Cuckoo Search Strategies for Solving Combinatorial Problems (Solving Substitution Cipher: An Investigation)

Chapter · January 2018

DOI: 10.1007/978-981-10-5828-8_56

Ashish Jain
Manipal University Jaipur
20 PUBLICATIONS **149** CITATIONS

Jyoti Grover
Malaviya National Institute of Technology Jaipur
48 PUBLICATIONS **544** CITATIONS

Tarun Jain
Manipal University Jaipur
36 PUBLICATIONS **76** CITATIONS

Cuckoo Search Strategies for Solving Combinatorial Problems (Solving Substitution Cipher: An Investigation)

Ashish Jain, Jyoti Grover and Tarun Jain

Abstract Approximate algorithms have been well studied in order to solve combinatorial problems. This paper addresses cryptanalysis of the substitution cipher which is an interesting combinatorial problem. For this purpose, we utilize one of the latest approximate algorithms which is referred to as cuckoo search. Here, we point out that the proposed cuckoo search algorithm is not only an effective and efficient approach for solving the considered cryptanalysis problem, rather it can be a true and efficient choice for solving similar combinatorial problems.

Keywords Evolutionary computing · Cuckoo search · Scatter search · Genetic algorithm · Tabu search · Automated cryptanalysis

1 Introduction

Cryptology is very important in day-to-day life of common people, because cryptology is one set of techniques that provides information security. Cryptology field is commonly divided into two subfields cryptanalysis and cryptography.

Cryptography is the study of art and science for designing cryptosystems (or ciphers). A crucial component of the cipher is known as encryption algorithm. The encryption algorithm uses a secret key to transform a given plaintext into a ciphertext. Historically, cryptography is commonly connected with surveillance, warfare, and the like. However, with the advent of the information civilization and the digital

A. Jain (✉)
Discipline of Information Technology, Manipal University Jaipur, Jaipur, India
e-mail: ashish.jain@jaipur.manipal.edu

J. Grover · T. Jain
Discipline of Computer Science and Engineering, Manipal University Jaipur, Jaipur, India
e-mail: jyoti.grover@jaipur.manipal.edu

T. Jain
e-mail: tarun.jain@jaipur.manipal.edu

A.K. Somani et al. (eds.), *Proceedings of First International Conference on Smart System, Innovations and Computing*, Smart Innovation, Systems and Technologies 79, https://doi.org/10.1007/978-981-10-5828-8_56

revolution, cryptography is more and more important also in the peaceful lives of common people, e.g., when buying something over the Internet through credit card, withdrawing money from the ATM machines using smart-cards, and locking and unlocking luxury cars.

Cryptanalysis is related to finding flaws or oversights in the design of cryptosystems. The person who performs cryptanalysis is known as the cryptanalyst. The aim of cryptanalyst is to systematically recover the plaintext and/or secret key by attacking the cipher [1]. The attack involves the use of intelligent computer algorithms, just the ciphertext (i.e., encrypted plaintext) and/or some plaintexts. In the case of "ciphertext-only" attack, the attacker recover the plaintext and/or determine secret key by utilizing only the given ciphertexts. This type of attack is most challenging because we have just the knowledge of some ciphertexts and using that ciphertexts we need to determine secret key. This paper presents such a challenging cryptanalytic attack on a classical substitution cipher.

Cryptanalysis problems are considered as combinatorial search problems where the search space is consists of all possible combinations of key elements. The cryptanalytic attack via exhaustive (brute-force) searching, in theory, can be used against any encrypted text. However, in the worst case, this would involve traversing the entire search space. For instance, if the period of a substitution cipher is 26, then there are $26! = 4.03 \times 10^{26}$ permutation keys. If a supercomputer is available that could verify one billion million, i.e., 10^{15} "26-tuple" keys per second, would, in theory, require about 12788.28 years. Hence, these ciphers are secure from brute-force attack because the keyspace size is so large the resources and time are not available for searching the key exhaustively. On the other hand, search heuristics are capable of efficiently reducing the search space to a considerable extent. In the literature, several search heuristics have been applied that presents successful automated attacks on various classical ciphers. The advantage of automated attacks is that they run without time-consuming interaction of humans with a search process and finish when the key is determined. Thus, many cryptanalyst are interested in developing automated attacks of cryptographic algorithms. In this paper, we consider the automated cryptanalysis of the simple substitution cipher which is a most general form of classical ciphers.

