USING GENETIC ALGORITHMS TO BREAK A SIMPLE TRANSPOSITION CIPHER

A. S. AL-KHALID
mudariben@yahoo.com

S. S. OMRAN
Omran_Safaa@ymail.Com

Dalal. A. HAMMOOD
Alsaady_Dalal@Yahoo.Com

Foundation Of Technical Education
College Of Elec. & Electronic Techniques
(Baghdad-IRAQ)

Abstract

Genetic Algorithms are search algorithm. They use a metaphor where an optimization problem takes the place of an environment and feasible solutions are considered as individuals living in that environment.

This paper focuses on using GAs to cryptanalyse a transposition cipher. A certain method is used to find the possible key length. It is shown that such algorithm could be used to discover the key for transposition cipher. The focus is to be on a bi- and Tri-gram frequency of letters.

The frequency analysis is used as an essential factor in the objective function. The length of text was 3000 letters. The population size are 10, 20, 30, 40, and 50. The key size is 10 letters.

Two equations are used to find optimal soulution. The first one depends on the letter frequency of bigram, and the second one depends on score table. The result clarifies that the second equation gave the optimal solution for the population size 30, and 40, where the number of correct letters was 10 out of 10 letters.

A lot of papers are published to cryptanalyse a transposition cipher using GA, but they used score table. This paper presents two equations to see which one has a best solution. Different population size, key size is 10 letters, crossover and mutation rates are 0.2 are used.

Keywords: Genetic Algorithms, Transposition, Cryptanalyse, Key Search.

1. INTRODUCTION

Cryptology can be subdivided into two disciplines. *Cryptography* concerns itself with the design of cryptosystems, while *cryptanalysis* studies the breaking of cryptosystems. These two aspects are closely related; when setting up a cryptosystem, the analysis of its security plays an important role[1,2,3]. Cryptography is classified into two types Symmetric and Asymmetric[4].

The focus will be on transposition cipher as bi and Tri- gram letters.

There is a lot of papers that have been published about the using of genetic algorithms to cryptanalyse transposition cipher.

In 1993 **Matthwes** [5] for the first time presented a genetic algorithm approach for the cryptanalysis of transposition cipher, using the fitness scoring as in equation (1).

$$F_L = L\sum^{Q}(PiSi)/100\cdots\cdots(1)$$

In 1994 **Clark** [6] presented a genetic algorithm approach for the cryptanalysis of transposition cipher using genetic algorithm, using fitness weight in equation (1).

In 2003 Grundlingh and Vuuren [7] presented a genetic algorithm approach for the cryptanalysis of transposition cipher using genetic algorithm. He used fitness weight as in equation(2).

$$F\,(L,N,T,k) = \frac{2(N-1-\delta^{N}_{\min}(L)) - \sum_{i,j=A}^{Z}\left|\delta^{N}_{ij}(L) - f^{T(N)}_{ij}(k)\right|}{2(N-1-\delta^{N}_{\min}(L))}\cdots\cdots(2)$$

In 2007 **Toemeh and Arumugam** [8] presented a genetic algorithm approach for the cryptanalysis of transposition cipher using genetic algorithm. They used fitness weight in equation (1).

This paper is organized as follows: A brief description of cryptanalysis, description of attacks on transposition and algorithmic description of the attack on a simple transposition cipher using a genetic algorithm. The fitness weight as is shown in equation (3 and 4). The results of the genetic algorithm attack are given finally.

2. TRANSPOSITION CIPHER

A transposition is not a permutation of alphabet characters, but a permutation of places[9].

Transposition or Permutation cipher works by breaking a message into fixed size blocks, and then permuting the characters within each block according to a fixed permutation, say P. The key to the transposition cipher is simply the permutation P. So, the transposition cipher has the property that the encrypted message i.e. the cipher text contains all the characters that were in the plain text message. In the other word, the unigram statistics for the message are unchanged by the encryption process[2,9,10].

In this method, the message is written in a rectangle, row by row. Reading the message off, row by row, but permuting the order of the columns. The order of the columns then become the key to the algorithm. For example[10,11]:

The plain text is: breaking transposition cipher. Key : P ={4 1 3 5 7 6 2}

In this case, the message is broken into block of seven characters, and after encryption the fourth character in the block will be moved to position 1, the first is moved to position 2, the third remains in position 3, the fifth to position 4, the seventh to position five, the sixth remains in position 6, the two is moved to position 7 [10].

Figure 1.a shows the plaintext and Fig. 1.b shows the encryption process of the previously described transposition cipher.

