Abstract

This paper covers the canonical genetic algorithm as well as more experimental forms of genetic algorithms, including parallel island models and parallel cellular genetic algorithms. The theoretical foundations of genetic algorithms are reviewed, include the schema theorem as well as recently developed exact models of the canonical genetic algorithms.

Keywords: Genetic Algorithms, Search, Parallel Algorithms, Evolutionary Computing, NP, Co-NP.

Introduction

Genetic Algorithm is a part of Evolutionary Computing, which is a rapidly growing area of artificial intelligence. Genetic algorithms are designed from the inspiration of Darwin's theory about evolution [6,9,11-14].

Rechenberg, in his work "Evolution strategies" (Evolutionstrategie in original) first introduced the idea of evolutionary computing in the 1960's. His idea was then developed and modified by other researchers. Genetic Algorithms (GAs) in its present form were first invented by John Holland, his students and colleagues. This lead to Holland's book "Adoption in Natural and Artificial Systems" that was published in 1975[6,9,14]. Then in 1992 John Koza has used genetic algorithm to evolve programs to perform certain processes. He introduced his method as "genetic programming" (GP). He used LISP programs, because programs in this language can be expressed in the form of a "parse tree", which is the object the GA [5,6,9].

Background of GA

Chromosome

All living organisms consist of billions and billions number of cells. In each cell there is the same set of chromosomes. Chromosomes are nothing but the strings of DNA and serves as a basis structure for the whole organism. A chromosome consists of genes, blocks of DNA[9]. Each gene represents the encoded form of a particular protein. Basically it can be said, that each gene encodes a trait, e.g. color of eyes. Possible settings for a trait are called alleles. Each gene has its own identification position in the chromosome. This identification position is called as locus[9].

Complete set of all chromosomes is called genome. Particular set of genes in genome is called genotype. The genotype is with later development after birth base for the organism's phenotype[9].

Reproduction

During the time of reproduction, first occurs recombination (or crossover). Genes from parents then formed in some way to whole new chromosome. The newly created offspring can then be mutated. Mutation means, that the elements of DNA are a bit changed. These changes are mainly caused by errors in copying genes from parents[1-5]. The fitness of an organism is determined by the success of the organism in its life[6,9].

Search Space

If we are given to solve some problems, we are usually looking for some solutions, which will be the best feasible among others. The space of all feasible solutions (i.e. the objects among those of the desired solutions) is called search space (also the state space). Each feasible solution can be "marked" by its value or fitness value for the problem. We are looking for our solution, which is one point (or more) among feasible solutions – i.e. one point in the search space [9-14]. Searching for a solution is then equivalent to a looking for some optimum (minimum or maximum) in the search space. The search space may be completely known at the time of solving a problem, but usually we are given only a few points in it and we are generating other points as the process of finding solution[9,12].

The problem is that this kind of search can be very complicated. One does not know where to look for the solution, where to start and where to stop searching. There are many methods, how to find some suitable solution (i.e. not necessarily the best solution), for example hill climbing, tabu search, simulated annealing, genetic algorithm etc. The solution found by this method is often considered as a good solution, because it is not
often possible to prove which one is the real optimum[6,9,13].

**Nature of the Problems which GA deals with**

Examples of difficult problems, which cannot be solved in the "traditional" way, in polynomial time are either NP-Problems or Intractable problems [15-17]. NP stands for nondeterministic polynomial and it means that it is possible to guess the solution (by some nondeterministic algorithm) and then check it, both in polynomial time [15-17].

There are many problems for which we know fast (polynomial) algorithms. There are also some problems that are not possible to be solved algorithmically. For some problems it was proved that they are neither solvable in polynomial time deterministically nor they can decide the existence of its solution in polynomial time, they are NP-Heard problems. There are many problems, for which it is very difficult to find a solution, but once we have it, it is easy to check the solution in polynomial time. This fact led to NP-Complete problems. If we had a machine that can guess, we would be able to find a solution in some reasonable time[15-17].

Some of the well known NP problems are Satisfiability Problem, Traveling Salesman Problem, Knapsack Problem, etc[15-17].

Today for some problems nobody knows if some faster exact algorithm exists. Proving or disproving these remains as a big task for new researchers. Today many people think, that such a global optimum solution algorithm does not exist and so they are looking for some alternative methods (soft computing techniques) one of these methods are genetic algorithms[15,16].

**Outline of the Basic Genetic Algorithm**[9]

1. [Start] Generate random population of \( n \) chromosomes (suitable solutions for the problem)
2. [Fitness] Evaluate the fitness \( f(x) \) of each chromosome \( x \) in the population
3. [New population] Create a new population by repeating following steps until the new population is complete
4. [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
5. [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.
7. [Accepting] Place new offspring in a new population
8. [Replace] Use new generated population for a further run of algorithm
9. [Test] If the end condition is satisfied, stop, and return the best solution in current population
10. [Loop] Go to step 2

**Operators of GA**

The crossover and mutation are the most important part of the genetic algorithm. The performance is influenced mainly by these two operators.