Substitution cipher: A key of the substitution cipher is denoted as a permutation of plaintext letters (e.g., if 26 English alphabets represent plaintext letters, then a permutation of these 26 plaintext letters usually forms a substitution cipher key). Upon encryption, each letter of the plaintext message is replaced by the corresponding key element results in a ciphertext of length equal to the length of the plaintext. The original plaintext from the ciphertext is then recovered by intended recipient using decryption process. For a detailed description on the substitution cipher the interested reader can refer [2].

1.1 Related Work and Our Contributions

In literature, various search techniques have been applied for successfully attacking classical ciphers (e.g., substitution and transposition ciphers). Here, we mention only some of the previous research work where the main aim was to develop efficient automated attacks using search heuristics. Simulated annealing (SA) has been utilized by Forsyth and Safavi-Naini [3] to demonstrate attacks on the substitution ciphers. Another attack on these ciphers has been reported by Spillman et al. [4] using a genetic algorithm (GA). Clark [2, 5] has presented tabu search (TS) attack on the classical ciphers along with enhancement in previously proposed SA and GA attacks. Garici and Drias [6] have investigated the efficiency of scatter search (SS) for breaking the substitution cipher. In 2014, Boryczka and Dworak [7, 8] extended the study of attacks on the transposition ciphers using evolutionary algorithms to speed up the cryptanalysis process. The main deficiency in the above-mentioned attacks (i.e., in [6–8]) is that they fail to balance effectiveness and efficiency property of the GA attack that was proposed by Clark [2]. In other words, no cryptanalytic attacks of substitution cipher have been proposed in the literature that satisfies all the following objectives:

1. Number of ciphertext characters required by the proposed attack should be lesser among other attacks. In addition, the success rate should not be decreased. That is a more effective attack than other reported attacks.
2. Time required by the proposed attack should be lesser among other attacks. That is a more efficient attack that other reported attacks.
3. Number of keys examined by the proposed attack in finding the best solution should be lesser than other reported attacks.

This paper achieves the above-stated objectives by presenting two novel attacks based on the cuckoo search (CS). The authors of this paper are first researchers determining the efficiency of CS technique in developing automated attacks of the substitution cipher. In order to optimize the CS technique, some of its parameters are fine-tuned. Some of th efficient attacks of the substitution cipher that are based on the GA and TS techniques are also implemented. It should be noted that the SS attack presented by Garici and Drias [6] is not efficient with respect to the time complexity, however, this attack is considered in order to obtain a significant comparison between various attacks. Note that the CS algorithm is being designed in such a way that can be easily modified and can be used to solve similar combinatorial problems.

1.2 Fitness Function for Assessing Candidate Substitution Key

For determining the secret key of the substitution cipher, we start with a pool of candidate keys. Afterwards, we find suitability of each candidate key using following strategy. First of all, the known ciphertext is decrypted using candidate key. Afterward comparison is obtained between n-gram statistics of the decrypted text and language statistics which are considered to be known. Generally, these statistics are compared using the following equation:

$$Cost_k = \alpha\left(\sum_{i\in\mathscr{A}} |\mathcal{K}_i^u - \mathcal{D}_i^u|\right) + \beta\left(\sum_{i\in\mathscr{A}} |\mathcal{K}_i^b - \mathcal{D}_i^b|\right) + \gamma\left(\sum_{i\in\mathscr{A}} |\mathcal{K}_i^t - \mathcal{D}_i^t|\right), \quad (1)$$