It can be noticed that the random string "X" was appended to the end of message to enforce a massege length, which is a multiple of the block size[10,12].

1	2	3	4	5	6	7
b	r	e	a	k	i	n
g	t	r	a	n	s	p
o	s	i	t	i	o	n
c	i	p	h	e	r	x

-a- Plain text

4	1	3	5	7	6	2
a	b	e	k	n	i	r
a	g	r	n	p	s	t
t	o	i	i	n	o	s
h	c	p	e	x	r	i

-b- Cipher text

Fig. 1: Example of the transposition cipher encryption process

The cipher text is: bekniragrnpsttoiinoshcpexri

3. GENETIC ALGORITHMS

The genetic algorithm is based upon Darwinian evolution theory. In 1975 Holand was first to suggest the genetic algorithms for the problem solving[13].

In genetic algorithms, individuals are binary digits or of some other set of symbols drawn from a finite set. As computer memory is made up of array of bits, anything can be stored in a computer and can also be encoded by a bit string of sufficient length. Each of the encoded individual in the population can be viewed as a representation, according to an appropriate encoding of a particular solution to the problem. For Genetic Algorithms to find a best optimum solution, it is necessary to perform certain operations over these individuals[14,15].

A genetic algorithm contains three operators: Selection , Crossover and Mutation. To explain each operator as following[14 - 19,20]:

3.1 Selection

Selection is the process of choosing parents from the population for mating.

Selection is a method that randomly picks chromosomes out of the population according to their evaluation function. The higher the fitness function, the more chance an individual has to be selected. The selection pressure is defined as the degree to which the better individuals are favored. The higher the selection pressured, the more the better individuals are favored. This selection pressure drives the GA to improve the population fitness over the successive generations[14,21,22].

3.2 Crossover

Is the process of taking two parent solutions and producing from them two childs. After the selection process, the population is enriched with better individuals[14,21,23].

3.3 Mutation

Mutation is used for randomly altering a apart of an individual to produce a new individual[14,24,20].

Fig. 2 shows the cycle of genetic algorithms. Each cycle in Genetic Algorithms produces a new generation of possible solutions for a given problem[12,15].

Fig. 3 shows the flowchart for the proposed algorithm of transposition cipher by using GAs.

