©Journal of the Technical University – Sofia
Plovdiv branch, Bulgaria
"Fundamental Sciences and Applications" Vol. 19, 2013
International Conference Engineering, Technologies and System
TECHSYS 2013
BULGARIA

ИЗСЛЕДВАНЕ ЕФЕКТИВНОСТТА НА ГЕНЕТИЧНИТЕ АЛГОРИТМИ ПРИ ОПТИМИЗИРАНЕ НА ТЪРСЕНЕТО НА НАЙ-ДОБРА КОМПЮТЪРНА КОНФИГУРАЦИЯ

ДИЛЯНА БУДАКОВА

Резюме: *Целта на тази статия е да се изследва ефективността на генетичен алгоритъм за оптимизиране избора на компютърни компоненти (CPU, HDD, Mother Board, RAM, Video Card and Mark) за конструиране на най-добра компютърна конфигурация. Предложен е модел, подходящ за симулиране на еволюция. Разработена е програмна система, която използва този модел. Проведените експерименти с разработената програмна система доказват ефективността от прилагането на генетичните алгоритми за решаването на задачи от този клас.*

Ключови думи: *генетични алгоритми, оптимизация, методи за търсене, машинно обучение, компютърни конфигурации*

INVESTIGATION OF THE EFFICIENCY OF GENETIC ALGORITHMS IN OPTIMIZING THE SEARCH FOR THE BEST QUALITY COMPUTER CONFIGURATION

DILYANA BUDAKOVA

Abstract: *The aim of this paper is to investigate the efficiency of using a genetic algorithm in optimizing the selection of computer components (CPU, HDD, Mother Board, RAM, Video Card and Mark) for constructing the best quality computer configuration. A model, appropriate for simulation of evolution has been proposed. A software system, based on this model, has been developed. The experiments, conducted with the developed software system, confirm the efficiency of implementation of genetic algorithms for solving problems of this order.*

Key words: *genetic algorithms, optimization, search methods, machine learning, computer configuration*

1. Introduction

Genetic algorithms (GAs), introduced by Holland in 1975 [4], are inspired by natural evolution and the magnum opus „The Origin of Species", published in 1859 by Charles Darwin. They are search methods based on the evolutionary concept of natural mutation and the survival of the fittest individuals. Given a well-defined search space they apply three different genetic search operations, namely, selection, crossover, and mutation, to transform an initial population of chromosomes, with the objective to improve their quality.

Experimental analysis illustrated that the GAs design constantly outperforms the greedy method in terms of solution quality.

GAs have been used for **problem-solving and for modeling**[9]. GAs are applied to many scientific, engineering problems, in business and entertainment, including [2,3,6,9]:

Optimization: numerical optimization, combinatorial optimization problems such as traveling salesman problem (**TSP)**.

Machine and robot learning: including classification and prediction, and protein structure prediction. GAs have also been used to design neural networks, to evolve rules for learning classifier systems or symbolic production systems, and to design and control robots.

Economic models. Immune system models. Ecological models: GAs have been used to model ecological phenomena such as biological arms races, host-parasite co-evolutions, symbiosis and resource flow in ecologies.

Models of social systems: GAs have been used to study evolutionary aspects of social systems, such as the evolution of cooperation [2], the evolution of communication, and trail-following behaviour in ants.

In this paper the efficiency of a genetic algorithm in optimizing the selection of computer components (CPU, HDD, Mother Board, RAM, Video Card and Mark) for constructing the best quality computer configuration is under investigation. A model, suitable for evolution simulation is proposed. A programming system, using this model, is developed. The experiments, conducted with the developed programming system the efficiency of genetic algorithms in solving problems of this order.

2. Examples of evolution simulation

In GAs the following specifications are adhered [7]: A chromosome is a representation in which: There is a list of elements called genes. The chromosome determines the overall fitness manifested by some mechanism that uses the chromosome's genes as a sort of blueprint.

