University of South Carolina - Upstate

**Genetic Algorithm**

Jonathan Porter Sullivan

Design and Analysis of Algorithms – CSCI 421

Dr. Lewis, Jerome

April 24, 2015

The genetic algorithm was originally invented in the 1970s by a computer scientist known as John Holland (I. Introduction). In general, genetic algorithms are based on a subpart of Charles Darwin’s theory of evolution. Specifically, genetic algorithms are based on natural selection. According to Wikipedia, a genetic algorithm is, “a search heuristic that mimics the process of natural selection (Genetic Algorithm).” The genetic algorithm is an algorithm that mimics natural selection by using naturally occurring events such as, evaluation, selection, crossover, and mutation.

To fully understand the concept behind a genetic algorithm one must first understand natural selection and “basic” biological terms such as, genes, chromosome, and population. Natural selection is the **“**process that results in the adaptation of an organism to its environment by means of selectively reproducing changes in its genotype, or genetic constitution (Natural Selection | Biology | Encyclopedia Britannica)”. In biology, genes are “segment[s] of DNA that are responsible for the physical and inheritable characteristics or phenotype of an organism (Gene).” A chromosome is “a structure within the cell that bears the genetic material (Chromosome).” A population is “the whole number of inhabitants occupying an area and continually being modified by increases (births and immigrations) and losses (deaths and emigrations) (Population | Biology and Anthropology)”. In a genetic algorithm the population is considered to be the search space containing possible solutions to the problem at hand. Each possible solution within the population is known as a chromosome or individual, which holds the genes and a fitness score. The terms chromosome and individual are used interchangeably. For clarity, the term chromosome will be used throughout this explanation. The setup for a genetic algorithm is simple. To aid in the understanding and process of a genetic algorithm a “Hello World” example will be used. Below are the steps needed to develop a genetic algorithm.

1. **INITIALIZATION**

Initialize a population of any size containing chromosomes with randomly generated encoded genes. Before the initialization step is complete, each chromosome’s genes must be encoded. The type of encoding used can change the complexity of the search meaning that different types of encoding can either increase or decrease the search complexity. There are several different types of encoding such as, binary encoding, permutation encoding, value encoding, and tree encoding. The type of encoding selected depends solely on the problem.

1. **Binary encoding** is the process of converting the genes to a binary string representation. The conversion can be accomplished by converting each character in the string to its corresponding ASCII character code. Then, converting the ASCII character code to its binary representation. This type of encoding is the most common in genetic algorithms due to the fact that the first genetic algorithm used this type of encoding. Binary encoding is not always the best type of encoding for a problem because after crossover and mutation corrections may need to be made to the genes. An example of a problem that could benefit from binary encoding would be the Knapsack problem. Each bit within the chromosome’s genes would represent whether a given item is present (e.g. 1 representing true and 0 representing false).

|  |  |
| --- | --- |
| Chromosome | Genes |
| Chromosome A | 101000011110 |
| Chromosome B | 000111010101 |

1. **Permutation encoding** is the process of representing the chromosome’s genes as a string of numbers. For instance, the traveling salesman problem would be a good fit for this type of encoding. Each number within the chromosome’s gene would represent the distance between each city. Permutation encoding is generally used for ordering problems or task ordering problems.

|  |  |
| --- | --- |
| Chromosome | Genes |
| Chromosome A | 15 35 22 18 29 |
| Chromosome B | 66 80 33 21 12 |

1. **Value encoding**, also known as direct value encoding, is the process of representing the chromosome’s genes as a string of complicated values. For instance, a chromosome’s genes would be represented as H1$%K@()%\*FJ. An example of a problem that would benefit from value encoding would be finding weights for Artificial Neural Networks (ANNs)[[1]](#footnote-1). Each gene in the string representation of a chromosome’s genes represents a given weight for a given input.

|  |  |
| --- | --- |
| Chromosome | Genes |
| Chromosome A | H1$%K@()%\*FJ |
| Chromosome B | @#D%” ^J;|Q!G |

1. **Tree encoding** is the process of representing the chromosome’s genes as a tree of objects. For example, a problem that would benefit from tree encoding would be Artificial Creativity[[2]](#footnote-2), also known as Computational Creativity. Take a moment to imagine a computer that could create jokes. The chromosome’s genes would contain a tree of words that could represent a possible joke. Tree encoding is generally used for evolving programs or expressions for evolutionary programming.

