The Future and Applications of Genetic Algorithms

Randall, Marcus

Published in:
Proceedings: Electronic Technology Directions to the Year 2000

THE FUTURE AND APPLICATIONS OF GENETIC ALGORITHMS

Marcus Randall
Faculty of Engineering and Applied Science
Griffith University - Gold Coast
PMB 50 Bundall Mail Centre, Q 4217
Ph: (075) 948666, Fax: (075) 948679
Email: EASMARCUS@ipnet.ins.gu.edu.au

Abstract **- This paper presents a method of producing solutions to difficult problems based on the laws of natural selection. The method, known as the genetic algorithm, is described in detail and applied to the cart pole control problem. The future of genetic algorithms is discussed in terms of potential commercial application.**

Key Words/Phrases: genetic algorithms, chromosomes, genes, crossover, mutation, generic genetic algorithm engines, cart pole problem

1.0 INTRODUCTION

Producing solutions to standard business problems (such as developing and maintaining payroll systems) have long been the staple activity of the computer industry. However as more of these systems are prepared, this is leaving more difficult problems to solve using computer implementation. Difficult problems are those which cannot be readily solved using conventional techniques/algorithms. Such problems involve finding solutions which are non-linear (i.e. there may not be a discernible relationship between input and output). As these problems are often quite complex, they are dealt with by artificial intelligence / heuristic techniques. One such subclass of techniques search for an optimum solution in the space formed by assigning each of the problem's parameters to a spatial dimension. As a result of the non-linearity, the space may be multimodal so that it contains many local minima as well as the global minimum (the optimal solution).

There are many such techniques for searching this space. One of these relies on the laws of natural selection in order to breed solutions to problems using genetic modification techniques. These are known as genetic algorithms.

2.0 HOW GENETIC ALGORITHMS WORK

Genetic algorithms, as the name implies, are based on the premise of the biological concept of genetic reproduction. Genetic algorithms essentially manipulate chromosomes which are vectors of numbers or values. Each of these values is referred to as a gene. A number of these chromosomes is generated and applied to a particular problem. Each chromosome is then evaluated to determine how well it satisfies the problem by a problem specific fitness function. The chromosomes that produce the most suitable results are then selected to form the basis of a new generation of chromosomes. This process can be likened to Darwinian Selection. The objective is that from initial populations of chromosomes containing random values, within acceptable problem defined limits, a solution or perhaps many solutions to the problem can be obtained after many generations have been formed.

Reproduction is carried out in order to produce a new generation of chromosomes. This process involves the selection of chromosomes that will form a mating pool and the application of the reproduction operators. The aim of the reproductive process is to form a new generation of chromosomes whose fitness is greater than that of the previous generation. There are numerous means of reproduction. However, a widely used technique of reproduction will be described here and consequently used in this study. This is known as the "remainder stochastic sampling without replacement" technique [3].

The application of remainder stochastic sampling without replacement is described by Goldberg [3] and involves the following steps:

1. *Calculate the fitness of each chromosome*

The fitness of each chromosome is the measure of how well the chromosome performs on the problem and is referred to as f_i. The method of remainder stochastic sampling without replacement makes use of this fitness, f_i, of the chromosome compared to the proportion of fitness it accounts for over the whole population. This will be the chromosome's probability of being represented in the mating pool and is:

$$pselect_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \quad (1)$$

where:

$pselect_i$: The probability of selection of chromosome *i*

f_i: The fitness of the chromosome *i*

n: The population of the chromosomes in each generation

2. *For each chromosome, calculate the expected number of copies of that chromosome that will be present in the mating pool.*

The mating pool consists of chromosomes that best satisfy the problem conditions and will hence form the basis of the next generation. The expected number of each chromosome in the pool is determined according to its probability of selection, $pselect_i$, and the number of chromosomes in the population, *n*:

$$e_i = n \times pselect_i \quad (2)$$

where:

e_i is the expected number of copies of chromosome *i* in the mating pool

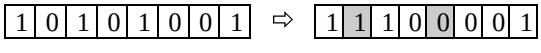
3. *Determine the mating pool constituency*

The integer component of e_i determines how many copies of chromosome *i* will be definitely present in the mating pool. The chromosome then also has the probability of the fractional component of e_i of having another copy of that chromosome present in the mating pool.

