Evolutionary Operators-Based Particle Swarm Optimization (EOPSO) to Attack Classical Cryptography Methods

Ahmed Tariq Sadiq
Computer Sciences Department, University of Technology, Baghdad, Iraq
E-Mail : drahmaed_tark@yahoo.com
Received 2/2/2014 – Accepted 23/2/2014

الخلاصة

تعتبر أمثلية حشد الجزيئات من طرق الامثلية المعتمدة على المجتمع، والتي تعتبر سهلة التنفيذ والتطبيق لحل مشاكل الامثلية وبعض مشاكل من نوع NP-Complete. هذا البحث يقدم أمثلية حشد الجزيئات المطورة باستخدام عمليتين من العمليات التطويرية وهما : التضارب والطفرة، لذلك سميت الخوارزمية المقترحة بأمثلية حشد الجزيئات المعتمدة على العمليات التطويرية (EOPSO). الفائدة من العمليتين لأمثلية حشد الجزيئات هي اعطاء زخم وتنوع للمجتمع. الخوارزمية المقترحة تهاجم نوعين من التشفير التقليدي (التعويضي والابدالي). نتائج التجارب اظهرت ان نسبة استرجاع المفتاح باستخدام الخوارزمية المقترحة أفضل من الاصلية (PSO) وأفضل من خوارزميات مطورة مثل (2-op PSO) و (SAPSO).

ABSTRACT

Particle Swarm Optimization (PSO) is a population-based optimization tool, which could be implemented and applied easily to solve various function optimization problems and some NP-complete problems. This paper present a benefit developed PSO using two evolutionary operators: crossover and mutation, so it called Evolutionary Operators-based PSO (EOPSO). The benefit of these two operators in PSO is use as momentum and diversity tool in the population. EOPSO used to attack the two types of classical cryptography (substitution and transposition). Experimental results of EOPSO appear that the amount of recovered key of classical ciphers and fitness function values are best than with PSO, improved 2-opt PSO and simulated annealing PSO.

Keywords: PSO, Evolutionary Operators, Crossover, Mutation, Attack Classical Cryptography.

INTRODUCTION

Two fundamental goals in computer science are finding algorithms with provably good run times and with provably good or optimal solution quality. A heuristic is an algorithm that gives up one or both of these goals; for example, it usually finds pretty good solutions, but there is no proof the solutions could not get arbitrarily bad; or it usually runs reasonably quickly, but there is no argument that this will always be the case. Often, one can find specially crafted problem instances where the heuristic will in fact produce very bad results or run very slowly; however, these instances might never occur in practice because of their special structure. Therefore, the use of heuristics is very common in real world implementations [1].

Population-based metaheuristics share many common concepts. They could be viewed as an iterative improvement in a population of solutions. First, the population is initialized. Then, a new population of solutions is generated. Finally, this new population is integrated into the current one using some selection procedures. The search process is stopped when a given condition is satisfied (stopping criterion). Algorithms such as

evolutionary algorithms (EAs), genetic algorithms (GA), scatter search (SS), estimation of distribution algorithms (EDAs), particle swarm optimization (PSO), bee colony (BC), and artificial immune systems (AISs) belong to this class of metaheuristics [2].

In the other side, cryptanalysis can be described as the process of searching for flaws or oversights in the design of cryptosystems (or ciphers). A typical cipher takes a clear text message (known as the plaintext) and some secret keying data (known as the key), as its input and produces a scrambled (or encrypted) version of the original message (known as the ciphertext) [6]. The use of automated techniques in the cryptanalysis of cryptosystems is desirable as it removes the need for time-consuming (human) interaction with a search process. Making use of computing technology also allows the inclusion of complex analysis techniques, which can quickly be applied to a large number of potential solutions in order to weed out unworthy candidates [7].

This paper presents an attempting to use meta-heuristic techniques in the attacking of classical cipher with two type substitution and transposition. Section 2 includes the related works. Evolutionary operators types in section 3. Section 4 presents a brief overview of the classical cipher. Section 5 presents the principle of particle swarm optimization (PSO). Section 6 contains the proposed EOPSO to attacks the classical cipher. Section 7 includes the results of computational tests to evaluate the performance of the PSO and EOPSO algorithms. The conclusions are reported in section 8.

LITERATURE REVIEWS

In [20], two new variants of Particle Swarm Optimization (PSO) called AMPSO1 and AMPSO2 are proposed for global optimization problems. Both the algorithms use adaptive mutation using Beta distribution. AMPSO1 mutates the personal best position of the swarm and AMPSO2, mutates the global best swarm position. The performance of proposed algorithms is evaluated on twelve unconstrained test problems and three real life constrained problems taken from the field of Electrical Engineering. The numerical results show the competence of the proposed algorithms with respect some other contemporary techniques.

In [21], an improved algorithm for Particle Swarm Optimization (PSO) named Elite Particle Swarm Optimization with Mutation (EPSOM) is proposed in this paper. Elite particles and bad particles are distinguished from the swarm after some initial iteration steps. Bad particles are replaced with the same number of elite particles, and a new swarm is generated. To avoid losing diversity of the swarm and to decrease the risk of trapping in local optimum, mutation operation is introduced in evolution process. The results of several simulations for

different benchmark functions illustrate that EPSOM algorithm has the ability of local exploitation and global exploration. EPSOM algorithm outperforms the Linearly Decreasing Weight Particle Swarm Optimization (LDW-PSO) and Random Mutation Particle Swarm Optimization (RM-PSO) in respects of calculation accuracy and convergence.

In [22], propose an advanced PSO algorithm with mutation operator. By adding the mutation operator to the algorithm, the advanced algorithm can not only escape from the local minimum's basin of attraction of the later phase, but also maintain the characteristic of fast speed in the early convergence phase. By the contrast experiments of three multimodal test functions and an example whose problem space is non-convex set, it has been proved that the advanced PSO algorithm can improve the global convergence ability, greatly enhance the rate of convergence and overcome the shortcoming of basic PSO algorithm.

In [23], with high dimension of solution vectors, PSO also gets into trouble for finding optimal solutions. With mutation mechanism frequently used in GA into PSO algorithms, it shows improved performance for finding solution with high dimension optimization problems can be achieved compared with PSO without mutation.