|  |  |
| --- | --- |
| Chromosome | Genes |
| Chromosome A |  |

1. **EVALUATION**

Evaluate each chromosome using a fitness function to calculate a fitness score. The fitness score represents how well a chromosome or possible solution fits the desired solution. As a common practice, a lower fitness score is more desirable than a higher fitness score. For instance, if our target genes were “Hello World!” and our genetic algorithm used value encoding, the fitness score for the example chromosome’s genes, H1$%K@()%\*FJ, would be 12 because the gene “H” is the correct value and correctly positioned in comparison to our target genes. If the genes for the chromosome being evaluated were “Hel1@ ]ol5d#” the chromosome’s fitness score would be 6. As with many of the operations in a genetic algorithm, the fitness function is problem dependent meaning the fitness function is designed specifically for the problem.

1. **SELECTION**

Select two parents from the population using a selection function such as roulette wheel selection, tournament selection, rank selection, elitism, stochastic universal sampling (SUS), and random selection. Selection is the process of choosing two chromosomes from the population to crossover or mate and create two offspring that may be better solutions to the problem thus increasing the fitness of the population overall. Depending on the problem, one selection function may be more suitable than another.

1. **Roulette Wheel selection**, also known as fitness proportionate selection, equates the probability of chromosome being selected as a parent to each chromosome’s fitness score by calculating the total fitness score of all chromosomes, generating a random number, and then iterating over the population while calculating an accumulated normalized fitness score. The normalized fitness score is obtained by dividing a single chromosomes fitness score by the sum of the total fitness scores’ of all chromosomes. When the accumulated normalized fitness score is greater than the random number, the chromosome at that point is then chosen as one of the parents for crossover. This means that the probability of a chromosome being selected as a parent is proportionate to its absolute fitness score. For example, if chromosomes with a higher fitness score are favored they will have a higher chance of being selected than a chromosome with a lower fitness score.

1. **Tournament selection** retrieves N random chromosomes from the population. N represents the number of competitors, or chromosomes, in the tournament as defined by the user. The fitness scores of the retrieved chromosomes are then compared to one another to find the fittest chromosome to be used as a parent for crossover.
2. **Rank selection** is, “similar to fitness-proportionate selection except that selection probability is proportional to relative fitness rather than absolute fitness (Rank Selection).” Chromosomes are selected based on their rank, which is assigned based on the chromosome’s fitness score. For example, if we assume that chromosomes with a lower fitness score are more desirable, the first chromosome could have a rank of ½ and a fitness score of 1. The second chromosome could have a rank of 1/3 and a fitness score of 3 and the fourth chromosome could have a rank of ¼ and a fitness score of 10, etc. The probability of a chromosome being selected as a parent depends solely on its rank, however every chromosome has the same probability of being selected. For instance, a chromosome with a fitness score that’s 10x more desirable than the weakest chromosome will have the same probability of being selected as a parent as the weakest chromosome.
3. **Elitism** is not necessarily a selection function in the sense that it returns parent chromosomes for crossover. Generally, elitism is used alongside other selection functions. Elitism retrieves N percent of the fittest chromosomes from the population and adds them to the next generation. N is a percentage, defined by the user, of the fittest chromosomes in the population. Thus, the rate at which a possible solution is found may increase.

1. **Random selection** selects two random parent chromosomes from the population to use for crossover.
2. **CROSSOVER**

Crossover the two parent chromosomes to create two offspring. Crossover is used to combine genes of the two parent chromosomes to develop a solution that may be a better or fitter solution to the problem. In essence, crossover functions mimic the naturally occurring event between real creatures, sex. The crossover functions described below are also known as Genetic Operators[[3]](#footnote-3). The type of the genetic operator used in a genetic algorithm depends on the type of encoding. For instance, one point, two point, uniform, and arithmetic crossover cannot be used for a genetic algorithm that uses tree encoding.