Create a chromosome from a given list of elements – in this case the constructor might be called the genesis constructor. The multiplicity of candidate solutions, processed by the genetic algorithms at each step is called population. Mutate one or more genes in one or more of the current chromosomes, producing one new offspring for each chromosome mutated. Mate one or more pairs of chromosomes. Add the mutated and offspring chromosomes to the current population.

Create a new generation by keeping the best of the current population's chromosomes, along with other chromosomes selected randomly from the current population. Bias the random selection according to assessed fitness.

Here we have a number of exemplary models, used for evolution simulation in accordance with this terminology.

The model, developed for solving the traveling salesman problem **(TSP)** with the help of a genetic algorithm is of special interest. This is a typical optimization problem, aimed at finding a Hamilton cycle with minimum length at a given weighed complete graph G(V,E) with weight of the edges real (positive) numbers. It is NP-complete and in solving it by full running of a complete graph with n vertices, n! Hamilton cycles must be checked, which is unacceptable for big graphs with more than 50 vertices.

For solving this problem with GAs [1][9] an initial population with arbitrary Hamilton cycles – chromosomes, as well as with genes – the vertices of the graph under consideration, is built. They are stored as permutations of the numbers from 1 to n, according to the order in which the vertices are visited in any Hamilton cycle. Population development is guided by the objective function, by which only a number of the generated Hamilton cycles with the smallest length survive to the next generation, and those with greater length drop out. The genetic algorithm builds a Hamilton cycle with optimal or close to the optimal length for only a few steps, avoiding the necessity of considering all solutions [1][9].

In http://www.boxcar2d.com/index.html [8] a model is developed with the terms of GAs, aiming at evolution simulation in order to study a programming system for constructing high quality 2D cars.

Each car is a set of 8 randomly chosen vectors: direction and magnitude. All the vectors radiate from a central point (0,0) and are connected with triangles. For each wheel it randomly chooses a vertex to put the axle on, and picks an axle angle from 0 to 2 pi.

For the purposes of GAs the following model is developed [8]. Each car represents one chromosome and has 22 variables such as: vertex, axle angle, and radius, each represented as a real number (or integer) with varying ranges. For the selection process two algorithms are implemented: Roulette-Wheel Selection and Tournament Selection. The authors use two point crossover, which means two random points along the chromosome are selected and everything in between is swapped. In addition to the crossover, in each generation the chromosomes go through mutation. This means there is a probability that each aspect of the car (or variable in the chromosome) will change, as determined by the mutation rate slider set by the user. When a variable mutates, a new value is randomly chosen in the desired range.[8]

Another example is the model developed for solving the problem of minimizing network traffic for fast dissemination and access of information in large distributed systems, such as the Internet with GAs [5]. The decision of what to replicate where, requires solving a constraint optimization problem, which is NP-complete in general. The replication in large static distributed systems is considered and it is aimed at finding the appropriate allocation of the replicas in the distributed system so that the network traffic is minimized. The developed in [5] Genetic replication algorithm (GRA) and Adaptive GRA are a good example. In their model [5] a chromosome encoding a replication scheme is a bitstring consisting of M genes (one for each site). Every gene is composed of N bits (one for each object). A **1** value in the **k**th bit of the **i**th gene, denotes that **i**th site holds a replica of **k**th object, otherwise it is **0**. Using this encoding the total length of a chromosome is MN bits. [5] The main merits of using a genetic algorithm approach in the dynamic case lies in the proposed adaptive GA that uses existing knowledge about replica distribution in order to quickly define a new scheme.[5].

