Genetic algorithm in cryptography

Ing. Martin Javurek[1], Ing. Michal Turčaník, PhD[2], doc. Ing. Marcel Harakal, PhD[2],

[1] ICT Department
Armed Forces Academy of gen. M. R. Štefánik
Liptovský Mikuláš, Slovakia
martin.javurek@aos.sk

[2] Department of Informatics
Armed Forces Academy of gen. M. R. Štefánik
Liptovský Mikuláš, Slovakia
michal.turcanik@aos.sk, marcel.harakal@aos.sk

Abstract. The genetic algorithm can be used in cryptography, mostly for break the cipher but it can be used for random number generator. Therefor this article shows an overview about genetic algorithm, where it shows some example of crossovers and mutations. Next present example of using genetic algorithm as a random number generator, then using genetic algorithm in cryptanalysis and for training artificial neural networks. At the end there are summary of advantages and disadvantages of genetic algorithm.

Keywords: GA, genetic algorithm, chromosome, Fitness function, Crossover, Mutation, Selection, TPM, ANN

1 Introduction

The genetic algorithm is a search algorithm based on the mechanics of natural selection and natural genetics. A population of individuals should behave like a natural system in order to adapt to some environment. Survival and reproduction of an individual is promoted by the elimination of useless features and by rewarding useful behavior. The genetic algorithm belongs to the family of evolutionary algorithms, along with genetic programming, evolution strategies, and evolutionary programming. Evolutionary algorithms can be considered as a broad class of stochastic optimization techniques. An evolutionary algorithm maintains a population of candidate solutions for the problem at hand. The population is then evolved by the iterative application of a set of stochastic operators. The set of operators usually consists of mutation, recombination, and selection or something very similar. Globally satisfactory, if sub-optimal, solutions to the problem are found in much the same way as populations in nature adapt to their surrounding environment.

There are some groups of problems for which a GA can be very useful. The application of a genetic algorithm (GA) in the field of cryptography is rather unique. Only few works exist on this topic. This nontraditional application is investigated to determine the benefits of applying a GA to a cryptanalytic problem, if any. If the GA-based approach proves successful, it could lead to faster, more automated cryptanalysis

techniques. However, since this area is so different from the application areas where GAs developed, the GA-based methods will likely prove to be less successful than the traditional methods [15].

2 Genetic algorithm

The Genetic algorithm is based on the Charles Robert Darwin's theory of evolution. It is adaptive heuristic exploration algorithm based on mechanics of the theory of natural selection and natural genetics [1], [2].

The main idea of genetic algorithm is to replicate the randomness from nature where population of individuals adapts to its surroundings over natural selection process and behavior of natural system. Survival and reproduction of an individual is promoted by the elimination of unwanted characteristics [2].

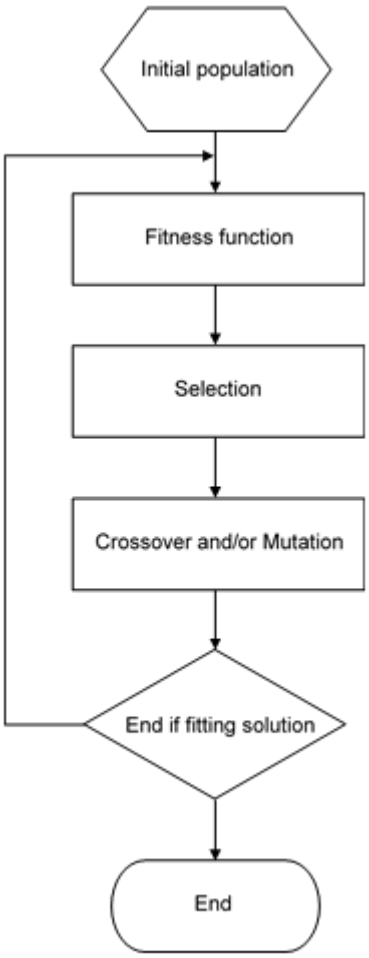


Fig. 1. Genetic algorithm

Genetic algorithm required to initialize a population of suitable size and suitably selected fitness function which is essential to achieve a suitable outcome. And subsequently used by three operators to transform a population (chromosomes) into the new population with better fitness:

1. Selection.
2. Crossover.
3. Mutation.

Initial population is also called chromosomes. Genetic algorithm needs initial population with some population size which remains constant from generation to generation. Determining the size of the population is a crucial factor. Too small population size increases risk of converging prematurely to local minima. Initial population can be determined randomly or by using some heuristic [2].