4. *Apply the reproduction operators*

The two most frequently used reproduction operators are crossover and mutation. Crossover involves combining one part of a chromosome with another part of a chromosome (both of which are in the mating pool) to form a new individual. Mutation is the process where the genes of certain chromosomes in the new generation randomly change to another value. Crossover and mutation are represented diagrammatically in figure 1.

Mutation



Crossover

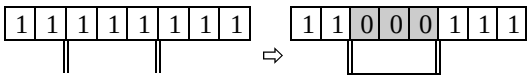


Figure 1: Common genetic reproduction operators

Chromosomes are selected at random from the mating pool to engage in crossover. The offspring of this activity form the new generation. The crossing over of pairs of chromosomes continues until the new generation is full. After this, mutation occurs.

The process of chromosome fitness evaluation and reproduction continues until the problem is seen as being adequately satisfied (meets a certain condition) or a fixed number of generations have been formed and evaluated. As there can be a large number of chromosomes to evaluate and reproduce, genetic algorithms are computationally intensive [2].

A complication of coding genetic algorithms is that the chromosomes should be represented as binary strings [3], even though most real world problems involve floating point numbers. This is because the implementation of crossover and mutation (as well as other reproduction operators) becomes simpler and the solution quality better [3].

There are many variations of the operation of genetic algorithms in terms of which operators to apply as well as the exact mechanics of those operators (for instance the mutation probability rate). For a comprehensive survey of these, the reader is referred to [3].

3.0 AN APPLICATION OF THE GENETIC ALGORITHM : THE CART POLE PROBLEM

Genetic algorithms may be applied to a wide range of optimisation problems. Hence there are many areas within such disciplines as engineering and business where they are finding use.

Determining the weights of a neural network is an optimisation problem. To demonstrate the effectiveness of genetic algorithms in finding a set of appropriate weights for a neural network and its optimisation ability in general, the well demonstrated cart pole control problem will be utilised. Below is a description of the cart pole problem and its neural network implementation.

The cart pole problem is derived from the broom balancing (inverted pendulum) problem where a person attempts to balance a broomstick with the palm of their hand. In the case of the cart pole problem, a pole hinges on a moving platform known as a cart so that it can pivot in the plane of cart motion only. The objective is for the cart to balance the pole for as long a time period

(measured in discrete time steps) as possible. In order to achieve this, the cart controller must constantly issue appropriate control instructions to the cart to prevent the pole from exceeding predefined failure angles (from the vertical axis) or the cart overshooting the limits of a finite length track. The cart pole problem has been considered by most researchers in a uni-dimensional context. Therefore the cart may either move left or right in order to balance the pole as in the system illustrated below:

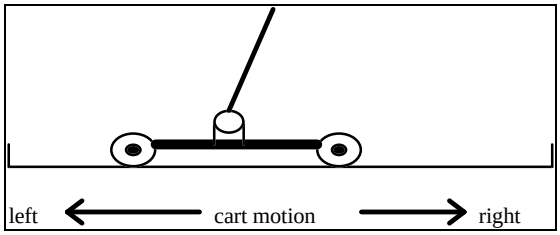


Figure 2: The cart balancing the pole on a finite length track

The problem of balancing the pole on the cart can be solved using control theory, although it is a difficult problem and a solution by this method is unwieldy and largely impractical. However, if the controller is implemented by a heuristic technique (such as the genetic algorithm), it can *learn* how to solve the task rather than necessarily relying upon supplied or prepogrammed domain knowledge.

It has been determined by Sitte and Geva [2] that the following four parameters are required by the cart pole system:

- cart position
- cart velocity
- pole angle
- pole angular velocity

These parameters can be used as inputs to a single artificial neuron. This neuron simply sums the products of the weights by these inputs and thresholds this value to produce either a left or right control action of constant magnitude. While other studies [1] have used complex neural architectures, the cart pole controller can be implemented on a single neuron. The problem then becomes one of finding a weight for each of the above parameters. The following shows the cart pole neuron:

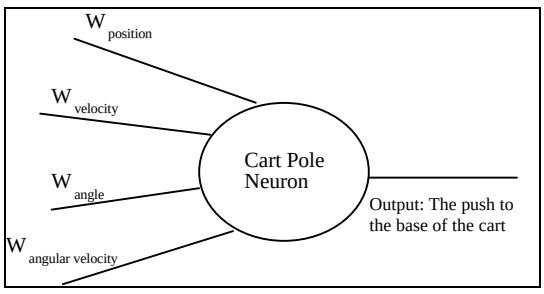


Figure 3: The single neuron and weights that can be used to implement the cart pole controller

