I J C T A, 9(16), 2016, pp. 8295-8302

An Improved Cryptanalysis Technique Based on Tabu Search for Knapsack Cryptosystem

Tamoghna Mandal* and Malay Kule**

ABSTRACT

With the exponential increasing of networked system & networked application, the demand of effective network security is increasing day by day. Cryptology is the science and study of systems for secret communication. Cryptology is the combination of two complementary fields of study: Cryptography and Cryptanalysis. In this paper, we propose a cryptanalysis method based on Tabu Search to break knapsack cipher. We will also compare and analyze the performance of proposed algorithm on a knapsack cipher with that of Genetic Algorithm. The experimental results shows that our proposed algorithm using Tabu Search is better than that of genetic algorithm based method.

Keywords: Cryptology, Cryptography, Cryptanalysis, Genetic Algorithm, Tabu Search, Knapsack Cipher.

1. INTRODUCTION

The demand for effective internet security is increasing exponentially day by day. Businesses have an obligation to protect sensitive data from loss or theft. Such sensitive data can be potentially damaging if it is altered, destroyed, or if it falls into the wrong hands. So they need to develop a scheme that guarantees to protect the information from the attacker. Cryptology is at the heart of providing such guarantee. The basic aim of Cryptography is to allow the intended recipients of a message to receive the message properly while preventing eavesdroppers from understanding the message. Cryptology is the science of building and analyzing different encryption and decryption methods. Cryptology consists of two subfields; Cryptography [6, 7] and Cryptanalysis [9]. Cryptography is the science of building new powerful and efficient encryption and decryption methods. Cryptanalysis is the science and study of method of breaking cryptographic techniques i.e. ciphers. In other words it can be described as the process of searching for flaws or oversights in the design of ciphers. In our paper we have performed cryptanalysis of Knapsack cipher using tabu search [13]. The results are compared with that of using genetic algorithm [5]. The rest of this paper is organized as follows: Section 2 consists of related work, Section 3, 3.1 and 3.2 deals with the overview of Knapsack cipher, genetic algorithm and tabu search respectively, section 3.3 represents the proposed algorithm, section 4 gives the experimental results with their analysis and Section 5 is the conclusion of the paper.

2. RELATED WORK

The cryptanalysis of Knapsack Cipher has many methods for decryption of cipher data. Cryptanalysis of knapsack ciphers using genetic algorithms in paper [4] considers the genetic algorithm simply as a possibility for the cryptanalysis of knapsack ciphers. The paper introduces genetic algorithms using binary chromosomes. The applicable process of selection, mating and mutation are described. The representation used here is of binary chromosomes, which represent the summation terms of the knapsack, i.e., a1 in the chromosomes

* Indian Institute of Engineering Science and Technology, Shibpur, Howrah-711103, India, *Email: tamoghna.iiests@gmail.com*

** Indian Institute of Engineering Science and Technology, Shibpur, Howrah-711103, India, *Email: malay@cs.iiests.ac.in*

indicates that the term should be included when calculating the knapsack sum. The evaluation function is determined once a representation scheme is selected. Here the function measures how close a given sum of terms is to the target sum of the knapsack. Function range between 0 and 1, with 1 indicating an exact match. Chromosomes that produce a sum greater than the target should have a lower fitness, in general, than those that produce a sum less than the target.

An Enhanced Cryptanalytic Attack on Knapsack Cipher Using Genetic Algorithm in paper [3] represents how genetic algorithm can be applied to Markel-Hellman Knapsack cipher. This is a meta-heuristic approach for the cryptanalysis of Knapsack cipher. The genetic algorithm used involves following steps like encoding, initialization, evaluation, selection, crossover and mutation. During encoding each character is converted into ASCII code. During initialization a random population of chromosomes is generated. During evaluation a fitness function is used to calculate the fitness of each chromosome. Then the best fit chromosomes are selected using any selection algorithm (rank selection, roulette wheel). Then crossover and mutation is applied on the newly selected population. This process is continued until the value of the fitness calculated is 1. When this happens for a particular chromosome this is the required ASCII code of the plain text. In order to improve the efficiency of genetic algorithm attack on knapsack cipher, the previously published attack was enhanced and reimplemented with variation of initial assumptions and results are compared with Spillman's [4] results. Supravo Palit et al [1] used binaryfirefly algorithm in cryptanalysis of Knapsack cipher. Saptarshi Neil Sinha et al [2] proposed a cryptanalytic attack on Knapsack cipher using differential evolution algorithm.