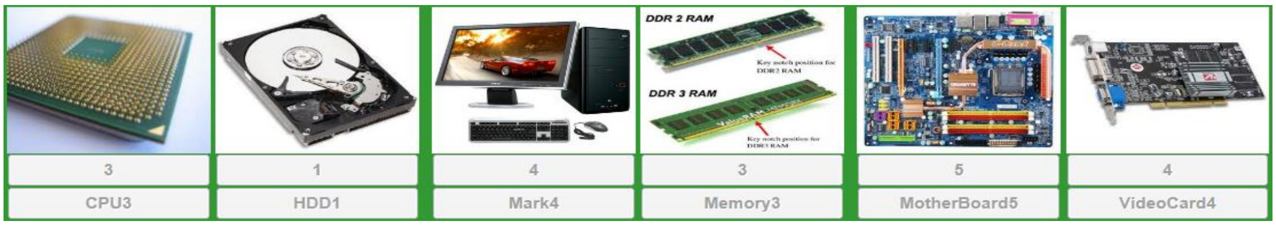

Fig. 1. *A chromosome in the computer configuration world consists of six numbers, which act as gene analogs. They determine quality(from 1 to 5) of CPU, HDD, Mark, Memory, Mother Board, and Video Card to use.*

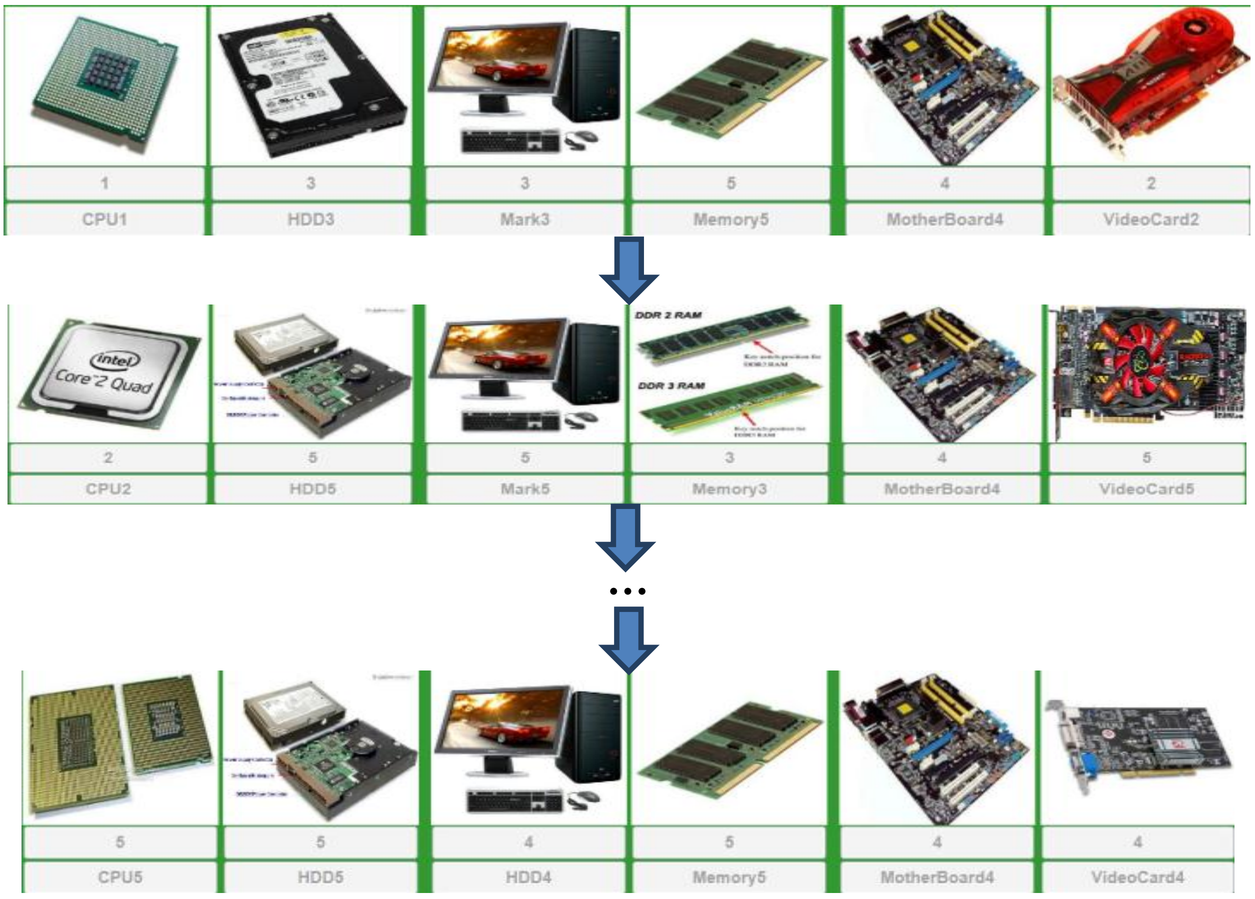

Fig.2 *A chromosome undergoing a series of mutations, each of which changes genes by adding or subtracting 1. The original chromosome is 1-3-3-5-4-2 chromosome which produces a computer configuration of quality 18. The final chromosome is a 5-5-4-5-4-4 chromosome, yielding a quality 27 computer configuration.*

3. Model and procedures constructing

The design of the chromosome (or individual) is probably the most important step in making a successful genetic algorithm.

In the computer configuration world each computer configuration is an "individual". The chromosome consists of six "genes" each of which is a data structure with a number, quality from 1 to 5, description and a picture (Fig.1) The genes are as it follows: CPU, HDD, Mother Board, RAM, Video Card and Mark. The quality of the computer configuration ranges from 5 to 30.

The fitness of a chromosome is the probability that the chromosome survives to the next generation. Accordingly a formula is required to relate the fitness of the **i**th chromosome, $\mathbf{f_i}$, a probability ranging from **0 to 1** to the quality of the corresponding computer configuration, $\mathbf{q_i}$**,** a number ranging from **5 to 30**. The following formula, in which the sum is over all candidates, is one possibility[7]:

$$f_i = \frac{q_i}{\sum_j q_j} \qquad (1)$$

An example of calculated quality and fitness of the individuals from one population by ten computer configurations is given in Table 1.

Table 1

The fitness of a ten chromosome population

N:	CHROMOSOMES	QUALITY	STANDARD FITNESS
1	1-3-3-5-4-2	18	0.08
2	2-5-5-3-4-5	24	0.11
3	5-5-4-5-4-4	27	0.13
4	3-4-5-1-1-1	15	0.07
5	3-3-3-2-4-5	20	0.09
6	4-5-2-2-3-4	20	0.09
7	5-5-3-2-5-5	25	0.12
8	5-2-3-4-4-5	23	0.11
9	1-1-4-4-5-4	19	0.09
10	4-3-2-5-5-5	24	0.11

