Differential Evolution for the Cryptanalysis of Transposition Cipher

Gia S. Wulandari[1], Wahyu Rismawan[2], Siti Saadah[3]
[1,2,3]School of Computing, Telkom University, Bandung, Indonesia
[1]giaseptiana@telkomuniversity.ac.id, [2]wahyu.rismawan@outlook.com, [3]sitisaadah@telkomuniversity.ac.id

***Abstract*—Transposition cipher is a class of hystorical encryption algorithms that rearrange positions in plaintext based on some fixed permutation which is its secret key. In this research Differential Evolution was used to attack the transposition cipher, which is a permutation of integer problem. Despite the fact that Differential Evolution mostly used for problem with real numbers; this paper shows that Differential Evolution could be used to correctly decrypt ciphertext that has up to permutation length of 9, but started to have half of incorrect answers in 10 simulations done to permutation length of 10.**

I. Introduction

Cryptography is the art to keep information by creating a secret codes so that unwanted people can not see the information. There is also cryptanalysis, the art to break the code without knowing how the secret code is created [6]. Transposition cipher is one of some basic methods to create the secret code. Transposition cipher is created by using permutation.

As transposition cipher has the length of its permutation size as its public key, attacking transposition cipher can be done by trying all the permutation possibility of the length. However the process will take a long time. Therefore, heuristic methods can be used to minimize the time to attack. There are some heuristic methods applied in attacking Transposition Cipher, such as Simulated Annealing, Genetic Algorithm, Tabu Search, and Particle Swarm Optimization [1],[7],[9],[11]. Differential Evolution (DE) is an evolutionary algorithm that outperforms Genetic Algorithms on many numerical single objective optimization problems [5],[12]. However attacking transposition cipher is a discrete and permutation problem, which is not a common problem in DE that mostly works in real numbers that never been researched before. Representing real numbers given from DE equations into permutation of integer is still an ongoing task that has a lot of room for improvement. In this paper, it will be shown that DE also can attack transposition cipher for some permutation length.

II. Transposition Cipher

Transposition cipher is a method of encryption by rearrange positions held by units of plaintext to some fixed permutation which is its secret key. Public key of transposition cipher is its permutation size. For example consider a permutation cipher that has permutation length of 5 and a key K={3,5,2,1,4}. In this case, plaintext is broken into a block of five characters, and the ciphertext will have the third letter of each blocks on the plaintext as the first letter of the blocks, fifth letter on plaintext as second letter on cipher text, and so on. Table I shows the encryption process of the example.

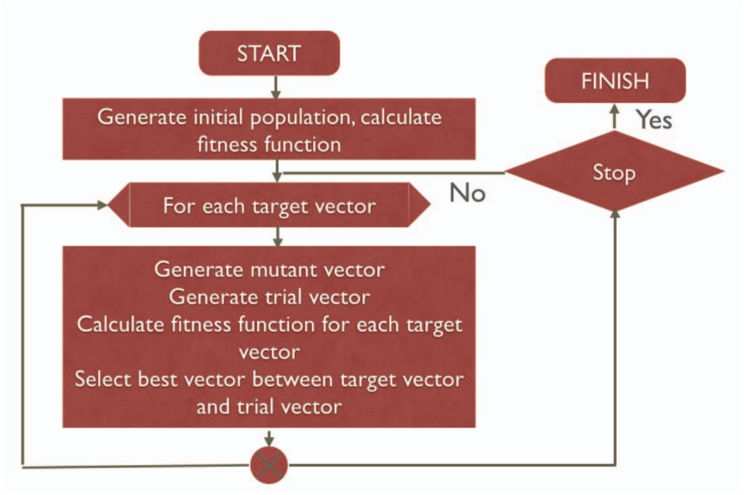

START
Generate initial population, calculate fitness function
For each target vector
Generate mutant vector
Generate trial vector
Calculate fitness function for each target vector
Select best vector between target vector and trial vector
Stop
No
Yes
FINISH

Fig. 1. General process of Differential Evolution

TABLE I. Example of Transposition Cipher

Plaintext	A	L	G	O	R	I	T	H	M	S
Position	1	2	3	4	5	1	2	3	4	5
Key	3	5	2	1	4	3	5	2	1	4
Ciphertext	G	R	L	A	O	H	S	T	I	M

III. Differential Evolution

Differential Evolution (DE) is an evolutionary algorithms by trying to improve candidate solution. Population on DE initialized by initial bounds. Figure 1 shows the general process of DE.