In this work we are trying to get improved methods which will take minimum iteration for finding desired output. Here we are using two methods Genetic Algorithm and Tabu Search methods for the same knapsack cipher.

3. KNAPSACK CIPHER

One of first knapsack cipher [2, 3] was proposed by Markle and Hellman in 1975 which utilized a NP-complete problem for its security. The knapsack problem is formulated as follows. Let us assume the values M1, M2……..,Mn and the sum S are given. Let it be necessary to compute values b1, b2,……,bn values, so that S = M1b1 + M2b2…… + Mnbn . The values of coefficient bi can be equal 0 or 1. The 1 value shows that object will fit into the knapsack, 0 values will not in the knapsack. The Markle-Hellman knapsack cipher encrypts a message as a knapsack problem. The plaintext block transforms into binary string (the length of block is equal number of elements in knapsack sequence). One value determines that an element will be in target sum. This sum is the ciphertext. Table 1 shows an example of solving the knapsack problem for the entry numbers sequence: 1, 3, 6, 13, 27 and 52.

The public/private key aspect of this approach lies in the fact that there are actually two different knapsack problems – referred to as the easy Knapsack and hard knapsack. The Markle-Hellman [2] algorithm is based on this property. The private key is a sequence of number for a superincreasing knapsack problem. The public key is a sequence of number for a normal knapsack problem with the same solution. Easy knapsacks have a sequence of numbers that are superincreasing - that is, each number is greater than the sum of previous numbers:

Table 1
Example of Knapsack Encryption

Plain text	Knapsack Sequence	Cipher text
1 1 1 0 0 1	1 3 6 13 27 52	1+3+6+52= 62
0 1 0 1 1 0	1 3 6 13 27 52	3+13+27= 43
0 0 1 1 0 1	1 3 6 13 27 52	6+13+52= 71

$$a_i > \sum_{j=1}^{i-1} \left(a_j \right) \text{ for i = 2, 3, ……, n}$$

(where a_i is i-th element of the sequence). For example {1, 3, 6, 13, 27, 52} is a superincreasing sequence but {1, 3, 4, 9, 15, 25} is not. The knapsack solution with the superincreasing sequence proceeds as follows. The target sum is compared with a greatest number in the sequence. If the target sum is smaller, than this number, the knapsack will not fill, otherwise it will. Then the smaller element is subtracted from the target sum, and the result of the subtraction, is compared with next element. Such operation is done until the smallest number of sequence is reached. If the target sum is reduced to 0 value, then solution exists. In other case solution doesn't exist. For example, consider a total knapsack target sum is 70 and the sequence of weights of {2, 3, 6, 13, 27, and 52}. The largest weight, 52, is less than 70, so 52 are in the knapsack, Subtracting 52 from 70 leaves 18. The next number 27 is greater than 18, so 27 is not in the knapsack. The next weight 13 is less than 18, so 13 is in the knapsack. Subtracting 13 from 18 leaves 5. The next weight, 6, is greater than 5, so 6 are not in the knapsack. Continuing this process will show that both 2 and 3 are in the knapsack and the total weight is brought to 0, which indicates that a solution has been found. The plaintext that resulted from a ciphertext value of 70 would be 110101. The superincreasing knapsack is easy to decode, which means that it does not protect the data. Anyone can recover the bit pattern from the target sum for a superincreasing knapsack if the elements of the superincreasing knapsack are known.

Merkle and Hellman suggested that such a simple knapsack be converted into a trapdoor knapsack which is difficult to break. The algorithm [2, 3] work as follows:

1. Select a simple knapsack sequence. Elements make a superincreasing sequence

$$A' = (a_1' + a_2' + \ldots\ldots\ldots\ldots + a_n')$$

2. Select an integer value m greater than sum of all elements of superincreasing sequence.
3. Select another inter w that the gcd(m, w) = 1, that is number m and w are reciprocally prime.
4. Find the inverse of the w mod m = ω^{-1}
5. Construct the hard knapsack sequence A = wA′ mod m i.e. $a_i = w' a_i \bmod m$

The trapdoor sequence A could be published as a public key (encryption key). The private (secret) key for this cipher consists of a simple knapsack sequence A′, so-called trapdoor, values m, w, w-1.

