Mathematical Theory and Modeling
ISSN 2224-5804 (Paper) ISSN 2225-0522 (Online)
Vol.4, No.9, 2014

Modification of Some Solution Techniques of Combinatorial Optimization Problems to Analyze the Transposition Cipher

Tariq S. Abdul-Razaq* and Faez H. Ali
Math. Dept, College of Sciences, University of Al-Mustansiriya, Baghdad, Iraq.
*dr.tariqsalih@yahoo.com

Abstract

In this paper we attempt to use a new direction in cryptanalysis of classical crypto systems. The new direction represented by considering some of classical crypto systems, like transposition cipher problem (TCP), as a combinatorial optimization problem (COP), then using the known solving methods of COP, with some modification, to cryptanalysis the TCP. In this work we investigate to use Branch and Bound (BAB) and one of swarm algorithms as a local search method.

The main aim of the research presented in this paper is to investigate the use of some optimization methods in the fields of cryptanalysis and cryptographic function generation. These techniques were found to provide a successful method of automated cryptanalysis of a variety of the classical ciphers.

Keywords: cryptography, cryptanalysis, Classical Ciphers, Transposition Cipher, Branch and Bound, Swarm Intelligence, Bees Algorithm.

1. Introduction

In 1993, an attack on the transposition cipher (TC) was proposed by Matthews (Matthews 1993) using a genetic algorithm. Clark in his Ph.D. thesis investigates the use of various optimization heuristics (simulated annealing, the genetic algorithm and the tabu search) in the fields of automated cryptanalysis. These techniques were found to provide a successful method of automated cryptanalysis of a variety of the classical ciphers (substitution and transposition type ciphers) (Clark 1998). Russell et al. investigated the use of Ant colony optimization for breaking TC (Russell et al. 2003). Ali, in his Ph.D. thesis, introduces a set of modifications to enhance the effective of particle swarm optimization to cryptanalyze the TC (Ali 2009). Ahmed et al. presented an improved cuckoo search algorithm for automatic cryptanalysis of TC's (Ahmed et al. 2014).

Cryptography studies the design of algorithms and protocols for information security. The ideal situation would be to develop algorithms which are provably secure, but this is only possible in very limited cases. The whole point of cryptography is to keep the plaintext (or the key, or both) secret from eavesdroppers. **Cryptanalysis** is the science of recovering the plaintext (PT) or the key. An attempted cryptanalysis is called an attack. Successful cryptanalysts may recover the PT or the key. They also may find weaknesses in a cryptosystem that eventually leads to the previous results (Mao 2004).

Swarm Intelligence (SI) is an Artificial Intelligence (AI) technique that focuses on studying the collective behavior of a decentralized system made up by a population of simple agents interacting locally with each other and with the environment (Xiaodong 2004).

The most fundamental criterion for the design of a cipher is that the key space (i.e., the total number of possible keys) be large enough to prevent it being searched exhaustively (Clark 1998).

The main goal of this paper is to exploit the more common method of COP, which is Branch and Bound (BAB) method, as an exact method, to solve the TCP. Next we will explore and illustrate the abilities of SI especially the usage of the Bees algorithm (BA), as an approximate method, for developing intelligent optimization tool colonies for the purpose of providing a "good" solution of TCP. Also, the work presented here studies and utilizes the use of mentioned cryptanalysis methods (BAB and BA) with help of successive rules to solve this problem within acceptable amount of time with a faster convergence and time reduction.

The paper is organized as follows: combinatorial optimization is described and given is section (2). In section (3) the TCP is given in details. The BA and its parameters are illustrated in section (4). The classical and the proposed cryptanalysis tools are shown in sections (5) and (6) respectively. In section (7) the implementation of exact methods (complete enumeration CE and BAB) and classical BA (CBA) to solve TCP are discussed and show the modification of CBA. Section (8) introduces the concept and the generating of subsequences from successive rules (SR). Lastly, in section (9) the exploitation of successive rules in solving TCP using CEM, BAB and modified BA methods is introduced.

2. Combinatorial Optimization (CO)

The aim of **combinatorial optimization** is to provide efficient techniques for solving mathematical and engineering related problems. Many problems arising in practical applications have a special discrete and finite nature, for examples to find the minimal value of COP: Shortest Path, Scheduling, Travelling Salesman Problem and many more. These problems are predominantly from the set of NP-complete problems. Solving such problems requires effort (eg., time and/or memory requirement) which increases dramatically with the size of the problem. Thus, for sufficiently large problems, finding the best (or optimal) solution with certainty is often infeasible. In practice, however, it usually suffices to find a "good" solution (the optimality of which is less certain) to the problem being solved.

Provided a problem has a finite number of solutions, it is possible, in theory, to find the optimal solution by trying every possible solution. An algorithm which tries every solution to a problem in order to find the best is known as a **brute force algorithm** (BF). Cryptographic algorithms are almost always designed to make a BF attack of their solution space (or key space) infeasible. CO techniques attempt to solve problems using techniques other than BF since many problems contain variables which may be unbounded, leading to an infinite number of possible solutions.

Algorithms for solving problems from the field of CO fall into two broad groups- exact algorithms and approximate algorithms. An exact algorithm guarantees that the optimal solution to the problem will be found. The most basic exact algorithm is a BF or what we called **complete enumeration**. Other examples are **branch and bound**, and the **simplex method**. The algorithms used in this paper are from the group of approximate algorithms. Approximate algorithms attempt to find a "good" solution to the problem. A "**good**" **solution** can be defined as one which satisfies a predefined list of expectations. For example, consider a cryptanalytic attack. If enough PT is recovered to make the message readable, then the attack could be construed as being successful and the solution assumed to be "**good**". Often it is impractical to use exact algorithms because of their prohibitive complexity (time or memory requirements). In such cases approximate algorithms are employed in an attempt to find an adequate solution to the problem. Examples of approximate algorithms (or, more generally, heuristics) are simulated annealing, the genetic algorithm and the tabu search (Clark 1998).

3. Transposition Cipher Problem (TCP)

Classical ciphers were first used hundreds of years ago. As far as security is concerned, they are no match for today's ciphers; however, this does not mean that they are any less important to the field of cryptology. Their importance stems from the fact that most of the ciphers in common use today, and utilize the operations of the classical ciphers as their building blocks (Mao 2004).

