LTI

Let's Solve

Genetic Programming

Author

Dinar Senjit

Technical Architect, LTI

A Larsen & Toubro Group Company

Abstract

Genetic programming is a bio-inspired programming technique where a problem statement is represented as set of chromosomes, that evolve and mutate to generate the most optimum executable program, according to fitness function. Genetic programming is particularly useful when the solution space is huge, and exhaustive search in the solution space is practically impossible.

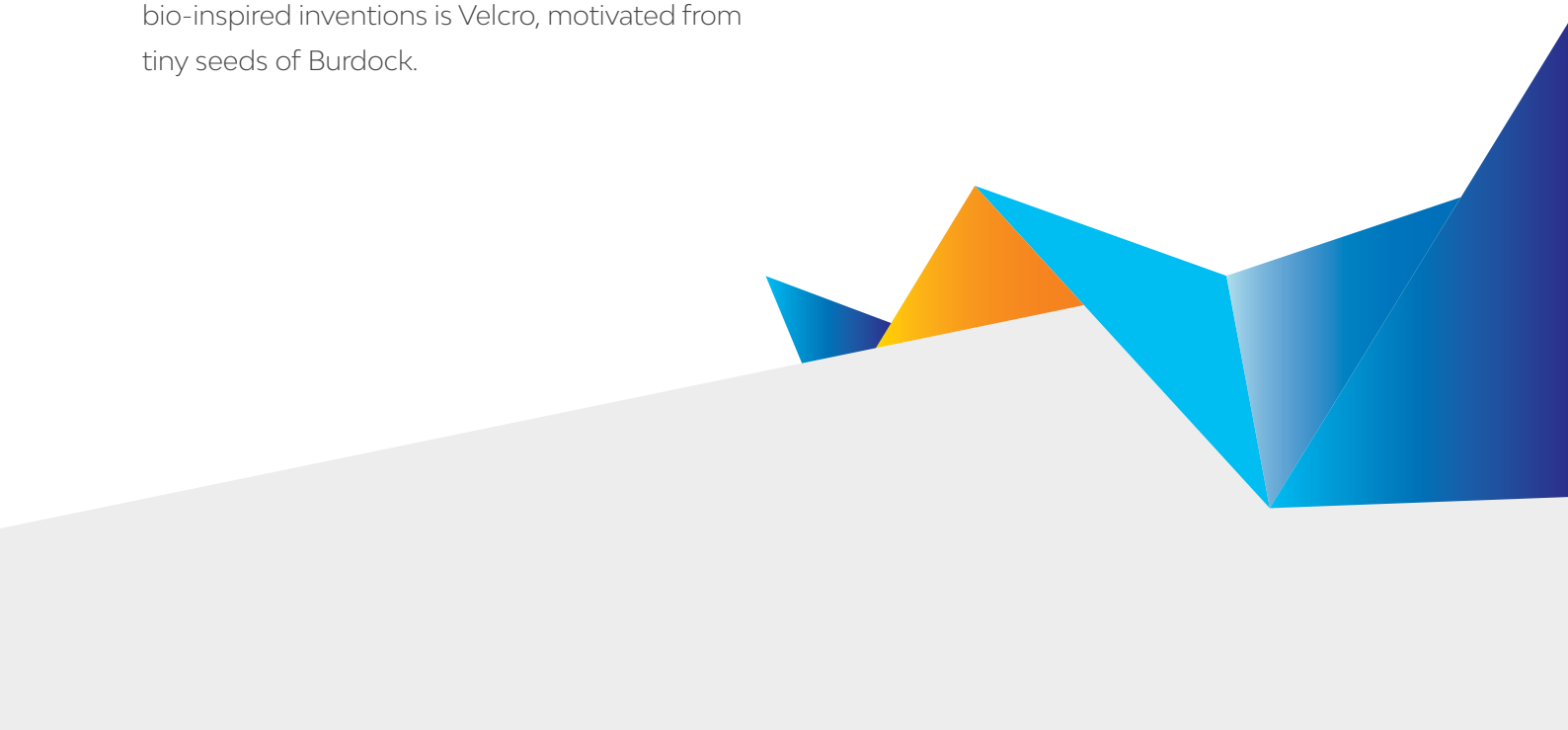
This document is for introduction to Generic Programming and overview of its major constituents.

Introduction to Genetic Programming:

There has been a metamorphosis in software programming techniques. The programming techniques have evolved from assembly language and procedural programming techniques to object-oriented and aspect-oriented programming. However, all the techniques fail when exhaustive search in the solution space is practically impossible.

Nature has been the motivation behind many human inventions. One of the most significant bio-inspired inventions is Velcro, motivated from tiny seeds of Burdock.

Genetic programming is such bio-inspired programming technique motivated from the process of natural selection and evolution. Genetic programming represents a problem statement in the form of chromosomes implemented as encoded string. These chromosomes undergo mutation and crossover to generate an optimal breed according to the fitness criteria for algorithm. Genetic Programming is a type of evolutionary algorithm, and is a subset of machine learning.



Key Concepts in Genetic Programming

Genetic Diversity

It is a characteristic of an ecosystem and a gene pool that describes an attribute, which is commonly held to be advantageous for survival.

Chromosomes

In Genetic Programming, a chromosome (also sometimes called a genome) is a set of parameters, which define a proposed solution to the problem that the Genetic Programming is trying to solve. The chromosome is often represented as a simple string, although a wide variety of other data structures are also used. For example, if the problem is to find the integer value of x between 0 and 255 that provides maximum result for f(x) = 2x, then our chromosome will be an 8 digit string. If the value of x is 132, then our chromosome will be 10000100.