Initial vector population in DE is chosen randomly and should cover the entire parameter space [8]. To generate new parameter vectors, three random vectors from target vectors are used. First, mutation operator is performed by adding a vector to the weighted difference between two other vectors. The mutated vector's parameters are then mixed with the parameters of the target vector to produce the trial vector. This operation is called crossover. The trial vector then be compared with the target vector in the selection process. Trial vector replaces the target vector in the next generation if the fitness function value of the trial vector is better than fitness function value of the target vector.

The optimization in DE is based on three main operators, that are mutation, crossover, and selection.

1) Mutation: Mutation process is done by selecting three random vectors from target vectors. For each target vector $x_{i,g}$, $i = 1, 2, 3, \cdots, NP$, a mutant vector in g-th generation is generated with the following formula:

$$v_{i,g+1} = x_{r_0,g} + F.(x_{r_1,g} - x_{r_2,g}). \tag{1}$$

NP is number of the population; r_0, r_1, and r_2 are choosen randomly from $\{1, 2, \cdots, NP\}$ and are mutually different. $F \in [0, 2]$ is real and constant factor that works as the weight of the difference of $x_{r_1,g}$ and $x_{r_2,g}$.

2) Crossover: Crossover generates trial vector $u_{i,g+1} = (u_{1i,g+1}, u_{2i,g+1}, \cdots, u_{Di,g+1})$, where D is the dimension of the vectors, from mutant vector $v_{i,g+1}$ and target vector $x_{i,g}$ with equation:

$$u_{i,g+1} = \begin{cases} v_{ji,g+1}, & \text{if } rand_j(0,1) \leq CR \text{ or } (j = j_{rand}); \\ x_{ji,g}, & \text{else.} \end{cases} \tag{2}$$

where $rand_j(0,1)$ is j-th evaluation from random number generator, CR is a crossover constant that is determined by the user, and j_{rand} is randomly choosen index from 1 to D.

3) Selection: In selection process, trial vectors and target vectors are compared to choose whether the trial vector should be a member of the next generation or not. If the trial vector is better than target vector, the target vector be replaced by the trial vector for the next generation. The comparation is done by comparing their fitness function value as bellow:

$$x_{i,g+1} = \begin{cases} u_{i,g+1}, & \text{if } f(u_{i,g+1}) \leq f(x_{i,g}); \\ x_{i,g}, & \text{else.} \end{cases} \tag{3}$$

IV. Attack on Transposition Cipher

Actually, there are many techniques for transposition cipher, but in this research we only focus on the technique described before. For other techniques, we need further analysis on its performance but for the attacking method itself will not have a lot of difference.

Differential Evolution mostly used for problems with real numbers, but the attack on transposition cipher works only with integer. Therefore we need some modification to general DE as described above. Lingjuan HOU [6], in his paper titled "A Novel Discrete Differential Evolution Algorithm for Stochastic VRSPD" proposed an adaptation from continue case to discrete problem by changing mutation process using definition in algebra. Cao Erbao also modified Differential Evolution to solve vehicle routing problem [3]. In this research, we used some modification proposed by Cao Erbao for mutation operator.

Initial population was generated randomly with the size between LB and UB, where LB is lower bound and UB is the upper bound for the population size. In our research, upper bound was set to $D(D - 1)$ and lower bound was set to $D(D \div 2)$, where D represents permutation length of transposition cipher. Initial population was generated randomly, and

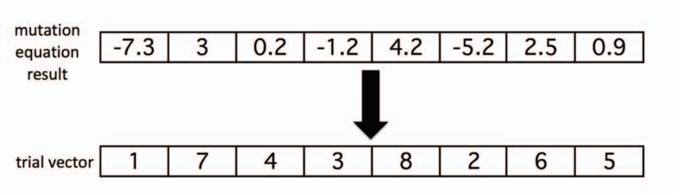


Fig. 2. Example of mutation operation

after we have the initial population set as target vector; the mutation, crossover, and selection process are performed to obtain target vector for the next generation. Selection process was performed using equation 3.