A TC is an encryption in which the letters of the message are rearranged. With a TC the goal is diffusion, spreading the information from the message or the key out widely across the ciphertext (CT). TC tries to break established patterns. Because a TC is a rearrangement of the symbols of a message, it is also known as a permutation.

A TC works by breaking a message into fixed size blocks, and then permuting the characters within each block according to a fixed permutation, say Π. The key to the TC is simply the permutation Π.

Let E and D=E^{-1} be encryption and decryption of two variables functions of TCP respectively. Let M be the PT which want to be encrypted using E function, then the CT C_m of TCP, where $1 \le m \le n!$, using arbitrary encryption key EK_m (Π of n-sequence) with length n is:

$$C_m = \mathrm{E}(M, EK_m) \qquad \text{(E)}$$

Let DK_m (σ of n-sequence) be the decryption key corresponding to the EK_m for CT C_m of TCP and M_m be the decrypted text using DK_m, is:

$$M = M_m = \mathrm{D}(C_m, DK_m) \qquad \text{(D)}$$

Its clear that C_m (and M_m) consists of n columns.

Example (1): Lets have the following PT message (showed in uppercase letters):

1	2	3	4
T	H	E	Q
U	I	C	K
B	R	O	W
N	F	O	X
J	U	M	P
S	O	V	E
R	T	H	E
L	A	Z	Y
D	O	G	X

The size of the permutation is known as the period. For this example a TC with a period of 4 is used. Let Π=(3,1,4,2) be encryption key. Then the message is broken into blocks of 4 characters. Upon encryption the 3rd character in the block will be moved to position 1, the 1st to position 2, the 4th to position 3 and the 2nd to position 4.

K		**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
P	:	T	H	E	Q	U	I	C	K	B	R	O	W	N	F	O	X	J	U	M	P
C	:	e	t	q	h	C	u	k	i	o	b	w	R	o	n	x	f	m	j	p	u
K	:	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
P	:	S	O	V	E	R	T	H	E	L	A	Z	Y	D	O	G	X	X	X	X	X
C	:	v	s	e	o	H	r	e	t	z	l	y	A	g	d	x	o	x	x	x	x

The resulting CT (in lowercase letters) would then be read off as:

etqhc ukiob wronx fmjpu vseoh retzl yagdx o

Notice also that decryption can be achieved by following the same process as encryption using the "inverse" of the encryption permutation. In this case the decryption key (*DK*), $\Pi^{-1}=\sigma$ is equal to (2,4,1,3).

4. Classical Bees Algorithm (CBA)

The artificial intelligence is used to explore distributed problem solving without having a centralized control structure. This is seen to be a better alternative to centralized, rigid and preprogrammed control. Real life SI can be observed in ant colonies, beehives, bird flocks and animal herds.

The most common examples of SI systems: Ant Colony Optimization, Particle Swarm Optimization and Marriage in Honey Bees Optimization.

MBO is a new development which is based on the haploid-diploid genetic breeding of honeybees and is used for a special group of propositional satisfiability problems. The main processes in MBO are: the mating flight of the queen bee with drones, the creation of new broods by the queen bee, the improvement of the broods' fitness by workers, the adaptation of the workers' fitness, and the replacement of the least fit queen with the fittest brood (Ashraf et al. 2006).

The challenge is to adapt the self-organization behavior of the colony for solving the problems. The **Bees Algorithm (BA)** is an optimization algorithm inspired by the natural foraging behavior of honey bees to find the optimal solution.

The algorithm requires a number of parameters to be set, namely:

n	:	Number of scout bees (*n*).
m	:	Number of sites selected out of n visited sites (*m*).
e	:	Number of best sites out of m selected sites (*e*).
nep	:	Number of bees recruited for best e sites (*nep*).
nsp	:	Number of bees recruited for the other (*m-e*) selected sites (*nsp*).
ngh	:	Initial size of patches (*ngh*) which includes site and its neighborhood and stopping criterion.

The pseudo code for the BA is shown below in its simplest form (Pham et al. 2006).

Bees Algorithm

INPUT: *n*, *m*, *e*, *nep*, *nsp*, Maximum of iterations.

```
Step1.  Initialize population with random solutions.
Step2.  Evaluate fitness of the population.
Step3.  REPEAT
Step4.      Select sites for neighborhood search.
Step5.      Recruit bees for selected sites (more bees for best e sites) and evaluate fitness’s.
Step6.      Select the fittest bee from each patch.
Step7.      Assign remaining bees to search randomly and evaluate their fitness’s.
Step8.  UNTIL stopping criterion is met.
OUTPUT: Optimal or near optimal solutions.
  END.
```

The advantages of Bees algorithm (Ashraf et al. 2006):

- BA is more efficient when finding and collecting food that is it takes less number of steps.
- BA is more scalable, it requires less computation time to complete the task.

5. Classical Cryptanalytic Tools for TCP

Before we discuss the TC cryptanalysis we have to know something about **Diagram** (DG) and **Trigram** (TG). Just as there are characteristic letter frequencies, there are also characteristic patterns of pairs of adjacent letters, called DG. Letter pairs such as "**RE**", "**TH**", "**EN**", and "**ED**" appear very frequently. Table 1 lists the (46) most common DG and TG (groups of three letters) in English.

The frequency of appearance of letter groups can be used to match up PT letters that have been separated in a CT. These counts and relative frequencies were obtained from a representative sample of English, not counting DG that consist of the last letter of one word and the first letter of the next word (Pfleeger 1998).

Table 1. Most common DGs and TGs (Pfleeger 1998).

	1	2	3	4	5	6	7	8	9	10
DG	AN	AR	AS	AT	EA	EN	ER	ES	ET	HA
	HE	HI	IN	IS	IT	ND	NG	NT	OF	OR
	OU	RE	SE	ST	TE	TF	TH	TI	TO	29
TG	THI	NTH	THE	ENT	AND	HER	ETH	DTH	WAS	TIO
	ONE	ION	THA	OUR	FOR	IVE	ING	17		

6. Proposed Cryptanalysis Studies on TCP

6.1 Proposed Cryptanalysis Tools

In general, the literatures assign that, in almost situations, there is no direct solution (decryption) for TCP when using any classical or modern cryptanalysis tools. As usual in cryptanalysis the TCP, the final obtained key, as a result from the cryptanalysis process, will be used to decrypt the CT, if the PT is not really correct we will still swapping between some wrong positions of the key until we gain good readable text, then we can say that we obtain the actual decipher key (*ADK*).

