# Genetic Programming: A study on Computer Language

Nilam Choudhary
*Department of Computer Science Engineering*
*Vivekananda Institute of Technology, Jaipur, Rajasthan, India*

Prof.(Dr.) Baldev Singh
*Department of Computer Science Engineering*
*Vivekananda Institute of Technology, Jaipur, Rajasthan, India*

Er. Gaurav Bagaria
*Department of Computer Science Engineering*
*Vivekananda Institute of Technology, Jaipur, Rajasthan, India*

**Abstract- this paper describes genetic programming in more depth, assuming that the reader is familiar with computer science, but not with machine learning or genetic programming. Genetic programming has automatically produced the result that are competitive to human intelligence and performance Genetic programming was developed and popularized by John Koza. It helps us in providing interface between the program and memory, freeing the program of memory management details which are left to the data structures to implement. The main result presented herein is that GP can automatically generate stacks and queues.**

**Keywords – genetic programming, crossover, mutation**

## I. INTRODUCTION

In today's world of technology personal computers have become more powerful and ubiquitous, they are called upon to perform and solve an ever increasing number of tasks and problems. In order to solve these problems or perform tasks a computer must execute a computer program. The act of writing such a computer program can be very complex and difficult. Under the pressure of an ever-growing need for computer programs, genetic programming/evolutionary computing emerged as a promising method in which the computer itself performs this often daunting task of writing software and helps in decreasing complexity.
Genetic programming [Genetic programming is a collection of methods for the automatic generation of computer programs that solve carefully specified problems, via the core, but highly abstracted principles of natural selection[1].Genetic programming has automatically produced the result that are competitive to human intelligence and performance . E.g.: creation of better than classical quantum algorithm for the deutsch –jozsa "early promise problem", creation of a novel variant of quantum dense coding [2] . The study of genetic programming investigates the efficacy and potential of using genetic algorithms for multiscale materials modelling and addresses some of the challenges involved in designing competent algorithms that solve hard problems quickly, reliably and accurately. We propose the use of genetic programming (GP)—a genetic algorithm that evolves computer programs—for bridging simulation methods across multiple scales of time and/or length [3]. It is the compound breeding of(initially random), computer programs where only the relatively more successful individuals pass on genetic materials(program and program fragments) to the next generation.[1] in 1964, Lawrence J. Fogel, one of the earliest practitioners of the gp methodology, applied evolutionary algorithms to the problem of discovering finite-state automata[4]. Lisp is a famous language of genetic programming .it is based upon two principles:

1. It is often easier to write an algorithm that can measure the amount of success a given program has at solving a problem, than to actually write the successful program itself.
2. A less-than-fully-successful program often contains components, which, appropriated in the right way, could contribute to designs for more successful programs.[1]

There are some practical applications of genetic programming. According to J. Koza, these problems have some of the following characteristics:

1. conventional mathematical approach cannot be applied
2. approximate solution is acceptable or is the only one possible,
3. relationships between variables are poorly understood,
4. problem of determining the size and shape of solution,
5. existence of large amount of data that requires computer analysis, classification and integration
6. problems where optimization is highly prized [6]

## I. ATTRIBUTES OF GP

- starts with “what needs to be done”;
- tells us “how to do it”;
- produces a computer program;
- automatically determines the number of steps in the program;
- supports code reuse (subroutines);
- supports parameterized code reuse;
- supports code reuse in the form of iterations, loops, and recursions;
- supports reuse of the results of executing code in the form of memory and internal storage;
- automatically determines the use of
- subroutines, iterations, loops, recursions, and memory;
- automatically determines the hierarchical arrangement of subroutines, iterations, loops, recursions, and memory;
- supports a wide range of useful programming constructs;
- is problem-independent;
- applies to a wide variety of problems;
- is scalable; and
- produces human-competitive results.[5]

There are some advantages and disadvantages of Genetic programming

Advantages are:

1.) It does impose any fixed length of solution, so the maximum length can be extended up to hardware limits
2.) In genetic programming it is not necessary for an individual to have maximum knowledge of the problem and to their solutions [7]

Disadvantages of Genetic programming are :

1.) In GP, the number of possible programs that can be constructed by the algorithm is immense. This is one of the main reasons why people thought that it would be impossible to find programs that are good solutions to a given problem.
2.) Although GP uses machine code which helps in providing result very fast but if any of the high level language is used which needs to be compile, and can generate errors and can make our program slow.
3.) There is a high probability that even a very small variation has a disastrous effect on fitness of the solution generated.[7]

## III. HOW GP WORKS?

Genetic programming is a branch of genetic algorithms where genetic algorithms are one of the best ways to solve a problem for which little is known. They are a very general algorithm and so will work well in any search space. Genetic algorithms use the principles of selection and evolution to produce several solutions to a given problem. 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 are 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 is 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 The main difference between genetic programming and genetic algorithms is the representation of the solution. Genetic programming creates computer programs in the lisp or scheme computer languages as the solution. Genetic algorithms create a string of numbers that represent the solution [7].