Fitness function is very important for guiding genetic algorithm. It can help to explore the search space more effectively and efficiently, but if the fitness function is bad, it can easily make genetic algorithm get trapped in a local optimum solution and lose the discovery power. Fitness function can be classified as constant fitness function and mutable fitness function [2].

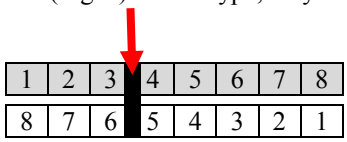
Selection is the stage of the genetic algorithm in which individual chromosomes are chosen from a population for reproduction. It is a quantitative criterion based on fitness value. The chromosome with higher fitness value will be considered better in order to implement proportionate random choice. Selection can be classified into different types [1], [2], [4]:

1. Roulette-wheel selection. If procedure is repeated until there are enough selected individuals.
2. Stochastic Universal Sampling. If instead of single pointer spun multiple times, there are multiple, equal pointers on a wheel that is spun once.
3. Tournament selection. It refers to repeatedly selecting the best individual of a randomly chosen subset.
4. Truncation selection. It involves taking the best half, third and another proportion of the individuals.
5. etc.

Crossover is a genetic operator which takes two chromosomes and new child is generated by taking some attributes of first chromosome and rest from second chromosome. Crossover can be classified into different types [1], [2] [3], [4]:

1. Single point crossover (Fig. 2). In this type, only one crossover point is chosen,

Parents:



Children:

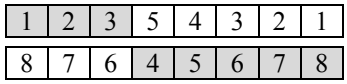
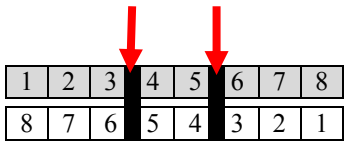


Fig. 2. Single point crossover

2. Two point crossover (Fig. 3). The crossover is improved by using two crossover points,

Parents:



Children:

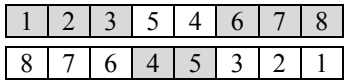
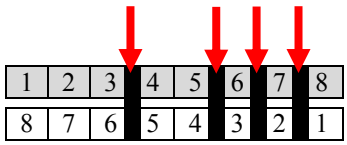


Fig. 3. Two point crossover

3. Multipoint crossover (Fig. 4). The crossover is improved by using more crossover points,

Parents:

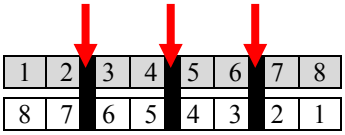


Children:

Fig. 4. Multipoint crossover

4. Uniform crossover (Fig. 5). There are bits of child uniformly taken from both the parents,

Parents:



Children:

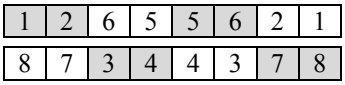
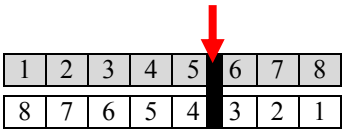


Fig. 5. Uniform crossover

5. Arithmetic crossover (Fig. 6). Arithmetic recombination between parents is used,

Parents:

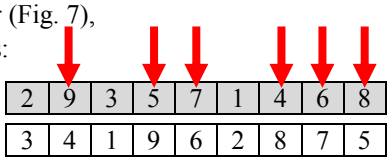


Children:

Fig. 6. Arithmetic crossover

6. Cycle crossover (Fig. 7),

Parents:



Children:

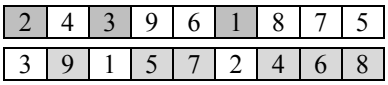
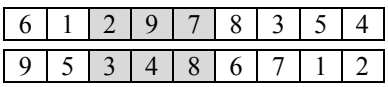


Fig. 7. Cycle crossover

7. Partially mapped crossover (Fig. 8),

Parents:



Children:

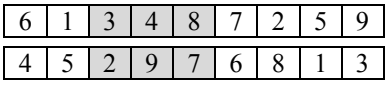


Fig. 8. Partially mapped crossover

8. Order crossover (Fig. 9),

Parents:

2	1	5	4	6	8	7	3	9
3	6	4	8	1	7	5	2	9

Children:

4	6	3	8	1	7	9	2	5
1	7	5	4	6	8	2	9	3

Fig. 9. Order crossover

9. Heuristic crossover, etc.

Mutation is a genetic operator which changes one or more bits in the chromosome. It is used to maintain genetic diversity from one generation to the next. It is similar to biological mutation. It is performed on the child after crossover. Mutation allows the algorithm to avoid local minima by preventing the population chromosomes from becoming too similar to each other [1], [2]. Mutation can be classified into different types [2], [4]:

1. Flipping of bits (Fig. 10). It involves selecting one or more bits of chromosome and inverting it,

Before mutation:

0	1	1	0	0	1	0	0

After mutation:

0	1	1	0	0	0	0	0

Fig. 10. Flipping of Bits

2. Swap mutation (Fig. 11). It randomly picks two bits in string and swaps their values,

Before mutation:

1	2	3	4	5	6	7	8

After mutation:

1	6	3	4	5	2	7	8

Fig. 21. Swap mutation

3. Insert mutation (Fig. 12). It picks two bits and moving one so that it is next to the other,

Before mutation:

1	2	3	4	5	6	7	8

After mutation:

1	2	6	3	4	5	7	8

Fig. 32. Insert mutation

4. Scramble mutation (Fig. 13). Here the entire string or some randomly subset of values are chosen and their positions are scrambled.

Before mutation:

1	2	3	4	5	6	7	8

After mutation:

1	3	5	4	2	6	7	8

Fig. 43. Scramble mutation

5. Inversion mutation (Fig. 14). Here randomly selecting two positions in the string and reversing the order.

Before mutation:

1	2	3	4	5	6	7	8

After mutation:

1	5	4	3	2	6	7	8

Fig. 54. Inversion mutation

6. Boundary mutation. It involves randomly replacing chromosome with either lower or upper bound,
7. Non-Uniform mutation. It is used to increase the probability that amount of mutation will go to 0 with the next generation,
8. Uniform mutation (Fig. 15). A chosen chromosome cell is replaced with a uniform random value whose range is selected,

Before mutation:

0	1	1	0	0	1	0	0	0

After mutation:

0	1	0	0	0	0	0	0	1

Fig. 65. Uniform mutation

9. Gaussian mutation. It involves adding a unit Gaussian random value to a chromosome cell, etc.

We do not need use both operators but there is a question what is better crossover or mutation? There are some pluses and minuses which can help us to decide [4]:

1. Only crossover can combine information from two parents.
2. Only mutation can introduce new information.
3. Crossover does not change the frequencies of the population.

According to these advantages and disadvantages, the easiest answer is to use both operators.

Selection is the stage of the genetic algorithm in which individual chromosomes are chosen from a population for reproduction. It is a quantitative criterion based on fitness

value. The chromosome with higher fitness value will be considered better in order to implement proportionate random choice. Selection can be classified into different types [1], [2], [4]:

6. Roulette-wheel selection. If procedure is repeated until there are enough selected individuals.
7. Stochastic Universal Sampling. If instead of single pointer spun multiple times, there are multiple, equal pointers on a wheel that is spun once.
8. Tournament selection. It refers to repeatedly selecting the best individual of a randomly chosen subset.
9. Truncation selection. It involves taking the best half, third and another proportion of the individuals.
10. etc.