In this paper, we suggest our own cryptanalysis tools in order to analyze the TCP from a sense of COP or in another word; we treat the TCP as a COP. In this manner, we introduce a new study about the DG, TG and quadgram (QG) (4 contagious letters) frequency of PT letters with length L=10000 letters, we called these frequencies as **desired frequencies**. We take in consideration the most frequent samples in PT, so we use the following notations:

D_i^d	:	Desired Frequency of DG (d-gram) of letter *i*.
O_i^d	:	Observed Frequency of DG (d-gram) of letter *i*.
D_i^t	:	Desired Frequency of TG (t-gram) of letter *i*.
O_i^t	:	Observed Frequency of TG (t-gram) of letter *i*.

D_i^q : Desired Frequency of QG (q-gram) of letter *i*.

O_i^q : Observed Frequency of QG (q-gram) of letter *i*.

Where *i*='*a*','*b*',…,'*z*'.

Remark (1): Its important to mentioned that all frequencies for the three samples are calculated with overlap and not calculated for the beginnings and ends of the words.

In general let P(X_i^j) be the probability of *X* (=*D* or =*O*) desired or observed frequency of *j*-gram (*j*=*d*, *t* and *q*), for the letter i s.t.

$$P(X_i^j) = \frac{X_i^j}{L^j} \qquad (1)$$

$$\overline{X}^j = \frac{\sum_{i='a'}^{'z'} X_i^j}{L^j} \qquad (2)$$

where L^j=*L*-*j*+1, *j*=*d*,*t*,*q*, $\overline{X}^j$ is the arithmetic mean of X_i^j frequency.

The results of frequency and ration of the three mentioned samples of our study for the English language are illustrated in tables (2), (3) and (4).

Table 2. Desired frequency of the most common DG (80 DGs).

		Samples															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DG	S.	AB	AC	AL	AN	AR	AS	AT	BE	CA	CE	CH	CO	DE	DI	EA	EC
	D_i^t	49	41	83	152	67	64	136	47	51	61	46	78	40	46	95	58
	S.	ED	EE	EI	EL	EM	EN	EO	ER	ES	ET	FT	HA	HE	HI	HO	IC
	D_i^t	88	44	44	49	58	114	62	160	136	100	41	88	252	44	49	64
	S.	IL	IN	IO	IS	IT	LE	LI	LL	LY	MA	ME	NE	NG	NI	NE	NG
	D_i^t	54	210	58	83	113	72	68	54	64	67	66	49	64	41	49	64
	S.	NI	NS	NT	OA	OF	OM	ON	OR	OT	OU	PE	PR	RA	RE	RM	RO
	D_i^t	41	48	128	46	105	66	136	82	52	77	42	78	53	147	60	99
	S.	SA	SE	SI	SO	SS	ST	TA	TE	TH	TI	TO	TT	US	UT	VE	WE
	D_i^t	63	96	62	67	54	134	69	85	312	161	101	46	50	46	63	60

$$\overline{D}^d = \frac{\sum_{i='aa'}^{'zz'} D_i^d}{L^d} \qquad (3)$$

From table 2, $\overline{D}^d$ =0.6471 for *L*=10000 PT letters and for the most 80 DG frequency, these 80 DG filtered when $P(F_i^d)$ ≥TD, where TD is a threshold DG chosen by experiences (in this paper we chose TD=0.004).

Table 3. Desired frequency of the most common TGs (52 TGs).

		Samples												
		1	2	3	4	5	6	7	8	9	10	11	12	13
TG	S.	ABI	ALL	AND	ARE	ATI	BAB	BIL	COM	CON	ECO	EDI	EIN	EMA
	D_i^t	31	27	66	28	62	31	31	29	23	23	23	22	23
	S.	ENT	ERE	ERS	EST	ETH	FTH	HAT	HEM	HER	ILI	ING	INT	ION
	D_i^t	50	32	28	23	38	35	37	28	44	31	45	42	52
	S.	IST	ITI	ITY	LIT	MAT	NCE	NTH	OBA	OFT	OME	OTH	OUT	PRO
	D_i^t	35	28	28	35	29	35	41	31	34	25	27	24	53
	S.	REA	ROB	SOF	STA	STH	STO	THA	THE	THI	TIC	TIO	TIS	TTH
	D_i^t	24	37	34	22	33	25	38	219	23	23	45	22	27

$$\overline{D}^t = \frac{\sum_{i='aad'}^{'zzz'} D_i^t}{L^t} \qquad (4)$$

From table 3, $\overline{D}^t = 0.1901$ for L=10000 PT letters and for the most 52 TG frequency, these 52 filtered when $P(D_i^t) \geq TT$, where TT is a threshold TG chosen by experiences (in this paper we chose TT=0.0022).

Table 4. Desired frequency of the most common QGs (21 QGs).

		Samples										
		1	2	3	4	5	6	7	8	9	10	11
QG	S.	ABIL	ATIO	BABI	BILI	ETHE	FTHE	ILIT	INTH	LITY	NTHE	OBAB
	D_i^q	31	26	31	31	28	33	31	22	24	27	31
	S.	OFTH	OTHE	PROB	ROBA	STHE	THAT	THEM	THEO	THER	TION	
	D_i^q	30	22	37	31	23	29	28	20	34	45	

$$\overline{D}^q = \frac{\sum_{i='aaad}^{'zzzz'} D_i^q}{L^q} \qquad (5)$$

From table 4, $\overline{D}^q = 0.0614$ for the L=10000 PT letters and for the most 21 QG frequency, these 21 filtered when P(D_i^q)≥TQ, where TQ is a threshold QG chosen by experiences (in this paper we chose TQ=0.002).

6.2 Proposed Cryptanalysis Objective Functions

From equations (3-5), the **sum of the most high frequency** (SMHF) for PT and CT are calculated in equation (6) and (7) respectively.

$$\text{SMHF(P)} = \overline{D}^d + \overline{D}^t + \overline{D}^q \qquad (6)$$

for $\overline{D}^d \geq 0.004$, $\overline{D}^t \geq 0.0022$ and $\overline{D}^q \geq 0.002$

$$\text{SMHF(C)} = \overline{O}^d + \overline{O}^t + \overline{O}^q \qquad (7)$$

The $\overline{O}^j$ of letters of CT are the corresponding to the letters of PT frequency $\overline{D}^j$ mentioned in equation (6), where *j*=*d*, *t*, *q*.

