IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 3, March 2016

Engineering Applications using Genetic Algorithm

Mashal Alenazi

University of Bridgeport, Biomedical Engineering Department, Bridgeport, CT USA

Abstract: In this paper, a variant of the Genetic Algorithm is used to place sensors optimally on a Large Space Structure for the purpose of modal identification. The selection and reproduction schemes of the Genetic Algorithm are modified and a new operator called forced mutation is introduced. These changes are shown to improve the convergence of the algorithm and to lead to near optimal sensor locations. genetic programming combined with neural networks could be incredibly slow, thus impractical. As with many problems, you have to constrain what you are attempting to create.

Keywords: Engineering, Applications, Technology, Rastrigin.

I. INTRODUCTION

The basic theory behind genetic algorithms is quite simple. First, a series of binary strings (chromosomes) is randomly generated. Sections of these chromosomes, or genes, are taken to represent variables. These genes are used as parameters in a system (real or simulated) and the relative success of that system when compared to a desired goal is rated. This step is re-iterated using each of the binary chromosomes and the suit abilities are sorted to determine the best two. From this point on, an optimization cycle begins. The two most suitable chromosomes are taken from the previous run and the rest are erased (killed). These two chromosomes are split at random points and recombined to create a new generation of chromosomes. A small percentage of genes is randomized (mutated) to ensure that new solutions can arise (evolve). This new generation is then run through the system to evaluate the suitability of each. Just as in the case of living organisms, environmental factors drive binary chromosomes to continually search for the best solution for a given environment. Occasionally a mutant arises that will either die off, or outshine its peers to pass its genes on to future generations. As the environment changes, so do the factors that determine who will live and who will die. While the presence of mutants prevents the determination of a "perfect" solution, it also ensures that the system can stay flexible enough to adapt to changing conditions.

The most common type of genetic algorithm works like this: a population is created with a group of individuals created randomly. The individuals in the population are then evaluated. The evaluation function is provided by the programmer and gives the individuals a score based on how well they perform at the given task. Two individuals are then selected based on their fitness, the higher the fitness, the higher and the chance of being selected. These individuals then "reproduce" to create one or more offspring, after which the offspring are mutated randomly. This continues until a suitable solution has been found or a certain number of generations have passed, depending on the needs of the programmer.

Over the last few years, there has been an ever increasing interest in the area of artificial immune systems (AIS) and their applications. Among the many works in this new field of research, we can detach those of Ishida (1996); Hunt & Cook (1996); Dasgupta (1999) and Hofmeyr &Forrest (1999). The AIS aim at using ideas gleaned from immunology in order to develop systems capable of performing different tasks in various areas of research.

The Genetic Algorithm Toolbox is a collection of functions that extend the capabilities of the Optimization Toolbox and the MATLAB® numeric computing environment [1].

Lymphocytes, in addition to proliferating and/or differentiating into plasma cells, can differentiate into long-lived B memory cells. Memory cells circulate through the blood, lymph and tissues, and when exposed to a second antigenic stimulus commence to differentiate into large lymphocytes capable of producing high affinity antibodies, pre-selected for the specific antigen that had stimulated the primary response. Figure 1 depicts the clonal selection principle. These algorithms enable you to solve a variety of optimization problems that lie outside the scope of the Optimization Toolbox.

All the toolbox functions are MATLAB M-files made up of MATLAB statements that implement specialized optimization algorithms. You can view the MATLAB code for these functions using the statement type function_name. You can extend the capabilities of the Genetic Algorithm by writing your own M-files, or by using the toolbox in combination with other toolboxes, or with MATLAB or Simulink®. Learning in the immune system involves raising the population size and affinity of those lymphocytes that have proven themselves to be valuable by having recognized any antigen. While doing technology, it's one's desire to solve any kind of problem using a minimal amount of resources. Hence, we need the engineering tools to seek high quality and parsimonious solutions. In our model, we do not intend to maintain a large clone for each candidate solution, but to keep the single best individual. A clone will be temporarily created, according to the clonal selection theory, and those progenies with low affinity will be discarded [2].