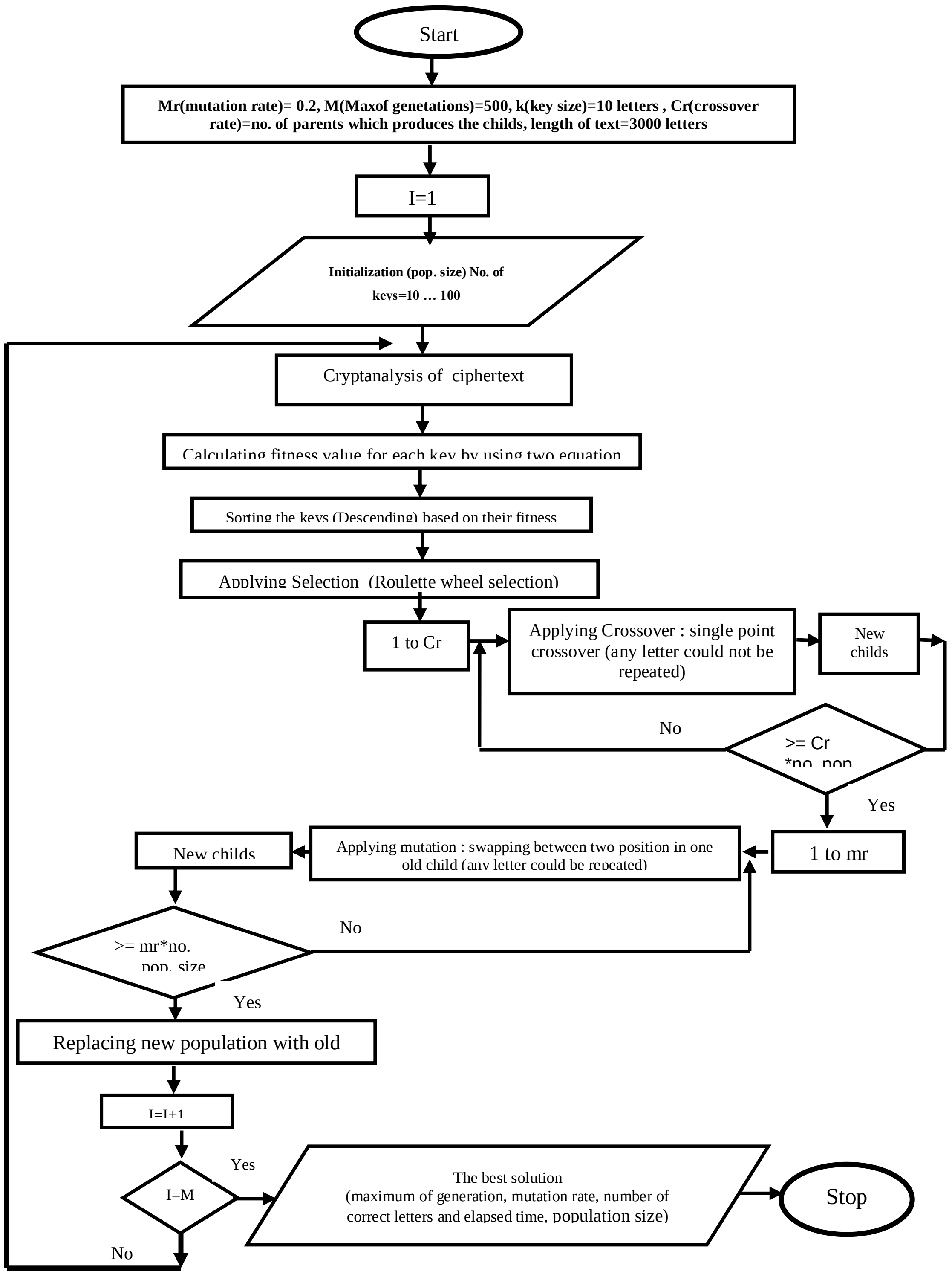

Start
Mr(mutation rate)= 0.2, M(Maxof genetations)=500, k(key size)=10 letters , Cr(crossover rate)=no. of parents which produces the childs, length of text=3000 letters
I=1
Initialization (pop. size) No. of keys=10 … 100
Cryptanalysis of ciphertext
Calculating fitness value for each key by using two equation
Sorting the keys (Descending) based on their fitness
Applying Selection (Roulette wheel selection)
1 to Cr
Applying Crossover : single point crossover (any letter could not be repeated)
New childs
No
>= Cr *no. pop
Yes
1 to mr
Applying mutation : swapping between two position in one old child (any letter could be repeated)
New childs
No
>= mr*no. pop. size
Yes
Replacing new population with old
I=I+1
Yes
I=M
No
The best solution (maximum of generation, mutation rate, number of correct letters and elapsed time, population size)
Stop

Fig.3: Flowchart Of Transposition Cipher By Using GAs

4. FITNESS MEASURE

4.1 First Equation

The technique used to compare candidate keys is to compare di-gram statistics of the decrypted message with those of the language (which is assumed known).

Table (1) shows the expected number of digram letters occurrences in English language text of length 10000 characters[7].

The following equation is a general formula used to determine the suitability of a proposed key (k)

$$F_{key} = 1 - [\beta \sum_{i,j \in A} | k^{b}_{i,j} - D^{b}_{i,j} | + \gamma \sum_{i,j,k \in A} | k^{t}_{i,j,k} - D^{t}_{i,j,k} |] \cdots (3)$$

Where *Fkey* is fitness value to find optimal solution, $K^{b}_{i,j}$ and $K^{t}_{i,j,k}$ are the known language bi-gram and tri-gram statistics as shown in table 1, and $D^{b}_{i,j}$ and $D^{t}_{i,j,k}$ are the bigram and trigram statistics of the message decrypted with key *K*. The weights β , and γ can be varied to allow more or less emphasis on particular statistics**.**