**Encoding of a Chromosome**

The chromosomes contain information about solution which it represents. The most used way of encoding is a binary string [9,17]. The chromosome then could look like this:

<table>
<thead>
<tr>
<th>Chromosome 1</th>
<th>1101100100110110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome 2</td>
<td>1101110000111110</td>
</tr>
</tbody>
</table>

Each chromosome has one binary string. Each bit in this string can represent some characteristic of the solution.

Obviously there are many other ways of encoding. This depends mainly on the given problems. For example, one can encode directly integer or real numbers; sometimes it is useful to encode some permutations[6,9,15].

**Crossover**

After the decision of what encoding we will use, we can make a step to crossover. Crossover selects genes from parent chromosomes and creates a new offspring. The simplest way 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[6,9]. Crossover can then look like this [9]:

<table>
<thead>
<tr>
<th>Chromosome 1</th>
<th>1101100100110110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome 2</td>
<td>1101110000111110</td>
</tr>
<tr>
<td>Offspring 1</td>
<td>1101111000111110</td>
</tr>
<tr>
<td>Offspring 2</td>
<td>11011100100110110</td>
</tr>
</tbody>
</table>

There are other ways to make crossover, for example we can choose more crossover points. Crossover can be rather complicated and very much dependant on encoding of chromosome. Specific crossover made for a specific problem to improve in solution [9].

**Mutation**

After a crossover is performed, mutation takes place. This is done to prevent falling all solutions in population into a local optimum of the problem. Mutation changes randomly the newly created 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 [9,15]:

The mutation depends on the encoding as well as the crossover. For example when we are encoding permutations, mutation could be exchanging two genes[6,9,15].

Parameters of GA
Crossover and Mutation Probability
There are two basic parameters of GA, crossover probability and mutation probability. Crossover probability says how often will be crossover performed. If there is a crossover, offspring is created from parts of parent’s 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[9]. Crossover rate generally should be high, about 80%-95%. However some results show that for some problems crossover rate about 60% is the best[9]. Mutation probability says how often will be parts of chromosome mutated. 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 optima, but it should not occur very often, because then GA will in fact change to random search[1,6,9]. Mutation rate should be very low. Best rates reported are about 0.5%-1%.

Population Size
Another important parameter of GA is population size. Population size says how many chromosomes are in population (in one generation). If there are less number of chromosomes, GA have a less probability to perform crossover and only a small part of search space is explored. If there are a huge number of chromosomes present, GA slows down. Research shows that there must be an upper bound of generation of population (which depends mainly on encoding and the problem) to make solving process faster. The problem is again the same - looking for extreme of a function[9].

Selection
Chromosomes are selected from the generated population to be parents for next crossover. The problem is how to select these chromosomes. According to Darwin’s evolution theory the best ones should survive and create new offspring. There are many methods present to select such best chromosomes, for example roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection etc.[15,17].

Elitism
While creating new population by crossover and mutation there is a chance, that we will lose the best chromosome. Elitism is name of method, which first copies the best chromosome (or a few best chromosomes) to new population. The rest is done in classical way. Elitism can very rapidly increase performance of GA by capturing the best found solutions[17].

Encoding
Encoding of chromosomes is one of the problems. Encoding is essentially dependant on the problem. There are several encoding techniques present[9,15].

Binary Encoding
Binary encoding is the most common (mainly because of fast work) in GA. In binary encoding every chromosome is a string of bits, 0 or 1[9,15].

Permutation Encoding
Permutation encoding can be used in ordering problems (such as Traveling Salesman Problem). Even for this problem, to make the chromosomes consistent, some types of crossover and mutation corrections must be made. In permutation encoding, every chromosome is a string of numbers, which represents number in a sequence[9,17].

Value Encoding
In some problems where some complicated value, such as, real numbers, are used, direct value encoding are used. Use of binary encoding for this type of problems would be very difficult. In value encoding, every chromosome is a string of some values. Values can be anything connected to problem, form numbers, real numbers or chars to some complicated objects[9,15].
For this encoding it is often necessary to develop some new crossover and mutation specific for the problem[15].

**Tree Encoding**

Tree encoding is used mainly for evolving programs or expressions, for genetic programming. In tree encoding every chromosome is a tree of some objects, such as functions or commands in programming language[9].

Programming language LISP is often used to this, because programs in it are represented in this form and can be easily parsed as a tree, so the crossover and mutation can be done relatively easily[9].

**Applications of GA**

Genetic algorithms are used for difficult problems, for machine learning and also for evolving simple programs. They are also used for some art, for evolving pictures and music. Advantage of GA is in its parallelism. GA is traveling in a search space with more individuals. So they are less likely to get stuck into local optima like some other methods. The only disadvantage of GA is in their computational time. They can be slower than some other methods. Though, it is not a big problem for our present day super computer[9,15,17].

Here is a list of some important applications which GA deals with:

1. Nonlinear dynamical systems - predicting, data analysis
2. Designing neural networks, both architecture and weights
3. Robot trajectory
4. Evolving LISP programs (genetic programming)
5. Strategy planning
6. Finding shape of protein molecules
7. TSP and sequence scheduling
8. Functions for creating images

**References**


[369-372]