Imperial Journal of Interdisciplinary Research (IJIR)
Vol-2, Issue-11, 2016
ISSN: 2454-1362, http://www.onlinejournal.in

Genetic Algorithm: A Brief Use in Solving Mathematical Equations

Purushottam Lal Bhari
Research Scholar under Guidance of Dr. Ashok Jetawat.
Dept. of Computer Science and System Studies, Mewar University, Chittorgarh

Abstract— *In this paper mainly pointed to study of Genetic Algorithm procedure. A Genetic Algorithm has many things like step-by-step solution procedure for a problem, pseudo code and flow chart of solution procedure. What kind of functions used during solves the problem by an algorithm? Genetic Algorithms are widely used for solving mathematical problems. A new evolutionary technique for evolving mathematical equations called Mathematical Equations Evolver (MEE) is proposed in this paper. Each MEE chromosome is a string of operands and operators (on alternative positions). Each sub equations encoded in a chromosome is considered a potential solution of the problem. The best of these equations is chosen to represent the chromosome for find solution. In this way variable length equations can be stored in a fixed length chromosome. Several numerical experiments for solving symbolic regression problems are performed. Results show that MEE outperforms Genetic Programming on the considered examples.*

Keywords- *Genetic Algorithms, Mathematical Equations, Mathematical Equation Evolver (MEE)*

1. Introduction

Linear Equation system mostly occurs in various fields of Science, Social Sciences and Engineering to find solutions to some problems. There are different types of known conventional algorithms for solving linear equations and they are based on some theoretical facts. Finding solution to these equations through the evolutionary process of genetic algorithm is a new and developing research area of interest in the sector of Artificial Intelligence with Mathematics. The Genetic Algorithm methods follow the concept of solution evolution by stochastically developing generations of solutions population using a definite fitness function to determine the best fit solution to the problem. Genetic Algorithm has been applied to several scientific problems such as Scheduling, timetabling, travelling salesman's problem. It has been successfully exploited in many optimization problems but its application in solving systems of equations is still being researched into.

This study investigates the applicability and effectiveness of Genetic Algorithm to finding the solutions of equations, which involve a search for optimal values for the unknown variables in the equations that best fit the systems of linear equations. We compare the competence and efficiency of Genetic Algorithm with the numerical methods.

Evolving mathematical expressions is a well-established field in Genetic Programming. Many evolutionary techniques which can be applied for detecting Mathematical equations have been proposed in the recent past. Genetic Programming (GP), Multi Expressions Programming (MEP), Linear Genetic Programming (LGP), Grammatical Evolution (GE), Gene Expression Programming, Formula Prediction using Genetic Algorithms (FPEG) are some of them. A new evolutionary algorithm for detecting mathematical expressions is proposed in this paper. This algorithm is called Mathematical Equations Evolver (MEE). A MEE solution is a string of operators and operands (on alternative positions). Each sub-equations encoded in a chromosome is considered that the problem have a potential. The best of these equations is chosen to represent the chromosome for next generation.

2. Basic Philosophy of Equation

Equations mostly used for every field and genetic algorithms also widely used for solutions. Genetic algorithm developed by Goldberg. He was inspired by Darwin's theory of evolution which states that the survival of an organism is affected by "the strongest species that survives" rule and follow it. The process of reproduction, crossover and mutation can be maintains the survival of an organism, it is suggest by Darwin. Darwin's concept of evolution is then adapted to computational algorithm to find solution to a problem and in natural fashion it is called objective function. When a genetic algorithm generates any solution, it is called a chromosome, while collection of chromosome is called population. A chromosome is completely composed from genes and its value can be either binary, numerical, symbols or characters depending on the problem want to be solved by us. We can say that the genetic algorithms mainly depending the working of chromosomes. These chromosomes will undergo a process called fitness function to measure the suitability of solution generated by Genetic Algorithm with problem. In population chromosomes will mate

through process called crossover thus producing new chromosomes named offspring. So the offspring's genes compositions are the combination of their parent or last generation chromosomes. Although in a generation, a few chromosomes will also mutation in their gene. The number of chromosomes which will undergo crossover and mutation is controlled by crossover rate and mutation rate value of genetic process. In the population chromosome will maintain for the next generation will be selected based on Darwinian's evolution rule, the chromosomes are selected again in the next generation on the bases of higher fitness value and greater probability. Later then several generations, the chromosome value will converge to a certain value which is the best solution for the problem.