3 Genetic algorithm as random number generator

Genetic algorithm in cryptography can be used for generating the key. Key generation in cryptography is the most important part of encoding data. If the key is randomly chosen and non-repeating used than this cypher is called one time pad (or one time system). The one time pad is theoretically unbreakable [1], [5]. The one of the most used one time pad is in Vernam cipher. Vernam cipher is a stream cipher where plaintext is converted into cipher text by using XOR operation between plaintext and the key [1], [5].

One of the possible method of generating the key is described in the work [5]. It consists of:

1. Generating binary population. For this step it can be used pseudo random number generator.
2. Selection. Where the two parents will be chosen for reproduction.
3. Crossover. From parents by using crossover operator we gain child.
4. Mutation. After crossover we applied mutation operator.
5. Fitness function. If value from fitness function is satisfactory random chromosome is selected as the key else process is repeated.

Example from work [5]:

Original population is 5x5 and the cells are generated randomly (Fig. 16):

1	0	0	1	0
1	0	1	0	1
1	1	1	0	1
0	0	1	0	1
0	1	0	1	1

Fig. 76. Original population 5x5

The population is read vertically and new array would be (Fig. 17):

1	1	1	0	0
0	0	1	0	1
0	1	1	1	0
1	0	0	0	1
0	1	1	1	1

Fig. 87. The population is read vertically

Any two random numbers are generated from 1 to 5, for example 2nd and 4th chromosome (Fig. 18):

0	0	1	0	1
1	0	0	0	1

Fig. 98. Two random chromosome

One point crossover is performed and the chromosome becomes (Fig. 19):

0	0	0	0	1

Fig. 109. Chromosome after one point crossover

Then mutation is done by selecting a random chromosome and flipping from amongst those chromosomes (Fig. 20):

1	1	1	1	0

Fig. 20. Chromosome after mutation

The Number of crossover is to be performed is given by the formula (1):

$$\boldsymbol{NC = \frac{Cell * Ch * Cr}{100}} \quad \textbf{(1)}$$

Where ***NC*** is Number of crossovers, ***Cell*** is Number of cells in each chromosome, ***Ch*** is Numbers of chromosomes and ***Cr*** is Crossover Rate.
The Number of mutation is to be performed is given by the formula (2):

$$\boldsymbol{NM = \frac{Cell * Ch * Mr}{100}} \quad \textbf{(2)}$$

Where ***NM*** is Number of mutations, ***Cell*** is Number of cells in each chromosome, ***Ch*** is Numbers of chromosomes and ***Mr*** is Mutation Rate.

Once the task is accomplished, then Coefficient of correlation is calculated by taking ***k=1, 2*** and ***3***. The coefficient of autocorrelation is defined as follows. Given measurements $\boldsymbol{Y_1, Y_2, \ldots, Y_N}$ at time $\boldsymbol{X_1, X_2, \ldots, X_N}$, the lag ***k*** autocorrelation function is defined as formula (3):

$$r_k = \frac{\sum_{i=1}^{N-k}(Y_i - \overline{Y})(Y_{i+k} - \overline{Y})}{\sum_{i=1}^{N}(Y_i - \overline{Y})^2} \quad (3)$$

If the coefficient of Correlation is satisfactory, then random chromosome is selected which is taken as the key otherwise, the process is repeated.

That was only example but there are a lot of procedures with changes in generating initial population, fitness function and chose type of crossover, mutation and selection operator. For example [2], [6]. In work by "Swati Mishra, Siddharth Bali, Public key cryptography using genetic algorithm" [2], they used ring crossover and for fitness function used Shannon Entropy, Chi Square and Coefficient of Autocorrelation. There are a lot of possibilities to create own genetic algorithm for generate random key.