1.) Recognizing that the solution to a wide variety of problems can easily be recast as a search for a computer program, GP conducts its search in the space of computer programs. It operates by progressively breeding a population of computer programs over a series of generations using the Darwinian principles of evolution and natural selection.
2.) A run of GP starts with a primordial ooze of thousands of randomly created computer programs. It evaluates pairs of programs from the population to determine which one is better for solving the problem at hand. It then probabilistically selects programs from the population based on this partial order and modifies the programs using crossover (sexual recombination), mutation, and architecture-altering operations. The architecture-altering operations automatically add and delete subroutines, subroutine parameters, iterations, loops, recursions, and memory in a manner patterned after gene duplication and gene deletion in nature. These operations automatically arrange the program’s elements into a hierarchy.

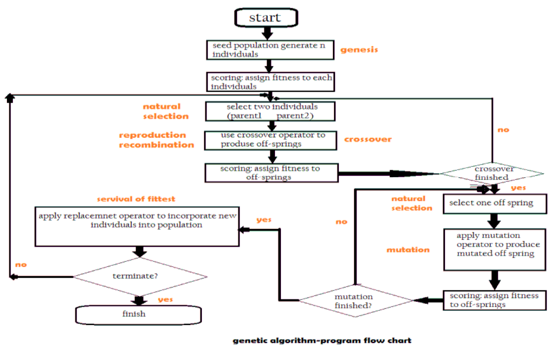
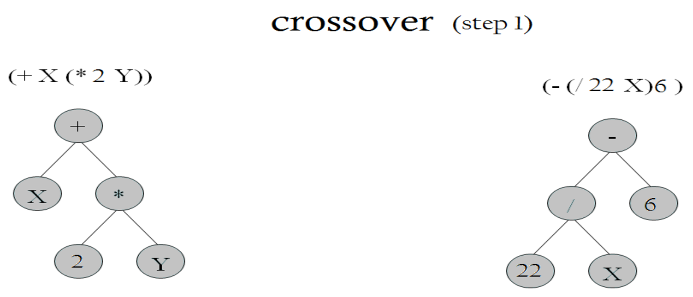
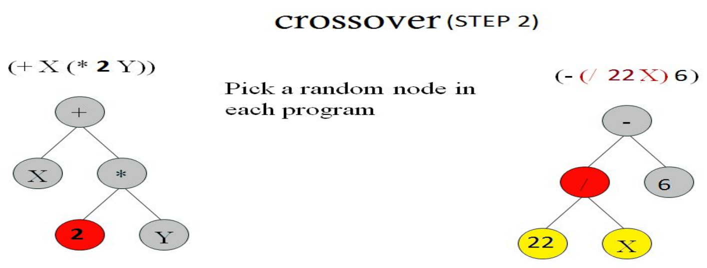

start
seed population generate n individuals
genesis
scoring: assign fitness to each individuals
natural selection
select two individuals (parent1 parent2)
reproduction recombination
use crossover operator to produse off-springs
crossover
scoring: assign fitness to off-springs
crossover finished
no
yes
servival of fittest
apply replacemnet operator to incorporate new individuals into population
yes
no
natural selection
select one off spring
mutation
apply mutation operator to produce mutated off spring
scoring: assign fitness to off-springs
mutation finished?
terminate?
no
yes
finish
genetic algorithm-program flow chart

