysis of some common methods used for solving transcendental equations, their disadvantages and cases of failures are discussed below.

a) Newton Raphson method: This is a widely used method for solving transcendental equations. The method makes use of the slope of the curve at different points. Therefore, if the function is non differentiable at points or has a point of inflexion, the method is not able to find the root. Secondly, if the function changes its slope very quickly (frequently achieves slope of zero), or is discontinuous, cannot be solved by this method. If the function is discrete, the derivative has no meaning for it and this method cannot be used. Also there is no straightforward way to find all the roots in an interval or even ascertain the number of roots in the interval.

b) Bisection method: This method needs two points on the graph such that $f(a)*f(b)<0$. There is no straightforward analytical method to find these points. Another problem lies in choosing the distance between the points a and b. For this method to work, a and b should be close enough, such that the function behaves monotonously in these limits. At the same time, a small difference in values of a and b makes it difficult to search the sample space.

c) Method of False Position: This method suffers from same problems as Bisection method. Hence it can be concluded that analytical methods cannot find all the roots of a transcendental equation reliably. These limitations of analytical procedures can be overcome by using global optimization techniques like genetic algorithm (GA).

The Genetic Algorithm (GA) is a stochastic search method based on the mechanics of

natural selection and genetics analogous to natural evolution. The GA has been employed in a wide variety of problems related to pattern recognition, image processing, medical image registration, image segmentation, contour recognition and so on. A fair amount of research work has been found in literature for the solution of mathematical problems using GA. Kavanagh and Kelley have solved some non-linear equations using GA [1]. P.C. Barman and R. Ahmed have given a comparison of GA and bisection method in the numerical solution of transcendental equations [2]. S. Shahid, M.N. Bhuiyan and M. M. Haque have solved some non-linear equation using GA with dynamic mutation rate [3]. Almost all of the papers found in literature use GA to solve mathematical equations in traditional way.

This paper investigates the application of genetic algorithm to search for the roots of mathematical equations. We use iteratively fitness values to select chromosomes using roulette wheel but with fuzzy inference. The implication of fuzzy logic for the "survival of the fittest" chromosome selection procedure has proliferated the solution process with more success and efficiency.

2. Genetic Algorithm

Genetic algorithms are a class of algorithms inspired by evolution. These algorithms encode solutions to a specific problem on a simple chromosome like data structure and apply recombination operators to these structures so as to preserve critical information [3].

Central to the idea of GA is a population of individuals, each represents a possible solution to the given problem [2]. Each individual, known as chromosome, usually represented by a bit string consisting of 0s and 1s, is assigned to a fitness value based on how good its solution to the problem is. The individuals then evolve through successive iterations called generations [4]. During one generation, highly fit individuals are given the opportunity to mate with other individuals in the population. Since the least fit individual in the population are less likely to get selected for mating, they disappear from future generations. As a result, the population of individuals converges to an optimal solution to the problem.

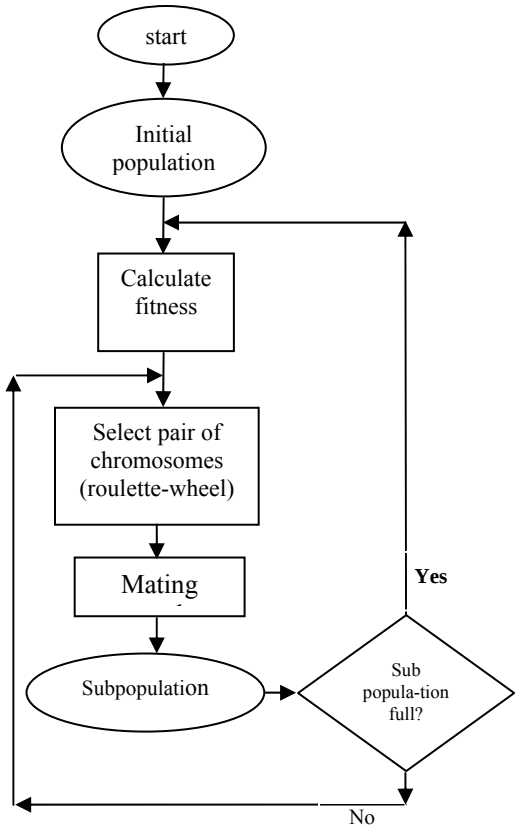

Figure 1. Flowchart for genetic algorithm.