In [43], MPSO have been proposed to attack the classical cryptography, the mutation operator used to enhance the performance of PSO to find the largest amount of classical cryptography keys. There are good results are obtained because mutation operator provides diversity to the PSO candidate solutions.

EVOLUTIONARY OPERATORS

Genetic algorithms have been developed by J. Holland in the 1970s (University of Michigan, USA) to understand the adaptive processes of natural systems [3]. Then, they have been applied to optimization and machine learning in the 1980s [4, 5]. GAs are a very popular class of EAs. Traditionally, GAs are associated with the use of a binary representation but nowadays one can find GAs that use other types of representations. A GA usually applies a crossover operator to two solutions that plays a major role, plus a mutation operator that randomly modifies the individual contents to promote diversity. GAs uses a probabilistic selection that is originally the proportional selection. The replacement (survivor selection) is generational, that is, the parents are replaced systematically by the offsprings. The crossover operator is based on the *n*-point or uniform crossover while the mutation is bit flipping [2].

The role of crossover operators is to inherit some characteristics of the two parents to generate the offsprings. As for the mutation operator, the design of crossover operators mainly depends on the representation used.

Unlike single-metaheuristics, here the search operators are always unary, this part has been independently developed in the evolutionary computation community. Some important points must be taken into account in the design or use of a crossover operator:

• Heritability: The main characteristic of the crossover operator is *heritability*. The crossover operator should inherit a genetic material from both parents. An operator is a pure recombination operator (strong heritability) if two identical individuals generate identical offsprings. Hence, the mutation and crossover operators are complementary. In this case, mutation introduces some diversification in the individuals by introducing some missing values in the current individuals of a population. A crossover operator Ox is *respectful* if the common decisions in both parents are preserved, and is *assorting* if the distance between the parents (p1,p2) and the offspring o is lower or equal to the distance between the parents [8]:

$$d(p1, o) \leq d(p1, p2) \text{ and } d(p2, o) \leq d(p1, p2), \forall o \in O(p1, p2, Ox)$$

where O(p1, p2, Ox) is the set of all possible offsprings generated by the crossover operator Ox.

• Validity: The crossover operator should produce valid solutions. This is not always possible for constrained optimization problems [2].

The crossover rate pc ($pc \in [0, 1]$) represents the proportion of parents on which a crossover operator will act. The best parameter value for pc is related to other parameters among them such as the population size, the mutation probability, and the selection procedure. The most commonly used rates are in the interval [0.45, 0.95]. Adaptive techniques for the crossover rate may also be useful. For linear representations excluding permutations, the well-known crossover operators are the 1-point crossover, its generalized form the *n*-point crossover, and the uniform crossover.

For real-valued representations, in addition to the 1-point, n-point, and the uniform crossover operators, the following two types of crossover operators are commonly used: mean-centric recombination and parent-centric recombination. In mean-centric recombination, the offsprings are generated closer to the centroid of their parents. One may use the following crossover operators [9, 10, 11, 12, 13, 14, 15]:

• Intermediate crossover.
• Geometrical crossover.
• Unimodal normal distribution crossover (UNDX).
• Simplex crossover (SPX).
• Simulated binary crossover (SBX).
• Parent-centric crossover (PCX).

Applying classical crossover operators to permutations will generate solutions that are not permutations (i.e., nonfeasible solutions). Hence,

many permutation crossover operators have been designed as, Order crossover (OX), Partially mapped crossover (PMX), maximum preservative crossover [16], cycle crossover (CX) that preserves the absolute positions of the elements [17], merge crossover [18], and position-based crossover (POS) [19].

Mutation operators are unary operators acting on a single individual. Mutations represent small changes of selected individuals of the population. The probability pm defines the probability to mutate each element (gene) of the representation. It could also affect only one gene too. In general, small values are recommended for this probability ($pm \in [0.001, 0.1]$). Some strategies initialize the mutation probability to 1/k where k is the number of decision variables, that is, in average only one variable is mutated [2].

Some important points that must be taken into account in the design or use of a mutation operator are as follows [2]:

• **Ergodicity:** The mutation operator should allow every solution of the search space to be reached.

• **Validity:** The mutation operator should produce valid solutions. This is not always possible for constrained optimization problems.

• **Locality:** The mutation should produce a minimal change. The size of mutation is important and should be controllable. Locality is the effect on the solution (phenotype) when performing the move (perturbation) in the representation (genotype). When small changes are made in the genotype, the phenotype must reveal small changes. In this case, the mutation is said to have a strong locality. Hence, an evolutionary algorithm will carry out a meaningful search in the landscape of the problem. Weak locality is characterized by a large effect on the phenotype when a small change is made in the genotype. In the extreme case, the search will converge toward a random search in the landscape.

There are several types of mutation representation, such as [2]:

• **Mutation in binary representation:** The commonly used mutation is defined as the flip operator.

• **Mutation in discrete representation:** It consists generally in changing the value associated with an element by another value of the alphabet.

• **Mutation in permutations:** Mutation in order-based representations are generally based on the *swapping*, *inversion*, or the *insertion* operators.

CLASSICAL CIPHERS

Generally, there are two types of classical cipher methods, substitution and transposition as below.

1. Substitution Cipher

The simple substitution cipher (sometimes referred to as the monoalphabetic substitution cipher to distinguish it from the

polyalphabetic substitution cipher). Each symbol in the plaintext maps to a (usually different) symbol in the ciphertext. The processes of encryption and decryption are best described with an example, such as the one following.

For example, a simple substitution cipher key can be represented as a permutation of the plaintext alphabet. Table 1 gives a sample key. Using this representation the ith letter in the plaintext alphabet is encrypted to the i^{th} element of the key. The string "the money is in the bag" is encrypted using the key in the Table 1.