1. **One Point crossover** is a function that generates a random pivot point in the interval [0, N]. N represents the length or number of genes. Any genes after or before the pivot point, depending on preference, are then used to create two offspring. One point crossover can be applied to genetic algorithms that use binary, permutation, or value encoding (Crossover and Mutation).

|  |
| --- |
| Pivot Point |
| 4 |

|  |  |
| --- | --- |
| Parent Chromosome | Genes |
| Chromosome A | 11010001 |
| Chromosome B | 00101111 |

|  |  |
| --- | --- |
| Offspring Chromosome | Genes |
| Chromosome C | 11011111 |
| Chromosome D | 00100001 |

1. **Two Point crossover** is the same as One Point crossover except that it has two pivot points instead of one. Two point crossover can be applied to genetic algorithms that use binary or value encoding (Crossover and Mutation).
2. **Uniform crossover** is similar to one and two point cross over except that genes are crossed over based on a user specified probability. The points of crossover are randomly selected. For instance, assuming a uniform crossover probability of 50%, a random number of pivot points are selected. Then, 50% of the first parent chromosome’s genes between the pivot points and 50% of second parent chromosome’s genes between the pivot points are crossed over to create two offspring. Uniform crossover can be applied to genetic algorithms that use binary or value encoding (Crossover and Mutation).

|  |  |
| --- | --- |
| Parent Chromosome | Genes |
| Chromosome A | 11010001 |
| Chromosome B | 00101111 |

|  |  |
| --- | --- |
| Offspring Chromosome | Genes |
| Chromosome C | 00011101 |
| Chromosome D | 11100011 |

1. **Arithmetic crossover** performs some type of arithmetic operation on the parent genes. For example the two parent genes could be ANDed or ORed together to create two offspring. Arithmetic crossover can be applied to genetic algorithms that use binary or value encoding (Crossover and Mutation).

|  |  |
| --- | --- |
| Parent Chromosome | Genes |
| Chromosome A | 11010001 |
| Chromosome B | 00101111 |

|  |  |
| --- | --- |
| Offspring Chromosome | Genes |
| Chromosome C | 00000001 |
| Chromosome D | 11111111 |

1. **MUTATION**

Depending on the mutation probability, as defined by the user, mutate one or more of the genes in the newly created offspring. Mutation is used to add genetic diversity in the population. Without mutation, the genes of future generations would eventually be the same. Like crossover functions, mutation functions are also known as genetic operators. There are several types of mutation functions such as, Bit String Mutation, Order Changing, Boundary Mutation, Non-Uniform, and Uniform (Mutation (Genetic Algorithm))." The type of mutation function used is not only dependent on the problem but the type of encoding used. For example, if binary encoding is used bit string mutation “flips” a one in the chromosome’s genes to a zero or vice versa. Therefore, bit string mutation is a viable option if, and only if, binary encoding is used.

1. **EVOLUTION**

Finally, repeat the process starting at evaluation to evolve the population until a chromosome meets or exceeds a desired solution. For example, assuming that a lower fitness score is more desirable, a terminating condition for the “Hello World!” problem would exist when a chromosome’s fitness score is equal to zero meaning that the genes are equal to “Hello World”. As an alternative, a terminating condition could be when a maximum number of generations have been achieved.

1. **ANALYSIS**

Due to the stochastic[[4]](#footnote-4) nature of genetic algorithms and their heavy dependence on the problem a time complexity classification can be misleading. However, given a specific problem, a fixed population size, a fixed number of generations, a specific selection function, and a specific crossover function, one can derive a general approximation of the time complexity classification. Although, to best approximate the time complexity of a genetic algorithm one must determine what is known as the convergence time. To determine the general approximation of the time complexity classification one must first define a set of fixed variables. For example, if the terminating condition for a genetic algorithm were 2048 generations, had a population size of 400, used roulette wheel selection, and used one point crossover, we can assume a general time complexity classification of O(generations \* (selection + crossover)) where selection and crossover is O(N\*M). N represents the size of the population and M represents the size of the chromosomes.

1. **APPLICATIONS**

Genetic algorithms are applied and elegantly used in a wide variety of fields. According to Brains.org, genetic algorithms are used for, but not limited to:

1. Automotive Design
2. Engineering Design
3. Robotics
4. Evolvable Hardware
5. Joke and Pun Generation
6. Biomimetic Invention
7. Computer Gaming
8. Encryption and Code Breaking
9. Optimizing Chemical Kinetic Analysis
10. Marketing and Merchandising

In conclusion, a genetic algorithm is not just one algorithm but a set or family of algorithms designed to mimic natural selection by way of naturally occurring events such as, evaluation, selection, crossover, and mutation. Genetic algorithms can be used to solve problems with large, complex search spaces to find a local or global maxima or minima. However, genetic algorithms are not a one-for-all solution to any problem due to the limited knowledge of how the inner workings of genetic algorithms find solutions. As with any problem, the type of algorithm used as a solution must be researched, analyzed, and used with skepticism, before being implemented.