2.1 GA for the present problem

To apply GA, let the transcendental equation is $f(x) = 3 + 2x - \cos x$. There is no strict rule for encoding the solutions to the problem. Generally binary coded solutions are used, though lately, real coded chromosomes are also being used. We have used binary encoding for this research. The advantage of using binary encoding is that the maximum number of schemes is investigated [5]. An approximation has been taken that the root of the above equation exists within the range [1.2,1.8].

2.1.1 Chromosome Encoding

The chromosomes are represented by binary vectors to represent the real values of the x. The length of the vectors depends on the required precision [6]. This means that minimum 10 bits are required as a binary vector to represent each chromosome. We use 20 bits for each chromosome for better effect of crossover and mutation.

The binary string (c_0, c_1, c_2, …,c_{19}) maps into a real number x.

a) Convert the binary string $<c_0, c_1, c_2, \ldots, c_{19}>$ from the base 2 to base 10.
i=19
$<c_0, c_1, c_2, \ldots, c_{19}>_2 = (\sum c_i .2^i)_{10} = x'$
i=0
Find a corresponding real number x:
$x = 1.2 + x'.(1.8 = 1.2)/(2^{20} - 1),$
(1)
where 1.2 is the left boundary of x and 1.8 is the right boundary of x. The chromosomes (00000000000000000000) and (11111111111111111111) represent boundaries of the domain 1.2 and 1.8, respectively.

2.1.2 Initial population

The initialization process is very simple. Here we assume that 20 chromosomes are used as population. A population of chromosomes has been created as follows

- Generate 20 random numbers in the range [0, (2^{20}-1)].
- Convert each random number into 20 bits binary vector and assign each binary vector to one of 20 chromosomes.
- Now each chromosome is a binary vector of 20 bits.

2.1.3 Evaluation function

The evaluation function for a chromosome c is equivalent to function f; evaluate(c) = f (x), where
x=1.2+ binarytodecimal(c).(1.8-1.2)/(2^{20} -1). (2)

In order to identify the best individual root (*real* x) during the evolutionary process, a function needs to assign a degree of fitness to each chromosome in every generation. So in order to determine largest region where the best root lies in that region, we have to compute fitness of particular chromosome.

The fitness of a chromosome is defined as the function that represents that how much $f(x)$ is close to zero. That is, for each chromosome n whose corresponds to real input value x, fitness function is defined as follows:

For each chromosome, n=0,1, …,19 compute

$$f(x_n) = 3 + \cos(x_n) - 2x_n \quad (3)$$

Add all values

$$\text{sum} = \sum_{n=0}^{19} absolute(1/f(x_n)) \quad (4)$$

Now compute fitness values for each chromosomes

$$\text{Fitness Area}(n) = \frac{absolute(1/f(x_n))}{\text{sum}} \times 100\% \quad (5)$$

where x_n is the real value corresponding to the n-th chromosome.

Fitness area represents the fitness of the root(real value of x). That is, absolute[$f(x_n)$] is more closer to zero, x_n is more close to the actual final root. So it has been shown that the larger fitness area is the result for the more fittest root. Thus survival of the fittest is maintained.

2.1.4 Selection

Good chromosomes that contribute their gene-inherited knowledge to breed for the next generation are chosen. Here we use conventional elitist selection scheme to select an elitist chromosome with the highest fitness value, which is copied directly into the new population of next generation [7]. The other chromosomes are selected by roulette-wheel selection process, where the selection probability of each individual is proportional to its fitness value. In this process on a wheel, all the samples are placed allotting them space proportional to their fitness. N markers are put around the wheel, where N is the size of the new population. The wheel is spun and samples under each marker are selected [1].