It's clear that for *L*=10000 PT letters the SMHF(P)=0.6471+0.1901 +0.0614=0.8986.

For the CT of TCP with *n*=4 (*n* is the key length), the frequencies of the DG, TG and QG are really different, and that is the weak point we exploited to differentiate between PT and CT.

Of course we are interest in SMHF for PT and CT. Table 5 shows the values of SMHF for PT and CT for text with length *L*=(1,(1),10) $\times 10^3$ letters.

Table 5. The values of SMHF(P) and SMHF(C) for text with different L.

SMHF	Text ×1000 letters										Av.
	1	2	3	4	5	6	7	8	9	10	
SMHF(P)	1.375	0.979	0.929	0.894	0.868	0.862	0.887	0.923	0.930	0.899	0.955
SMHF(C)	0.848	0.587	0.549	0.527	0.512	0.498	0.476	0.485	0.488	0.491	0.546

From the above table, especially in Av. column, the SMHF(P) is always greater than SMHF(C), so in order to obtain real PT from decrypted CT, using specific key, we look for the maximum SMHF. Notes that the worst case for SMHF(P) function is 0.862 at *L*=6000 and best case for SMHF(C) function is 0.8482 at *L*=1000 (shaded cells). Figure 1 shows the behavior of SMHF function for different *L* of PT and CT.

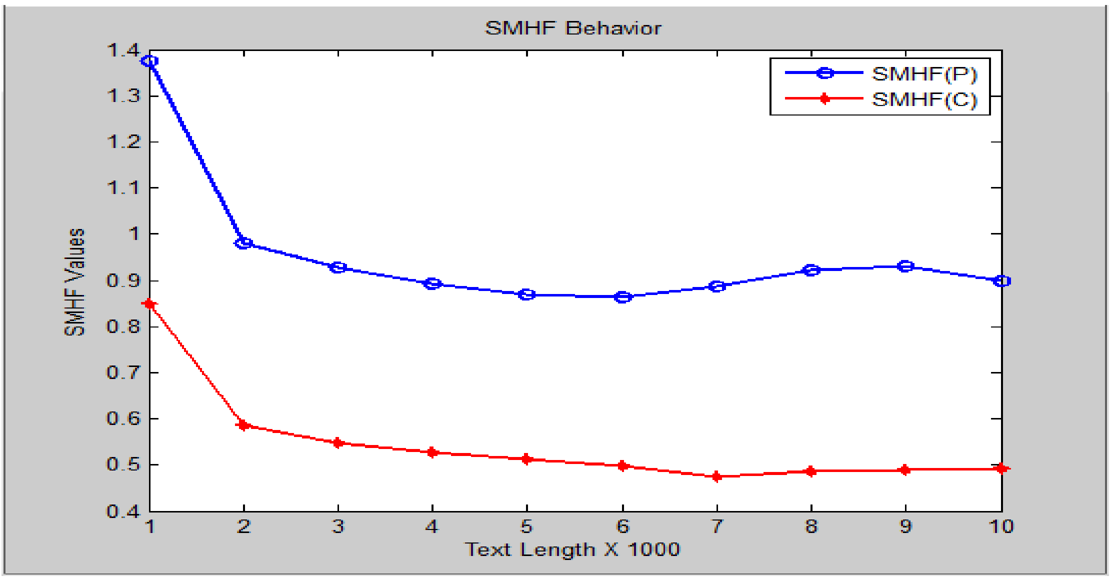

Figure 1. The behavior of SMHF function for different *L* of PT and CT.

Now we have another measure to diagnose the TC which is called the **coincidence of desired frequency** (CDF) for PT. This value can be calculated as follows:

$$\text{CDF}=\sum_{i='aa'}^{'zz'}\left|\bar{D}_i^{d}-\bar{O}_i^{d}\right|+\sum_{i='aad'}^{'zzz'}\left|\bar{D}_i^{t}-\bar{O}_i^{t}\right|+\sum_{i='aaad}^{'zzzz'}\left|\bar{D}_i^{q}-\bar{O}_i^{q}\right| \qquad (8)$$

The frequency in equation (8), are for all combinations of two, three and four letters sample in PT or CT. Table 6 shows the CDF values for different *L* for PT and CT.

Table 6. The CDF values for different *L* for PT and CT.

CDF	Text ×1000 letters										Av.
	1	2	3	4	5	6	7	8	9	10	
CDF(P)	0.352	0.225	0.206	0.176	0.159	0.117	0.080	0.044	0.030	0.000	0.139
CDF(C)	0.627	0.593	0.600	0.590	0.593	0.594	0.594	0.597	0.597	0.598	0.598

Note that the CDF(P) values, for different text lengths, is in continuous decreasing, while the CDF(C) values are in stable context. Notes that the worst case for CDF(P) function is 0.3523 for *L*=1000 and best case for CDF(C) function is 0.5899 for *L*=4000 (shaded cells). Figure 2 shows the behavior of CDF function for different *L* of PT and CT.