Only half of the most adapted individuals survive to the next generation (computer configurations – CC). As it is seen from Table 1, these are the CCs with numbers 2,3,7,8, and 10. These are the CCs which will participate in mutation and crossover for obtaining new CCs, whose quality will be evaluated again etc. up to reaching the best quality CC.

Fig. 2 shows the process of mutation. A chromosome is undergoing a series of mutations, each of which changes genes by adding or subtracting 1. The original chromosome is 1-3-3-5-4-2 chromosome, which produces a quality 18 computer configuration. The final chromosome is a 5-5-4-5-4-4 chromosome, yielding a quality 27 computer configuration.

In Fig. 3 two chromosomes are undergoing crossover, each of which is cut in the 2,4, and 6 genes and reattached to the other chromosome. One of the original chromosomes is a 2-5-1-5-4-5 chromosome and the other is a 5-2-4-1-5-2 chromosome. One of the two new chromosomes is a 5-5-4-5-5-4 chromosome, which yields quality 28 computer configuration.

For the work of the GAs a number of basic functions of the programming system are developed in order to realize the following possibilities:

- function for random computer configuration creation
- function for selection of the 5 or 10 or 15 of the best generated computer configurations of the generating population.
- function for viewing the best generated computer configuration
- function for realization of the mutation of the selected genes. The step of mutation is +1 or -1.
- funciton for realization of the crossbreed of the genes of the chromosomes.
- function for realization of the crossbreed between 1-2-3 genes from one of the chromosomes and 4-5-6 genes from another chromosome.
- function for realization of the crossbreed between 2-4-6 genes from one of the chromosomes and 1-3-5 genes from another chromosome

The software system is implemented using Visual Studio .NET, ASP.NET - server- side Web application framework designed for Web development to produce dynamic Web pages; AJAX (Asynchronous JavaScript and XML) - a group of interrelated web development techniques used on the client-side to create asynchronous web applications and the programming language C#.