Table 1: Diphthong frequencies for English language

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	6	18	24	26	9	8	13	41	39	0	15	79	36	193	1	12	0	68	61	83	11	29	11	0	34	3
B	12	2	0	0	57	0	0	0	6	0	0	11	0	0	15	0	0	16	2	1	21	0	0	0	10	0
C	27	0	4	0	29	0	0	36	8	0	8	6	0	0	32	0	0	6	0	5	6	0	0	0	1	0
D	56	15	7	8	44	12	8	22	42	3	2	7	11	10	45	5	0	15	31	65	5	1	20	0	13	1
E	105	24	41	92	36	37	18	61	46	5	12	61	55	102	40	32	1	149	135	84	5	25	47	3	43	2
F	24	3	2	2	18	10	5	8	26	5	1	6	6	1	56	2	0	18	5	47	5	0	2	0	11	1
G	22	2	1	2	16	2	1	26	12	0	1	4	2	3	34	1	0	10	11	14	3	0	3	0	5	0
H	135	5	3	2	373	4	2	10	105	1	1	2	4	2	72	2	0	10	9	37	9	0	30	0	10	0
I	15	4	24	34	23	15	18	5	0	1	5	52	33	151	16	5	1	27	92	81	0	18	8	1	0	1
J	4	0	0	0	11	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	6	0	0	0	0	0
K	4	1	0	0	27	1	0	1	19	0	0	1	1	6	3	0	0	0	4	4	0	0	1	0	1	0
L	35	12	4	30	58	8	3	5	35	1	3	88	5	6	32	3	0	2	18	26	2	5	5	0	17	0
M	52	10	1	1	59	3	1	4	20	1	0	1	7	2	27	6	0	1	8	15	5	0	6	0	20	0
N	37	6	16	160	54	6	72	15	23	3	4	4	6	5	56	3	0	2	35	105	4	1	10	0	14	0
O	14	10	7	30	6	121	6	15	12	2	11	22	48	92	19	16	0	98	27	54	139	13	35	1	12	1
P	13	1	0	0	25	1	0	11	7	0	0	17	1	0	12	6	0	19	3	10	5	0	1	0	1	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
R	47	7	7	28	126	9	8	13	54	1	6	6	10	16	57	5	0	6	39	45	12	7	9	0	18	0
S	83	12	12	7	84	13	5	68	35	1	4	7	8	10	64	18	0	12	41	88	13	1	24	0	10	0
T	41	9	5	6	54	8	3	416	49	2	2	7	7	4	110	3	0	21	29	41	12	0	20	0	22	0
U	11	6	6	10	4	2	11	5	8	0	1	17	6	24	1	11	0	51	47	43	0	0	4	0	1	0
V	7	0	0	0	75	0	0	0	15	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0
W	35	1	0	1	54	1	0	55	55	0	0	2	1	9	21	0	0	3	4	6	0	0	2	0	1	0
X	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Y	35	6	4	5	12	6	3	8	10	0	1	4	6	3	85	6	0	3	21	17	1	0	9	0	3	0
Z	2	0	0	0	3	0	0	0	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

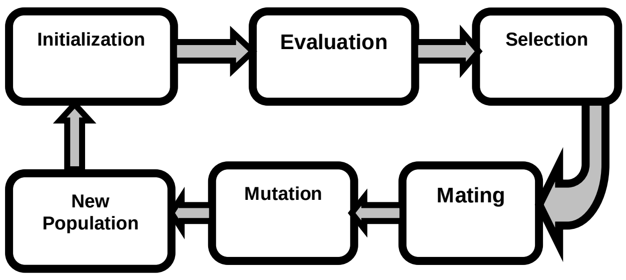


Fig. 2: Cycle of genetic algorithms

Table 2: The fitness scoring

Bigram	score	Bi/trigram	score
TH	+2	ED	+1
HE	+1	THE	+5
IN	+1	ING	+5
ER	+1	AND	+5
AN	+1	EEE	-5

$$F_L = \sum^{Q} (PiSi) \cdots\cdots (4)$$

4.2 Second equation: Fitness Scoring

F is fitness value, *Pi* is the percentage frequency of that bi- or tri-gram in the text, *Si* is the fitness score to the ith bi- or tri-gram tested for, and the summation is over the Q bi- and tri-grams checked[6].

Table (2) shows the expected number of digram and Tri-gram letters.

It is clear that the most score (+2) for bi-gram is TH letters, and (+5) for the tro giram are THE, ING, and AND. the most bi or tri gram could be TH(the, that, then,their, this, these, those, etc) , THE (the, there, these, them, their, etc), ING(and word with ing for example: teacing, finding, reading, etc,), and AND.