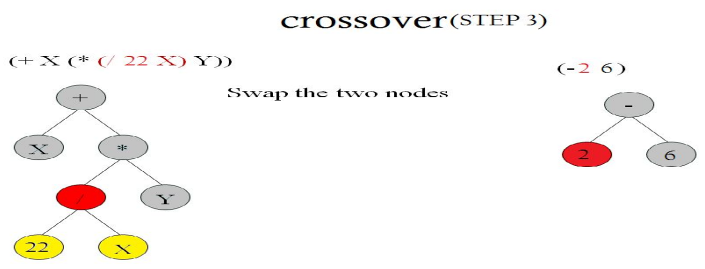
Fig.1.1

## II. GENETIC OPERATIONS IN GP

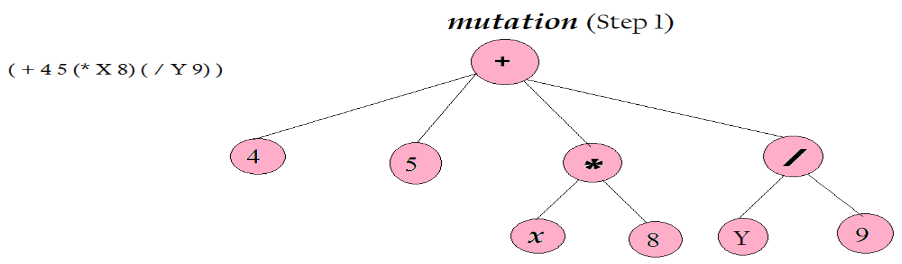
*A.* 1. Crossover is applied on an individual by simply switching one of its nodes with another node from another individual in the population. In the crossover operation, two solutions are sexually combined to form two new solutions or offspring. The parents are chosen from the population by a function of the fitness of the solutions. With a tree-based representation, replacing a node means replacing the whole branch. This adds greater effectiveness to the crossover operator. Illustrating crossover with an example :

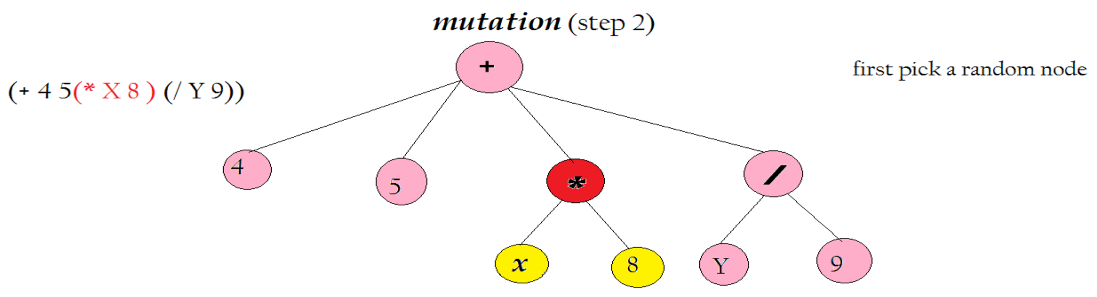


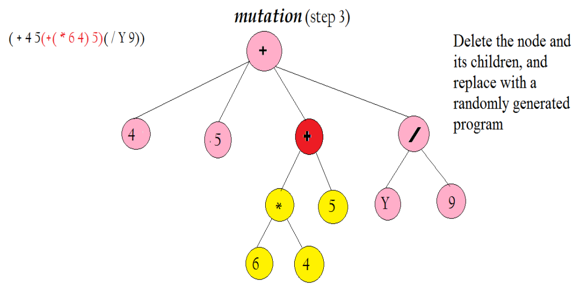




2. Mutation affects an individual in the population. It can replace a whole node in the selected individual, or it can replace just the node's information. The mutation operation potentially can be beneficial in reintroducing diversity in a population that may be tending to prematurely converge.[1]






mutation (step 3)
( + 4 5(+( * 6 4) 5)( / Y 9))
Delete the node and its children, and replace with a randomly generated program
+
4
5
+
/
*
5
Y
9
6
4

## V. Conclusion

Genetic programming methods are easy to apply to a wide range of problems, from optimization problems like the travelling salesperson problem, to inductive concept learning, scheduling, and layout problems. It is also believed that this work provides a strong basis for future work in several key areas .The results can be very good on some problems, and rather poor on others. If only mutation is used, the algorithm is very slow. Crossover makes the algorithm significantly faster. GA is a kind of hill-climbing search; more specifically it is very similar to a randomized beam search. In this report, we have placed more emphasis in explaining the use of GP in many areas of engineering and commerce with the help of its two of the basic operations performed; mutation and crossover. We believe that, through working out these interesting examples, one could grasp the basic idea of GP with greater ease.

## REFERENCES

[1] Alan Robinson. 2011, Genetic Programming: Theory, Implementation, and the Evolution of Unconstrained Solutions
[2] Spector, Barnum, and Bernstein 1998, Spector and Bernstein 2003http://www.genetic programming.com/humancompetitive.html
[3] Sastry, K., Johnson, D. D., Goldberg, D. E., Bellon, P. (2004). http://www.kumarasastry.com/2004/12/25/genetic-programming-for-multiscale-modeling/
[4] Ömer UZEL, Erdem KOÇ (2012), Basics of Genetic Programming
[5] *Genetic Programming III: Darwinian Invention and Problem Solving*, http://www.genetic- programming.com/attributes.html
[6] Aleksandra Takač 2003, Genetic Programming in Data Mining: http://ii.fmph.uniba.sk/~takaca/thesis/thesis.pdf
[7] Ömer UZEL & Erdem KOÇ 2012 Basics of Genetic Programming http://mcs.cankaya.edu.tr/proje/2012/ guz/omer_erdem/Rapor.pdf