In Eq. 1, $\mathscr{A}$ represents the alphabet of the language (for example, in English language: A, B, ..., Z), and $\mathcal{K}$ and $\mathcal{D}$ represent known language statistics and decrypted text statistics, respectively. The symbols u, b and t represent the unigram statistics of the language, bigram statistics of the language and trigram statistics of the language, respectively. Different value in 0.0, 0.1,..., 1.0 can be assigned to α, β and γ. However, $\alpha + \beta + \gamma = 1.0$ condition should be satisfied to keep the number of combinations of α, β and γ workable. Note that the substitution cipher attack is more effective by utilizing the cost functions that are designed using only bigrams than one which utilizes trigrams only [2]. Due to this fact, the cost function given below is used in this work which is purely based on the bigrams.

$$Cost_k = \sum_{i\in\mathscr{A}} |\mathcal{K}_i^b - \mathcal{D}_i^b| \quad (2)$$

1.3 Implementation of GA and TS Attacks of the Substitution Cipher

It is surprising that in last two decades no cryptanalytic attacks of the substitution cipher (except CS attack proposed by Jain and Chaudhari in [9]) have been proposed in the literature that can give results better than GA and TS algorithms of Clark [2]. Therefore, our task is to first implement GA and TS attacks and then compare two different CS attacks (one which is developed in this paper and another that was reported in [9]).

GA Attack: GA technique is initialized by a pool of n solutions (i.e., by n candidate keys of the substitution cipher). The main components of the GA attack are: fitness function which is already defined in the previous section, selection method which is a tournament selection method that selects a best candidate key from five randomly chosen candidate keys, crossover operator is defined in [2], and mutation operator is simply based on swapping two random elements of the investigated candidate key. Lack of space prevents us to present the pseudocode of GA attack in this paper. However, for a detailed description of GA attack, the interested reader can refer [2, 9].

First of all, a pool of n candidate keys called tabu is initialized. The TS attack maintains this tabu list as a short-term memory list. In each iteration, the investigated key is appended in the tabu list, and the key remains in tabu for a fixed number of iterations. Lack of space prevents us to present the pseudocode of TS attack in this paper. However, for a detailed description of TS attack, the interested reader can refer [2, 9].

2 Proposed CS Attacks: CS Attack1 and CS Attack2

The "standard" CS technique has been proposed in 2009 by Yang and Deb [10, 11] which is a new nature-inspired population-based search heuristic. In this paper and in [9], we have utilized the standard CS technique and incorporated our attack idea in the standard template. The "standard" template has already been presented in [9, 10]. For pseudocode of CS attack1, we refer the reader to [9]. For pseudocode of CS attack2, the reader can refer Algorithm 1 of this paper. The main difference between CS attack1 and CS attack2 are as follows: (1) we repeat Steps (6) to (8) of Algorithm 1 n times in the case of CS attack2. (2) We abandon the 0.02 fraction of worst nests in the case of CS attack2, while in the case of CS attack1 we abandon 0.01 fraction of worst nests. These changes have been introduced to identify the better CS attack. The set of equations that are employed in Algorithm 1 are as follows:

$$\mathbf{x}_j(t+1) = \mathbf{x}_j(t) + \mu\, l \tag{3}$$

$$l = \frac{u}{|v|^{1/\lambda}} \tag{4}$$

$$\sigma_u(\lambda) = \left[\frac{\Gamma(1+\lambda)\sin(\pi\lambda/2)}{\Gamma((1+\lambda)/2)\lambda 2^{(\lambda-1)/2}}\right]^{1/\lambda} = 0.696575 \text{ and } \sigma_v(\lambda) = 1 (\text{if: } \lambda = 1.5), \tag{5}$$