2.1.5 Cross-over

This operator randomly chooses a crossover point where two parent chromosomes 'break', and then exchanges the chromosome parts after that point [2]. As a result, two offspring are generated by combining the partial features of two chromosomes. If a pair of chromosomes does not cross over, then the chromosome cloning takes place, and the offspring are created as exact copies of each parent [8]. Here we have studied single point cross-over, two point cross-over and uniform cross-over operators. The cutting points are selected randomly within the chromosome for exchanging the contents. In this experiment, the

cross-over rate was chosen as 0.7 for all cases. For example, the strings 10000100 and 11111111 could be crossed over after the third locus in each to produce the two offspring 10011111 and 11100100.

2.1.6 Mutation

Mutation, which is rare in nature, represents a change in the gene and aids us in avoiding loss of genetic diversity. Its role is to provide a guarantee that the search algorithm is not trapped on a local optimum.

This operator alters a randomly selected gene of chromosome with a very low probability, p_M. For each chromosome, generate a random value between [0,1]. If the random value is less than p_M, then choose a bit at a random location to flip its value from 0 to 1, or 1 to 0. The mutation rate for our method was chosen as 0.05. For example, the string 00000100 might be mutated in its second position to yield 01000100. Mutation can occur at each bit position in a string with some probability, usually very small (e.g.,0.001) [9].

2.1.7 Replacement

After generating the subpopulation (offspring), two representative strategies have been used for the replacement of old generation [2].

3. Fitness Evaluation Fuzzy

This section introduces a new concept for the evaluation of the fitness value of each chromosome.

The fuzzy logic based GA has been designed with the following notions:

1. Create a fuzzy function (S-function) for a set of chromosomes.
2. Now rating the fitness area based on the fuzzy function values to 7 fuzzy rating such as Extremely good (more than 80% area), Very good, Good, More or less good, Not so good, Not bad, Bad (less than 5% area).
3. Extremely good is for the highest fitness area (more than 80% area).
4. Bad is for lowest fittest area (less than 5% area).
5. Other ratings are covered for intermediate areas.
6. Now when a selection is made after roulette-wheel spun, if bad fittest area is selected, the selection is rejected and again roulette-wheel will be spun for the selection. The process is repeated until other than bad fittest area is selected.
7. At the first iteration all area can be badly fittest. So only for 1st iteration above rule is not followed (General rule followed.).

4. Experimental Results

The effectiveness of this method has been justified over some experiments. Experiments were carried out on a Pentium IV 1.3 GHz PC with 256 MB RAM. The algorithm has been implemented in Visual C++. In any iteration, the evaluated values of the given equation and corresponding fitness values are depicted against chromosomes.

The survival value i.e., evaluated value of the function and the fitness of each individual chromosome for the first and second iterations are shown in Fig. 2, and Fig. 3, respectively.

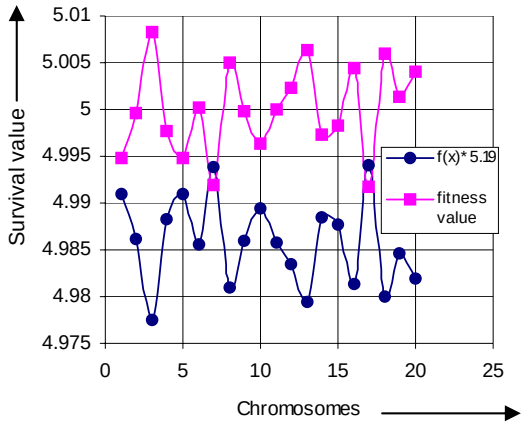

Figure 2. Survival value of chromosomes (1st iteration)

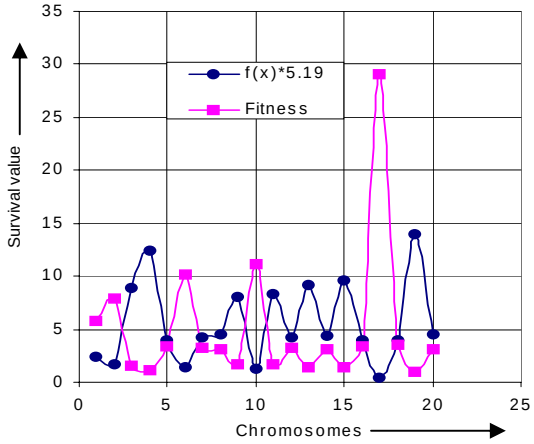

Figure 3. Survival value of chromosomes (2nd iteration).

