Genetic Algorithms

Diwakar Yagyasen
Department of Computer Science
BBDNITM

Visit dylycknow.weebly.com for detail

Genetic Algorithms

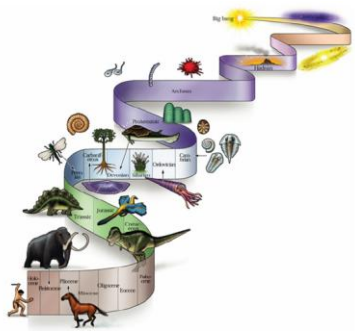


Image from http://www.geo.au.dk/besoegsservice/foredrag/evolution

Genetic Algorithms

- 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

Image adapted from http://today.mun.ca/news.php?news_id=2376

Genetic Algorithms

- 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**

Genetic Algorithms

- **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.

Genetic Algorithms

- With every evolutionary step, known as a *generation,* the individuals in the current population are *decoded* and *evaluated* according to some predefined quality criterion, referred to as the *fitness* or *fitness function*.
- **The degree of *fitness* is related to the strength of the genes in an individual.**
- When new child is born, the parents' genes are inherited sometimes directly, without any modification, if they are strong enough. This process is known as **duplication** (exact resemblance of the child to its parent) in new generation.
- **Strong genes are normally represented in the new generation.**

Genetic Algorithms

- **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.**

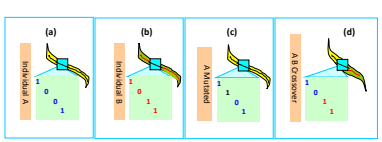


Figure : Operations in the process of natural selection

Genetic Algorithms

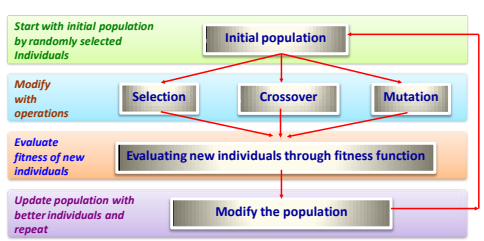


Figure : Genetic cycle

Genetic Algorithms

Encoding Strategy

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:

Individual 1	0	1	0	1	0	1	0	0	1	0	0	1	0	1	1	1
Individual 2	1	1	0	0	0	1	1	0	0	0	0	1	0	0	1	0

Genetic Algorithms

Mutation

Individual 1	0	1	0	1	0	1	0	0	1	0	0	1	0	1	1	1
New Individual 1	0	1	0	1	1	1	0	1	1	0	0	1	0	0	1	1
Individual 2	1	1	0	0	0	1	1	0	0	0	0	1	0	0	1	0
New Individual 2	0	0	0	1	0	1	1	0	0	1	0	1	0	0	1	0

Crossover

Individual 1	0	1	0	1	0								0	1	0	1	1	1
Individual 2	1	1	0	0	0	1					0	0	0	0		
New Individual 1	1	1	0	0	0	0	0	0	1	0	0	1	0	1	1	
New Individual 2	0	1	0	1	0	1	1	0	0	0	0	1	0	0	1	0

Substrings of same length

Images from wpclipart.com

Genetic Algorithms

Selection

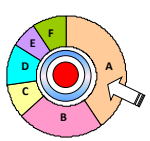
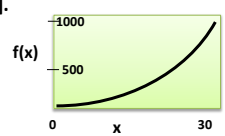


Figure: Roulette wheel selection

Genetic Algorithms

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



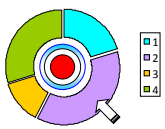
- **Consider the following four strings in initial population:**
  01101
  11000
  01000
  10011
- **Use a simple genetic algorithm** composed of three operators(Reproduction, Crossover and Mutation) **to optimize** the aforementioned function in given interval.

Genetic Algorithms

Sr. No. of individual	Value of x	Decimal value of x	Fitness f(x) = x*x (Value in decimal)	Roulette wheel selection count
1	01101	13	169	1
2	11000	24	576	2
3	01000	08	064	0
4	10011	19	361	1

Table : First Generation of f(x)=x*x

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



Genetic Algorithms

Sr. No. of selected individual	Value of x	New individual after mutation and cross-over	Operation site	Fitness f(x) = x*x (Value in decimal)
1	01101	01100	5	12*12=144
2	11000	11001	5	25*25=625
3	11000	11011	3	27*27= 729
4	10011	10000	3	16*16= 256

Table : Mutation and crossover after reproduction for f(x)=x*x

Sr. No. of Individual	Value of x	Decimal value of y	Fitness f(x) = x*x (Value in Decimal)
1	01100	12	144
2	11001	25	625
3	11011	27	729
4	10000	16	256

Table: Second generation of the function f(x)=x*x

Genetic Algorithms

f(x,y) = x + y

Sr. No. of individual	Value of x	Value of y	Bit string	Fitness f(X) = x+y (Value in decimal)	Roulette wheel selection count
1	01011	01010	0101101010	10+11= 21	1
2	11000	01100	1100001100	24+12=36	2
3	01001	00101	0100100101	9+5=14	0
4	01011	00111	0101100111	11+7=18	1

Table: First generation of f(x,y)= x+y

- 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.