Due to lack of space we cannot give details here, therefore, for a detailed description of terms used in the above set of equations, the reader can refer [9]. Here we describe the main idea of the attack. First of all, we map major parts of the CS technique that is nest, egg and Lévy flights to the considered problem. In almost all problems where CS technique is applied, usually it is assumed that each nest has one egg. But, in the case of cryptanalysis of the substitution cipher, we consider each nest has N distinct eggs/elements (i.e., N distinct characters of a key: $n_1, n_2, ..., n_n$, where, $N = 26$ (i.e., A–Z character). For the sake of simplification, we consider a unique number ($\in [1, N]$) is associated with each of the eggs. It means there must be a unique identity of each element in the nest, i.e., a substitution cipher key cannot have two similar characters. The difficulty is how to preserve distinctness property of the key elements, because it can be observed that the updating Eq. (3) will disturb the distinctness property of the key elements during update. It should be noted that the importance of Eq. (3) is that it build the new solution from an existing solution via Lévy flights which is an efficient approach, because the step-size is heavy-tailed and any large step is possible. That is, the candidate key has more chance to get

Algorithm 1 : CS Attack2 [proposed in this paper]

```
 1: Initialization: Generate a pool of $n$ host nests randomly, where host nests are
    actually the candidate keys of the substitution cipher. Call this pool $Existing_{keys}$,
    where each key is a vector.
 2: repeat
 3:   Initialize a counter COUNT=1.
 4:   Compute the cost of each of the keys of the $Existing_{keys}$ using Eq. (2).
 5:   repeat
 6:     Select a lowest cost key from the $Existing_{keys}$, call this key $BEST_{key}$.
 7:     Choose a key randomly from $Existing_{keys}$, e.g., $Existing^{i}_{key}$. Construct a new
        key called $NEW_{key}$) via Lévy flights as: $NEW_{key}$=$Existing^{i}_{key}$+$\mu$ $l$, where $l$
        is evaluated using Eq. (4) with $\lambda$=1.5 and $\mu$=0.01.
 8:     Compute the cost of $NEW_{key}$ using Eq. (2). If the cost of $NEW_{key}$ is lesser
        than the cost of $BEST_{key}$ then it becomes the new $BEST_{key}$,
 9:   until (COUNT>$n$)
10:   Update the $Existing^{i}_{key}$ using $BEST_{key}$ and $NEW_{key}$ as follows.
11:   Generate a random number in the range [1, 26], call this number $m$.
12:   repeat
13:     Find the "next" element in $NEW_{key}$ that has the identity less than $m$. Assum-
        ing that the element is found at $P_1$ position. In the case of first iteration, read
        "first" instead of "next".
14:     Note the identity of the element which is located at position $P_1$ in the
        $BEST_{key}$. Call this identity $n_1$.
15:     Assuming that $P_2$ is the position in the $Existing^{i}_{keys}$ where the element of
        identity $n_1$ is located. Swap elements of the $Existing^{i}_{nests}$ that are located at
        positions $P_1$ and $P_2$. This operation will store the element of identity $n_1$ at
        position $P_1$ because $P_1$ is the best position of the same identity element in
        the $BEST_{key}$. It is important to note that this swapping operation is carried
        out in order to preserve the element that have already placed in $Existing^{i}_{keys}$
        to its best position.
16:   until ($NEW_{key}$ list has been traversed completely)
17:   If $BEST_{key}$ and updated $Existing^{i}_{keys}$ are identical, then swap the elements that
      are located at two different positions in $Existing^{i}_{keys}$, where positions are deter-
      mined randomly.
18:   Abandon a fraction ($p_a$) of worst nests and create new nests, i.e., create new
      keys as replacement of abandon keys. The new keys are created again from
      the best key by swapping two randomly chosen elements of the best key. In
      the experiment, we fixed $p_a$=0.02.
19: until (Maximum-Iterations (or) All key characters are determined correctly)
20: Output the best fitness key from the $Existing_{keys}$.
```

converted in the exact key in less computations. Hence, we do not change this equation, reasonably we apply its efficiency in improving existing keys using the best key and new keys generated by Eq. (3) (see Algorithm 1, Steps (8) to (13)).