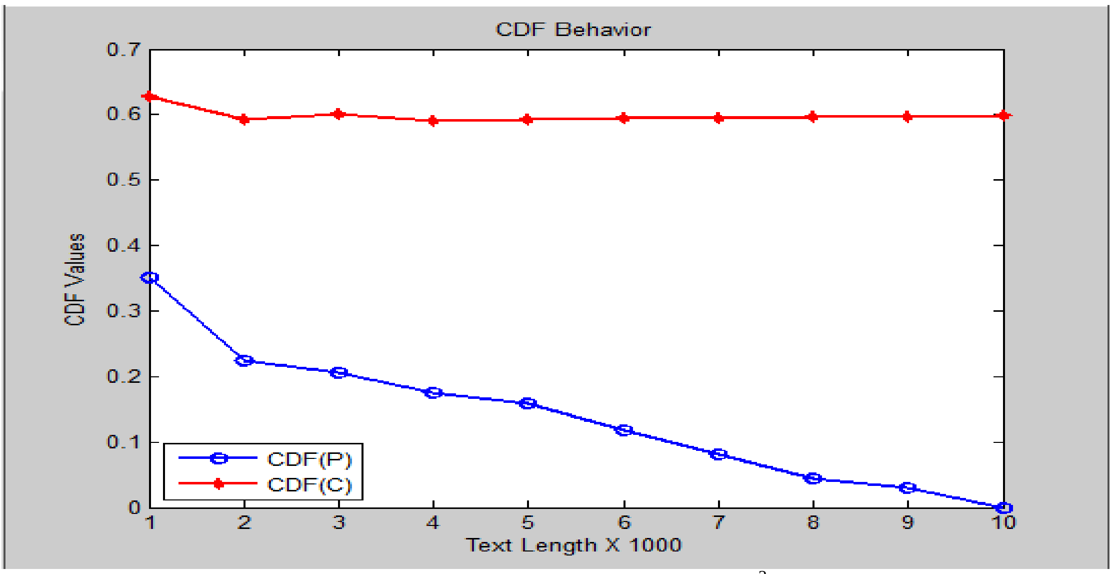

Figure 2. The behavior of CDF function for L=(1,(1),10)×10^3 letters of PT and CT.

In figure 3 the behavior of SHMF and CDF functions for different *L* of PT is shown.

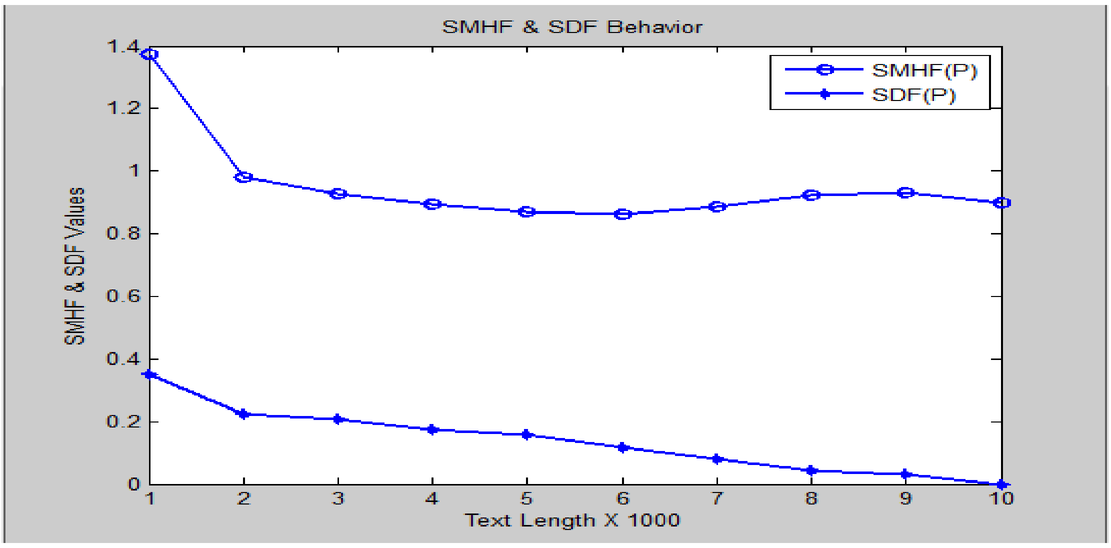

Figure 3. The behavior of SMHF and CDF functions for different *L* of PT only.

6.3 TCP Formulation

The cryptanalysis of TCP can be treated as a COP with two objective functions (TOF). Suppose that the *EK* treated as a sequence σ=(1,2,…,*n*), where *n* is the length of the *EK* (or *DK*), s.t.

$$\text{TOF }(\sigma)=\{\text{Max SMHF, Min CDF}\} \qquad (9)$$

The bicriteria approach enables constraints of different nature, expressed in different ways, to be treated simultaneously. The two objectives aggregated into one composite (single) objective function (SOF). Since 0≤CDF≤1 and Min{CDF}≡ Max{1-CDF}, hence the TCP can be written as follows:

$$\left.\begin{array}{l}
\text{SOF}(\sigma)=\text{ Max }\{\text{SMHF}+1-\text{CDF}\} \\
\text{s. t.} \\
\text{SMHF(C)} \geq \overline{O}^{d}+\overline{O}^{t}+\overline{O}^{q} \\
CDF=\sum_{i='aa'}^{'zz'}\left|\overline{D}_i^{d}-\overline{O}_i^{d}\right|+\sum_{i='aad'}^{'zzz'}\left|\overline{D}_i^{t}-\overline{O}_i^{t}\right|+\sum_{i='aaad}^{'zzzz'}\left|\overline{D}_i^{q}-\overline{O}_i^{q}\right| \\
\overline{D}^{d},\overline{D}^{t},\overline{D}^{q},\overline{O}^{d},\overline{O}^{t},\overline{O}^{q}\geq 0
\end{array}\right\} \qquad (P)$$

7. Solving TCP using Exact and BA Methods

In this section, for TCP, we will apply the exact methods which represented by complete enumeration and branch and bound methods, and a local search (LS) method represented by the classical BA. These methods are chosen to solve the TCP using the proposed cryptanalysis tools.

7.1 Solving TCP using Complete Enumeration Method (CEM)

As known, to pass all the states of any permutation with length *n*, we use the Complete Enumeration Method (CEM) with *n*! states. The CEM results for *n*=3 (6 states) with *L*=3000 letters viewed in table 7 for the TOF and SOF.

Table 7. The CEM results for n=3 with L=3000 letters.

state	Applying CEM results			
	Decrypted 20 letters text	σ=DK	TOF(σ)	SOF(σ)
1	HTESUENITIHSUPBILCTA	(3,2,1)	(0.6092,0.6076)	1.0016
2	HETSEUNTIISHUBPICLTI	(3,1,2)	(0.5762,0.4679)	1.1083
3	THEUSEINTHISPUBLICAT	(2,3,1)	(0.9288,0.2059)	1.7229
4	TEHUESITNHSIPBULCIAI	(2,1,3)	(0.6342,0.6142)	1.0200
5	ETHEUSTINSHIBPUCLIIA	(1,2,3)	(0.5685,0.4755)	1.0930
6	EHTESUTNISIHBUPCILIT	(1,3,2)	(0.6469,0.5926)	1.0543

The shaded cell represents the PT obtained from decrypting the CT using the *ADK*. Table 8 shows the SOF values showed for different *L* and also shows the difference between the correct and incorrect decrypted texts for *n*=4 using CEM.