3.1 COMPUTER IMPLEMENTATION OF THE CART POLE SYSTEM AND THE GENETIC ALGORITHM

The genetic algorithm and cart pole system were simulated in software using Borland Pascal for MS-DOS. The following set of parameters (which are for the cart pole system and the genetic algorithm) is used in this study:

PARAMETER	VALUE
Track Length	± 2.4 m
Failure Angles	± 0.21 rad
Gravity	9.8 m/s^2
Length of Pole	1 m
Mass of cart	1.0 kg
Mass of pole	0.1 kg
Control Force	+10 N or -10 N
Starting condition[1]	{1,1,0.17,0.18}
Time step (update interval)	0.02 s
Population size	200
Number of allowed generations	50
Mutation probability	1 in 100 genes

Table 1: Cart pole and genetic algorithm parameters

3.2 RESULTS

It is difficult to compare the results generated in this study with that of other studies as it uses a unique parameter set and starting condition (in accordance with the recommendations set out in [2]). For a standardised comparison of common adaptive techniques, the reader is referred to [5].

The genetic algorithm found a solution (a cart pole controller) which could balance the pole for 1000 seconds (the test recommended by Sitte and Geva [2]) and longer time periods. The solution was found by the sixth generation as seen in the following:

[1] The starting condition is in terms of cart position (m), cart velocity (ms^{-1}), pole angle (rad) and pole angular velocity (rad^{-1}).

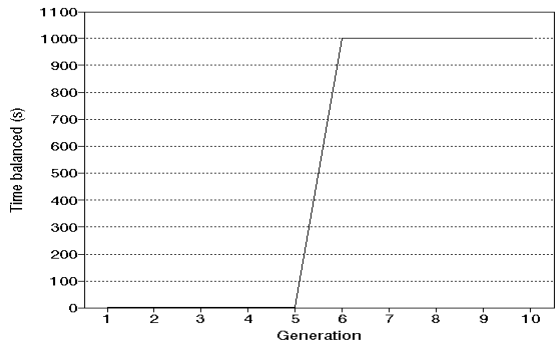


Figure 4: Graph of the balancing time of the best controller in each generation

The following graphs show the first 20 seconds of the genetic algorithm controller's balancing of the pole. The starting condition used in this study is difficult for controllers to overcome [5]. However, as seen by figure 5, the genetic algorithm's controller initially moved the cart nearly to the failure point (the extreme right edge of the track) and then stabilised its movement around the centre of the track for the rest of the time for which it was balanced. This is similarly reflected in the pole's movement (as seen by figure 6).

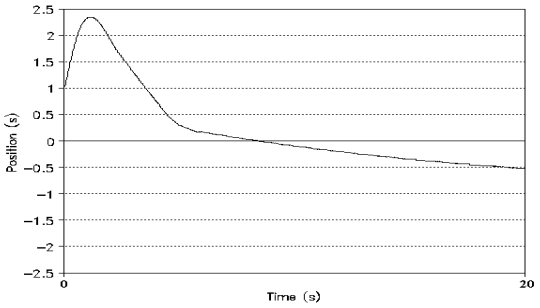


Figure 5: Graph of the best genetic algorithm controller - position versus time

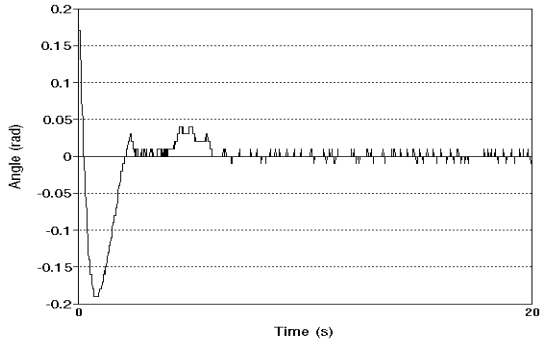


Figure 6: Graph of the best genetic algorithm controller - angle versus time

The controller is therefore considered to be proficient at the task because it could overcome the initial difficult starting condition and then stabilise the pole at the centre of the track with the pole oscillating only slightly around the vertical axis. This is an unexpected result, as the fitness function is only in the form of balancing time, not stabilising ability.