4 Genetic algorithm in cryptanalysis

One of the most used genetic algorithm is in cryptanalysis. Cryptanalysis is the process of recovering the plaintext or key from a cipher [7]. Genetic algorithm can be used in genetic attack in neural cryptography and often in attacks on the transposition cipher. In breaking transposition cipher are used four main attack models where difficulty of a successful attack relates to the quantity of information that attacker has. Those models are sorted by the amount of information that the attacker has [8]:

1. Ciphertext only.
2. Known plaintext.
3. Chosen plaintext.
4. Chosen ciphertext.

4.1 Know Plaintext Attack

The main idea in this model is that attacker knows a sample plaintext for a corresponding encrypted text. Attacker now needs to reconstruct key, knowing encryption function, decryption function, substring of the plaintext and substring of the ciphertext, in order to read all encrypted messages. Given the complexity of encryption and decryption function it is not a trivial problem [8].

4.2 Chosen Plaintext Attack

There attacker know substring of the plaintext, chosen him and get access to encryption that uses the key, without knowing key, thus generating the corresponding substring of the ciphertext. . Attacker now needs to reconstruct key, knowing encryption function, decryption function, substring of the plaintext and substring of the ciphertext, in order to read all future encrypted messages. And again, encryption and decryption function are very complex, however due to the ability to select substring of

the plaintext have useful statistical or mathematical properties. In the Chosen Plaintext Attack, attacker may use the useful properties of substring of the plaintext in order to implement the attack, as is the case with differential cryptanalysis. The difference is input as a parameter to the system and the system automatically generates a plaintext of that difference [8].

4.3 Brute Force Attack

All technique for breaking a cipher must be compared to a Brute Force Attack. The main idea in the Brute Force Attack is that all possible keys are trying to break the cipher. Practically every cipher is vulnerable to this attack type on the key, but the time for these attacks to be successful is unacceptable. Average at least half of the key space must be tried before the key is found. The goal of security through cryptography is not to keep a message secure forever, but to make the cost of breaking the cipher in terms of time so large that the data is worthless when found, or so costly in terms of computing resources that the gains from the intelligence are less than the cost expended for breaking it. This principle is known as a cipher being computationally secure [8].

4.4 Genetic algorithm for breaking transposition cipher

So Brute Force attack has high computational complexity. In order to overcome this complexity, the Meta heuristic search techniques like Genetic Algorithm are used [7]. It consists of Initialize algorithm variables [7], [9]:

1. The maximum numbers of generations to consider.
2. The solution pool size.
3. And any other problem dependent variables.
4. Generate an initial solution pool containing candidate solutions.
5. Using the current pool for number of generation iterations.

Then the genetic algorithm continues by selecting a pool from the current solution and pair parents. For each pair of parents generate a pair of child using suitable crossover function and apply a mutation operator. Then evaluate the fitness function for each of the children and on the base on the fitness of each of the children and the fitness of each of solutions in the pool, decide which solution will be placed in the new solution pool. Next replace the current solution pool with the new one. And at the end, choose the fittest solution of the final generation as the best solution. The technique to compare candidate key is next used in genetic attack. The main idea in this technique is to compare n-gram statistic of the decrypted message with those of the language [7]. General formula (4) is used to determine the suitability of a proposed key [7].

$$\boldsymbol{C^k = \beta . \sum_{i,j \in A} \left| K^b_{(i,j)} - D^b_{(i,j)} \right| + \gamma . \sum_{i,j,k \in A} \left| K^t_{(i,j,k)} - D^t_{(i,j,k)} \right|} \quad \textbf{(4)}$$

Where A is the using language alphabet, K is known language statistics, D is decrypted message statistic, b is bigram statistics, t is trigram statistics. And the value of β and γ allows assigning of different weights to each of the two n-gram types.

The complexity of determining the fitness is $O(N^3)$ (where N is the alphabet size). To calculate the cost of associated with the transposition cipher key the proposed key is used to decrypt the ciphertext and then the statistics of the decrypted message are then compared with statistics of the language. Improvement of this method is in used only subset of the most common bigrams and trigrams instead of using all possible bigrams and trigrams [7].

4.5 Genetic algorithm for attack on neural cryptography

Safety of the neural exchange protocol is based on the fact that two ANN (A, B) communicating with each other are synchronized faster than the third network (E) (attacker`s) which is trained only to capture the input and outputs from the synchronizing ANN from the public channel. The attacker does not know the topology of the ANN and what output values are in the each hidden neuron so that the most of attacks are based only on estimate status of hidden neurons. There are four basic attacks: Simply attack, Geometric attack, Majority attack and Genetic attack.