Table-1: Example Substitution Cipher

KEY	
Plaintext	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ciphertext	PQOWIEURYTLAKSJDHFGMZNXBCV
ENCRYPTION	
Plaintext	T H E M O N E Y I S I N T H E B A G
Ciphertext	M R I K J S I C Y G Y S M R I Q P U

Using the key representation in Table 1, the original message can be discovered by reversing the encryption procedure. That is, the ciphertext character at position I in the key decrypts to the i^{th} character in the plaintext alphabet. For an alphabet of 26 characters, there are 26! or greater than 4×10^{26} possible keys for a simple substitution cipher. This number is far too large to allow a brute force attack even on the fastest of today's computers. However, because of the properties of the simple substitution cipher they are relatively easy to cryptanalysis [24, 25].

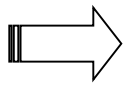
2. Transposition Cipher

A transposition cipher [1], also sometimes called a permutation cipher, is one for which applying *E* to plaintext produces ciphertext with the same symbols as the plaintext, but rearranged in different positions. We must divide the plaintext message into message blocks. The size of the permutation is known as the period. Let us consider an example of a transposition cipher with a period of five, and a key {4, 2, 1, 5, 3}. In this case, the message is broken into blocks of five characters, and after encryption the fourth character in the block will be moved to position 1, the second remains in position 2, the first is moved to position 3, the fifth to position 4, and the third to position 5. Figure. 1 illustrates an example for this type.

Plaintext : "MYPSOALGORITHMSAREANEFFICIENT"

Encrypted using the key (32514)

M	Y	P	S	O
A	L	G	O	R
I	T	H	M	S
A	R	E	A	N



P	Y	O	M	S
G	L	R	A	O
H	T	S	I	M
E	R	N	A	A

E	F	F	I	C		F	F	C	E	I
I	E	N	T	X		N	E	X	I	T

Ciphertext: "PGHEFN YLTRFE ORSNCX MAIAEI SOMAIT"

Figure-1: An Example for Transportation Cryptography

PARTICLE SWARM OPTIMIZATION (PSO)

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995 [26], inspired by social behavior of bird flocking or fish schooling and swarm theory. We have used this technique in the cryptanalysis of simple transposition ciphers.

1. The Principle

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. Compared to GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust. PSO has been successfully applied in many areas; function optimization, artificial neural network training and fuzzy system control. PSO simulates the behavior of bird flocking, suppose a group of birds are randomly searching food in an area, not all the birds know where the food is. The effective strategy is to follow the bird that is nearest to the food. In PSO, each single solution is a "bird" in the search space. We call it "particle". All particles have fitness values that are evaluated by the fitness function to be optimized, and have velocities, which direct the flying of the particles [26].

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generation. In each generation, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. This value is called (pbest), another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and is called gbest. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called (*l*best).

After finding the two best values, the particle updates its velocity and position with the following equations [26]:

$$V_i[t+1]=wV_i[t]+C_1*r1*(pbest_i[t]-present_i[t])+C_2*r2*(gbest_i[t]-present_i[t]) \quad (1\text{-}a)$$

$$present_i[t]=present_i[t]+V_i[t] \quad (1\text{-}b)$$

where, i =1,2,…..,*N*; *w* is the inertia weight, V[t] is the particle's velocity, present[t] is the current particle (solution), pbest and gbest are defined as stated before, r1 and r2 are two random numbers between (0,1), C_1 and C_2 are learning factors. However these values of C_1,C_2 are problem dependent. These are very essential parameters in PSO. They particles' velocities on each dimension are clamped to a maximum velocity Vmax [14].

2. Binary PSO

The original PSO is for continuous population, but is later extended by Eberhart and Kennedy [27] to the discrete valued population. In the binary PSO thus emerged, the particles are represented by binary values (0 or 1). The velocity and particle updating for binary PSO are the same as in the case of continuous one. However, the final decisions are made in terms of 0 or 1. Sigmoid function in [28] is used to restrict the decision in the range [0,1].

3. Discrete PSO

Several adaptations of the method to discrete problems, known as Discrete Particle Swarm Optimization (DPSO). Since, in words of the inventors of PSO, it is not possible to "throw to fly" particles in a discrete space [26], several Discrete Particle Swarm Optimization (DPSO) methods have been proposed.

A DPSO whose particles at each iteration are affected alternatively by its best position and the best position among its neighbors was proposed by Al-kazemi and Mohan [29]. Pampara et al. [30] solved binary problems by combining continuous PSO and Angle Modulation with only four parameters. Furthermore, several PSO variants applied to problems, where the solutions are permutations were considered in [31, 32]. The multi-valued PSO (MVPSO) proposed by Pugh and Martinoli [33] deals with variables with multiple discrete values.

Another DPSO was proposed in [34] for feature selection problems, which are problems their solutions are sets of items. A new DPSO proposed in [35, 36] does not consider any velocity since, from the lack of continuity of the movement in a discrete space, the notion of velocity loses sense; however kept the attraction of the best positions.

4. Parameters of PSO

The convergence and performance of PSO are largely dependent upon chosen parameters. *w* is termed as inertia weight [28] and is incorporated in the algorithm to control the effect of the previous velocity vector of the swarm on the new one. It facilitates the trade-off between the local and the global exploration abilities of the swarm and may result in less number of iterations of the algorithm while searching for an optimal

solution. It is experimentally found that inertia weight *w* in the range [0.8, 1.2] yields a better performance [37]. The velocity lies in the range [- Vmax , Vmax] where, - Vmax denotes the lower range and Vmax is the upper range of the motion of the particle. The roles of C_1 and C_2 are not so critical in the convergence of PSO, however, a suitably chosen and fine tuned value can lead to a faster convergence of the algorithm. A default value of $C_1 = C_2 = 2$ is suggested for general purpose, but somewhat better results are found with $C_1 = C_2 = 0.5$ [38]. However, the values of cognitive parameter, C_1 larger than the social parameter C_2 are preferred from the performance point of view with the constraint $C_1 + C_2 \leq 4$ [39]. The parameters r1 and r2 used to maintain the diversity of the population in equation (1-a).