II. APPLICATIONS OF GENETIC ALGORITHMS

There are many applications of genetic algorithms. A lot of problems in real world have been solved via genetic algorithms. In general, Genetic Algorithms can be applied to virtually any problem that has a large search space and considering applications of genetic algorithms it can be viewed as problem solvers, as challenging technical puzzle, as basis for competent machine learning, as computational model of innovation and creativity, as computational model of other innovating systems and as guiding philosophy. Below given are some of applications of genetic algorithms. A variant of the Genetic Algorithm is used to place sensors optimally on a Large Space Structure for the purpose of modal identification. The selection and reproduction schemes of the Genetic Algorithm are modified and a new operator called forced mutation is introduced. These changes are shown to improve the convergence of the algorithm and to lead to near optimal sensor locations. Two practical examples are investigated; sensor placement for an early version of the Space Station and an individual Space Station photovoltaic array. Simulated results are also compared with previous results obtained by the Effective Independence method. The Genetic Algorithm based sensor configuration estimates the target mode response more accurately [4].

Application of the genetic algorithm requires that the problem be coded in parameter strings (called genes), and that there be an evaluation criterion which can determine a fitness value associated with each gene. As usually implemented, there are two phases to the genetic algorithm. During the initial phase, characterized by a highly diverse genetic pool and dominated by mating of highly diverse parents, the parameter estimates move rapidly towards the desired parameterization. In the second phase, when the gene pool is highly uniform and the incorporation of new genetic material must rely on the mutation process, the motion of the parameters is painfully slow. Our modification essentially monitors the diversity of the gene pool, and forces a mass "extinction and immigration" whenever the diversity has fallen below a preset value. Several examples illustrate the method, including linear and nonlinear, FIR and IIR identification. The method is also applied to identify the parameters in layered feed forward and feedback (recurrent) neural network structures.
Genetic algorithms have been used in a wide variety of optimization tasks, including numerical optimization, and combinatorial optimization problems such as traveling salesman problem, circuit design, job shop scheduling and video & sound quality optimization.

Genetic algorithms have been used to model processes of innovation, the development of bidding strategies, and the emergence of economic markets. Genetic algorithms have been used to model various aspects of the natural immune system, including somatic mutation during an individual's lifetime and the discovery of multi-gene families during evolutionary time. Genetic algorithms have been used to model ecological phenomena such as biological arms races, host-parasite co-evolutions, symbiosis and resource flow in ecologies. Genetic algorithms have been used to study questions in population genetics, such as under what conditions will a gene for recombination be evolutionarily viable Genetic algorithms have been used to evolve computer programs for specific tasks, and to design other computational structures, for example, cellular automata and sorting networks. Genetic algorithms have been used for many machine- learning applications, including classification and prediction, and protein structure prediction. Genetic algorithms have also been used to design neural networks, to evolve rules for learning classifier systems or symbolic production systems, and to design and control robots [3].

Genetic algorithms have successfully been used to evolve various aspects of Genetic algorithms - the connection weights, the architecture, or the learning function. You can see how Genetic algorithms are perfect for evolving the weights of a neural network - there are immense number of possibilities that standard learning techniques such as back-propagation would take thousands upon thousands of iterations to converge to. Genetic algorithms could (given the appropriate direction) evolve working weights within a hundred or so iterations.

Evolving the architecture of neural network is slightly more complicated, and there have been several ways of doing it. For small nets, a simple matrix represents which neuron connection which, and then this matrix is, in turn, converted into the necessary 'genes' and various combinations of these are evolved. Many would think that a learning function could be evolved via genetic programming. Unfortunately, genetic programming combined with neural networks could be incredibly slow, thus impractical. As with many problems, you have to constrain what you are attempting to create. For example, in 1990, David Chalmers attempted to evolve a function as good as the delta rule. He did this by creating a general equation based upon the delta rule with 8 unknowns, which the genetic algorithm then evolved [5].