3. Introduction to Genetic Algorithms

Genetic Algorithms (GAs) is a branch of Artificial Intelligence based the theory of biology. It were first presented by J. H. Holland in his book "Natural and Artificial Systems" in the year 1975 and developed further by his students. According to time, many changes and improvements have been suggested for genetic algorithms. Here, we discuss the present form of implementation of Genetic Algorithms.

Genetic algorithms are a category of algorithms inspired by evolution. Genetic algorithms provide solutions to a specific problem on a simple chromosome like data structure and apply recombination operators to these structures so as to keep safe critical information. A procedure of a genetic algorithm begins with a population of (typically random) chromosomes. Then these structures are evaluated and allocated reproductive opportunities in such a way that those chromosomes which represent a better solution to the problem are given more chances to reproduce than those chromosomes which are poorer solutions of problems. The "goodness" of a solution provided by Genetic Algorithms is typically defined with respect to the current population.

A Genetic Algorithm has some basic steps are as follows:

1. Build an initial population of samples (solutions) created randomly or using some initialization method.
2. Calculate the fitness (measure of being provided reproductive opportunities) of all the samples and select individuals for the reproduction process. The selection of the individual is though based on fitness method, but the fitness is a probabilistic mechanism. Some selection methods are used as well as Roulette Stochastic Universal Selection, wheel selection, Rank Tournament Selection.
3. Apply the genetic operators of crossover, mutations, inversions, etc. after fitness process to the selected individuals to create new individuals and thus a new generation. Crossover exchanges some of the bits of the two chromosomes and mutation inverts any bit(s) of the chromosome depending on a probability. Crossovers are a distinguishing feature of Genetic Algorithms of Artificial Intelligence. Many early developed algorithms were earlier used, but they basically worked on the method of mutations and no crossovers took place. In GAs crossovers 'explore' around the already found good solutions and mutations help 'exploiting' the search space for new solutions.

Then again Step2 is followed till the algorithm is reached in ending condition. Some most important issues are properly encoding of problem, selection operator categories and fitness functions categories have been discussed and experimented at large by researchers to reach better results.

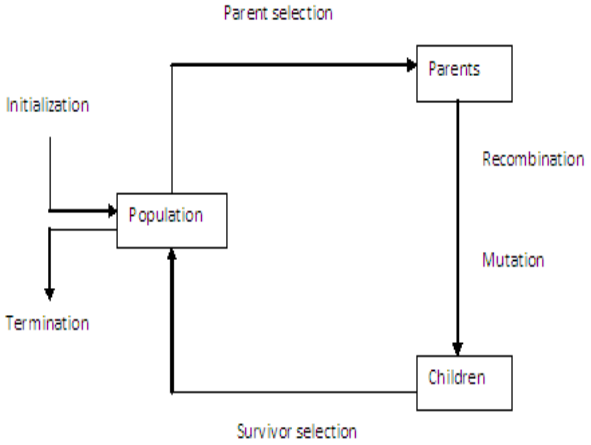

Figure 1 - The general scheme of Genetic Algorithm

4. Working Process of a Genetic Algorithm

Some important components of a Genetic Algorithm are representation (definition of individuals), evaluation function (or fitness function), population, parent selection mechanism, variation operators (crossover and mutation) and survivor selection mechanism (replacement).

Objects forming possible solution within original problem context are called phenotypes, their encoding and the individuals within the Genetic Algorithms are called genotypes. The mapping from the phenotypes onto a set of genotypes is specified by the representation step. Candidate solution, phenotype and individual are used to denote points of the space of possible solutions and this space is called phenotype space. In a genetic process chromosome and individual can be used for points in the genotype space. Elements of a chromosome are called genes and a value of a gene is called an allele.