EOPSO FOR ATTACKING CLASSICAL CIPHERS

PSO is rather a successful method for the continuous optimization problems; however, it is very difficult to adapt it for the discrete case; many studies have been done on it. The focus of this paper is to apply a PSO algorithm to the problem of searching through the key space of a simple transposition cipher. The type of transposition ciphers, which want to be attacked, encrypts text according to the following classical two-stage algorithm:

- A key of length *N* takes the form of a permutation of the integers 1 to *N*. The plaintext, of L characters, is written beneath the key to form a matrix *N* characters wide and at least L mod *N* characters deep.
- The text is then enciphered by reading it off in columns in the order dictated by the integers making up the key.

A. Representation of a Candidate Solution

Since 1995, Particle Swarm Optimization has been successfully applied in continuous problems, and a transposition cipher is the new field of PSO algorithm in discrete space. If the PSO algorithm is designed to tackle discrete problem, a direct correlation must be found between the particle vector and the solution representation of classical cipher.

To attack the classical ciphers using PSO we must define the position vector of a particle represents a feasible solution, that is to say, the position vector means the permutation of the solution key and specifies the order of key to be executed. The vector elements must be encoded as integer limited in [1, *N*], and *N* is the key size (especially in transposition). In addition to, in substitution a key for decoding a cipher is given by an ordered list of the 26 letters of the alphabet. The order expresses the underlying substitution. Every integer in the scale appear only once in feasible solution.

B. Initial Population

Like other evolutionary algorithms, the PSO algorithm starts from an initial population. The particles are randomly generated between the maximum and the minimum operating limits of the generators.

C. Fitness Function Assessment

Fitness function values of the particles are evaluated using equation (3). These values are set as the pbest value of the particles. The fitness rating helps the transposition cryptanalysis algorithm achieve breaking by awarding scores according to the number of times two and three letter combinations (bi-and tri-grams) commonly found in English actually occur in the decrypted text. The more columns correctly put next to the other by this algorithm, the higher the fitness rating ascribed to that trail permutation. Whenever these combinations were found in decrypted text, points were awarded, the highest being given for the appearance of trigrams, as their appearance suggested that three columns have been correctly aligned. A study has been made in English text consists of more than 4500 letters, taken the 29 high frequency diagram letters and 12 high frequency trigram letters in Table 2 gives 41 combinations letters and their percentage frequency in standard English text. Were the percent calculated by :

Let frq[D_i] denotes the frequency of the bi-gram letters, where $D_i \in \{TH,HE,\ldots,HA\}$, where $1 \le i \le 29$, and frq[T_j] denotes the frequency of the trigram letters, and $T_j \in \{ARE,THE,\ldots,DTH\}$, where $1 \le j \le 12$, then, percent(bi-gram) = frq[D_i] / (L-1) and percent(tri-gram) = frq[T_j] / (L-2), (we use (L-1) and (L-2) because the diagram and trigram used with overlap), and

L: is the text length, then:

$$\textit{Total percent (bi-gram)} = \sum_{i=1}^{29} \text{frq}[D_i] / \textit{(L-1)} \qquad \textit{(2-a)}$$

and,

$$\textit{Total percent (tri-gram)} = \sum_{j=1}^{12} \text{frq}[T_j] / \textit{(L-2)} \qquad \textit{(2-b)}$$

Therefore, the total percent will represent the fitness value of the transposition cryptanalysis :

$$\textbf{\textit{Fitness}} = \sum_{i=1}^{29} \text{frq}[D_i] / \textbf{\textit{(L-1)}} \sum_{j=1}^{12} \text{frq}[T_j] / \textbf{\textit{(L-2)}} \qquad \textbf{\textit{(3)}}$$

It is clear that the other non-plaintext fitness values ≤ 0.3811568, and it is important to mention that, the text fitness increased as the text length increased, and vice versa.

The most commonly used fitness function for breaking the simple substitution ciphers is based only on unigram and bigram analysis and has the form:

$$\textbf{\textit{Fitness}}=(1-\sum_{i=1}^{26}\left\{\left|SF[i]-DF[i]\right|+\sum_{j=1}^{26}\left|SDF[i,j]-DDF[i,j]\right|\right\}/4)^{8} \quad (4)$$

where, $SF[i]$ and $SDF[i, j]$ are standard frequencies of character i and the pair of symbols (i, j) in the original language, respectively, and $DF[i]$ and $DDF[i, j]$ are measured frequencies of character i and pair (i, j) in the "decrypted" text. The measured errors (i.e., the difference between standard frequencies and measured frequencies) are normalized and subtracted from 1, so that number closer to 1 represents the higher fitness. In addition, the constants 4 and 8 are used to reduce sensitivity to large errors and to amplify small differences, respectively. Some fitness functions give more weight to one of the factors (unigram or bigram) by inserting weight constants in front of the corresponding error calculation terms [1, 40, 41, 42].

Table-2: 29 Bi-Gram and 12 Tri-Gram Letters Combinations

29 Diagram Letters				12 Trigram Letters	
Diagram	Percent	Diagram	Percent	Trigram	Percent
TH	0.0250791	EA	0.0079078	ARE	0.0006778
HE	0.0237235	NG	0.0103931	THE	0.0153638
IN	0.0171713	AS	0.0112969	ING	0.0085856
AN	0.0153638	OR	0.0117488	AND	0.0085856
RE	0.0115228	TI	0.0092634	ENT	0.0042928
ED	0.0092634	IS	0.0092634	THA	0.0033891
ON	0.0108450	ET	0.0067781	NTH	0.0027113
ES	0.0131044	IT	0.0040669	WAS	0.0045188
ST	0.0151378	AR	0.0056484	HER	0.0011297
EN	0.0108450	TE	0.0070041	ETH	0.0038409
AT	0.0131044	SE	0.0079078	FOR	0.0020334
TO	0.0101672	HI	0.0063263	DTH	0.0009038
NT	0.0099413	OF	0.0074559	--------	---------
ND	0.0122006	HA	0.0079078	--------	---------
OU	0.0112969	---------	----------	---------	----------
Di-Total	Percent = 0.3246724			Tri-Total	0.0564844
Total	Percent = 0.3811568 = fitness of plaintext				

D. The Proposed Algorithm of EOPSO

The following is an algorithmic description of the attack on a classical cipher using particle swarm optimization (PSO):

1. The algorithm is given the cipher text (and its length), the key size (permutation size or period) N.
2. Initialize the algorithm parameters. They are:

a. C_1 (Self-confidence).
b. C_2 (Swarm confidence).
c. W (Inertia weight).
d. Pc (Crossover Probability).
e. Pm (Mutation Probability).
f. V_{max} (The maximum of velocity).
g. Swarm_Size (Number of particles in the swarm.)
h. Max-Iter (The maximum number of iterations).

