The basic purpose of a genetic algorithm (GA) is to mimic Nature's evolutionary approach. The algorithm is based on the process of natural selection—Charles Darwin's "survival of the fittest." GAs can be used in problem solving, function optimizing, machine learning, and in innovative systems. Historically, GAs were first conceived and used by John Holland (1975) at the University of Michigan. Genetic algorithms are useful and efficient when:

- The search space is large, complex, or poorly understood
- Domain knowledge is scarce or expert knowledge is difficult to encode to narrow the search space
- No mathematical analysis is available

According to Hales (2006), instead of just accepting a random testing strategy to arrive at a solution, one may choose a strategy like:

- Generate a set of random solutions
- Repeat
  - Test each solution in the set (rank them)
  - Remove any bad solutions from the set (bad solutions refer to solutions which are less likely to provide effective answers according to fitness criteria given by experts)
- Duplicate any good solutions or make small changes to some of them
- Until an appropriate solution is achieved
DNA is the building block of bio-cells and strings of DNA is chromosomes.

Chromosomes can be bit strings, real numbers, permutations of elements, lists of rules, and data structures.

On each DNA string, there is a set of genes responsible for some property of a human (or living thing) to which it belongs.

Such properties, to name a few, are height, skin color, hair color, and eye color.

A genotype is a collection of such genes representing possible solutions in a domain space.

This space, referred to as a search space, comprises all possible solutions to the problem at hand.

An initial population is considered to have a fixed number of individuals (also known as offspring) containing building blocks or chromosomes on which genes are set.

A fit individual with strong genes can reproduce itself.

Otherwise, in the next generation, the strong genes from one or more individuals can be combined, resulting in a stronger individual. Over time, the individuals in the population become better adapted to their environment.
The most common method of encoding is with a binary string. Each individual represents one binary string. Each bit in this string can represent some characteristic of the solution. Then the individual (or chromosome) can be represented as shown below:

Consider a simple function $f(x)=x^2$ on the integer interval $[0,30]$.

- Consider the following four strings in initial population: 01110 11100 01100 10011
- Use a simple genetic algorithm composed of three operators (Reproduction, Crossover and Mutation) to optimize the aforementioned function in given interval.
4/22/2014

**Function Optimization - 1**

**Function Optimization - 2**

Table: First Generation of \( f(x) = x^2 \)

<table>
<thead>
<tr>
<th>Sr. No. of Individual</th>
<th>Value of ( x )</th>
<th>Decimal value of ( x )</th>
<th>Fitness ( f(x) = x^2 ) (Value in decimal)</th>
<th>Roulette wheel selection count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01101</td>
<td>13</td>
<td>169</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>11000</td>
<td>24</td>
<td>576</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>01000</td>
<td>08</td>
<td>64</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>10011</td>
<td>19</td>
<td>361</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure: Roulette wheel presentation for the first generation shown in the \( f(x) = x^2 \) example

**Function Optimization - 3**

<table>
<thead>
<tr>
<th>Sr. No. of Individual</th>
<th>Value of ( x )</th>
<th>Decimal value of ( x )</th>
<th>New Individual after mutation and cross-over</th>
<th>Selection site</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01100</td>
<td>12</td>
<td>144</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>11001</td>
<td>25</td>
<td>625</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>11011</td>
<td>27</td>
<td>729</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>10000</td>
<td>16</td>
<td>256</td>
<td>1</td>
</tr>
</tbody>
</table>

Table: Mutation and crossover after reproduction for \( f(x) = x^2 \)

**Function Optimization - 4**

<table>
<thead>
<tr>
<th>Sr. No. of Individual</th>
<th>Value of ( x )</th>
<th>Value of ( y )</th>
<th>Bit string</th>
<th>Fitness ( f(x) = x+y ) (Value in decimal)</th>
<th>Roulette wheel selection count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01101</td>
<td>01010</td>
<td>01010101010</td>
<td>10+11=21</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>11000</td>
<td>01100</td>
<td>11000110010</td>
<td>24+13=37</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>01010</td>
<td>00101</td>
<td>01010010101</td>
<td>5+5=10</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>01111</td>
<td>00111</td>
<td>01110111111</td>
<td>11+7=18</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure: Roulette wheel presentation for the first generation shown in the \( f(x,y) = x+y \) example

*Maximize \( f(x,y) = x + y \) in the interval of \([0, 30]\) with binary encoding strategy of size 10, with the first 5 bits representing a value of the first variable \( x \), and the remaining bits representing a value of the second variable \( y \).*
Initial Population for TSP covering any five city from given six cities:

- (5,4,6,2)
- (2,4,6,3,5)
- (4,3,6,5,2)
- (2,3,4,6,5)
- (4,3,6,2,5)
- (5,4,2,3,6)
- (4,6,3,2,5)
- (3,4,2,6,5)
- (3,6,5,1,4)

Example Courtesy: Jim Cohoon and Kimberly Hanks

Generation of more Individuals:

- (5,3,4,6,2)
- (2,4,6,3,5)
- (4,3,6,5,2)
- (2,3,4,6,5)
- (4,3,6,2,5)
- (5,4,2,3,6)
- (4,6,3,2,5)
- (3,4,2,6,5)
- (3,6,5,1,4)

Mutation:

- (5,3,4,6,2)
- (2,4,6,3,5)
- (4,3,6,5,2)
- (2,3,4,6,5)
- (4,3,6,2,5)
- (5,4,2,3,6)
- (4,6,3,2,5)
- (3,4,2,6,5)
- (3,6,5,1,4)
**Finding an optimal ordering** for a sequence of N items.

- For example, in a traveling salesman problem, consider there are four cities 1, 2, 3, and 4.
- Each city is then, labeled by a unique bit string.
- A common fitness function for the problem is **length of the candidate tour**.
- A natural way is the permutation, so that 3214 is one candidate’s tour and 4123 is another.
- This representation is **problematic** for genetic algorithm because Mutation and crossover do not produce necessarily legal tours.
- For example, cross over between position 2 and 3 produces (in the example 3214 and 4123) produces the individuals 3223 and 4114, both of which are illegal tours.

- Adopting a different representations
- Designing a special crossover operators
- Penalizing the illegal solution with the proper fitness function.
For the problems, which are complex, noisy or dynamic, it is virtually impossible to predict the performance of a genetic algorithm.

Holland[1975], introduced the notion of schema to explain how genetic algorithms search for region of high fitness.

A schema is a template, defined over alphabet {0, 1, *} which describes a pattern of bit strings in the search space {0, 1}^l, where l is length of a bit string.

For example, the two strings A and B have several bits in common. We can use schemas describe the patterns these two strings A=101011 and B=010011 share:

\*0\*11
***11
**0**
**0**1

A bit string x that matches a schema S's pattern is said to be an instance of S. For example, A and B both are instances of the schemas.

Order of schema is the number of defined bits in it. For example, order of the first schema \**0**11 is 3.

Defining length \( \delta \) of a schema is the distance between the left most and right most defined bits in the schema.

For example, the defining length of \**0**11 is 3. Other examples are

- for \( S_1 = [01#1#] \), \( \delta(S_1) = 4 – 1 = 3 \)
- for \( S_2 = [##1#1010] \), \( \delta(S_2) = 8 – 3 = 5 \)

Programs written in subset of LISP language and any point in the space can be simultaneously an instance of two schemas.

This figure defines four hyper planes corresponding to the four different schemas.

Any point in the space gives some information about the average fitness of the 2^l different schemas (class) of which it is an instance.

The fitness of any bit string in the population gives some information about the average fitness of the 2^l different schemas (class) of which it is an instance.

So an explicit evaluation of population gives some information about the average fitness of the 2^l different schemas (class) of which it is an instance.

This is referred to as implicit parallelism.

Programs which can be represented in a tree are the candidate of evolution under genetic fashion/natural selection.

Population of random trees are generated and evaluated as in the standard genetic algorithm.

They have special crossover function.

Programs written in subset of LISP language and

Expression: \( s^x + 3y + z^2 \)

LISP: \( (+(*x*x)(*y*y)) \)
Human written programs are designed elegant and general, but the genetic generated programs are often needlessly complicated and not revealing the underplaying algorithm.

For $\cos 2x$ we write $(-1[2(\sin x)(\sin x)])$.

Genetic program discovers

- $\sin(-1(2[x^2])(\sin(\sin(\sin(\sin(\sin(\sin(\sin(\sin(\sin(\sin (1))))))))))))$.

The evolved programs are inelegant, redundant, inefficient, and difficult for human read.

They give adhoc type of solutions that evolve in nature through gene duplication, mutation, and modifying structures.

If successful, it leads to the design revolutions.

Optimization, combinatorial, and scheduling problems

- Automatic programming
- Game playing
- Self-managing and sorting networks
- Machine and robot learning
- Evolving artificial neural network (ANN), rules-based systems and other hybrid architectures of soft computing
- Designing and controlling robots
- Modeling natural systems (to model processes of innovation)
- Emergence of economic markets
- Ecological phenomena
- Study of evolutionary aspects of social systems, such as the evolution of cooperation, evolution of communication, and trail-following behavior in ants
- Artificial life models (systems that model interactions between species evolution and individual learning)

Reference