Table 8. SOF(σ), n=4, different L, the correct and incorrect decrypted texts by CEM.

$L\times10^3$	Incorrect decrypted text		Correct decrypted text	
	Text	SOF(σ)	Text	SOF(σ)
1	AKMRSRSEIEVCAKM	1.1391	MARKSSERVICEMAR	1.7194
2	IISMATLRRSEMVNE	1.3163	SIMILARTERMSEVE	1.6999
3	YREANTEODNIEIIT	1.2649	EYARENOTIDENTIF	1.6919
6	UHSCSOINTBTOTKE	1.2392	SUCHISNOTTOBETA	1.7215
8	NXAERSPEINSOFPO	1.2774	ANEXPRESSIONOFO	1.8145
10	ATNSWEOHHRTEROO	1.2838	NASTOWHETHERORN	1.8372

In table 9 we will discuss the required times (RT) to solve the CT for different *L* with *n*=3,…,10 for RT≤2750 sec. (≈45 minutes) using CEM.

Table 9. The RT for different L with n=3,…,10 using CEM.

n	n!	L letters					
		L=1000		L=5000		L=9000	
		SOF(σ)	RT	SOF(σ)	RT	SOF(σ)	RT
3	6	1.7194	0.016	1.6886	0.03	1.8395	0.03
4	24		0.04		0.07		0.08
5	120		0.15		0.36		0.38
6	720		0.84		1.52		2.10
7	5040		4.62		9.56		14.8
8	40320		29.1		54.8		88.3
9	362880		257		510		788
10	3628800		2719		2707*		4113*

The RT signed with * is the expected time which is interpolated by using Lagrange interpolation. We conclude that the applying of CEM when 3≤*n*≤9 will be suitable for reasonable time.

7.2 Solving TCP using Branch and Bound (BAB) Method

Branch and Bound (BAB) considered as the most common method to solve problems classified as COP, especially when CEM will be no more efficient in finding optimal solutions for large *n*.

In this paper, we will see how the BAB method is efficient in solving the TCP? First, since we want to maximize SOF(σ) for TCP so we have to find a suitable lower bound (LB), where LB= SOF(σ). In order to obtain a suitable LB we suggest to make this LB as a dynamic LB, in another words, the proposed LB changes its value in each level of BAB method.

The BAB procedure is usually described by means of search tree with nodes that corresponding to subsets of feasible solutions. To maximize an objective function (SOF) for a particular TCP, the BAB method successively partitions subsets using a branching procedure and computes an upper bound (UB) using upper bounding procedure and by these procedures excludes the nodes which are found not to include any optimal solution and this eventually leads to at least one optimal solution.

For each of the subsets (nodes) of solutions one computes UB to the maximum value of the objective function (SOF) attained by solutions belong to the subsets. If the UB calculated for a particular node is less than or equal to the LB (this LB is defined as the maximum of the values of all feasible solutions currently found), this node is ignored since any node with value greater than LB can only exist in the remaining nodes. One of these nodes with maximum UB is chosen, from which to branch. When the branching ends at a complete sequence of *n* letters, this sequence is evaluated and if its value greater than the current LB, this LB is reset to take that value. The procedure is repeated until all node have been considered (i.e., upper bounds of all nodes in the scheduling tree are less than or equal to the LB), a feasible solution with this LB is an optimal solution.

Now we suggest a new BAB method with new technique. The new technique depends first on assign LB to 1.0 after that calculate the UB for each node in each level, then searching for the best UB which is corresponding to the best sequence σ. Then improve the value of LB in each level by make it equal to the mean of the set of good UB's (UB's≥LB) in that level. The best UB (≈1.7) is the fitness of ADK to solve TCP. The new BAB is called modified BAB (MBAB) method. The MBAB algorithm is shown below.

MBAB algorithm

```
STEP(1): INPUT CT, $L$, $n$;
        LB=1.0, $\ell$=0, $s\sigma=(1,2,\dots,n)$, $ND=n$, [FOR $k=1,\dots,n$ $SEQ(k)=k$];
STEP(2): $\ell=\ell+1$, $m=0$;
        FOR $k=1,\dots,ND$
            Branching from node last digit $\ell$ in $SEQ$;
            $UNSEQ = s\sigma$ without $SEQ$;
            $\sigma$ = concatenate($SEQ$,$UNSEQ$);
            Calculate $UB_k$= SOF($\sigma$)                    {in level –$\ell$ }
            IF $UB_k \geq$ LB THEN
                $m = m + 1$,
                $LIST(m, :) = \sigma$; $SUB(m) = UB_k$;
            END;
        END;
STEP(3): LB=mean {$SUB$};
        BestFit = $\max_{1\le i\le m}\{SUB\}$ , BestDK= $LIST(i)$;
        $SEQ$=cut from $LIST$ first $\ell$ digits, $LIST=\Phi$, $SUB=\Phi$; $ND=m$;
        IF $\ell=n-1$ STOP ELSE GOTO STEP(2);
        IF BestFit $\geq$ 1.68 STOP;
STEP(4): OUTPUT BestFit, BestDK;
```

In table 10 we will discuss the required CPU times and number of nodes to decrypt the CT for different L for n=5,…,12 using MBAB method compared with CEM, RT ≤ 2400 sec. (=40 minutes).

Table 10. RT and ND for different L for n=5,…,12 using MBAB compared with CEM.

n	***L*×1000 letters**								
	***L*=1, SOF(*σ*)=1.7194**			***L*=5, SOF(*σ*)=1.6886**			***L*=9, SOF(*σ*)=1.8395**		
	CEM	**BAB**		**CEM**	**BAB**		**CEM**	**BAB**	
	RT	**RT**	**ND**	**RT**	**RT**	**ND**	**RT**	**RT**	**ND**
5	0.15	0.15	2	0.36	0.6	15	0.38	0.43	20
6	0.84	0.33	14	1.52	0.5	20	2.10	0.67	34
7	4.62	1.3	98	9.56	2.7	150	14.8	4.1	146
8	29.1	1.4	30	54.8	4	270	88.3	4.8	196
9	257	9	640	510	10.6	645	788	27	1341
10	2719*	23	1558	-----	-----	-----	-----	-----	-----
11	10991*	71	2456	-----	-----	-----	-----	-----	-----
12	34638*	311	17421	-----	-----	-----	-----	-----	-----

Where RT cells assign with * is the expected RT when using CEM calculated by using Lagrange interpolation. We conclude that the applying of MBAB when 5≤*n*≤12 will be suitable for reasonable time for *L*=1000 only. Notice from table 10 the big difference between the RT for CEM and MBAB.