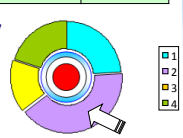
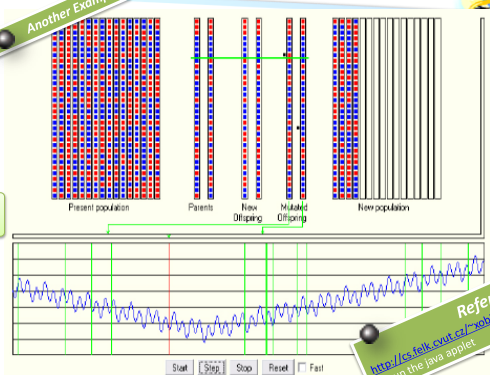


Figure : Roulette wheel presentation for the generation shown in the f(x,y)= x+y example

Genetic Algorithms

Another Example



Reference
http://cs.felk.cvut.cz/~xobitko/ga/
To run the java applet

Genetic Algorithms

Introduction
Fundamentals
Algorithm
Function Optimization - 1
Function Optimization - 2
Ordering Problems
Edge Recombination
Schema
Genetic Programming

Traveling Salesperson Problem

Initial Population for TSP

covering any five city from given six cities

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

Genetic Algorithms

Introduction
Fundamentals
Algorithm
Function Optimization - 1
Function Optimization - 2
Ordering Problems
Edge Recombination
Schema
Genetic Programming

Traveling Salesperson Problem

Generation of 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)	(3,4,5,2,6)
(3,5,4,6,2)	(4,5,3,6,2)	(5,4,2,3,6)
(4,6,3,2,5)	(3,4,2,6,5)	(3,6,5,1,4)

(3,4,5,6,2)

Genetic Algorithms

Introduction
Fundamentals
Algorithm
Function Optimization - 1
Function Optimization - 2
Ordering Problems
Edge Recombination
Schema
Genetic Programming

Traveling Salesperson Problem

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

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

Genetic Algorithms

Introduction
Fundamentals
Algorithm
Function Optimization - 1
Function Optimization - 2
Ordering Problems
Edge Recombination
Schema
Genetic Programming

Traveling Salesperson Problem

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

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

Genetic Algorithms

Traveling Salesperson Problem

Elimination of the worst tours

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

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

Genetic Algorithms

- 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 leagal 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.**

Genetic Algorithms

Solution

- Adopting a different representations
- Designing a special crossover operators
- Penalizing the illegal solution with the proper fitness function.

Genetic Algorithms

3 6 2 1 4 5
5 2 1 3 6 4

Original Individuals

→ 3 6 4 1 2 5

New Individual

Key	Adjacent Keys
1	2, 2, 3, 4
2	1, 1, 5, 6
3	1, 6, 6
4	1, 5, 6
5	2, 4
6	2, 3, 3, 4

Genetic Algorithms

- For the problems, which are complex, noisy or dynamic, it is virtually impossible t 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 to describe the patterns these two strings A=100111 and B=010011 share:

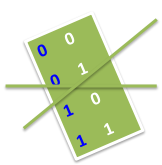
  **0*11
  ****11
  0*
  01

Genetic Algorithms

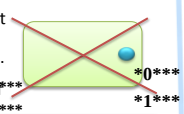
- A bit string x that matches a schema S's pattern is said to be an instance of S. For example, A and B both are instances of the schemas.
- Order of schema is the number of defined bits in it. For example, order of the first schema **0*11 is 3.
- Defining length δ of a schema is the distance between the left most and right most defined bits in the schema.
- For example, the defining length of **0**1 is 3. Other examples are
  - for S_1 = [01#1#], $\delta(S_1) = 4 - 1 = 3$
  - for S_2 = [##1#1010], $\delta(S_2) = 8 - 3 = 5$

Genetic Algorithms

- Schema defines hyper plane in the search space $\{0,1\}^l$.
- This figure defines four hyper planes corresponding to the four different schemas.



- Any point in the space can be simultaneously an instance of two schemas.
- 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 of M individual strings is also an implicit (implied) evaluation of a much larger number of schemas.



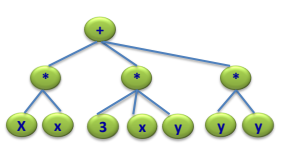
*is an instance of 1**** and *0****.*

- This is referred to as implicit parallelism.

Genetic Algorithms

- Programs written in subset of LISP language and
- Programs which can be represented in a tree are the candidate of evolution under genetic fashion/natural selection.
- Population of a random trees are generated and evaluated as in the standard genetic algorithm.
- They have special crossover function.

Expression:	$x^2+3xy+y^2$
LISP:	(+(*xx)(*xy)(*yy))



Genetic Algorithms

- Human written programs are **design elegant** and **general** but the genetic generated programs are **often needlessly complicated** and not revealing the underplaying algorithm.
- **For cos2x we write (-1(*2(*sin x)(sin x)))**
- Genetic program discovers
- (sin(-(-2(*x2))(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.**

Genetic Algorithms

Research Directions

- **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)**

Genetic Algorithms

Reference

- *"Knowledge-based systems"*, **Akerkar RA and Priti Srinivas Sajja, Jones & Bartlett Publishers, Sudbury, MA, USA (2009)**
- **Allen B. Tucker,Jr. The Computer Science and Engineering handbook, CRC Press, 1997,** (pp. 557 to pp.569)