Changing in a random gene or unary variation operator is called mutation. . In general, mutation is supposed to cause a random unbiased change. It is applied to one genotype and delivers a modified mutant, the child or offspring of it. It can guarantee that

the space is connected; it is a theoretical role of mutation.

Recombination or Crossover is a binary variation operator. The process merges information from two parent genotypes into one or two offspring genotypes is known crossover. Similarly to mutation, crossover is a stochastic operator: the choices of what parts of each parent are combined, it depends on random drawings and it is a good way. The simple principle behind crossover is: by mating two individuals with different but required features, we can produce an offspring which combines features both of those.

Parent selection is a most important mechanism to distinguish among individuals based on their quality to allow the better individuals to become parents of the next generation. It is probabilistic process. According to probability, high quality individuals get a higher chance to become parents than those with low quality.

The role of survivor selection is to distinguish among individuals based on their quality. In Genetic Algorithm, the population size is (almost always) constant, thus which individuals will be allowed in the next generation is a choice based on their fitness values, favoring those with higher quality. It is just opposed to parent selection, survivor selection is often deterministic, for particular, ranking the unified multi-set of parents and offspring and selecting the top segment (fitness biased), or selection only from the offspring (age-biased).

A population is a multi-set of genotypes. The role of the population is to hold possible solutions. Basically the population size is fixing, not changing during the evolutional search in all Genetic Algorithm applications.

The working process of Genetic Algorithms is explained below.

4.1 Initialization

An initial population is generated randomly and all genetic algorithms are generally stated with an initial population. Some research has been conducted into using special techniques to produce a higher quality initial population in genetic algorithms. Such an approach is designed to give the GA a good start and speed up the evolutionary process. Some authors propose a GA for exam timetabling problem. The GA works only with feasible solutions in this problem, when implying that the initial population must also be made up of feasible solution. Then the fitness of the initial population is improved by the running Genetic Algorithm.

Example 1: A simple exam timetabling problem.

We can use a non-binary bit string representation to represent the chromosome in a simple exam timetabling problem because it is easy to understand and represent. In this problem we use six positions representing six exams with each position's value as the time slot assigned to the exam. To assign each exam a timeslot, we can generate the population randomly.

Day	**AM**		**PM**	
	Time1	**Time2**	**Time3**	**Time4**
Day1			E1	E3
Day2		E5	E6	E2,E4

If we randomly generate six numbers 7, 6, 7, 7, 2, 2 as six timeslots for E1-E6, then the chromosome is 7 6 7 7 2 2.

If the population size is 5, an initial population can be generated randomly as follows:

Index	Chromosome	Fitness
1	7 6 7 7 2 2	0.020
2	7 3 7 6 1 3	0.062
3	5 3 5 5 5 8	0.006
4	3 8 4 8 6 7	0.005
5	1 7 4 5 2 2	0.040

Table 1 - Chromosomes

4.2 Reproduction

There are two kinds of reproduction as follow as:

4.2.1 Generational Reproduction

The whole of a population is potentially replaced at each generation in generational reproduction. The most often used procedure is to loop N/2 times, where N is the population size. In this method select two chromosomes each time according to the current selection procedure and producing two children from those two parents, then the finally producing N new chromosomes.

4.2.2 Steady-state Reproduction

Selects two chromosomes by the steady-state method according to the current selection procedure and performs crossover on them to obtain one or two children, perhaps applies mutation as well, and installs the result back into that population; the least fit of the population is destroyed.

4.3 Parent Selection mechanism

This selection procedure is stochastic; it does not imply GA employs a directionless search. Each parent being selected is in some way related to its fitness. The effect of selection is to return a probabilistically selected parent.

4.3.1 Fitness-based selection

Roulette Wheel selection or fitness-based selection is the standard, original method for parent selection. According to this kind of parent selection method each chromosome has a chance of selection that is directly proportional to its fitness. The range of fitness values in the current population is the main key of this effort.

Example 2: After calculation if fitness the range from 5 to 10, then the fittest chromosome is twice as likely to be selected as a parent than the least fit.