7.3 Solving TCP using Classical BA (CBA)

7.3.1 Classical BA Description

The attack of the TCP is to find the permutation of the *n*-key integers (*ADK*). In this paper we will try to apply BA in such as optimization of a function. This TCP was chosen according to different factors such as representation of the problem can be applied more efficiently. Furthermore, this problem chosen since its own a high complexity (the size and the shape of the search space), which its, cannot be solved using traditional known searches, like exhaustive search method.

First, we should address an important question connected with **BA representation**, we should leave a bee to be an integer vector, or permutation of (1,...,*n*) integers.

For the **initial population**, the cryptanalysis process begins with a random set of *n* bees consisting of permutations of the integers 1…*n*, then it applies such key to the CT, and assesses the “fitness” of each bee by determining the extent to which the attempted decryption matches certain characteristics of plain English.

The **fitness rating** helps the cryptanalysis algorithm for TCP achieve attacking by awarding scores according to the number of times. Two, three and four letter combinations (DG, TG and QG) commonly found in English actually occurs in the decrypted text.

The correlation between fitness and the correctness of a bee is not perfect, nor can it ever be. It is always possible that BA can award relatively high fitness to bees that chance produces relatively high numbers of DG, TG and QG.

Nevertheless, the fitness rating of BA does correlate well with the correctness of a bee. To show this we applied the following steps:

1. A target bee was used to encrypt a text.
2. A sample of random bees used for decryption.
3. Each of these bees was then rated for its similarity to the target, as measured by their BA allotted fitness.
4. These ratings were then compared to the actual similarity of the target, as measured manually by awarding points, an integer, making up the bee.

In this manner, the fitness function of BA algorithm for TCP is the SOF mentioned in problem (P) s.t. Fitness (SOF) = SMHF+1−CDF.

For **parameters selection**, as the goal of this study is to verify the impact of the choice of social topologies in the behavior of the algorithm, the tuning parameters are fixed. They are set to the values that are widely used by the community and that are deemed the most appropriate ones. Table 11 shows the different parameters which are used in cryptanalysis TCP.

Table 11. Parameters selection of cryptanalysis TCP.

Parameters	Symbol	Value
Length of key	n	≥ 5
Length of text	L	≥ 500
Population size (Number of Bees)	NB	10 – 20
Number of selected sites	m	3 – 5
Number of elite sites out of m selected sites	e	2
Number of bees for elite sites	nep	5
Number of bees other selected points	nsp	3
Number of Generations	NG	≥ 100

Here we demonstrate some related subalgorithms in order to implement BA on TCP:

```
1. Subalgorithm Initialization (Start)          {Initialize population with random solutions}
   FOR $i$=Start : $NB$
       $Key_i$ = RANDOM(1..$n$)
   END;
2. Subalgorithm CAL-Fitness            {Calculate fitness function for each solution $Key_i$}
   FOR $i$=1: $NB$
      $Fit_i$= (SMHF+1-CDF)($Key_i$)
   END;
3. Subalgorithm Improve-Keys($np$,$me$)          {Improve each solution $Key_i$ for m sites}
   FOR $i$=1: $me$
     FOR $j$=1: $np$
      $r$=RANDOM($n$-1);
      $New\_Key_i$=SWAP($r$,$r$+1);
      $New\_Fit_i$ = (SMHF+1-CDF)($Key_i$)
      IF $New\_Fit_i > Fit_i$ THEN
         $Fit_i = New\_Fit_i$;
         $Key_i = New\_Key_i$;
      END;
     END;
   END;
```

The description of the Classical BA (CBA) which is be used to attack the TCP is shown below.

<u>**Algorithm CBA**</u>

```
Step (1): READ cipher text with length ($L$),
          READ $n$, $NB$, $m$, $e$, $nep$, $nsp$, $NG$.              {BA parameters}
Step (2): CALL Initialization (1). (keys of TC)             {population of Bees}
Step (3): REPEAT
Step (4):       CALL CAL-Fitness.
Step (5):       CHOOSE Best $m$ fitness Keys.
Step (6):       CALL Improve-Keys ($nep$,$e$);  CALL Improve-Keys ($nsp$,$m-e$).
Step (7):       CALL Initialization ($m$+1).
          UNTIL (Reach $NG$) OR ($Fit_i$=Fitness of Plain).
Step (8): OUTPUT: Best $Key_i$.
```

7.3.2 Experimental Results of Solving TCP using CBA

In this section, a number of experimental results are presented which outline the effectiveness of CBA attack described above. Firstly, the best way to illustrate the BA in operation is to look at the development of the population over time. We apply CBA for any chosen n, we have population of *NB* random σ *n*-sequence, in chosen iterations, we choose a sequence (*DK*) with best fitness in the population. Table 12 shows the growth of fitness value till approach the fitness (1.7234) of *ADK* using *NG*=200, *NB*=20 for *n*=8 and *L*=1000 for one run.

Table 12. The growth of fitness value for n=8 and L=1000.

Iter.	Best DK in iter.	Fitness	Time/sec
1	5 4 8 1 7 3 2 6	1.2234	0.07
2	2 3 5 6 1 8 7 4	1.3019	0.16
4	3 5 7 6 4 1 2 8	1.3388	0.29
11	4 8 6 2 3 5 7 1	1.4091	0.71
29	5 4 8 2 3 7 6 1	1.4162	1.72
72	8 4 1 2 3 5 7 6	1.4198	4.64
91	4 8 3 2 5 7 6 1	1.4655	5.69
135	7 6 1 4 8 2 3 5	1.5101	9.30
151	4 8 2 3 5 7 6 1	1.7234	12.78

The shaded cell means this key is the *ADK*.

Remark (2): From the above table:

1. The key #8 (7,6,1,4,8,2,3,5) is only a left-shifting by 3 to ADK (4,8,2,3,5,7,6,1).
2. Notice that the key # 7 (4,8,3,2,5,7,6,1) has high similarity (6 positions in sequence from 8) to the ADK (key # 9) (4,8,2,3,5,7,6,1). For fixed *NG* the ratio of similarity will be decrease as *n* increase.

