Genetic Algorithms

Sayad Alizadeh **Maryam Shamsadini**,

Meyandouab Isamic Azad University

Summary
Genetic algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence.
Genetic algorithms are inspired by Darwin's theory about evolution simply said; solution to a problem solved by genetic algorithms is involved.

Key words:
GA, Fitness, Crossover, Mutation

Introduction

Genetic algorithms are so far generally the best and most robust kind of evolutionary algorithms. For most problems you don't have any formula for solving the problem because it is too complex, or if you do, it just takes too long to calculate the solution exactly.
Genetic algorithms are different from other heuristic methods in several ways. The most important difference is that a GA works on a population of possible solutions, while other heuristic methods use a single solution in their iterations. Another difference is that GA s are probabilistic (stochastic), not deterministic.
An implementation of a genetic algorithm begins with a population of (typically random) chromosomes. All living organisms consist of cells. In each cell there is the same set of chromosomes. Chromosomes are strings of DNA and serves as a model for the whole organism.
A chromosome consists of genes, blocks of DNA. Each gene encodes a particular portion. Basically can be said, that each gene encodes a trait, for example color of eyes has it is own position in the chromosomes. This position is called locus.
One individual might have these genes:"1100101011", another has these: "0101110001" (just examples).
The values (0 or 1) and their position is a point in a search space and a possible solution. The individuals in the population are then made to go through a process of evolution.

Manuscript received April 5, 2011
Manuscript revised April 20, 2011

Search space

If we are solving some problem, we are usually looking for some solution, which will be the best among others.
The space of all feasible solution (it means objects among those the desired solution is) is called search space (also state space). Each feasible solution can be marked by it is value of fitness for the problem. We are looking for our solution, which is one point (or more) among feasible solutions that are one point in the search space.
The search space can be whole known by the time of solving a problem but usually we know only a few points from it and we are generating other points as the process of finding solution continues.

General structure of genetic algorithm

[Start] Generate random population of n chromosomes (suitable solutions for the problem)
[Fitness] Evaluate the fitness of each chromosome in the population
[New population] Create a new population by repeating following steps until the new population is complete
[Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
[Crossover] With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
[Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).
[Accepting] Place new offspring in a new population
[Replace] Use new generated population for a further run of algorithm
[Test] If the end condition is satisfied, stop, and return the best solution in current population
[Loop] Go to step 2
Algorithm is started with a set of solutions (represented by chromosomes) called population. Solution from one population are taken and used to form a new population. This is motive by a hope, that the new population will be better than the old one.

Fitness

Associated with each individual is a fitness value. This value is a numerical quantification of how good of a solution to the optimization problem the individual is. Individuals with chromosomal strings representing better solutions have higher fitness values, while lower fitness values are attributed to those whose bit strings represent inferior solutions.
Solutions which are selected to form new solutions (offspring) are selected according to their fitness the more suitable they are the more chances they have to reproduce.
This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

Selection

The selection phase plays an important role in search towards better individuals. During each successive generation, a proportion of the existing population is selected to breed a new generation. The selection process is the step that guides the genetic algorithm towards ever-better solutions.

Reproduction

Main articles: crossover (genetic algorithm) and mutation (genetic algorithm)
The next step is to generate a second generation population of solutions from those selected through genetic operators: crossover (also called recombination), and/or mutation.
For each new solution to be produced, a pair of parent solutions is selected for breeding from the pool selected previously. By producing a child solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its parents. New parents are selected for each child, and the process continues until a new population of solutions of appropriate size is generated.

Crossover

After we have decided what encoding we will use, we can make a step to crossover. Crossover selects genes from parent chromosomes and creates a new offspring (One crossover thus create two new individuals, called offspring).
Crossover can then look like this:

Chromosome 1	11011 | 00100110110
Chromosome 2	11011 | 11000011110
Offspring 1	11011 | 11000011110
Offspring 2	11011 | 00100110110

The simplest way how to do this is to choose randomly some crossover point and everything before this point copy from a first parent and then everything after a crossover point copy from the second parent.
These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions, for reasons already mentioned above.
Holland explains that, "the purpose of crossing strings in the genetic algorithm is to test new parts of target regions rather than testing the same string over and over again in successive generation". [1]
There are other ways how to make crossover, for example we can choose more crossover points. Crossover can be rather complicated and very depends on encoding of the encoding of chromosome. Specific crossover made for a specific problem can improve performance of the genetic algorithm.