3 Results

The attack algorithms GA, TS, and CS have been implemented in Java 2.0 on an Intel Quad-Core processor i3 CPU (@2.20 GHz). The input to each of the algorithms is bigram statistics of the English language, known ciphertext, and ciphertext length. As an output, each algorithm determines full or partial key that has been used in the substitution cipher for encrypting plaintexts. Parameters such as population size in the case of GA attack, tabu list size in the case of TS attack and a pool of host nests in the case of CS attack have been fine-tuned by performing extensive experiments. The fine-tuning has been carried out separately for each of the techniques in order to optimize the cryptanalysis process. In some scenario, guidelines helps us, e.g., in the case of CS attack, we choose $\mu = 0.01$ and $\lambda = 1.5$ that has been published in [11] as a generic choice. For obtaining a fair comparison between presented attacks, the guidelines suggested by Clark [2] is followed, i.e., three criteria are used: number of ciphertext characters are available for attack, number of keys examined for finding the best solution, and time needed to find the best key.

We tested each of the attacks on 200 different known ciphertexts. For each ciphertext, each attack has been tested three times, i.e., a total of 600 times, and the best of the three has been recorded. In this way, 200 best-recorded results corresponding to each of the algorithms is then averaged. This recording and averaging process is repeated for known ciphertext of size 100, 200, 300, ..., 800 characters. The average results of each algorithm are shown in Figs. 1 and 2. From Fig. 1, we can observe that each algorithm perform well and approximate equally. However, the approximate algorithms should be compared based on the following two criteria for determining the accurate efficiency: state space searched (i.e., the number of keys examined before evolving the best result) and complexity of the attack (i.e., time taken to determine the best key). For these two criteria, we have tested each attack algorithm on 100 different known ciphertexts of 1000 characters length and recorded the number of keys examined and the time taken by the attack. The average results are shown in Figs. 3 and 4. Note that the SS method of Garici and Drias [6] is not implemented and therefore not considered in the Figs. 3 and 4, because Garici and Drias have concluded that the SS attack takes 75% more time than the GA attack, while the quality of the results is only 15% better. From Figs. 3 and 4, it can be observed that the average number of keys examined and average performance time, respectively in the case of TS attack are lesser than the GA attack. It is also clear from Figs. 3 and 4 that the CS attacks outperform TS and GA attacks in both respects.

Finally, we conclude that the CS attack1 is superior in performance than CS attack2.

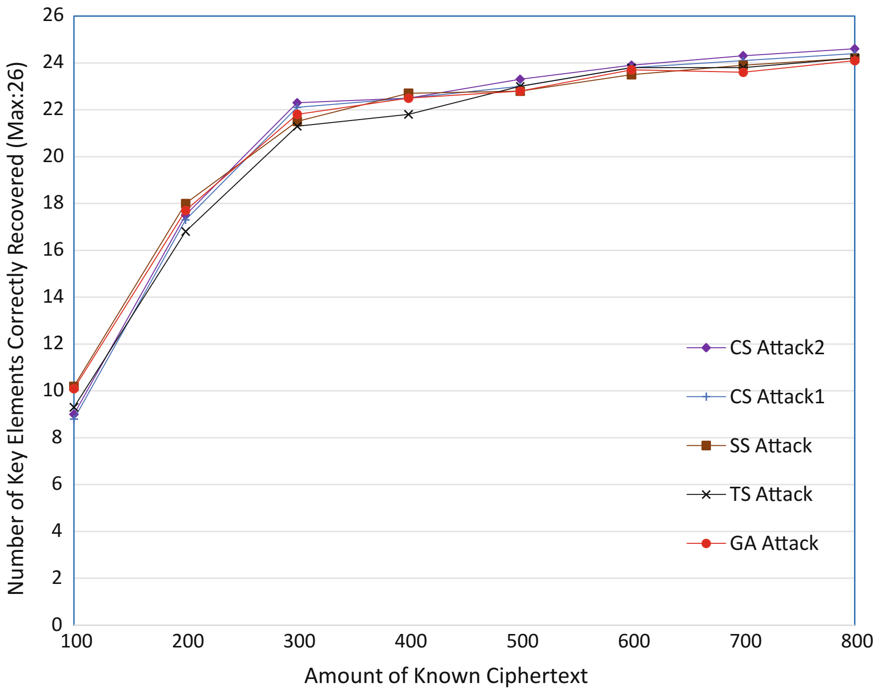
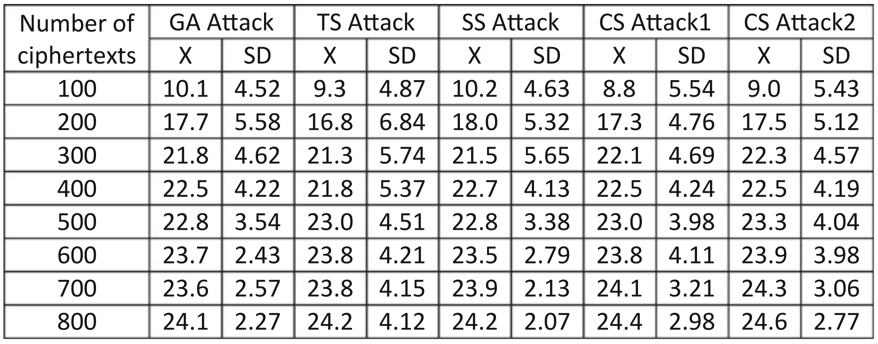