The encoding is done as follows. The message divides into n bits block (each block contains as many element as simple knapsack sequence). Values in the message block shows that the element will be in the target sum. The target sum of each block is ciphertext. The decoding consists of the following. Each number of the cipher text is multiplied through ω^{-1} mod m and the result of this operation is the plaintext.

3.1. Genetic Algorithm

In a genetic algorithm [5], a population of strings (called chromosomes or the genotype of the genome), which encode candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem, evolves toward bettersolutions. Traditionally, solutions are represented in binary as strings of 0s and 1s , but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

A typical genetic algorithm requires a genetic representation of the solution domain and a fitness function to evaluate the solution domain.In paper [3] and [4] the authors used genetic algorithm for cryptanalysis of knapsack cipher.

3.2. Tabu Search

The basic concept of Tabu Search [13] as described by Glover (1993) is a meta-heuristic superimposed on another heuristic. The overall approach is to avoid entrainment in cycles by forbidding or penalizing moves which take the solution, in the next iteration, to points in the solution space previously visited (hence 'tabu'). The Tabu method was partly motivated by the observation that human behavior appears to operate with a random element that leads to inconsistent behavior given similar circumstances. As Glover points out, the resulting tendency to deviate from a charted course might be regretted as a source of error but can also prove to be source of gain. The Tabu method operates in this way with the exception that new courses are not chosen randomly. Instead the Tabu search proceeds according to the supposition that there is no point in accepting a new (poor) solution unless it is to avoid a path already investigated. This ensures new regions of a problems solution space will be investigated in with the goal of avoiding local minima and ultimately finding the desired solution.

The Tabu Search begins by marching to local minima. To avoid retracing the steps used the method records recent moves in one or more Tabu lists. The original intent of the list was not to prevent a previous move from being repeated, but rather to ensure it was not reversed. The Tabu lists are historical in nature and form the Tabu search memory. The role of the memory can change as the algorithm proceeds. At initialization the goal is make a coarse examination of the solution space, known as 'diversification', but as candidate locations are identified the search is more focused to produce local optimal solutions in a process of 'intensification'.

As we just mentioned, TS is an extension of classical LS methods. In fact, basic TS can be seen as simply the combination of LS with short-term memories. It follows that the two first basic elements of any TS heuristic are the definition of its search space and its neighborhood structure.

The search space of an LS or TS heuristic is simply the space of all possible solutions that can be considered (visited) during the search. It is also important to note that it is not always a good idea to restrict the search space to feasible solutions; in many cases, allowing the search to move to infeasible solutions is desirable. Closely linked to the definition of the search space is that of the neighborhood structure. At each iteration of LS or TS, the local transformations that can be applied to the current solution, denoted S, define a set of neighboring solutions in the search space, denoted N(S) (the neighborhood of S). Formally, N(S) is a subset of the search space defined by:

$$N(S) = \{\text{solutions obtained by applying a single local transformation to } S\}.$$

Tabus are one of the distinctive elements of TS when compared to LS. As we already mentioned, tabus are used to prevent cycling when moving away from local optima through non-improving moves. The key realization here is that when this situation occurs, something needs to be done to prevent the search from tracing back its steps to where it came from. This is achieved by declaring tabu (disallowing) moves that reverse the effect of recent moves. Tabus are also useful to help the search move away from previously visited portions of the search space and thus perform more extensive exploration.

Tabus are stored in a short-term memory of the search (the tabu list) and usually only a fixed and fairly limited quantity of information is recorded. In any given context, there are several possibilities regarding the specific information that is recorded. One could record complete solutions, but this requires a lot of storage and makes it expensive to check whether a potential move is tabu or not; it is therefore seldom used. The most commonly used tabus involve recording the last few transformations performed on the current solution and prohibiting reverse transformations.

3.3. The Proposed Method

3.3.1. Fitness Function (Cost)

Fitness (cost) = 1-((|Target-Sum|/Target) ^0.5) if (Sum<=Target)

Fitness(cost) =1-((|Target-Sum|/MaxDiff)^0.167) if (Sum > Target)

Let M = {m_1, m_2 ,……. ,m_n), m_i € {0, 1} be an arbitrary solution and the public key

A = (a_1, a_2, … , a_n), Sum = $\sum_{j=1}^{n} ajmj$,

FullSum = $\sum_{j=1}^{n} aj$

Target = Cipher text,

MaxDiff = max {Target, FullSum – Target}

3.3.2. Proposed Algorithm

The proposed algorithm for Cryptanalysis based on Tabu Search for Knapsack Cryptosystem is represented in a simple way in Figure 1.