3. Randomly generate the initial particles (keys of the cipher) to form a swarm.
4. For i=1 To Max-Iter, do
   a. Calculate the fitness function of each of the particles (keys) using equation (2 and 3).
   b. If the current position of the particle is better than the previous history, updating the particles indicated this fact.
   c. Find the best particle of the swarm. Updating the positions of the particles by using equation (1a and 1b).
   d. If no update or no regression then recombine using crossover then mutates the two best particles (depend on Pc and Pm respectively).
5. Copy the best key obtained so far in the output key variable and exit.

RESULTS AND DISCUSSION

In this paper 2 type of classical cipher have been taken, substitution and transposition. So the results will divide into these 2 types. The results of EOPSO compare with PSO, 2-op PSO and SAPSO [1, 38].

1. Experimental Results for Substitution Cipher

EOPSO algorithm was applied to cipher text created using a simple substitution cipher and the attack was run a number of times with a variety of parameter values. In general it was found that (200) iterations were usually enough to break the cipher text and the algorithm was fast enough that this took a few seconds.

The experiments shown in Figure. 2 were performed on simple substitution cipher using basic PSO and improved PSO algorithm. The fitness function used was in equation (4). As we can see, the fitness value has started from about 0.51 and come up to 0.89 in about 200 iterations using EOPSO. By using PSO, 2-op PSO and SAPSO the fitness value was less than the EOPSO. In addition, we can see a rapid increase in fitness function at the beginning and the rate of increases as we run the experiments for longer period. This is because as the fitness value increases, it becomes more and more difficult to find a better key.

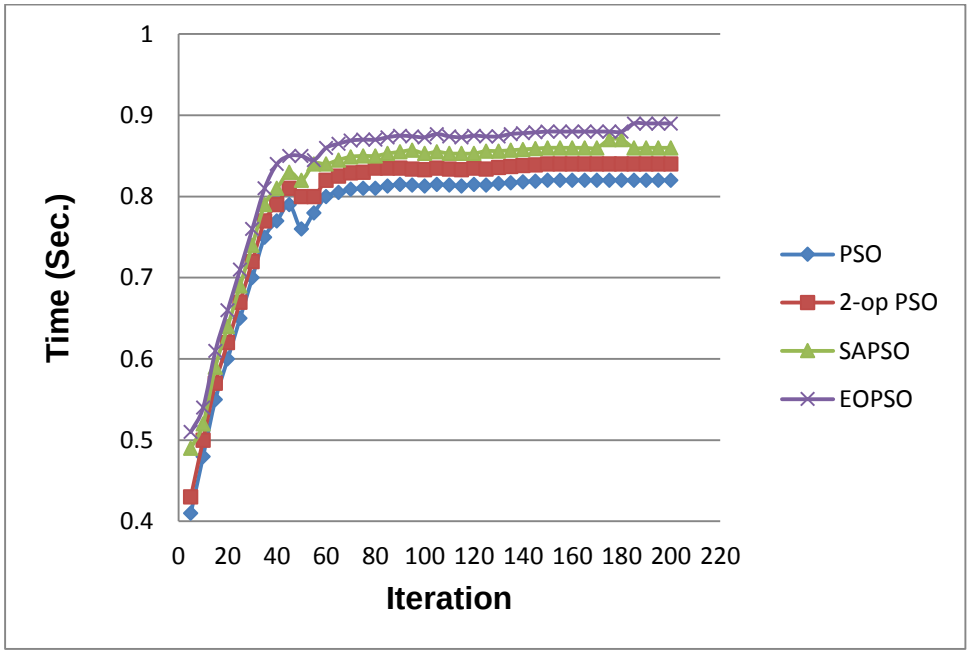


Figure-2: Performance of PSO, 2-op PSO, SAPSO and EOPSO for Substitution Attack

The best parameters value of PSO, 2-op PSO, SAPSO [1] and EOPSO to attack substitution cipher show in Table 3.

Table -3: Parameters Value of PSO, 2-op PSO, SAPSO and EOPSO to Attack Substitution Cipher

Parameter	Symbol	Value
Self-confidence	C_1	2
Swarm confidence	C_2	2
Inertia weight	W	1.2
Crossover Probability	Pc	0.65
Mutation Probability	Pm	0.05
The maximum of velocity	V_{max}	26
Number of particles in the swarm	Swarm_Size	25-50
The maximum number of iterations	Max-Iter	300

Figure. 3 illustrate the time comparison for EOPSO, PSO, 2-op PSO and SAPSO to attack substitution cipher.