Fig. 1 A comparison based on the amount of known ciphertext



Number of ciphertexts	GA Attack		TS Attack		SS Attack		CS Attack1		CS Attack2	
	X	SD	X	SD	X	SD	X	SD	X	SD
100	10.1	4.52	9.3	4.87	10.2	4.63	8.8	5.54	9.0	5.43
200	17.7	5.58	16.8	6.84	18.0	5.32	17.3	4.76	17.5	5.12
300	21.8	4.62	21.3	5.74	21.5	5.65	22.1	4.69	22.3	4.57
400	22.5	4.22	21.8	5.37	22.7	4.13	22.5	4.24	22.5	4.19
500	22.8	3.54	23.0	4.51	22.8	3.38	23.0	3.98	23.3	4.04
600	23.7	2.43	23.8	4.21	23.5	2.79	23.8	4.11	23.9	3.98
700	23.6	2.57	23.8	4.15	23.9	2.13	24.1	3.21	24.3	3.06
800	24.1	2.27	24.2	4.12	24.2	2.07	24.4	2.98	24.6	2.77

Fig. 2 Mean (X) and standard deviation (SD) corresponding to Fig. 1

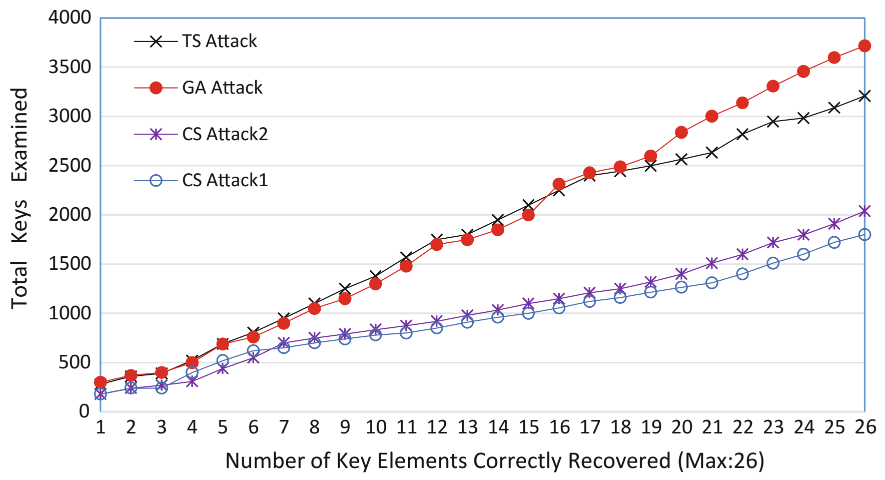


Fig. 3 A comparison based on the number of keys examined

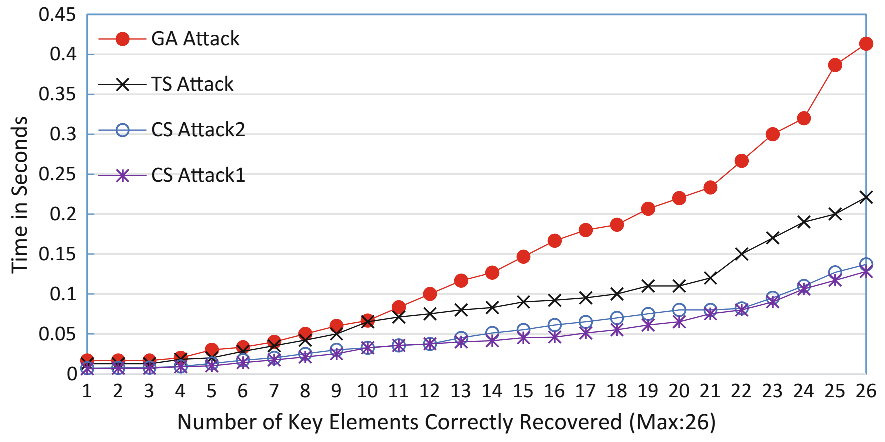


Fig. 4 A comparison based on time

4 Conclusion and Open Problems

This paper has demonstrated various attacks on the substitution cipher. The attacks have been developed by utilizing search heuristics, namely, GA, TS, SS, and CS. Results indicate the performance of CS attacks are superior than GA, TS, and SS attacks. It is important to note that in this work, we have to fine-tuned lesser number of parameters for both type of CS attacks than GA, TS, and SS attacks. This study indicates that the CS strategy is capable to produce results that are much better than the previously proposed GA, TS, and SS techniques. Therefore, the proposed CS strategies are true and efficient alternatives for solving this kind of combinatorial problems, for example, for optimal placement of phaser measurement units in a power system, for graph coloring, for solving Boolean satisfiability problem, etc., but, at what extent the proposed CS strategy can be utilized we left as an open problem.