When the score +1 for HE, IN, ER, AN, ED. it could be guess the pair of letters HE (he, she, here, the,etc), ER (are, there, here, ..etc), AN (and, an, another, ..etc), and ED (any word in past simple and past participle)

5. USING GENETIC ALGORITHMS TO ATTACK TRANSPOSITION CIPHER

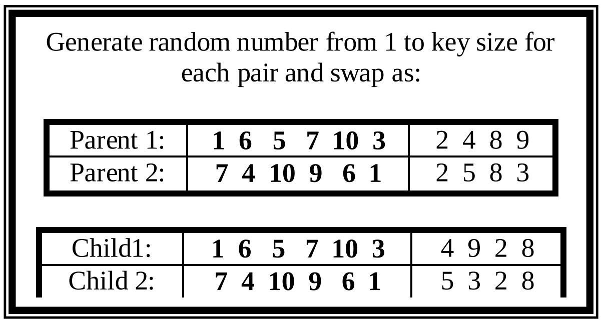
The attack is implemented by generating an initial candidate key pool *p(0)* of even cardinality, consisting of permutations of the set [1 to key size]. The first generation is generated randomly using a simple uniform random generator. Thereafter, the cipher text is decrypted using each permutation as a key, enabling us to assign a measure of fitness by using equation (2) to each candidate key. Pairs of candidate key are then selected for producing offspring after applying a method of crossover to each pairs.

The Roulette wheel selection is used in this paper. It is most common selection method used in genetic algorithms for selecting potentially useful individuals (solutions) for crossover and mutation.

In Roulette wheel selection, each member of the population is allocated a section of an imaginary roulette wheel. Unlike a real roulette wheel the sections are different sizes, proportional to the individual's fitness, such that the fittest candidate has the biggest slice of the wheel and the weakest candidate has the smallest. The wheel is then spun and the individual associated with the winning section is selected [7,14,21,22].

. Single point Crossover has been used in this method. The first part of first child is the first part of the first parent and second part of first child is the remaining digits as the order of second parent. The first part of second child is the first part of second parent and second part is the remaining digits as the order of first parent as shown in fig. 4.

After crossover, some keys are subjected to mutation. Mutation prevents the algorithm to be trapped in a local minimum. The mutation operation used in this cipher randomly selects two elements in the child and swapa those elements as shows in Fig. 5.



Generate random number from 1 to key size for each pair and swap as:

Parent 1:	**1 6 5 7 10 3**	2 4 8 9
Parent 2:	**7 4 10 9 6 1**	2 5 8 3

Child1:	**1 6 5 7 10 3**	4 9 2 8
Child 2:	**7 4 10 9 6 1**	5 3 2 8

Fig. 4: Applying Crossover

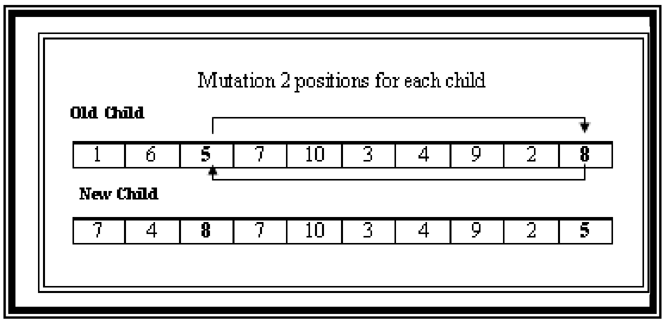


Fig. 5: Applying Mutation

6. RESULTS

The attack to transposition cipher was implemented for different numbers of populations, different numbers of generations and two equations (3 and 4).

$$F_{key}=1-[\beta\sum_{i,j\in A}|k^{b}_{i,j}-D^{b}_{i,j}|+\gamma\sum_{i,j,k\in A}|k^{t}_{i,j,k}-D^{t}_{i,j,k}|]\cdots(3)$$

$$F_L=\sum^{Q}(PiSi)\cdots\cdots(4)$$

In this method different population sizes and different lengths of key have been used for mutation 0.2 and two different equations were used to calculate the fitness value.

First: Using equation 3

Figure 6 shows the relation between the population size (10, 20,30 40 and 50) and number of generations for a key size of 10 letters. It is clear that the best number of correct letters is reached after 300 generation. Table 3 shows the number of correct letters for population size (10, 20, 30, 40 and 50). The number of correct letters is 7 for population size 30 and 40 which represents the best solution after 300 generations.