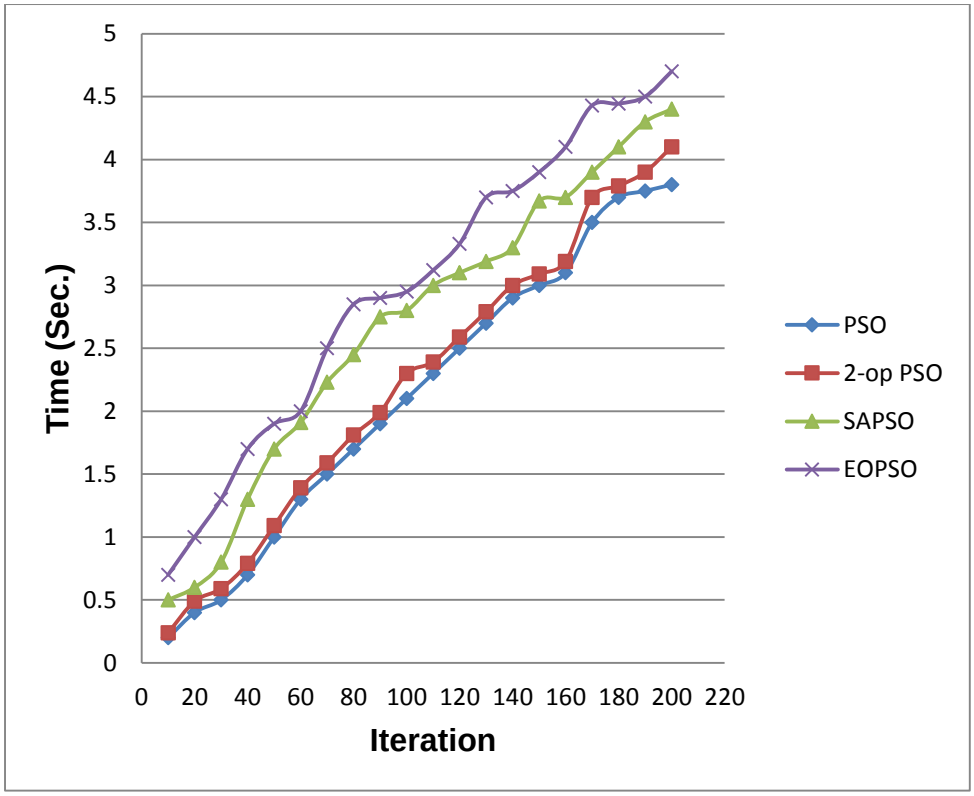


Figure-3: Curve of Time for PSO, 2-op PSO, SAPSO and EOPSO for Substitution Attack

2. Experimental Results for Transposition Cipher

The results of transposition cipher are presented in Table 4. Here the algorithm was run on different amounts of cipher text. The results in Table 2 represent the average number of key elements correctly placed for a key size of 20 with 100-1500 characters as a cipher text by using proposed EOPSO compare with PSO, 2-op PSO and SAPSO.

Table-4: The Amount of Recovered Key Versus Available Cipher Text, Transposition With Size 15 Using PSO and Improved PSO

Amount of Cipher Text	Amount of Recovered Key (PSO)	Amount of Recovered Key (2-op PSO)	Amount of Recovered Key (SAPSO)	Amount of Recovered Key (EOPSO)
100	7.75	9.75	10	10
150	10	10	11	11
250	10.50	10.75	12	12.25
400	11.25	11.25	12.5	12.5
600	12	12.25	12.75	13
800	12.75	12.75	13	13
1000	13	13.25	13.25	13.25
1250	13.25	13.25	13.5	13.5
1500	13.25	13.25	13.5	14

The amount of recovered keys using 1500 characters in known cipher text illustrate in Table 5. The table shows that the EOPSO attack was the

most powerful. For a transposition cipher of period 25 the improved PSO attack was better than basic PSO of the key elements, on the average.

Table-5: The Amount of Recovered Keys Versus Transposition Size Using 1000 Known Cipher Text Characters

Transposition Size	PSO	2-opt PSO	SAPSO	EOPSO
7	6.75	7	7	7
9	7.75	8	8	8
11	9	9	9	10
15	13	13.25	13.5	13.5
20	16.25	16.25	17	17.75
25	20	21.25	22	23

The best parameters value of PSO, 2-op PSO, SAPSO [1, 38] and EOPSO to attack transposition cipher show in Table 6.

Table-6: Parameters Value of PSO, 2-op PSO, SAPSO and EOPSO to Attack Transposition Cipher

Parameter	**Symbol**	**Value**
Self-confidence	C_1	2
Swarm confidence	C_2	2
Inertia weight	W	1.2
Crossover Probability	Pc	0.72
Mutation Probability	Pm	0.075
The maximum of velocity	V_{max}	26
Number of particles in the swarm	Swarm_Size	25-50
The maximum number of iterations	Max-Iter	550

Figure. 4 shows the performance of EOPSO compare with PSO, 2-op PSO and SAPSO. The EOPSO as compared to basic PSO, 2-op PSO and SAPSO performs better in terms of percentage of successful attacks, and we need to tune various parameters of the algorithm leading to simple program development effort.

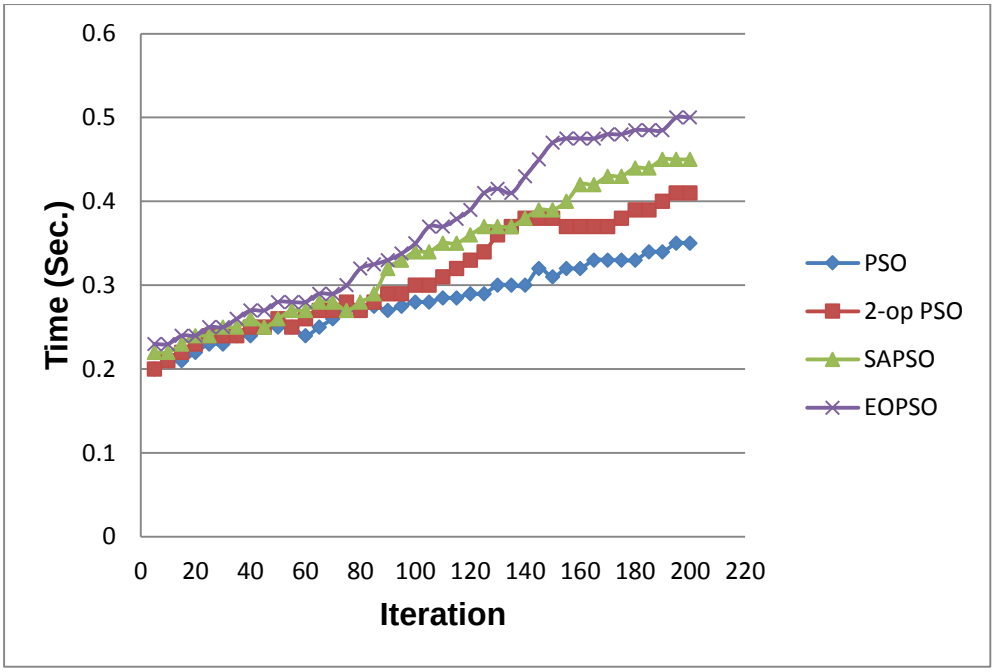


Figure.4. Performance of PSO, 2-op PSO, SAPSO and EOPSO for Transposition Attack

Figure. 5 illustrate the time comparison for EOPSO, PSO, 2-op PSO and SAPSO to attack substitution cipher.