The alphabets are represented as strings of 0's and 1's by considering their ASCII values. In detail the proposed algorithm is described as follows:

1. The inputs to the algorithm are the known (intercepted) ciphertext, key size, P.
2. Set the size of the tabu list S_TABU, and S_POSS, the size of the possibilities list. Initialize the tabu list with a list of random and distinct key (Binary Strings of 0's & 1's) and calculate the cost associated with each of the key in the tabu list.

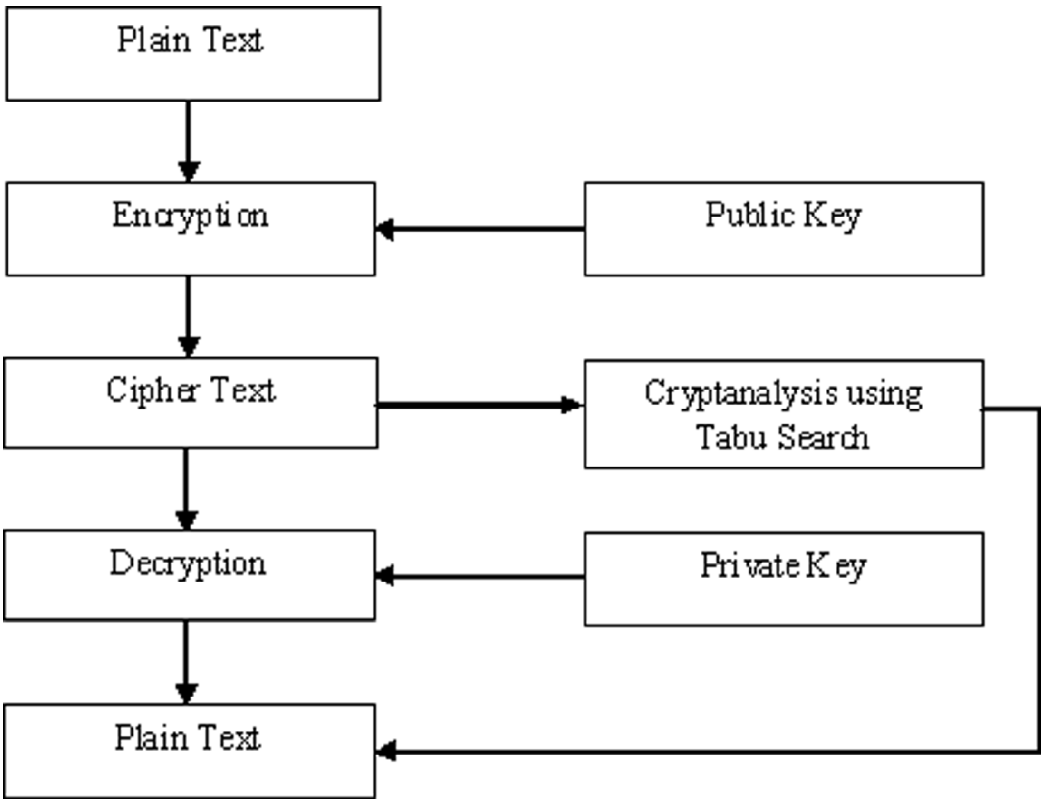

Figure1: Cryptanalysis using Tabu Search

3. While cost !=1 , do

(a) Find the best key in the tabu list which has the highest cost associated with it.

Call this key *KBEST*.

(b) For j = 1……, S_TABU, do

i. Choose $n1, n2 \in [1, P]$ $n1 \neq n2$.

ii. Create a possible new key *KNEW* by swapping the elements n1 and $n2$ in K*BEST*.

iii. Check that *KNEW* is not already in the list of possibilities for this iteration or the tabu list. If it is return to Step 3(b)i.

iv. Add K*NEW* to the list of possibilities for this iteration and determine its cost.

(c) From the list of possibilities for this iteration find the key with the highest cost – call this key *PBEST*.