4.0 GENETIC ALGORITHM CHARACTERISTICS AND POSSIBLE IMPROVEMENTS

When the genetic algorithm was applied to the cart pole problem, it produced stable controllers which could balance the pole for 1000 seconds and longer time periods. However for other problems, genetic algorithms will produce near optimal solutions [6]. This is one of this technique's inherent weaknesses. Therefore genetic algorithms can be used as a tool for finding good solutions to a problem, in which total optimality (i.e. the perfect solution) is not necessary. However in most problem cases, if the solution to this problem was exhaustively searched for (i.e. each possible solution is tested to determine the best solution) computer processing time would be in the order of years whereas with genetic algorithms and similar heuristic techniques, acceptable solutions can be found in only minutes or hours.

As previously mentioned, researchers have used a number of variations on the genetic algorithm formulation in order to improve performance. However it is believed that dramatic performance increase can only be gained by combining genetic algorithms with more traditional forms of function optimisation from operations research science and/or other heuristic techniques such as simulated annealing and tabu search.

5.0 THE FUTURE OF GENETIC ALGORITHMS

Genetic algorithms can be applied to a wide range of problems which are NP (Nondeterministically Polynomial) complete. This means that if an enumerative search were to be carried out, it would take an exponential amount of time [4]. One of the more well researched examples is the travelling salesperson problem, where a salesperson must find the shortest route through a number of cities starting and ending at a base city. However more practical applications include strategy planning, scheduling / time tabling and machine learning [7].

To date, the approach to the use of genetic algorithms has been to develop specific programs which encode both:

- the problem information

- a genetic algorithm engine (usually incorporating characteristics inherent in the particular problem)

As a result of this, finding solutions to optimisation problems can involve very intensive and costly development. Increasingly however, software is being written which separates out these two characteristics so that the problem information is input by the user rather than being hard coded. In order for there to be a wider acceptance of genetic algorithms for industrial use, the existing generic genetic algorithm engines need to be able to deal with a range of different business and scientific problems. Awareness of the power of genetic algorithms also needs to be promoted .

Some examples of these products include Genesis and Genitor [6]. To date however, these have been mainly of academic interest. In order to allow more commercial accessibility, these programs would need to be easier to operate (running under a GUI interface such as MS Windows) and allow a wide variety of problems to be solved.

6.0 SUMMARY

The genetic algorithm approach to solving difficult problems is outlined in this paper. The cart pole problem was utilised with the genetic algorithms and produced stable controllers which could balance the pole for more than 1000 seconds.

Genetic algorithm products are beginning to be produced commercially. However, ease of use and the ability for these products to be used over a wide variety of problem types will determine their success.

ACKNOWLEDGMENT

The author wishes to thank Murray Bourne for his critical reading of the paper.

BIBLIOGRAPHY

[1] Dominic, S.; Das, R.; Withely, D and Anderson, C. (1991): *Genetic Reinforcement Learning for Neural Networks*. Proceeding of the International Joint Conference on Neural Networks (July). pp 71-76.

[2] Geva, S. and Sitte, J. (1993): *The cart pole experiment as a benchmark for trainable controllers*. IEEE Control Systems Magazine. Vol 13, Num 5, pp 40-51.

[3] Goldberg, D.E. (1989): *Genetic Algorithms in Search, Optimisation & Machine Learning. Addison Wesley*. Reading, MA. 412 pages.

[4] Luger G; Subblefield (1983): *Artificial Intelligence. Addison Wesley*. Reading, MA. 740 pages.

[5] Randall, M.C., Thorne, C.E. and Wild, C (1994): *A Standard Comparison of Adaptive Controllers to Solve the Cart Pole Problem.* Proceedings of the Second IEEE Australian and New Zealand Conference on Intelligent Information Systems, pp 61 - 65.

[6] Ribeiro, J.L. and Treleaven, P.C.(1994): *Genetic Algorithm Programming Environments.* Computer Vol 27, No 6, pp 28-43

[7] Srinivas M. and Patnaik, L.M.(1994): *Genetic Algorithms : A Survey.* Computer Vol 27, No 6, pp 17-26.