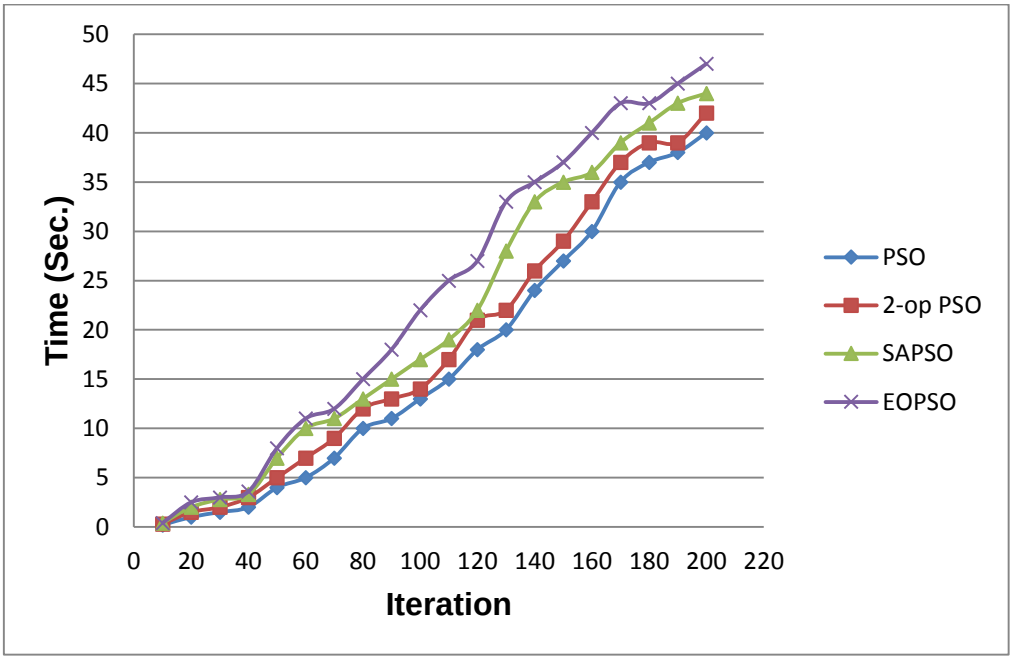


Figure.5. Curve of Time for PSO, 2-op PSO, SAPSO and EOPSO for Transposition Attack

CONCLUSIONS

The goal of this paper is to illustrate the feasibility of using EOPSO algorithm as a cryptanalysis tool and better than PSO and SAPSO. The proposed EOPSO algorithm is applied to classical ciphers (substitution

and transposition). The following points represent the important conclusions:

1- The performance (convergence) of EOPSO is better than PSO, 2-op PSO and SAPSO in both 2 types of classical ciphers (substitution and transposition).
2- The time of EOPSO is larger than PSO, 2-op PSO and SAPSO in both 2 types of classical ciphers (substitution and transposition) but the difference is small compare with increasing the performance.
3- Using the crossover operator gives a good diversity to the adjustment of PSO representation, so the convergence increases with a reasonable value.
4- Using mutation operator gives a momentum value to the adjustment of PSO parameters, so the convergence increases with small value. Also it provides population diversity, therefore it is useful to find the solutions.
5- The amount of recovered key in transposition cipher using EOPSO is larger than the other techniques (PSO, 2-op PSO and SAPSO).
6- The instability in EOPSO is less than PSO, 2-op PSO and SAPSO, so it is very useful technique in optimization problems.

REFERENCES

[1] Salih H. H., Ahmed Tariq Sadiq and Ismail K. Ali, "Attack on the Simple Substitution Ciphers Using Particle Swarm Optimization", Proceeding of 1st Conference of Software Eng., University of Technology, Baghdad, Iraq, (2009).

[2] El-Ghazali T., "Metaheuristics : From Design to Implementation", John Wiley & Sons Inc. Pub., USA, (2009).

[3] Holland J. H. "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, MI, (1975).

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

[5] De Jong K. A. "Genetic algorithms: A 10 year perspective", In International Conference on Genetic Algorithms, (1985), pp. 169–177.

[6] Mao, W., "Modern Cryptography: Theory & Practice". Upper Saddle River, NJ: Prentice Hall PTR, (2004).

[7] Giddy J. P. and Safavi-Naini R., "Automated Cryptanalysis of Transposition Ciphers". The Computer Journal, 37(5):429–436, (1994).

[8] Radcliffe N. J. and Surry P. D., "Fitness Variance of Form and Performance Prediction", In L. D. Whitley and M. D. Vose, editors, 3rd Workshop on Foundations of Genetic Algorithms, Morgan Kaufmann, (1994), pp. 51–72.

[9] Michalewicz Z., Nazhiyath G., and Michalewicz M., "A Note on Usefulness of Geometrical Crossover for Numerical Optimization Problems", In 5th Annual Conference on Evolutionary Programming, San Diego, CA. MIT Press, (1996), pp. 305–312.

[10] Ono I. and Kobayashi S., "A Real-Coded Genetic Algorithm for Functional Optimization Using Unimodal Normal Distribution Crossover", In ICGA-7, 7th International Conference on Genetic Algorithms, (1997), pp. 246–253.

[11] Tsutsui S., Yamamura M., and Higuchi T., "Multi-Parent Recombination with Simplex Crossover in Real-Coded Genetic Algorithms", In GECCO'99 Genetic and Evolutionary Computation Conference, (1999), pp. 657–664.

[12] Deb K. and Agrawal R. B., "Simulated Binary Crossover for Continuous Search Space", Complex Systems, 9:115–148, (1995).

[13] Deb K. and Joshi D., "A Computationally Efficient Evolutionary Algorithm for Real Parameter Optimization", Technical Report 003, KanGal, (2002).

[14] Davis L., "Job-shop Scheduling with Genetic Algorithms", In J. J. Grefenstette, editor, International Conference on Genetic Algorithms and Their Applications, Pittsburgh, PA, (1985), pp. 136–140.