According to apply fitness-based selection on the population given in example 1, we select the second chromosome 7 3 7 6 1 3 as our first parent and 1 7 4 5 2 2 as our second parent.

4.3.2. Rank-based selection

According to the rank-based selection method, selection probabilities are based on a chromosome's relative rank or position in the population, rather than absolute fitness.

4.3.3. Tournament-based selection

In original tournament selection method choose K parents at random and returns the fittest one of these.

4.4 Crossover Operator

The crossover operator is the most important part of Genetic Algorithm. Recombination of bit strings via an exchange of segments between pairs of chromosomes is a process of crossover. There are many kinds of crossover methods.

4.4.1. One-point Crossover

The one-point crossover procedure is to randomly generate a number (less than or equal to the chromosome length) as the crossover position. After that, keep the bits before the number unchanged and swap the bits after the crossover position between the two parents.

Example 3: We first taking two parents selected above and randomly generate a number 2 as the crossover position:

Parent1: 7 3 7 6 1 3
Parent2: 1 7 4 5 2 2

Then we get two children:

Child 1 : 7 3| 4 5 2 2
Child 2 : 1 7| 7 6 1 3

4.4.2. Two-point Cross Over

The procedure of two-point crossover is must select two positions and only the bits between the two positions are swapped. In this crossover method can the first and the last parts of a chromosome maintained in its existing state and just swap the middle part.

Example 4: We taking two parents selected above and randomly generate two numbers 2 and 4 as the crossover positions:

Parent1: 7 3 7 6 1 3
Parent2: 1 7 4 5 2 2

Then we get two children:

Child 1 : 7 3| 4 5| 1 3
Child 2 : 1 7| 7 6| 2 2

4.4.3. Uniform Crossover

Each gene of the first parent has a 1/2 probability of swapping with the corresponding gene of the second parent in the procedure of uniform crossover.

Example 5: We randomly generate a number between 0 and 1 for each position, for example, 0.2, 0.7, 0.9, 0.4, 0.6, 0.1. Child1 gets the gene from parent1 and child2 gets the gene from parent2, if the number generated for a given position is less than 0.5. Otherwise, vice versa.

Parent1: 7 *3 *7 6 *1 3
Parent2: 1 *7 *4 5 *2 2

Then we get two children:

Child 1 : 7 7* 4* 6 2* 3
Child 2 : 1 3* 7* 5 1* 2

4.5 Inversion

The inversion method performs operation as a kind of reordering technique. Mainly it is operates on a single chromosome and inverts the order of the elements between two randomly chosen points on the chromosome.

Example 6: If we randomly choose two positions 2, 5 from a given chromosome 3 8 4 8 6 7 and apply the inversion operator, then we get the new string: 3 6 8 4 8 7.

4.6 Mutation

Basically the mutation has an effect of ensuring that all possible chromosomes are reachable. The search is constrained to alleles with crossover and even inversion, which exist in the initial population. In a genetic algorithm the mutation operator can overcome this by simply randomly selecting any bit position in a string and changing it. This process is so much useful since crossover and inversion may not be able to produce new alleles if they do not appear in the initial generation.

Example 7: Assume that we have already used crossover to get a new string: 7 3 4 5 1 3. Let the mutation rate is 0.001. Next, we generate randomly a number between 0 and 1 for the first bit 7. If the number is less than the mutation rate (0.001), then the first bit 7 needs to mutate. After that we generate another number between 1 and the maximum value 8, and get a number (for example 2). So the first bit mutates to 2. The same procedure repeated for the other bits. We will get a new chromosome, if only the first bit mutates and the rest of the bits don't mutate, as below:

2 3 4 5 1 3

5. Use of Constraint Handling in Genetic Algorithms

Genetic Algorithms have many ways to handle constraints. At the high conceptual level we can distinguish two cases: indirect constraint handling and direct constraint handling.

The meaning of indirect constraint handling that we circumvent the problem of satisfying

constraints by incorporating them in the fitness function f such that f optimal implies that the constraints are satisfied, and use the power of Genetic Algorithm to find a solution.