Mutation

After a crossover is performed and before evaluating the new population, one final genetic operator is applied (takes place): mutation. This is to prevent falling all solutions in population into a local optimum of solved problem. Mutation changes randomly the new offspring. For binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1. Mutation can then be following:

Original offspring 1	1101111000011110
Original offspring 2	1101100100110110
Mutated offspring 1	1100111000011110
Mutated offspring 2	1101101100110110

The mutation depends on the encoding as well as the crossover. For example when we are encoding permutations, mutation could be exchanging two genes.
Note that the methods mentioned are just one way of making the GA work. There are many different methods of selection (how many parents to select? Kill the parents after cross-over? How to measure the fitness? etc.), many methods of cross-over (number of cross-over points, how to reassemble cross-over parts), and many methods of mutation (random noise, swapping values, no mutation, who to mutate, etc.).
The choice of methods will depend on the nature of the problem and your personal preferences.

Parameters Probability

Crossover probability says how often will be crossover performed. If there is no crossover, offspring is exact copy of parents. If there is a crossover, offspring is made from parts of parents' chromosome. If crossover probability is 100%, then all offspring is made by crossover. If it is 0%, whole new generation is made from exact copies of chromosomes from old population (but this does not mean that the new generation is the same!).
Crossover is made in hope that new chromosomes will have good parts of old chromosomes and maybe the new chromosomes will be better. However it is good to leave some part of population survive to next generation.
Mutation probability says how often will be parts of chromosome mutated. If there is no mutation, offspring is taken after crossover (or copy) without any change. If mutation is performed, part of chromosome is changed. If mutation probability is 100%, whole chromosome is changed, if it is 0%, nothing is changed.
Mutation is made to prevent falling GA into local extreme, but it should not occur very often, because then GA will in fact change to random search.
Population size says how many chromosomes are in population (in one generation). If there are too few chromosomes, GA has a few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many chromosomes, GA slows down. Research shows that after some limit (which depends mainly on encoding and the problem) it is not useful to increase population size, because it does not make solving the problem faster.

Advantages or Disadvantages

A GA has a number of advantages. It can quickly scan a vast solution set. Bad proposals do not affect the end solution negatively as they are simply discarded. The inductive nature of the GA means that it doesn't have to know any rules of the problem - it works by its own internal rules. This is very useful for complex or loosely defined problems. The disadvantage of it is that as you already know from the chapter about search space, problem solving can be often expressed as looking for extreme of a function. This is exactly what the problem shown here is.
Some function is given and GA tries to find minimum of the function. For other problems we just have to define search space and the fitness function which means to define the function, which we want to find extreme for.

Conclusion

If the conception of a computer algorithms being based on the evolutionary of organism is surprising, the extensiveness with this algorithms is applied in so many areas is no less than astonishing. These applications, commercial, educational and scientific, are increasingly dependent Genetic Algorithms. Its usefulness and gracefulness of solving problems has made it the more favorite choice among the traditional methods, namely gradient search, random search and others. GA s is very helpful when the developer does not have precise domain expertise, because GA s possesses the ability to explore and learn from their domain.

References

[1] Holland, John H. 2002. Genetic Algorithms. Scientific American, July, pp.
[2] Acquaintance with genetic algorithm by Mahmud Amin Tossi
[3] Http://www.lcc.uma.es/~ccottap/semEC/cap03/cap_3.html
[4] EvoNews 8 Posted: 14 August 2008
[5] Http://en.wikipedia.org/wiki/Genetic_algorithm
[6] Genetic Algorithms" by David Goldberg, Addison Wesley,
[7] Genetic Algorithms in Engineering and Computer Science , edited by G.Winter, J.Periaux & M.Galan, published by JOHN WILEY & SON Ltd.
[8] Jean-Philippe Rennard. Ph.D. May 2010http://www.rennard.org/alife alife@rennard.org
[9] Evolving solutions: an introduction to genetic algorithms by Greg Badros
[10] Genetic Algorithms Applications Edited by Lance Chambers pg102
[11] Artificial Intelligent Book by Dr Mehrdad Fahimi
[12] Computational Intelligence by John Wiley & Sons Ltd,2010 pg178