III. AN EVOLUTIONARY SYSTEM

The clonal selection functioning of the immune system can be interpreted as a remarkable microcosm of Charles Darwin's law of evolution, with the three major principles of diversity, variation and natural selection (Cziko, 1995).The two central processes involved in the production of antibodies, genetic recombination and mutation, are the same ones responsible for the biological evolution of sexually reproducing species. In these species, the same two processes are involved in providing the variations on which natural selection can work to fit the organism to the environment (Holland, 1995). As a consequence, cumulative blind variation and natural selection, which over many millions of years resulted in the emergence of mammalian species, remain crucial in the day-by-day ceaseless battle to survival of these species. Whereas adaptive biological evolution proceeds by cumulative natural selection among organisms, research on the immune system has now provided the first clear evidence that ontogenetic adaptive changes can be achieved by cumulative blind variation and selection within organisms. The clonal selection algorithm, to be

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 3, March 2016

described further in the text, aims at demonstrating that this cumulative blind variation can generate high quality solutions to complex problems [6].

IV. RASTRIGIN'S FUNCTION

How to find the minimum of Rastrigin's function used genetic algorithm. Rastrigin's function is defined as

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2).$$

The following figure shows a plot of Rastrigin's function

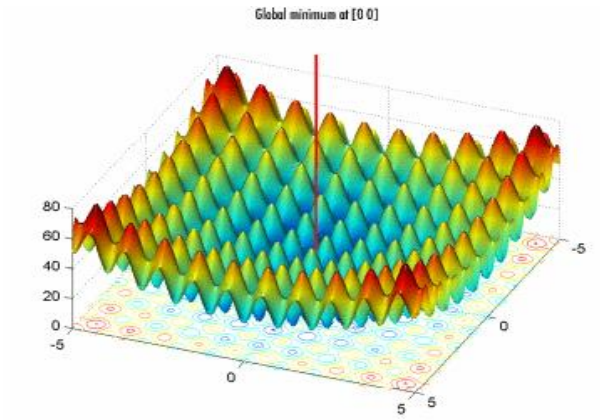

Figure 1: Rastrigin's Function

As the plot shows, Rastrigin's function has many local minima the "valleys" in the plot. However, the function has just one global minimum, which occurs at the point [0 0] in the x-y plane, as indicated by the vertical line in the plot, where the value of the function is 0. At any local minimum other than [0 0], the value of Rastrigin's function is greater than 0. The farther the local minimum is from the origin, the larger the value of the function is at that point. Rastrigin's function is often used to test the genetic algorithm, because its many local minima make it difficult for standard, gradient. based methods to find the global minimum. To find the minimum, do the following steps:

1. Enter optim tool ('ga') at the command line to open the Optimization Tool.
2. Enter the following in the Optimization Tool:
   - In the Fitness function field, enter @rastriginsfcn.
   - In the Number of variables field, enter 2, the number of independent variables for Rastrigin's function.

The Fitness function and Number of variables fields should appear as shown in the following figure.

While the algorithm is running, the Current iteration field displays the number of the current generation. You can temporarily pause the algorithm by clicking the Pause button. When you do so, the button name changes to resume. To resume the algorithm from the point at which you paused it, click Resume.

When the algorithm is finished, the Run solver and view results pane appears as shown in the following figure. The Run solver and view results pane displays the following information:

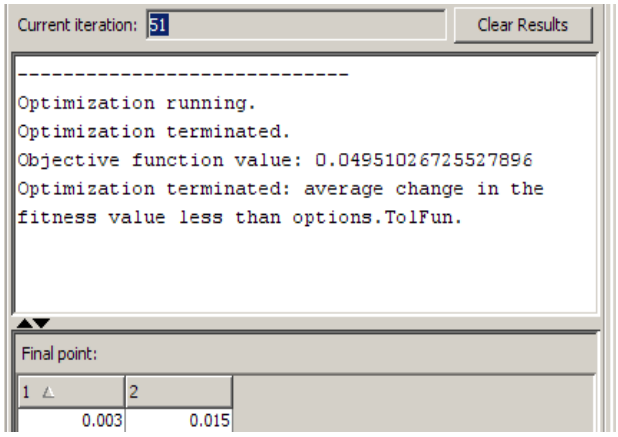

Figure 2: Current iteration

The Plot functions pane enables you to display various plots that provide information about the genetic algorithm while it is running. This information can help you change options to improve the performance of the algorithm. For example, to plot the best and mean values of the fitness function at each generation, select the box next to Best fitness, as shown in the following figure