The meaning direct constraint handling that we leave the constraints as they are and 'adapt' the Genetic Algorithm to enforce them.

Attention that direct and indirect constraint handling can be applied in combination, just like, in one application we can handle some constraints directly and others indirectly.

Basically, indirect constraint handling means transforming constraints into optimization objectives.

5.1 Direct constraint handling

Behaving constraints directly implies that violating them is not reflected in the fitness function, in this way there is no bias towards chromosomes satisfying them. Accordingly, the population will not become less and less infeasible w. r. t. these constraints. It means that we have to create and maintains feasible chromosomes in the population. In this case the regular operators are blind to constraints and mutating one or crossing over two feasible chromosomes can result in infeasible offspring, it is a basic problem. Typical approaches to handle constraints directly are the following:

• eliminating infeasible candidates
• repairing infeasible candidates
• special operators used to preserving feasibility
• decoding, i.e. transforming the search space

Eliminating infeasible candidates is very inefficient and consequently hardly applicable. A repair procedure that modifies a given chromosome by the using repairing infeasible candidates, that it will not violate constraints. This technique is as indicated problem dependent.

The preserving approach amounts to designing and applying problem-specific operators that do preserve the feasibility of parent chromosomes. The preserving approach requires the creation of a feasible initial population, which can be Non-Polynomial complete.

Decoding can simplify the problem search space and allow an efficient genetic algorithm. Basically, decoding can be seen as shifting to a search space that is different from the Cartesian product of the domains of the variables in the original problem formulation.

5.2 Indirect Constraint Handling

The optimization objectives replacing the constraints are viewed penalties for constraint violation hence to be minimized in the case of indirect constraint handling. In general penalties are given for violated constraints although some Genetic Algorithms allocate penalties for wrongly instantiated variables or as the distance to a feasible solution.

Benefits of indirect constraint handling are:

- generality
- reduction of the problem to 'simple' optimization
- weight is possibility of embedding user preferences

Indirect constraint handling has some disadvantages are:

- packing everything in a single number has loss of information
- indirect constraint handling does not work well with sparse problems

6. The Structure of a Genetic Algorithm

Step1: As a chromosome, representation of the problem variable domain of a fixed length, the size of a chromosome population is decided as well as the crossover probability.

Step 2: Fitness function is used for measuring the quality of an individual chromosome in the problem domain. Mainly a fitness function establishes the basis for selecting chromosomes that will be mated during reproduction. Generally, the quality of the represented solution is measured by the fitness function defined over the genetic representation. Fitness functions are problem dependent and in cases where their definition is hard or even impossible, then interactive genetic Algorithm is used.

Step 3: An initial population of fixed size chromosomes in random generation

Step 4: For each individual chromosome calculate fitness function.

Step 5: Selection of pairs of chromosomes for mating from the current population. In parent selection process parent chromosomes are selected with a probability related to their fitness. The higher probability chromosomes selected for mating.

Step 6: Application of the genetic operations (crossover and mutation) for creating pairs of offspring chromosomes.

Step 7: In the new population placement of the created offspring chromosomes

Step 8: Step 5 through step 7 repetition until the size of the new chromosome population becomes equal to the size of the initial population.

Step 9: Replacement of the initial (previous) parent chromosome population with the new offspring population.

Step 10: Repetition step 4 through step 9 until the termination criterion is satisfied.

6.1Pseudo-Code

BEGIN
INITIALIZE population with random candidate solution.
EVALUATE each candidate;
REPEAT UNTIL (termination condition) is satisfied DO

1. CALCULATE fitness
2. SELECT parents;
3. RECOMBINE pairs of parents;
4. MUTATE the resulting offspring;
5. SELECT individuals or the next generation;

END.