References

1. Stinson, D.R.: Cryptography: theory and practice. CRC press (2005)
2. Clark, A.J.: Optimisation heuristics for cryptology. PhD thesis (1998)
3. Forsyth, W.S., Safavi-Naini, R.: Automated cryptanalysis of substitution ciphers. Cryptologia **17**(4) (1993) 407–418
4. Spillman, R., Janssen, M., Nelson, B., Kepner, M.: Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers. Cryptologia **17**(1) (1993) 31–44
5. Clark, A.: Modern optimisation algorithms for cryptanalysis. In: Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on, IEEE (1994) 258–262
6. Garici, M.A., Drias, H.: Cryptanalysis of substitution ciphers using scatter search. In: Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach. Springer (2005) 31–40
7. Boryczka, U., Dworak, K.: Genetic transformation techniques in cryptanalysis. In: Intelligent Information and Database Systems. Springer (2014) 147–156
8. Boryczka, U., Dworak, K.: Cryptanalysis of transposition cipher using evolutionary algorithms. In: Computational Collective Intelligence. Technologies and Applications. Springer (2014) 623–632
9. Jain, A., Chaudhari, N.S.: A new heuristic based on the cuckoo search for cryptanalysis of substitution ciphers. In: Neural Information Processing, Springer (2015) 206–215
10. Yang, X.S., Deb, S.: Cuckoo search via lévy flights. In: Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, IEEE (2009) 210–214
11. Yang, X.S., Deb, S.: Engineering optimisation by cuckoo search. Internl. Journal of Mathematical Modelling and Numerical Optimisation **1**(4) (2010) 330–343