A. Mutation Operator

Mutation operator as in equation 1 works for real number if F is not integer. If integer F is choosen, that is $F \in \{0, 1, 2\}$, there is large posibility that the solution given for $v_{i,g+1}$ is not permutation. As we have permutation of integer problem, some modification for the operator is needed.

In this research, mutation operator using integer order criterion (IOR) described by Cao Erbao [3] was used. First, mutation operator as in the original equation is calculated; then the trial vector acquires the integer order of the calculation. For example, if equation 1 gives result [-7.3 3 0.2 -1.2 4.2 -5.2 2.5 0.9], then the trial vector is [1 7 4 3 8 2 6 5]; as can be seen in Figure 2.

B. Crossover

Crossover is performed to add diversity to the population. In this research, equation for crossover operator described in equation 2 was used with a little modification. If we have $rand_j$ value not bigger than crossover rate value, then we have an individual u_i identical to vector trial v_i. Otherwise, j^{th} element of u_i is the j^{th} element of v_i, and rest acquire original value of elements in the same index in x_i. However, identical elements might occur, so we need to change the value of one element to keep the permutation. For example as can be seen in Figure 3, when we have [1 7 4 3 8 2 6 5] as trial vector v_i and [3 8 6 7 2 4 5 1] as x_i. Let $rand_j$ greater than crossover rate and $j_{rand} = 5$, then we have a new vector [3 2 6 7 8 4 5 1], because if we only change the fifth index, we will have [3 8 6 7 8 4 5 1], which is not permutation. Therefor we need to change the remaining 8 with initial value of the fifth index, 2. So we have [3 2 6 7 8 4 5 1].

We also adapt time-varying crossover rate, also proposed by Cao Erbao [3]. The crossover rate was increased in the next generation to make sure that the solution is not trapped in local optimum. Crossover rate is calculated using equation 4.

$$CR = CR_{min} + g * \frac{CR_{max} - CR_{min}}{MaxGen}, \tag{4}$$

where CR_{min} and CR_{max} respectively are the proposed minimum and maximum crossover probability, g is the generation count at the time, and $MaxGen$ is the maximum iteration used. In this research, 0.5 and 0.9 are set as CR_{min} and CR_{max} respectively, and $MaxGen$ is set to 100.

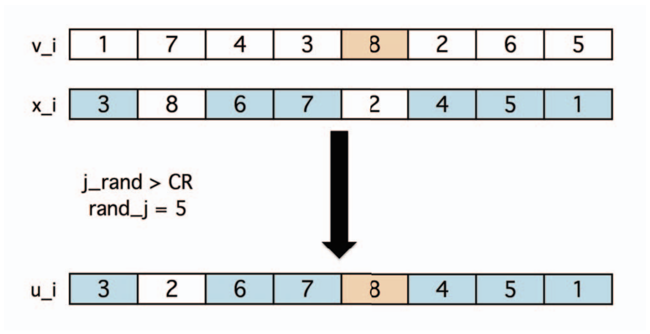


Fig. 3. Example of crossover operation

C. Fitness Function

In cryptanalysis, equation 5 mostly used to compare candidate keys. In the equation, the statistics of decrypted message using key k, D, is compared to statistics of known language, K. Statistics of English from Practical Cryptography (http://practicalcryptography.com/) is used in this research.

$$C_k = \beta \cdot \sum_{i,j \in A} |K^b_{(i,j)} - D^b_{(i,j)}| + \gamma \cdot \sum_{i,j,k \in A} |K^t_{(i,j,k)} - D^t_{(i,j,k)}| \tag{5}$$

Here, b and t detones bigram and trigram statistics, respectively, and the values of β and γ allow assigning different weight for bigram and trigram calculation. Some researcher used monogram also for the equation, but in this research, we did not use the monogram frequency as the frequency for each possible keys are equal. The weight used in this research are 0.4 and 0.6 respectively for β and γ.