6.2. The Flow-chart of Algorithm

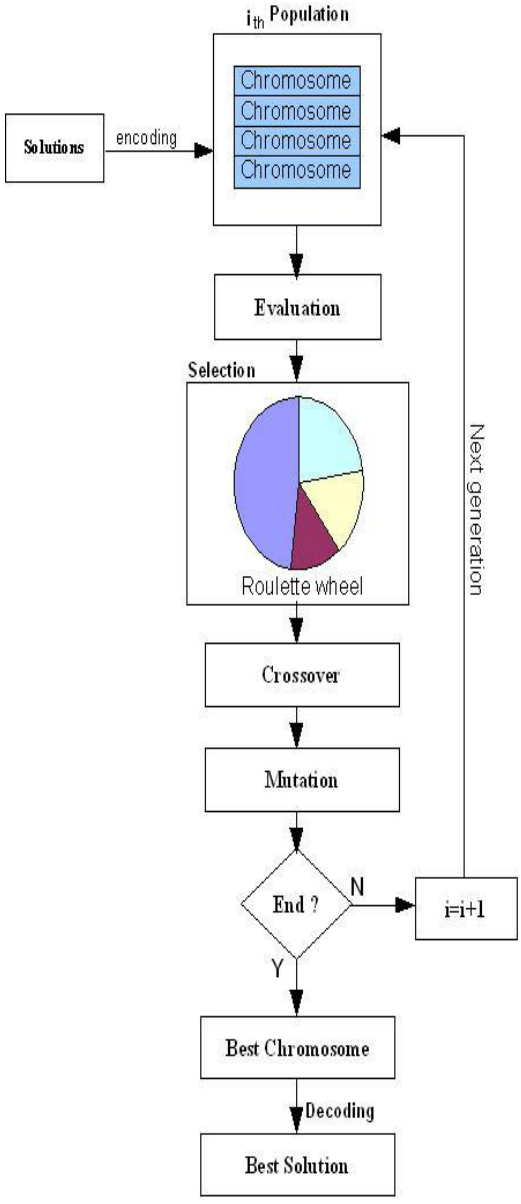

Figure 2 - Genetic algorithm flowchart

7. MEE Algorithm

It is a new evolutionary algorithm that can be applied for detecting mathematical equations. Genetic operators used are crossover and mutation in this algorithm. Solution representation and genetic operators are described as follows.

7.1 Solution representation

An equation is a set of operands and operators. Each operands and operators represented by individual (chromosome). Mostly in a equation odd positions of the chromosome consist of operands and the even positions consist of operators. The length of chromosome is constant during the search process.

For instance, if the operands set is {p, q} and the operators set is {+, -, *, /} than and example of chromosome with length 9 is:

$$C = p + q * q - p/q$$

Note: The chromosome length has to be an odd number because the last position of the chromosome cannot be an operator.

7.2 Genetic operators

The genetic operators used by this algorithm are crossover and mutation. Both of them are defined as well as:

Each gene of the chromosome with a given mutation probability can be effected by the mutation operators. For each gene a random number between 0 and 1 is generated. According to it, if this value is lower than mutation probability than the value of this gene is changed with another random generated value (if the value of the gene is an operator than another operator from the given set of operators is generated; if the value of a gene is an operand then another operand from the given set of operands is generated).

Let us suppose we have the chromosome like:

$$C = p + q * q - p / q$$

and the appliance of mutation operator affect the genes 3 and 6. Then a new obtained chromosome can be:

$$C = p + p * q + p / q$$

In above equation the value of the gene corresponding to third position has been changed from q to p and the value of the gene corresponding to sixth position (”-”) has been changed to the value (”+”).

The crossover operator’s specific to binary encoding (one cut point, two cut points, uniform, etc.) may be applied.

7.3 Fitness assignment

Each of the sub-expressions encoded in a MEE chromosome is considered as a potential problem solution in mathematical applications. For instance the chromosome

C = p + q * q – p / q

encodes the expressions:

E1 = p;
E2 = p + q;
E3 = p + q * q;
E4= p + q * q – p;
E5 = p + q * q – p / q;

The value of these equations may be evaluated by reading the chromosome from left to right. The chromosome fitness is usually defined as the fitness of the best expression detected in that chromosome.