(d) From the tabu list find the key with the lowest cost – call this key *TWORST*.

(e) While the cost of *PBEST* is greater than the cost of *TWORST*

i. Replace *TWORST* with *PBEST*.

ii. Find the new *PBEST*.

iii. Find the new *TWORST*

4. Output the best solution (i.e., the one with the cost 1) from the tabu list.

4. RESULTS

Experimental results for the two techniques were generated with 10 runs per data point using 'C' language in an Intel Core 2 Duo Processor with 2 GB of RAM. The best result for each message was averaged to produce the data point. We are comparing each of the two techniques on the basis of two criteria: number of iteration required and time requirement.

The first criterion is made upon the no of iteration executed to attack. These results are presented in Table 2.And the comparison of number of iteration required for genetic algorithm and tabu search for cryptanalysis is graphically represented in Figure 2.

The second criterion is made upon the time to execute the attack. These results are presented in Table 3.And the comparison of time requirement for genetic algorithm and tabu search for cryptanalysis is graphically represented in Figure 3.

Table 2
Number of iterations required

Character	*G.A*	*T.S*
M	472.9	464.5
A	162.3	143.1
C	383.7	273.3
R	265	232.6
O	276.1	260.3
Avg	312	274.76

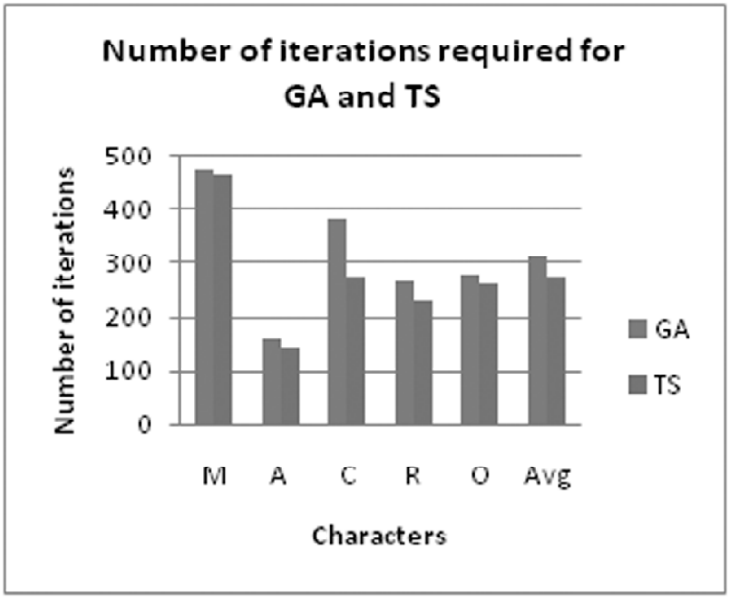

Figure 2: Comparison of number of iteration required for genetic algorithm and tabu search for cryptanalysis.

Table 3
Time to execute the attack

Character	*G.A*	*T.S*
M	1.8141	1.8015
A	1.2683	1.1867
C	1.8469	1.1142
R	1.0642	1.0505
O	1.3719	1.0613
Avg	1.47308	1.24284