Using the equation, we proposed a fitness function for this research as follows:

$$f(x_i) = \frac{1}{C_{x_i}} \tag{6}$$

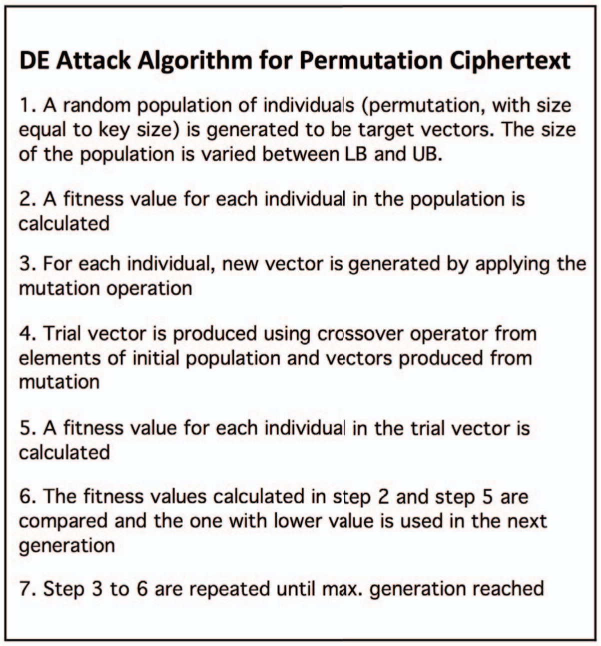
The steps needed to attack Transposition Cipher using Differential Evolution can be seen in Figure 4.

V. Experimental Test and Result

Experiments in this research were performed on text in alphabetic English, i.e. A-Z. All non-alphabet characters has been removed from the text before encryption, and some characters has been erased to match the text sized with the permutation size so that the ciphertext length can be divided by permutation length. The algorithm runs in python with different ciphertext and key size. Each ciphertext also has various text length.

There are five texts in various length and various topics used in this research. Information about the texts can be seen in Table II. Text length mentioned is the length after removing non-alphabet characters from the text.

To analyze whether the text size or topic matter in this algorithm, each data runs 10 times for permutation size of 5, 9, and 13. Table III shows the result of cryptanalys each data for three different permutation size; 5, 9, and 13. It is shown that for permutation size 5 and 9, neither text length nor topic give any differences to algorithm performance. However, differences can be seen at permutation size of 13. The longer text size is, the better result produced by the algorithm.



DE Attack Algorithm for Permutation Ciphertext

1. A random population of individuals (permutation, with size equal to key size) is generated to be target vectors. The size of the population is varied between LB and UB.
2. A fitness value for each individual in the population is calculated
3. For each individual, new vector is generated by applying the mutation operation
4. Trial vector is produced using crossover operator from elements of initial population and vectors produced from mutation
5. A fitness value for each individual in the trial vector is calculated
6. The fitness values calculated in step 2 and step 5 are compared and the one with lower value is used in the next generation
7. Step 3 to 6 are repeated until max. generation reached

Fig. 4. Algorithm of Attacking Transposition Cipher using Differential Evolution

TABLE II. INFORMATION OF USED TEXTS

Text	Text Length	Topic
Data01	665	Security
Data02	822	News of innovation
Data03	980	News of new product
Data04	2316	News of sports
Data05	3812	Research

TABLE III. NUMBER OF KEYS RECOVERED SUCCESSFULLY FOR FIVE DIFFERENT DATA

	Data01	Data02	Data03	Data04	Data05
Key size of 5	5	5	5	5	5
Key size of 9	9	9	9	9	9
Key size of 13	0.8	1	1.6	1.9	6.3

The reason of the failure in permutation size of 13 was because it was stuck in local minimum in spite we have prepared for it by using time dependent crossover rate. Other than that, fitness function also plays important role. For example, in Data02 with permutation size 13, a key with fitness value lower than the fitness value of correct key is produced. It means the fitness function also need another modification.

To know how long permutation length can be correctly decrypted by this algorithm, one Data03 has been executed 10 times for six different permutation size, that are: 5, 6, 7, 8, 9, and 10. Table IV shows the performance of the algorithm seen by averege amount of correctly recovered key in 10 executions.