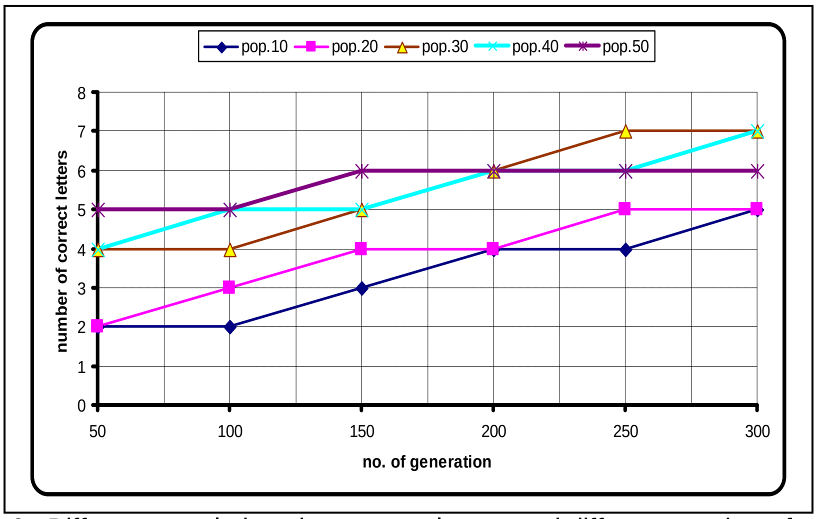


Fig. 6 : Different population size, correct letters and different number of generation

Bold lettering is used in table 6 for those letters in the population that appeared to be correct.
The true key is:
9 7 8 2 4 3 5 1 10 6

Figure 7 shows the relation between the population size and fitness values for 300 generations. It is clear that the best fitness value was 43.3767 for the number of population 20. Table 4 shows the fitness values for population size (10, 20, 30, 40 and 50).

Table 3: number of correct letters

population sizes	key										No. of correct letters
10	1	**7**	**8**	**2**	**4**	5	9	6	**10**	3	5
20	8	**7**	10	**2**	**4**	**3**	**5**	6	9	1	5
30	**9**	**7**	10	**2**	**4**	**3**	**5**	**1**	6	8	7
40	**9**	**7**	6	**2**	**4**	**3**	**5**	**1**	8	10	7
50	**9**	**7**	4	**2**	8	**3**	**5**	6	**10**	1	6

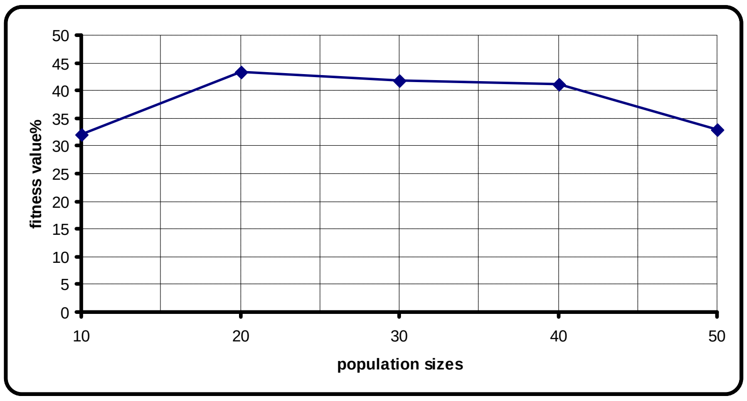


Table 4: Fitness values for different number of population, after 300 generation

No. of population	Fitness value %
10	32.1367
20	43.3767
30	41.7633
40	41.09
50	32.8633

Fig. 7: fitness values for different population size after 300 generation.

Figure 8 shows the time required (elapsed) to finish the algorithm for different number of generations and for different number of populations. It is clear that the time required is increased as the number of populations is increased.

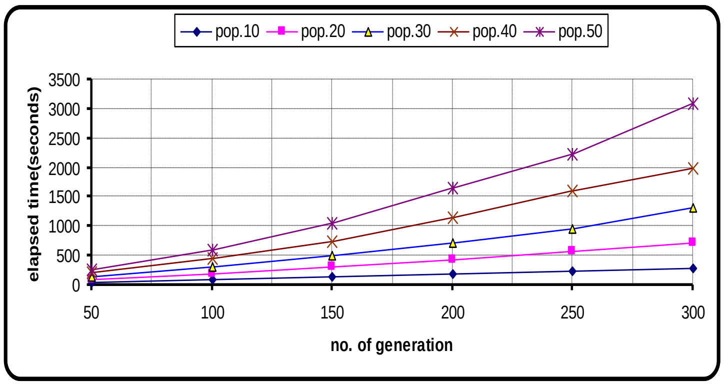


Fig. 8: Elapsed time for different number of population

Second: By using equation 4

In the previous method, the best solution was 7 letters from 10. It has depended on table 1 of bi-gram frequency letters of 10000 letters in the English language but this solution is not enough. Another equation is used (equation 3) to calculate the fitness value, where it depends on the frequency of the more frequency bigram and trigram which were shown in table 2. The single frequency letters have been ignored because they have no effect on the solution. Fig. 9 shows the relation between the number of generations and the number of correct letters for population size 10 and 20. It's clear that after 250 and 300 generation the number of letters has reached to 10 correct letters for population size 10. When the population size has increased to 20, the number of correct letters are 10 after 300 generations. Table 8 shows the number of correct letters. For different number of generations. Table 6 shows the number of correct letters for the number of population 10 and 20.