In the genetic attack, the attacker starts with only one TPM but is permitted to use M TPMs. Because the most challenging issue in the mutual learning process is to predict the internal representation of either TPM^A or TPM^B, the genetic attack directly deals with this difficulty. For a specific value of $o^{A/B}$, there are 2^{K-1} different internal representations to reproduce this value. The genetic attack handles all these possibilities in a parallel fashion. The genetic attack proceeds as follows [10], [11]:

- If $o^A = o^B$ and E has at most $M/2^{K-1}$ TPMs, 2^{K-1} TPMs are generated and each updates its weights based on one of the possible internal representations. This step is known in genetic algorithms as the mutation step.
- If E has more than $M/2^{K-1}$ TPMs, the mutation step will be essentially an overhead due to the exponential storage needed. Therefore, the attacker must discard some of the TPMs to avoid exponential storage increase. As a genetic algorithm, the discarding procedure is based on removing the TPMs with the least fittest function. The algorithm uses two variables U and V as the fitting functions. The variable U represents the number of correct prediction of o^A in the last V training steps.

And if $o^A \neq o^B$ the attacker`s networks remain unchanged, because A and B do not update the weights in their tree parity machines.

5 Genetic algorithm for training

Next genetic algorithm can be used for training or design artificial neural network. It has some advantages comparison with classical learning methods. For example, backpropagation has some disadvantages.

First is the scaling problem. Backpropagation works well on the simple training problems. However as the problem complexity increases the performance of backpropagation falls off rapidly. Second drawback is to compute a gradient requires differentiability. Therefore backpropagation cannot handle discontinuous optimally criteria or discontinuous node transfer functions. This precludes its use on some common node types and simple optimality criteria [12].

Genetic algorithms should not have problem with scaling as backpropagation. One reason for this is that they generally improve the current best candidate monotonically. They do this by keeping the current best individual as part of their population while they search for better candidates. Secondly, genetic algorithms are generally not bothered by local minima. The mutation and crossover operators can step from a valley across a hill to an even lower valley with no more difficulty than descending directly into a valley [12].

Genetic algorithm is algorithm for optimization and learning based loosely on several features of biological evolution. It requires five components [12]:

1. A way of encoding solutions to the problem on chromosomes. The weights (and biases) in the neural network are encoded as a list of real numbers.
2. An evaluation function that returns a rating for each chromosome given to it. Assign the weights on the chromosome to the links in a network of a given architecture, run the network over the training set of examples, and return the sum of the squares of the errors.
3. A way of initializing the population of chromosomes. The weights of the initial members of the population are chosen at random with a probability distribution. This is different from the initial probability distribution of the weights usually used in backpropagation, which is uniform distribution between -1 and 1.
4. Operators that may be applied to parents when they reproduce to alter their genetic composition. Included might be mutation, crossover (i.e. recombination of genetic material) and domain-specific operators.
5. Parameter settings for the algorithm, the operators and so forth. There are a number of parameters whose values can greatly influence the performance of the algorithm.

Given these five components a genetic algorithm operates according to the following steps [12]:

1. The population is initialized. The result of the initialization is a set of chromosomes.

2. Each member of the population is evaluated. Evaluations may be normalized and important thing is to preserve relative ranking of evaluations.
3. The population undergoes reproduction until a stopping criterion is met. Reproduction consist of a number of iterations. One or more parents are chosen to reproduce. Selection is stochastic, but the parents with the highest evaluations are favored in the selection. Then the operators are applied to the parents to produce children. And in the end, the children are evaluated and inserted into the population. In some version of the genetic algorithm, the entire population is replaced in each cycle of reproduction. In others, only subsets of the population are replaced.