If we want to solve symbolic regression problems, the fitness of each sub-expression E_i may be computed using the following formula:

$$f(E_i) = \sum_{k=1}^{n} |o_{k,i} - w_k|$$

Where: $o_{k,i}$ is the result obtained by the expression E_i for the fitness case k and w_k is the targeted result for the fitness case k.

In this case the fitness needs to be minimized. Basically the fitness of an chromosome is set to be equal to the lowest fitness of the expressions encoded in the chromosome:

$$f(c) = min f(E_i)$$

There is neither practical nor theoretical evidence that one of these equations is better than the others. Moreover, Wolpert and McReady proved that we cannot use the search algorithm's behavior so far for a particular test function to predict its future behavior on that function.

8. Algorithm Description

The MEE algorithm uses steady-state reproduction model as its underlying mechanism and starts by creating a random population of individuals. In algorithm the following steps are repeated until a given number of generation is reached: Using a standard selection procedure two parents are selected. Then the selected parents are recombined in order to obtain two offspring. Received offspring are considered for mutation. The best offspring from them O replaces the worst individual W in the current population if O is better than W.

The chromosome length is a constant of the search process, it is ensured by he variation operators. The algorithm returns as its answer the best equation evolved along a fixed number of generations.

9. Conclusion

In basic studies of Genetic Algorithms we understand about components it and they are work for a problem solving. A Genetic Algorithm's process can show by the using structure (figure 1), step-by-step process, flow chart or pseudo-code. A new evolutionary technique for detecting mathematical equations has been proposed in this paper. The defined algorithm encodes multiple equations in the same chromosome. We find that the fixed-length chromosomes are used to encode variable-length equations. Some numerical experiments for detecting quadratic and septic polynomial functions can be performed using this method. Results show that MEE performs better than Genetic Programming. Some other numerical experiment can be performed in order to analyze the relationship between success rate and population size, solution's number of iterations and other mathematical problems.

10. References

10.1. Papers and Publications

Abiodun, M., Olawale, N. and Adebowale, P. (2011), 'The Effectiveness of Genetic Algorithm in Solving Simultaneous Equations', International Journal of Computer Applications (0975 – 8887) Volume 14– No.8, February 2011, pp.38-41

Aggarwal, V. (2000), 'Solving transcendental equations using Genetic Algorithms', 104/ECE/2000 NSIT, Delhi, pp.1-12

Chakraborty, R. (2010), 'Fundamentals of Genetic Algorithms', Fundamentals of Genetic Algorithms: AI Course Lecture 39-40, notes, slides.

Grosan, C. (2007), 'Evolving mathematical expressions using Genetic Algorithms', Department of Computer Science Babe¸s-Bolyai University, Kogalniceanu, Cluj-Napoca, 3400, Romania.

Hermawanto, D. (2008), 'Genetic Algorithm for Solving Simple Mathematical Equality Problem', Indonesian Institute of Sciences (LIPI), INDONESIA

Khan, H., Khan, N., Inayatulla, S. and Nizami, T. (2009), 'Solving ISP Problem by Using Genetic Algorithm', International Journal of Basic & Applied Sciences IJBAS-IJENS Vol:09 No:10, pp.55-60

Mhetre, P. (2012), 'GENETIC ALGORITHM FOR LINEAR AND NONLINEAR EQUATION', International Journal of Advanced Engineering Technology E-ISSN 0976-3945, IJAET/Vol.III/ Issue II/April-June, 2012, pp.114-118

Shopova, G. and Bancheva, G. (2005), 'BASIC—A genetic algorithm for engineering problems solution', CACE-3228; pp.1-17

Rawat, S. and Rajamani, L. (2008), 'A TIMETABLE PREDICTION FOR TECHNICAL EDUCATIONAL SYSTEM USING GENETIC ALGORITHM', Journal of Theoretical and Applied Information Technology; pp.59-64

10.2. Web Sites

[1]www.elsevier .com
[2]www.technicaljournalsonline.com accessed
[3]www.myreaders.info/html/artificial_intelligence.htm
[4]www.cs.cmu.edu/Groups/AI/html/faqs/ai/genetic/part2/faq-doc-2.html
[5]www.jatit.org