Table 6 shows the fitness values for the number of population 10 and 20.

The second equation gave better solution than the first one.

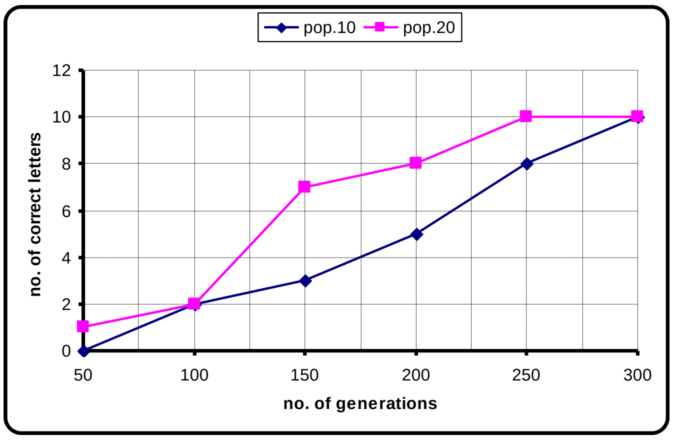


Fig. 9: Correct letters for population size 10 and 20 and different number of generation

Table 5: Number of correct letters for the number of population 20 and 40, after 300 generations

Number of generation	No. of correct letters	
	pop10	pop20
50	0	1
100	2	2
150	3	7
200	5	8
250	8	10
300	10	10

Table 6: Fitness values for the number population 20 and 40, after 300 generations

Number of generation	**Fitness value**	
	pop10	**pop20**
50	165	318
100	166	519
150	215	347
200	626	472
250	406	872
300	872	872

Figure 10 shows the time required (elapsed) to finish the algorithm for different number of population. The time required is increased as the number of population is increased.

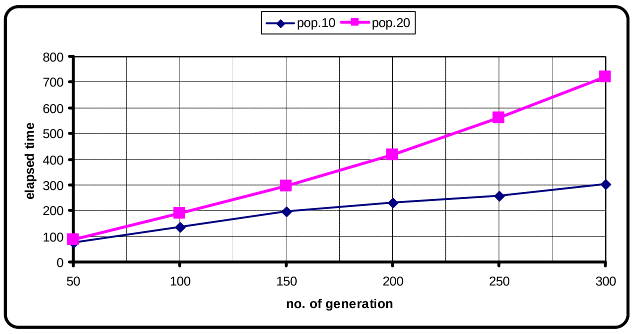


Fig. 10: Elapsed time for different number of generation

Table 7: Fitness values for different lengths of key, after 300 generation

Key length	Fitness value
5	295
6	131
8	108
10	872
12	106
15	33

To find the key size, the fitness value was measured for all possible key sizes ranging between 4 and 15. For example the possible key size for 3000 text letters are (5, 6, 8, 10, 12 and 15).

Table 7 shows fitness values for the possible key size of 3000 text letters. it is clear that the highest fitness value is obtained when the key size was 10.

7. CONCLUSIONS

1. Two different equations were used to calculate the fitness function to see which one gives best solution.
2. The first equation depends on the letter frequency of bigram
3. The algorithm was run for population size of 10, 20, 30, 40 and 50, and for a key size of 10 letters. The best number of correct letters was 7 from 10 letters after 300 generation for population size 30 and 40 which represent the best solution.
4. Another equation was used which depends on score table. In this equation, bi-grams and trigrams were used to represent the most ten bi-grams and trigrams letters.
5. The second equation to calculate the fitness value gives better results than the previous one. In this equation, 10 correct letters out of 10 was obtained. Then the algorithm was also run for different numbers of population, different size keys, different numbers of generations, and the text length of 3000 letters.
6. In both equations, the single letters frequency has been ignored because in this method, the number of each letter in the cipher text is the same as it is in the plain text. So it is not necessary to calculate single letters frequency.
7. The second equation gave the optimal solution, so different key sizes have been used to test which length is correct. For the 3000 letters, the key sizes are (5, 6, 8, 10, 12 and 15) and the high fitness has key size 10.
8. The elapsed time for 2nd equation was less than for the 1st equation.
9. The algorithms were run in MATLAB 2008 for pentuime 4, the processor 1.7GHz and RAM 512MB.