A number of experiments are conducted with the help of this system and the results from them will be presented in the next section.

3. The experimental results

Genetic algorithms generally involve many choices [7]:

How many chromosomes are to be in the population? If the number is too low, all chromosomes will soon have identical traits and crossover will do nothing; if the number is too high computation time will be unnecessarily excessive.

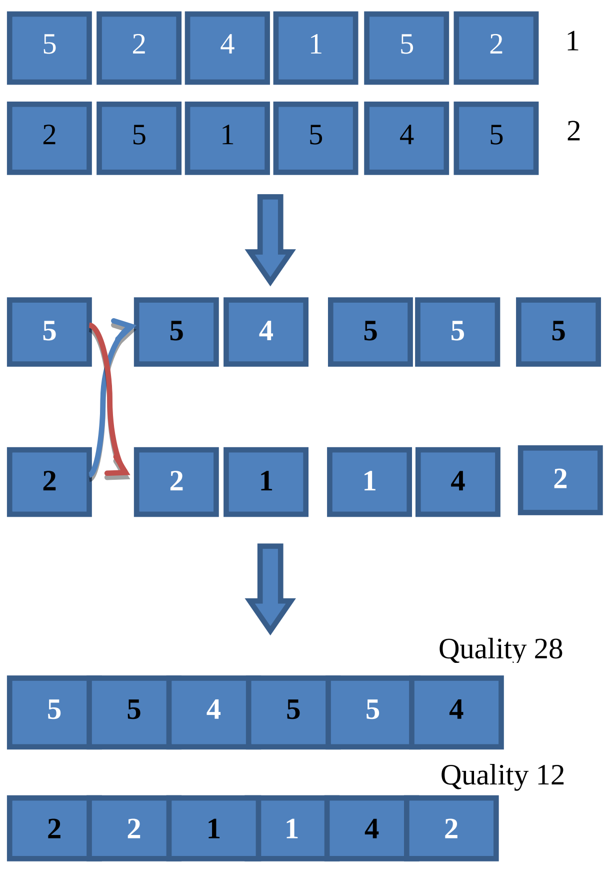

Fig. 3. *Two chromosomes undergoing crossover, each of which is cut in the 2,4, and 6 gene and reattached to the other chromosome. One of the original chromosomes is a 2-5-1-5-4-5 chromosome and the other is a 5-2-4-1-5-2 chromosome. One of the two new chromosomes is a 5-5-4-5-5-4 chromosome, which yields quality 28 computer configuration.*

What is the mutation rate? If the rate is too low, new traits will appear too slowly in the population; if the rate is too high, each generation will be unrelated to the previous generation.

Is mating allowed? If so, how are mating pairs selected, and how are crossover points determined?

Can any chromosome appear more than once in a population?

Based on these questions, a number of experiments are conducted to investigate the efficiency of the developed model:

- The search for the best quality computer configuration is realized by creating populations with different number of individuals, e.g. 10, 20 or 30 CC.
- The mechanisms of mutation and crossover are used in some of the experiments, while in others only mutation or only crossover is implemented. Then the procedures for quality assessment and natural selection are applied.
- When implementing crossover, experiments are conducted in which different positions for cutting the chromosomes are chosen, e.g. after the 2nd and the 4th gene or after the 1st, 3rd and 5th gene or after the 3rd one. It is investigated how the different choices influence finding the best solution.

The results show that, when crossover is used along with mutation, the best quality computer configuration is found much faster.

The small number of individuals in one generation and the use of only crossover (with no mutations) quickly leads to obtaining repeating individuals and cannot result in achieving the best solution.