It is shown that the algorithm works well until key size of

TABLE IV. AMOUNT OF KEYS RECOVERED SUCCESSFULLY FOR DATA03

Permutation size	5	6	7	8	9	10
Recovered key	5	6	7	8	9	5.5

9. The algorithm produced the exact answer key to decrypt the ciphertext for all 10 simulations. However, only five from 10 simulations for permutation length of 10 produced the answer key. Only a little amount of correct recovered keys was given by the other five simulations.

Logically thinking, time required for algorithm to run will be longer for longer key size. Table V shows how different the time required to execute the algorithm for Data03.

TABLE V. TIME REQUIRED FOR ALGORITHM TO RUN FOR DATA03

Permutation size	5	6	7	8	9	10
Time required (s)	10.57	45.47	63.04	64.65	77.6	91.16

VI. CONCLUSION

This paper presents Differential Evolution for attacking Transposition Cipher. Differential Evolution, that mostly used for problems with real numbers, could be used also to solve permutation of integers problem. With the ability to retrieve the correct key to decrypt the ciphertext, Differential Evolution proposed in this paper can be used to attack Transposition Cipher with short length of permutation, that is up to size of 9. For longer key, the algorithm still not doing well, specially for shorter text. Some other modification for mutation or crossover permutation might need to be done as it seems that the algorithm was trapped in local minimum. In addition, modification on fitness function also needed to get better result.

REFERENCES

[1] A.Dimovski and D.Gligoroski, ”Attacks on the Transposition Ciphers Using Optimization Heuristics” in *Proceedings of ICEST 2003*, 2003.

[2] Antonin Ponsich, Ma. Guadalupe Castillo Tapia, Carlos A. Coello Coello, ”Solving permutation problems with Differential Evolution: An Application to the Jobshop Scheduling Problem” in *Proceedings of 9th International Conference on Intelligent Systems Design and Applications*, 2009.

[3] Cao Erbao, Lai Mingyong, ”The open vehicle routing problem with fuzzy demands” in *Expert Systems with Applications* Vol. 37, pp. 24052411, 2010.

[4] E.C. Laskari, G.C. Meletiou, Y.C. Stamatiou, M.N. Vrahatisa, ”Evolutionary computation based cryptanalysis: A first study” in *Nonlinear Analysis* Vol. 63, pp. 823 - 830, 2005.

[5] Hegerty, Brian, Chih-Cheng Hung, and Kristen Kasprak. ”A comparative study on differential evolution and genetic algorithms for some combinatorial problems” in *Proceedings of 8th Mexican International Conference on Artificial Intelligence*, 2009.

[6] Lingjuan Hou, Hong Zhou, Jian Zhao, ”A Novel Discrete Differential Evolution Algorithm for Stochastic VRPSPD” in *Journal of Computational Information Systems* Vol. 6, pp. 2483-2491, 2010.

[7] Poonam Garg, ”Genetic Algorithms, Tabu Search and Simulated Annealing: A Comparison Between Three Approaches for the Cryptanalys of Transposition Cipher” in *Journal of Theoretical and Applied Information Technology*, pp. 387 - 392, 2009.

[8] Rainer Storn and Kenneth Price, ”Differential Evolution A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces” in *Journal of Global Optimization* Volume 11, pp. 341 - 359, 1997.

[9] SambasivaRao Baragada, and P. Satyanarayana Reddy, ”A Survey of Cryptanalytic Works Based on Genetic Algorithms” in *International Journal of Emerging Trends & Technology in Computer Science* Vol. 2 (5), pp. 18 - 22, 2013.

[10] Saptarshi Neil Sinha, Supravo Palit, Mostafiz Amin Molla, Atreyee Khanra, Malay Kule, ”A Cryptanalytic Attack on Knapsack Cipher Using Differential Evolution Algorithm”, IEEE, 2011.

[11] Sarab M. Hameed and Dalal N. Hmood, ”Particles Swarm Optimization for the Cryptanalys of Transposition Cipher” in *Journal of Al-Nahrain University*, Vol. 13 (4), pp. 211 - 215, 2010.

[12] Tea Tusar and Bogdan Filipic, ”Differential Evolution Versus Genetic Algorithms in Multiobjective Optimization” in *Lecture Notes in Computer Science* Vol. 4403, pp 257-271, 2007.