**Works Cited**

“Natural Selection | Biology | Encyclopedia Britannica.” *Encyclopedia Britannica Online.*

Encyclopedia Britannica, 14 May 2014. Web. 11 Apr. 2015. <<http://www.britannica.com/EBchecked/topic/406351/natural-selection>>.

“Gene.” *Gene – Biology-Online Dictionary.* Biology-Online.org, 4 June 2013. Web. 11 Apr.

2015. <http://www.biology-online.org/dictionary/Gene>.

"Chromosome." *Chromosome - Biology-Online Dictionary*. 5 Aug. 2008. Web. 11 Apr. 2015.

<http://www.biology-online.org/dictionary/Chromosome>.

Teitelbaum, Michael. “Population | Biology and Anthropology.” *Encyclopedia Britannica*

*Online.*Encyclopedia Britannica, 4 June 2013. Web. 11 Apr. 2015. <<http://www.britannica.com/EBchecked/topic/470303/population>>.

"Artificial Neural Network." *Artificial Neural Network,Artificial Neural Network Inventors |*

*Edubilla.com*. Edubilla.com. Web. 22 Apr. 2015. <http://www.edubilla.com/invention/artificial-neural-network/>.

“An Overview of Artificial Creativity.” *An Overview of Artificial Creativity | Think Artificial.*

Hrafn Th. Thorisson. Web. 11 Apr. 2015. <http://www.thinkartificial.org/artificial-creativity>.

Guervos, J.J. Merelo. "Genetic Operators." *Genetic Operators*. 22 Aug. 1997. Web. 22 Apr.

2015. <http://kal-el.ugr.es/GAGS/gags-tutorial/node3.html>.

“Rank Selection.” *Rank Selection.* Web. 11 Apr. 2015.

<http://watchmaker.uncommons.org/manual/ch03s03.html>.

“Crossover and Mutation.” *Crossover and Mutation.* Marek Obitko, 1998. Web. 20 Apr. 2015.

<<http://www.obitko.com/tutorials/genetic-algorithms/crossover-mutation.php>>.

"Stochastic." *Stochastic - Definition and More from the Free Merriam-Webster Dictionary*.

Merriam-Webster. Web. 20 Apr. 2015. <http://www.merriam-webster.com/dictionary/stochastic>.

"Mutation (Genetic Algorithm)." *Wikipedia*. Wikimedia Foundation, 26 Jan. 2015. Web. 22 Apr.

2015. <http://en.wikipedia.org/wiki/Mutation\_(genetic\_algorithm)>.

Obitko, Marek. "I. Introduction." *Introduction*. Marek Obitko, 1 Jan. 1998. Web. 23 Apr.

2015. <http://www.obitko.com/tutorials/genetic-algorithms/introduction.php>.

"Genetic Algorithm." *Wikipedia*. Wikimedia Foundation, 21 Apr. 2015. Web. 23 Apr. 2015.

<http://en.wikipedia.org/wiki/Genetic\_algorithm>.

"15 Real-World Uses of Genetic Algorithms." *15 Real-World Applications of Genetic*

*Algorithms*. Brainz.org. Web. 23 Apr. 2015. <http://brainz.org/15-real-world-

applications-genetic-algorithms/>.

1. **Artificial Neural Networks (ANNs)** – “A family of statistical learning algorithms inspired by biological neural networks (the central nervous systems of animals, in particular the brain) and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown (Artificial Neural Network).” [↑](#footnote-ref-1)
2. **Artificial Creativity** – “A branch of Artificial Intelligence that deals with the development and exploration of systems that exhibit creative behavior. This includes systems capable of such things as scientific invention, visual artistry, music composition, and story generation (An Overview of Artificial Creativity).” [↑](#footnote-ref-2)
3. **Genetic Operator** – “used in genetic algorithms to generate diversity (mutation-like operators) and to combine existing solutions into others (crossover-like operators). The main difference among them is that the former operate on one chromosome, that is, they are unary, while the latter are binary operators. (Genetic Operators).” [↑](#footnote-ref-3)
4. **Stochastic** – “involving chance or probability:  probabilistic (Stochastic).” [↑](#footnote-ref-4)