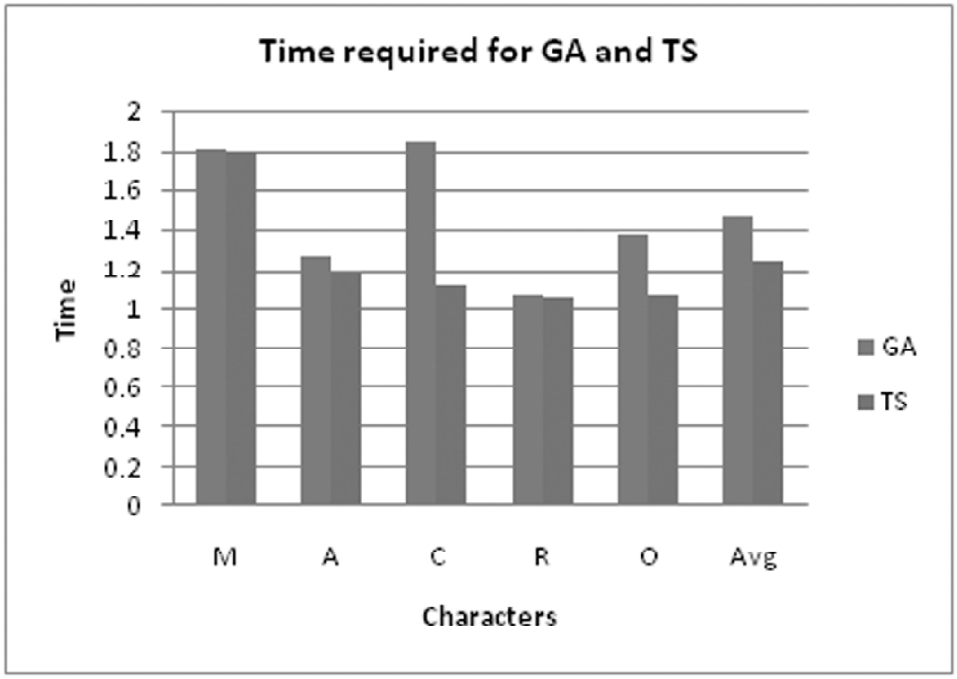

Figure 3: Comparison of time requirement for genetic algorithm and tabu search for cryptanalysis.

5. CONCLUSION

The paper explains cryptanalysis of the knapsack cipher using genetic algorithm and tabu search. It was found that for the knapsack cipher tabu search has the better result from the viewpoint of cryptanalysis. In paper [3] and [4] the authors used genetic algorithm as a tool for the cryptanalysis of Knapsack cipher. Our paper presents a new approach for the same using tabu search. It is producing better result both from the respect of iteration and time requirement.

It can be utilized to any highly secured environment to check the robustness of the algorithm as well as for cryptanalysis. One possible extension for our work is the incorporation of a new fitness function for getting better result.

REFERENCES

[1] Supravo Palit, Saptarshi Neil Sinha, Mostafiz Amin Molla, Atreyee Khanra, Malay Kule: A Cryptanalytic Attack on the Knapsack Cryptosystem using Binary Firefly Algorithm. IEEE International Conference on Computer & Communication Technology (ICCCT)-2011, Allahabad, India (2011) 428–432.

[2] Saptarshi Neil Sinha, Supravo Palit, Mostafiz Amin Molla, Atreyee Khanra, Malay Kule: A Cryptanalytic Attack on Knapsack Cipher using Differential Evolution Algorithm. IEEE RAICS2011, Bangalore, Kerala, India (2011) 317–320.

[3] P. Garg, A. Shastri, and Agarwal D.C.:An enhanced cryptanalytic attack on Knapsack Cipher using Genetic Algorithm.Proceedings of World Academy of Science, Engineering and Technology, Vol. 12 (2006) 145-148.

[4] Spillman, R: Cryptanalysis of Knapsack Ciphers Using Genetic Algorithms, *Cryptologia*, 17(4) (1993) 367-377.

[5] Goldberg, D.E.:Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading (1989).

[6] Behrouz A. Forouzan, Debdeep Mukhopadhay: Cryptography and Network Security,Second Edition, McGraw Hill Education (India) Private Limited, New Delhi (2014).

[7] Atul Kahate, Cryptography and Network Security,Second Edition, Tata McGraw-Hill Publishing Company Limited, New Delhi (2009).

[8] X.S.Yang:Firefly algorithms for multimodal optimization.Proceedings of the Stochastic Algorithms: Foundations and Applications (SAGA '09), Vol. 5792 of Lecture Notes in Computing Sciences, Springer, Sapporo, Japan (2009) 178–179.

[9] Helen F. Gaines: Cryptanalysis - A Study of Ciphers and Their Solutions (1989).

[10] Lukasik, S., Zak, S.: Firefly algorithm for continuous constrained optimization task. ICCCI 2009, Lecture Notes in Artificial Intelligence (Eds. N. T. Ngugen, R. Kowalczyk, S. M. Chen), (2009) 97-100.

[11] Poonam Garg: Genetic Algorithm & Tabu Search attack on mono alphabetic cipher.BIMC proceeding of Business Information Management Conference -2005, (2005) 322-328.

[12] Hertz A., de Werra D.:The Tabu Search Metaheuristic: how we used it. Annals of Mathematics and Artificial Intelligence 1, (1990) 111-121.

[13] Glover, F., Taillard, E.D., de Werra, D.: A user's guide to tabu search. Ann. Oper. Research 41, (1993) 3–28.