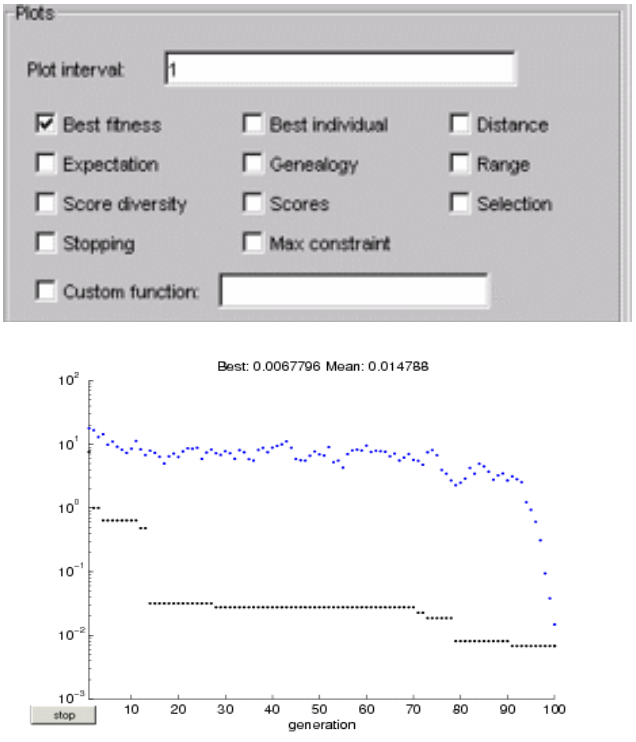

Figure 3: Optimization Result

The points at the bottom of the plot denote the best fitness values, while the points above them denote the averages of the fitness values in each generation. The plot also displays the best and mean values in the current generation numerically at the top. To get a better picture of how much

the best fitness values are decreasing, you can change the scaling of the y-axis in the plot to logarithmic scaling. To do so, typically, the best fitness value improves rapidly in the early generations, when the individuals are farther from the optimum. The best fitness value improves more slowly in later generations; whose populations are closer to the optimal point.

V. CONCLUSION

I proposed a general-purpose algorithm inspired in the clonal selection principle and affinity maturation of the immune response. The algorithm was verified to be capable of performing learning and maintenance of high quality memory and, it was also capable of solving complex problems, like multi-modal and combinatorial optimization. The major advantage of genetic algorithms is their flexibility and robustness as a global search method. They are "weak methods" which do not use gradient information and make relatively few assumptions about the problem being solved. Rastrigin's function has many local minima the "valleys" in the plot. However, the function has just one global minimum, which occurs at the point [0 0] in the x-y plane, as indicated by the vertical line in the plot, where the value of the function is 0. an optimization cycle begins. The two most suitable chromosomes are taken from the previous run and the rest are erased (killed). These two chromosomes are split at random points and recombined to create a new generation of chromosomes. A small percentage of genes is randomized (mutated) to ensure that new solutions can arise (evolve).

REFERENCES

[1] De Castro, L. N., & Von Zuben, F. J. (2000, July). The clonal selection algorithm with engineering applications. In Proceedings of GECCO (Vol. 2000, pp. 36-39).
[2] Dasgupta, D., & Michalewicz, Z. (Eds.). (2013). Evolutionary algorithms in engineering applications. Springer Science & Business Media.
[3] Mühlenbein, H., & Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm i. continuous parameter optimization. Evolutionary computation, 1(1), 25-49.
[4] Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. Structural and multidisciplinary optimization, 26(6), 369-395.
[5] Man, K. F., Tang, K. S., & Kwong, S. (1996). Genetic algorithms: concepts and applications. IEEE transactions on Industrial Electronics, 43(5), 519-534.
[6] Adeli, H., & Cheng, N. T. (1993). Integrated genetic algorithm for optimization of space structures. Journal of Aerospace Engineering, 6(4), 315-328.