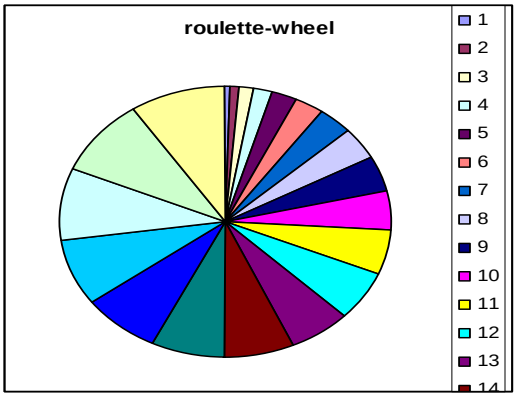

Figure 4. Roulette-wheel for first iteration.

The roulette-wheel construction for first iteration is shown in Fig. 4. Here the fittest root is: 1.201123 and largest area of the fittest root is 5.0083%. For the second iteration, fittest root is: 1.553666, and the largest area=29.12856%. Since the same areas correspond to same roots so in this experiment the sum of the same areas is taken into consideration to find the fittest root.

The error versus iteration graph for 100 iterations is furnished in Fig. 5, where error at any iteration i is defined as:

$$\text{Error}(i) = \text{abs}(\text{Th\_root}(i) - \text{Exp\_root}(i)), \quad (6)$$

where abs is the absolute function, Th_root(i) and Exp_root(i) are the theoreal and experimental values of the root at iteration i.

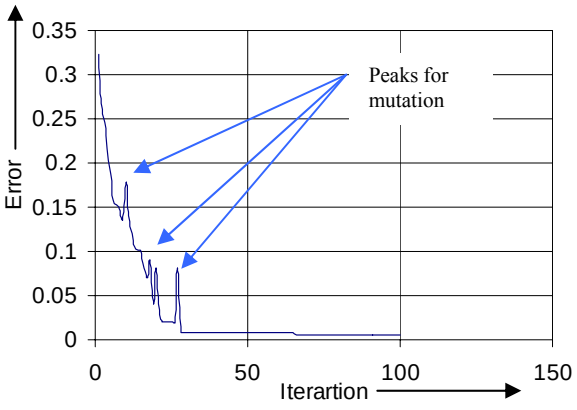

Figure 5. Error versus iteration graph.

From the resulting curve, we have found that the final root becomes stable approximately at 66th iteration and minimum error has been obtained over there. After 66th iteration, we have got more accurate final root as: 1.518475. The uneven peaks are caused by the mutation operation. By selecting appropriate position for mutation operation, we can avoid the local optimal value and proceed to global optimal value.

Now results of GA with fuzzy logic are furnished. The survival values for the 1st and 2nd iterations are shown in Fig. 6 and Fig. 7, respectively. For the first iteration, the graph is similar to that of general rule for GA. The fittest root is: 1.201, and largest area=5.005%. After 2nd iteration, the fittest root is: 1.696774 and largest area=15.84478%. The error versus iteration graph for the GA with fuzzy is shown in Fig. 8, which reveals that the final root becomes stable approximately at 40th iteration and the minimum error (.000432) has been obtained. After 40th iteration we have got more accurate final root as: 1.523167.