Genetic Operators

Genetic operators act on chromosomes to create a new generation (or set) of chromosomes. Mutation and crossover are the two genetic operators commonly used in Genetic Programming.

1) Mutation refers to random change in a selected part of chromosome. For example, the chromosome mentioned above for f(x) as 10000100 will mutate to 10110110

Role of mutation in Genetic Programming:
Mutation prevents any two chromosomes of a population from becoming too similar to each other.

2) Crossover creates a new chromosome by combining two or more chromosomes. For example, if two chromosomes for f(x) in above example are 10000100 and 10000111, then crossover will produce a new chromosome as 10000111. Crossovers are of following types:

i. Single point crossovers:
In this type, one crossover point is selected, a binary string from beginning of chromosome to the crossover point is copied from one parent, the rest is copied from the second parent. For example: 11001011+11011111 = 11001111

ii. Two point crossovers:
In this type, two crossover points are selected, a binary string from beginning of chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point is copied from the second parent, and the rest is copied from the first parent. For example: 11001011 + 11011111 = 11011111

iii. Uniform crossovers:
In this type, bits are randomly copied from the first or from the second parent. For example: 11001011 + 11011101 = 11011111

Role of crossover in Genetic Programming:
Crossover minimizes structural similarity between any two populations for a particular chromosome.

Fitness Function:
Fitness function represents the parameters to be optimized for the problem. For our example, fitness function for f(x) is f(x') ≤ 255.

Process of problem formulation using Genetic Programming

Following steps are involved in problem formulation:

1. **Define chromosomes**
2. **Define a fitness function**
3. **Create initial population of chromosomes**
4. **Apply mutations and crossovers to create most optimal breed**

We apply the above steps for the problem to find the integer value of x between 0 and 255 that provides maximum result for f(x) = 2x

Then our **chromosome** will be a bit string of length 8.

Fitness function will be f(x') ≤ 255

Let the initial population be {10000100, 10111100, 10000111 and so on..}

We apply mutation and crossover to generate different string till the chromosome satisfies the fitness function.

Tools for Implementing Genetic Programming

FCUBE (Amazon cloud) allows to execute genetic programming on AWS.

Encog is a popular tool for advanced machine learning. MATLAB also provides tools and add-ons for genetic programming.

Refer:
http://geneticprogramming.com/software/
for more elaborate listing.

Strengths of Genetic Programming

Genetic programming techniques are evolutionary. When exhaustive search of a problem domain is not possible, genetic programming techniques often prove to be useful. Most of the times, genetic evolution is modeled in terms of mathematical function and assume no training data and run unsupervised. This makes it a unique candidate of choice in such problems.

Inadequacies of Genetic Programming

Genetic programming techniques are evolutionary, therefore tend to be less efficient to achieve desired level of accuracy. Often, it is found that the other ML techniques like ANN (Artificial Neural Networks) are more efficient to reach desired degree of accuracy. So genetic programming needs to be applied with caution.

Key Concepts in Genetic Programming

Like any technology, Genetic Programming has its own strengths and weaknesses. Also, although Genetic Programming offers slower performance as compared to other ML techniques, Genetic Programming is highly parallelizable. When run in parallel, it is possible to gain substantial benefits.

Below are some of the aspects of successful Genetic Programming implementation:

1) Degree of Parallelism:

Genetic Programming is highly parallelizable compared to other machine learning techniques. This is because a set of chromosomes can be mutated/crossed-over independent of other set. With optimum amount of hardware, it is possible to yield better performance from Genetic Programming.

2) Fitness Function:

Genetic Programming is dependent on a Fitness function to select the next generation of chromosomes. More efficient fitness function implies more quickly, and we can achieve desired solution. One of the key challenges in creating an efficient fitness function is to know key attributes and trends in the solution space. Fitness function must test each and every aspect of solution.

3) Problem Domain Consideration:

Genetic Programming is better suited for non-deterministic problem. It may practically be impossible to narrow down on a deterministic solution using Genetic Programming. Instead, if it is used for non-deterministic problems, then it may yield better results. For example, Genetic Programming may not predict exact temperature for a given location, but instead it may be more accurate in determining temperature trend for a given period.

Possible adaptation in Mosaic

With LTI's Mosaic suite and its ability to utilize cloud topology, it is possible to remedy above aspects. And it can potentially become part of the suite. There is, of course a long way to go for this.

Conclusion

Genetic programming has a potential to solve challenges. However, it needs to be applied with caution. Other techniques like Artificial Neural Networks have proved to be efficient over genetic programming in many cases. But genetic programming has its own areas of strength.

References:

1. Goldberg, D. E.: Genetic Algorithms in Search, Optimizations and Machine Learning, Morgan Kaufmann, 1989.

2. Holland, J. H.: Adaptation in natural and artificial systems, The University of Michigan Press, Ann Arbor, MI, 1975.

3. Genetic Programming: On the Programming of Computers by Means of Natural Selection (Koza 1992)

4. Wikipedia

5. http://geneticprogramming.com/software/

About the Author

Dinar Senjit
Technical Architect, LTI

Dinar Senjit works as Microsoft Technical Architect specializing in .NET, SharePoint and Dynamics CRM. He has about 13 years of hands on experience. Equipped with extensive domain knowledge in Insurance industry, he has worked on various solutions and applications across US and UK. He is a technology evangelist with special interest in Machine Learning, Elastic Search and other emerging technologies.