It is apparent that genetic algorithm attack on simple cryptographic ciphers called transposition were implemented successfully.

8. REFERENCES

[1] Menezes, A., P.V. Oorschot, and S.Vanstone, *"Handbook of Applied Cryptography"*, Boca Raton: CRC Press, 1996.
[2] Jan Pelzl,C.P., *"Understanding Cryptography"*, Springer, 2010.
[3] Stinson, D.R., *"Cryptography Theory and Practice"*, 3rd Edition, Chapman and Hall/CRC, 2006.
[4] Denning, D.E., *"Cryptography and data security"*, Addison-Wesley Publishing Company, Reading, 1982.
[5] Mathews, R.A.J., *"The Use Of Genetic Algorithms In Cryptanalysis",* Cryptologia **Vol. (17) No.**(4), p. 187-201, 1993.

[6] Clark, A., *"Modern Optimisation Algorithms For Cryptanalysis*", in *Proceedings of the Second Australian and New Zealand Conference on Intelligent Information System IEEE(ANZII)*. p.258-262, Australian, 1994.

[7] Grundlingh, W.R. and J.H.V. Vuuren., *"Using Genetic Algorithms to break a simple cryptographic cipher"*, from http:// dip.sun.ac.za/ ~vuuren/abstracts/abstr_genetic.htm, 2003. The last visit is in (20-Feb-2011).

[8] Toemeh, R and S.Arumugam., *"Breaking Transposition Cipher with Genetic Algorithm*", Electronics and Elecrical engineering **7**(79): p. 75-78, 2007.

[9] Clark, A.J., *Optimization Heuristics for Cryptology. Thesis PhD*, in *Information Security Research Centre, Faculty of Information Technology, Queensland University of Technology*. February 1998.

[10] Stallings, W., "*Cryptography And Network Security, Principle And Practices*", 3rd Edition, Pearson Education, 2005.

[11] Rolf,O., "Contemporary Cryptography" Artech House Computer Security Series, Boston-London, 2005.

[12] Bergamann, K.P., *"Cryptanalysis Using Nature-Inspired Optimization Algorithms*", Msc. Thesis, *Department of computer science*. The University Of Calgary: Galgary, Alberta, August 2007.

[13] Holland, J., *"Adaptation in natural and artificial systems",* University of Michigan Press, Ann, Arbor, 1975.

[14] Sivanandam, S.N. and S.N. Deepa, *Introduction to Genetic Algorithms*. 2008: Springer.

[15] Erdogmus.P., A. Ozturk and S. Tosun, *Continious Optimization Problem Solution With Simulated Annealing and Genetic Algorithm*, in *5th International Advanced Technologies Symposium (IATS'09), May 13-15*. 2009, karabuk, Turkey

[16] Genetic Algorithms and Direct Search Toolbox^{TM} 2 , User's Guide , MATLAB. 2009.

[17] Chipperfield, A., A. Fleming, H. Pohlheim, and C. Fonseca, *"Genetic Algorithm TOOLBOX For Use With MATLAB*", Automatic Control & Systems Engineering - University Of Sheffielid, 2008.

[18] Delman, B., *Genetic Algorithms in Cryptography, Msc Thesis*, in *Computer Engineering ,Rochester Institute of Technology , Rochester, New York*. July 2004.

[19] Verma, A.K., M. Dave, and R.C. Joshi, *Genetic Algorithm and Tabu Search Attack on the Mono-Alphabetic Substitution Cipher in Adhoc Networks.* Journal of computer Science 2007. **3**(3): p. 134-137.

[20] Brownlee, J., "*Clever Algorithms, Nature-Inspired Programming Recipes"* , 1st edition, Jason Brownlee, 2011

[21] Melanie, M., *"An Introduction to Genetic Algorithms*". 5th Edition, Cambridge, Massachusetts • London, England: The MIT Press, 1999.

[22] Haupt, R.L. and S.E. Haupt, *"Practical genetic algorithms",* 2nd Edition, John wiley & Sons, INC, 2004.

[23] Fogel, G.B and D.W.Corne., *"Evolutionary Computation in Bioinformatics"*, Morgan Kaufmann publisher, 2003.

[24] Singiresin, S.R., *"Engineering Optimization, Theory and Practice"*, 3rd Edition, John Wiley and Sons, Inc. 1996.