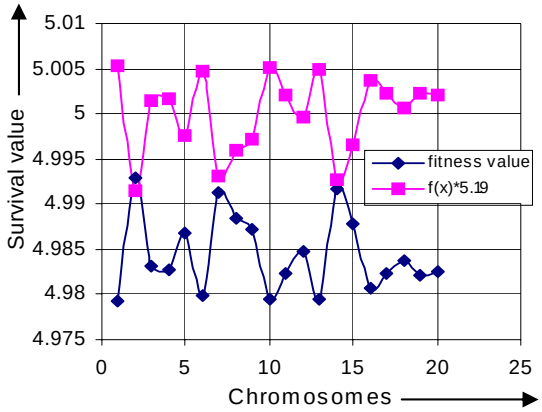

Figure 6. Survival value of chromosome for 1st iteration.

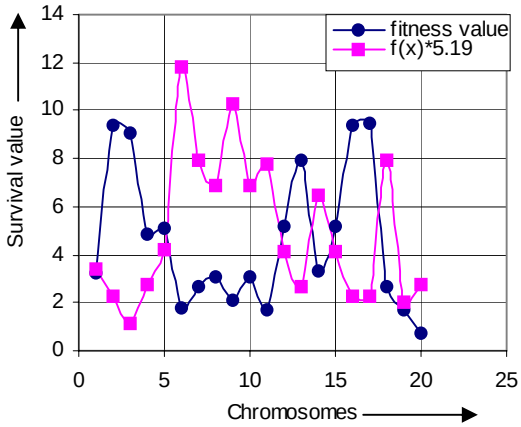

Figure 7. Survival value of chromosome for 2nd iteration.

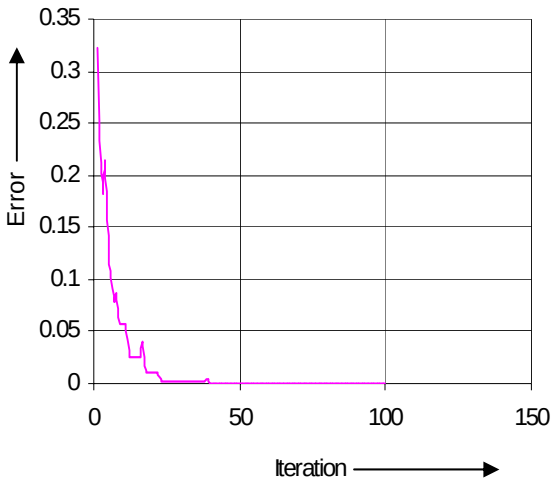

Figure 8. Error versus iteration graph for GA with fuzzy logic.

Experimental results demonstrate that GA with fuzzy logic provides much more accurate solution for transcendental equation. And also finds solution more quickly. A comparative graph for GA with fuzzy logic and without fuzzy logic is depicted in Fig. 9. It justifies that implication of fuzzy logic in GA to solve equations is more effective.

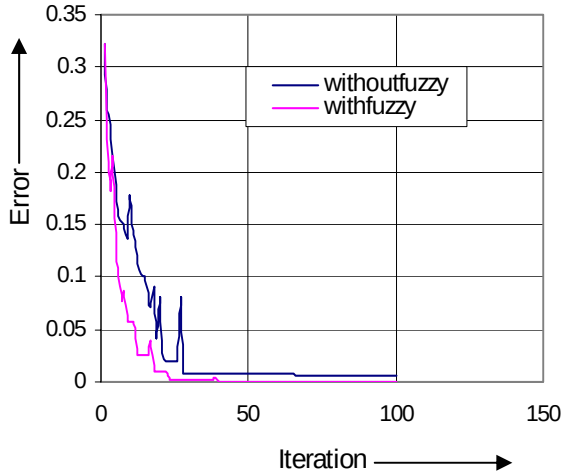

Figure 9. Comparison for error versus iteration

Mutation plays a great role on GA. We know that the MSB of a bit string has highest positional value and the LSB has lowest positional value. Effects of mutation operation in any bit except LSB and MSB have been justified over some experiments. The error versus iteration curve for GA with fuzzy logic is furnished in Fig. 10.