[15] Goldberg D. E. and Lingle R., "Alleles, Loci, and the Traveling Salesman Problem", In J. J. Grefenstette, editor, 1st International Conference on Genetic Algorithms and Their Applications. Lawrence Erlbaum Associates, (1985), pp. 154–159.

[16] Muhlenbein H., Schleuter M. G., and O. Kramer, "Evolution Algorithms in Combinatorial Optimization", Parallel Computing, 7:65–85, (1988).

[17] Oliver I. M., Smith D. J., and Holland J. R. C., "A Study of Permutation Crossover Operators on the Traveling Salesman Problem", In J. J. Grefenstette, editor, 2nd International Conference on Genetic Algorithms, Hillsdale, NJ, (1987), pp. 224–230.

[18] Blanton J. L. and Wainwright R. L., "Multiple Vehicle Routing with Time and Capacity Constraints Using Genetic Algorithms", In S. Forrest, editor, 5th International Conference on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, (1993), pp. 452–459.

[19] Syswerda G., "Schedule Optimization Using Genetic Algorithms", In Handbook of Genetic Algorithms. Van Nostrand Reinhold, (1991), pp. 332–349.

[20] Pant M., Thangaraj R. and Abraham A., "Particle Swarm Optimization Using Adaptive Mutation",19th International Workshop on Database and Expert Systems Application, DEXA '08, (2008).

[21] Wei J., Guangbin L. and Dong L., "Elite Particle Swarm Optimization with mutation", 7th International Conference on System Simulation and Scientific Computing, ICSC 2008, Nov. (2008).
[22] Li N., Qin Y., Sun D. and Zou T., "Particle swarm optimization with mutation operator", Proceedings of 2004 International Conference on Machine Learning and Cybernetics, (2004).
[23] Chou P. C., "Improved Particle Swarm Optimization with Mutation", Modeling, Simulation, and Identification, Edt. H. Ma and S. Narayanan, (2009).
[24] Douglas R. Stinson., "Cryptography: Theory and Practice". CRC Press, Boca Raton, Florida, USA, (1995).
[25] Henry B. and Fred P., "Cipher Systems: The Protection of Communications", Wiley-Inter Science, London, (1982).
[26] Kennedy J. and Eberhart R., "Particle Swarm Optimization", in Proceedings of the IEEE International Conference on Neural Networks, pp. 1942-1948, (1995).
[27] Eberhart R. C. and Kennedy J., "A New Optimizer Using Particle Swarm Theory", Proc. 6th Symposium on Micro Machine and Human Science, pp. 39–43, (1995).
[28] Khanesar M. A., Teshnehlab M., and Shoorehdeli M. A, "A Novel Binary Particle Swarm Optimization", Proc. 15th Mediterranean Conference on Control and Automation, (2007).
[29] Al-Kazemi, B and Mohan C., "Multi-phase Discrete Particle Swarm Optimization". In: Fourth International Workshop on Frontiers in Evolutionary Algorithms, Kinsale, Ireland, (2002).
[30] Pampara G, Franken N and Engelbrecht A., "Combining Particle Swarm Optimization with Angle Modulation to Solve Binary Problems". In: Proceedings of the IEEE Congress on Evolutionary Computing, vol 1, pp 89-96, (2005).
[31] Onwubolu G. and Clerc M., "Optimal Operating Path for Automated Drilling Operations by a New Heuristic Approach Using Particle Swarm Optimization". International Journal of Production Research 42(3):pp.473-491, (2004).
[32] Pang W, Wang K, Zhou C and Dong L., "Fuzzy Discrete Particle Swarm Optimization for Solving Traveling Salesman Problem". In: Proceedings of the 4th International Conference on Computer and Information Technology (CIT04), IEEE Computer Society, vol 1, pp 89-96, (2004).
[33] Pugh J, and Martinoli A., "Discrete Multi-Valued Particle Swarm Optimization". In: Proceedings of IEEE Swarm Intelligence Symposium, vol 1, pp 103-110, (2006).

[34] Correa E., Freitas A., Johnson C., "A New Discrete Particle Swarm Algorithm Applied to Attribute Selection in a Bio-informatic data set". In: Proceedings of GECCO 2006, pp35-42, (2006).

[35] Martinez-Garcia F., Moreno-Perez J., "Jumping Frogs Optimization: A New Swarm Method for Discrete Optimization". Tech. Rep. DEIOC 3/2008, Dep. of Statistics, O.R. and Computing, University of La Laguna, Tenerife, Spain, (2008).

[36] Moreno-Perez J., Castro-Gutierrez J., Martinez-Garcia F., Melian B, Moreno-Vega J. and Ramos J., "Discrete Particle Swarm Optimization for the p-median Problem". In: Proceedings of the 7th Metaheuristics International Conference, Montreal, Canada, (2007).

[37] Shi Y. and Eberhart R.C., "A Modified Particle Swarm Optimizer", Proc. IEEE Conference on Evolutionary Computation, (1998).

[38] Kumar A., and Zhang D., "Palm Print Authentication Using Multiple Representation", Pattern Recognition, vol. 38, pp.1695-1704, (2005).

[39] Carlisle A. and Dozier G., "An off-the-Shelf PSO", Proc .Particle Swarm Optimization Workshop, pp. 1–6, (2001).

[40] Clark, A., "Modern Optimization Algorithms for Cryptanalysis". In Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems, November 29 - December 2,pp. 258-262, (1994).

[41] Spillman, R., Jansses, M., and Kepner, M., "Use of Genetic Algorithm in the Cryptanalysis of Simple Substitution Ciphers", Cryptologia, vol. 17, issue 1, January 1993,pp. 31-34, (1993).

[42] Ismail K. Ali, "Intelligent Cryptanalysis Tools Using Particle Swarm Optimization", Ph. D. Thesis, Computer Sciences Department, University of Technology, Baghdad, Iraq, (2009).

[43] Sadiq A. T., "Evolutionary Operators-Based Particle Swarm Optimization (EOPSO) to Attack Classical Cryptography Methods", Journal of Advanced Computer Science and Technology Research, Vol. 2, No. 1, pp. 50-65, (2012).