The mutation leads to obtaining new genes, not seen before and correspondingly to appearance of new individuals with new quality. Therefore, the use of mutation is compulsory. For solving the problem in which it is impossible to achieve a generation with zero quality (0) and when there will be no surviving CCs to the next generation, the use of only mutation (with no crossover) and selection is sufficient for obtaining the best solution.

The efficiency of using GAs can be illustrated by calculating the total number of CCs, which can be built in solving the problem with full running,

In the model considered here 6 components are used for building a computer configuration, and copies from each component with quality 1, 2, 3, 4 and 5; then 15625 computer confirgurations can be made out of these components. These are all variations with repetition rV_n^k of n=5 elements of k=6 class

$$rV_n^k = 5^6 = 15625 \qquad (2)$$

variations with repetition.

In order to obtain the best quality computer configuration with the help of the genetic algorithm it is enough to build 50-100 computer configurations.

The genetic algorithm is efficient for building qualitative computer configurations because without building all 15625 computer configurations we manage to quickly build the best and approximately the most qualitative one.

4. Conclusion

The efficiency of using a genetic algorithm in optimizing the selection of computer components (CPU, HDD, Mother Board, RAM, Video Card and Mark) for building the best quality computer configuration is investigated in this paper.

A model is proposed, in which the computer components are analogues to the genes, and the computer configurations themselves are the analogue to the chromosomes. All genes can have quality from 1 to 5 and, consequently, the quality of the computer configurations can vary from 5 to 30.

A formula for quality assessment of each computer configuration is suggested.

Procedures, analogous to mutation, crossover and natural selection are developed. By means of them and based on the quality assessment, the process of evolution is modeled and the best solution is searched for.

The experiments, conducted with the developed software system confirm the efficiency and the benefits from implementing genetic algorithms for solving this type of problems.

The results from the conducted experiments show that the GAs are efficient in solving this type of problems and allow for avoiding a great number of poor quality computer configurations.

The results also show that when along with mutation crossover is also used, the best quality computer configuration is found much faster.

On the one hand, the small number of individuals in one generation and the use of only crossover (with no mutation) quickly lead to appearance of repeating individuals and cannot result in the best solution. On the other hand, the mutation leads to new genes, not seen before, and, consequently, to appearance of new individuals with new quality. Therefore, the use of mutation is required.

The efficiency of implementation of genetic algorithms is also confirmed by the fact that without being needed to go through full running (i.e., through building all 15645 computer configurations), it is sufficient to build only 5-6 generations of computer configurations with 10,20 or 30 individuals in order to find the best quality computer configuration.

References

1. Nakov,Pr., P. Dobrikov, Programming = ++Algorithms, TopTeam Co., ISBN: 954-8905-06-X, 2003,

2. Chughtai M., Determining Economic Equilibria using Genetic Algorithms, published by Imperial College – theses, September 1995

3. Goldstein M. Jonathan, Genetic Algorithm Simulation of the SHOP Scheduling Problem, published by An ICMS/Shell Oil Business Consultancy; Central Library of Imperial College (4 Management Thesis) – theses, September 1991

4. Holland J.H., Adaptation in natural and artificial systems, University of Michigan Press, Ann Arbor, MI, 1975.

5. Loukopoulos Thanasis, Ishfaq Ahmad, Static and adaptive distributed data replication using genetic algorithms, Journal of parallel and distributed computing; Elsevier, 0743-7315, 2004.

6. Schultz C. Alan, Learning Robot Behaviours using Genetic Algorithms , by. Navy Center for Applied Research in Artificial Intellignece. Central Library of Imperial, Obtained : web page - theses College, 2 Info. Desk MSc 1995.

7. Winston P., Artificial Intelligence, Addison-Wesley, 1992

8. http://www.boxcar2d.com/index.html

9. http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/tcw2/report.html#TSP

Department of Computer Systems and Technologies
Technical University–Sofia, Branch Plovdiv
25 Tsanko Dystabanov St.
4000 Plovdiv
BULGARIA
E-mail: dilyana_budakova@yahoo.com

Submitted on ………………

Reviewer ……………………………..