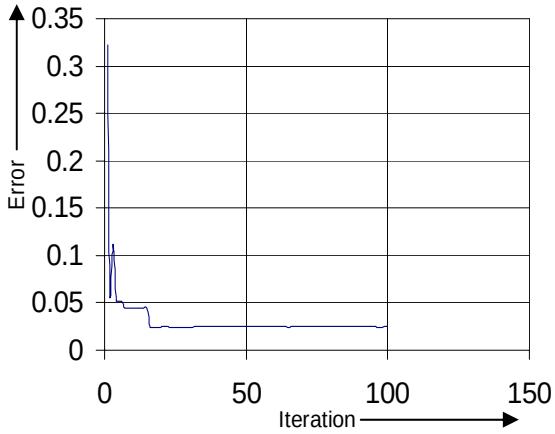

Figure 10. Error versus iteration graph for GA with fuzzy logic (no mutation at MSB or LSB).

Effects of mutation operation in any bit except LSB and MSB have been justified over some experiments and the result is furnished in Fig. 11. From this comparison curve, it reveals that mutation at MSB or LSB has significant effects to find a global optimal value.

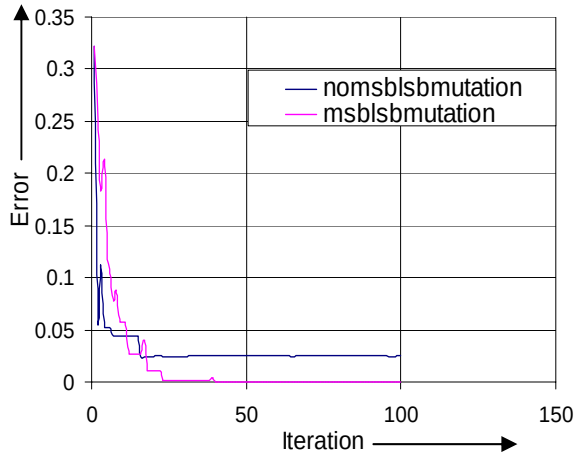

Figure 11. Error versus iteration comparison curve for GA with fuzzy.

5. Conclusion

This paper illustrates the implication of fuzzy logic in genetic algorithm for the selection of chromosome. Experiments were conducted to justify the expected results. The results reveal that GA with fuzzy logic provides more accurate results and that the fuzzy logic is more efficient way to find roots of mathematical equations. This paper also justifies the influences of mutations at MSB and LSB and

finds that mutation at MSB or LSB leads to dramatic change in the result. Our next target is to solve the real world mathematical equations using fuzzy logic.

References

[1] K. R. Kavanagh and C. T. Kelley, "Pseudo-transient Continuation for Nonsmooth Nonlinear Equations", *SIAM J. Numer. Anal., vol. 43*, 2005, 1385-1406.

[2] M. J. Uddin, A. M. Mondal, M. H. Chowdhury and M. A. Bhuiyan, "Face Detection using Genetic Algorithm", *Proceedings of the 6th ICCIT*, 2003, Dhaka, Bangladesh, pp. 41-46.

[3] M. A. Bhuiyan, V. Ampornaramveth, S. Muto, H. Ueno, "Face Detection and Facial Feature Localization for Human-machine Interface", *NII Journal, Vol. 5*, 2003, pp. 25-38.

[4] S. Kumar, E. David, M. Pelikan, *"IlliGAL report", No. 2001013*, January, 2001.

[5] M. Buulmer, *The mathematical theory of quantitative genetic*, Oxford Press, 1st Edition, 1980.

[6] D. Goldberg, K. Deb, and J. Clark, "Genetic Algorithms, Noise, and the Sizing of Populations", *Complex Systems, vol. 6*, 1992, pp. 333-362.

[7] D. Goldberg, *Evolutionary Design by Computers*, 1999, pp. 105-118.

[8] B. L. Miller, and D. Goldberg, "Genetic Algorithms, Selection Schemes and the Varying Effects of Noise", *Evolutionary Computation, vol. 4*, 1999, pp. 113-131.

[9] B. L. Miller, *Noise, sampling, and efficient genetic algorithms*, Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 1997.