6 Advantages and disadvantages of genetic algorithm

There are a lot of advantages and of course disadvantages for using genetic algorithm. The main pros are efficiently search the whole model space, so finding global minima. There is no needed linearization of the problem and no needed partial derivatives. So it needs only finite knowledge of the physical system. Genetic algorithm can help avoid a danger of trapping in local maximum or minimum and can be applied in wide variety of optimization problems [13], [14].

And most important cons are: there is no guaranty to find best solution because sometimes there is a problem find the exact global optimum. Next there is long time to evaluate the individuals, because genetic algorithm requires large number of fitness function evaluations [13], [14].

7 Conclusion

The genetic algorithm can be very helpful for solution of some groups of problems. One of them is application in the cryptography. They are usually used in cryptanalysis, but they can be used for the random number generators or for training and design artificial neural networks. The random number generator is very important part of any cryptographic system and artificial neural network is used in cryptography too. The main reason for using genetic algorithm in cryptanalysis is for their efficiently search the whole model space, so finding global minima. There is no needed linearization of the problem and no needed partial derivatives. Next these property increase suitability for training or design artificial neural networks.

References

1. Goyat, S.: Cryptography Using Genetic Algorithms (GAs). In IOSR Journal of Computer Engineering (IOSRJCE), 1(5), pp. 06-08Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195--197 (2012)
2. Mishra, S., Bali, S.: Public key cryptography using genetic algorithm. In International Journal of Recent Technology and Engineering, 2(2), pp. 150-154. (2013)
3. Leung, F. H., Lam, H. K., Ling, S. H., Tam, P. K.: Tuning of the structure and parameters of a neural network using an improved genetic algorithm. In Neural Networks, IEEE Transactions on, 14(1), pp. 79-88. (2003)
4. Eiben, A. E., Smith, J. E.: Introduction to evolutionary computing. Springer-Verlag Berlin Heidelberg, 2003. 300 p. ISBN 9783662050941 (2003)
5. Khan, F. U., Bhatia, S.: A novel approach to genetic algorithm based cryptography. In International Journal of Research in Computer Science, 2(3), pp. 7-10. (2012)
6. Soni, A., Agrawal, S.: Key Generation Using Genetic Algorithm for Image Encryption. In International Journal of Computer Science and Mobile Computing IJCSMC, 2(6). (2013)
7. Toemeh, R., Arumugam, S.: Breaking Transposition Cipher with Genetic Algorithm. In Elektronika ir Elektrotechnika, 79(7), pp. 75-78. (2015)
8. Brown, J. A., Houghten, S., Ombuki-Berman, B.: Genetic algorithm cryptanalysis of a substitution permutation network. In Computational Intelligence in Cyber Security, 2009. CICS'09. IEEE Symposium on. pp. 115-121. (2009)
9. Toemeh, R., Arumugam, S.: Applying Genetic Algorithms for Searching Key-Space of Polyalphabetic Substitution Ciphers. In Int. Arab J. Inf. Technol., 5(1), pp. 87-91. (2008)
10. Ruttor, A., Kinzel, W., Naeh, R., & Kanter, I.: Genetic attack on neural cryptography. In Physical Review E, 73(3), 036121. (2006)
11. Prabakan, N., Vivekanandan, P.: A New Security on Neural Cryptography with Queries. In Int. J. of Advanced Networking and Applications Volume: 02, Issue: 01, pp. 437-444 (2010)
12. Montana D. J., Davis L.: Training Feedforward Neural Networks Using Genetic Algorithms. In Proceedings of the International Joint Conference on Artificial Intelligence, San Francisco, Vol. 89, pp. 762-767. (1989)
13. Uzel, Ö., Koc, E.: Basics of Genetic Programming (Doctoral dissertation, Dissertaçao (Mestrado) - Cankaya University). (2012)
14. Sumathi, S., Hamsapriya, T., Surekha, P.: Evolutionary Intelligence: An Introduction to Theory and Applications with Matlab. 1. edition,. Springer, 2008. ISBN: 9783540751588 (2008)
15. Delman, B.: Genetic algorithms in cryptography. Thesis. Rochester Institute of Technology. Accessed from http://scholarworks.rit.edu/theses/5456 (2004)