We want to exploit these two notes to improve the performance of BA. The good performance for any LS means less number of iterations, less time and good convergence to the actual solution.

For *n*=5,…,10, the implementation of CBA to solve TCP has very reasonable time (0.09≤*RT*≤25.0 sec) and good AE (0≤AE≤0.0397) using *NG*=1500 for (NEx=5) examples with *L* = 1000.

In table 13, we will show the Best and average results of the implementation of CBA to solve TCP. We used *NG*=1500, *NB*=20 for *n*=11,12.

Table 13. Solving TCP using CBA for *n*=11,12 and *NEx*=5.

n	function	Fit.	Iter.	Time/sec.		*AE*
				BT	*RT*	
11	**Best**	1.7112	164	2.83	5.61	0.0000
	Av.	1.5917	642.80	27.85	41.85	0.1195
12	**Best**	1.5705	227	7.44	53.79	0.1482
	Av.	1.5043	686.40	35.22	69.73	0.2144

Where Av. is the average value of finesses of (5) examples and *AE* is the absolute error for Best and *Av*. fitness's compared with plain fitness and *BT* is the best time.

7.4 Solving TCP using Shifted Key BA

From note (1) in remark (2), to enhance the performance of CBA we suggest some modifications. These modifications summarized by applying shifting (by 1,…,*n*) to the generated keys which are generated randomly in subalgorithm Initialization, then pick the best fitness from n-shifted initial keys in order to start with good initial. We called the modified BA with the shifting initial keys Shifted Key BA (SKBA).

The following is a subalgorithm description of Shift-Key which is called by subalgorithm Shift-Key (Key_i):

Subalgorithm Shift-Key (Key_i)

```
FOR $j$=1 : $n$
    SHIFT($Key_i$) by $j$;
    CALL CAL-Fitness.
    IF MAX ($Fit_i$) THEN
        $TmpKey=Key_i$; $TmpFit=Fit_i$;
    END;
END;
Key_i=TmpKey; Fit_i=TmpFit;
```

Subalgorithm Sh_Initialization (*Start*) **= Initialization** (*Start*,*Sh*=1)

and

Algorithm Sh_Improve-Keys(*np*,*me*)= **Improve-Keys**(*np*,*me*,*Sh*=1)

Where **CALL Shift-Key** can be inserted after (Key_i = **RANDOM**(1..*n*)) statement in subalgorithm **Initialization** and **Improve-Keys(***np*,*me***)**

So algorithm SKBA is:

Algorithm SKBA=CBA(*Sh*=1**)**

Table 14 shows the growth of fitness value till approach the fitness (1.7234) of *ADK* using *NG*=200, *NB*=20 for *n*=8 and *L*=1000 in algorithm SKBA.

Table 14. The growth of fitness value for *n*=8 and *L*=1000 in SKBA.

Iter.	Best *DK* in iter.	SOF(*DK*)	*RT*
1	6 5 1 2 4 3 8 7	1.2234	0.04
2	1 5 2 4 3 6 8 7	1.4655	0.48
25	1 5 2 4 3 6 8 7	1.4822	6.95
27	2 5 3 1 4 6 8 7	1.4907	7.43
45	2 5 1 4 3 6 8 7	1.7234	12.01

In figure 4, the growth of fitness value of BA and SKBA is shown, it's clear that the performance of SKBA is better than CBA.

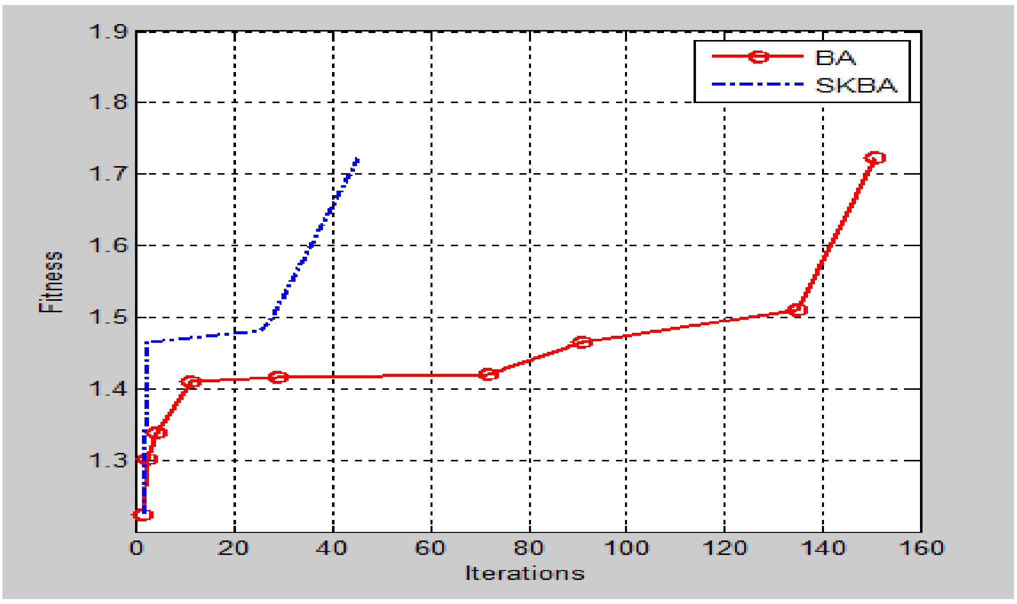

Figure 4. The growth of fitness value of BA and SKBA.

Table 15 shows the comparison results between the Best and *Av*. (some) results of table 13 and implementation of SKBA for fitness, iteration, *BT*, *RT* and *AE* for *n*=9,...,12.

Table 15. The comparison results between CBA and SKBA for n=9,...,12.

n	Fun.	Alg.	Fit.	Iter.	Time/sec.		*AE*
					BT	***RT***	
9	**Best**	**CBA**	1.7202	23	0.07	3.07	0.0000
		SKBA	1.7202	9	3.15	3.15	0.0000
	Av.	**CBA**	1.6823	131.6	4.65	6.82	0.0379
		SKBA	1.7202	85.20	15.81	18.91	0.0000
10	**Best**	**CBA**	1.7234	159	2.89	2.89	0.0000
		SKBA	1.7234	123	17.02	20.23	0.0000
	Av.	**CBA**	1.6550	413.60	21.50	25.40	0.0684
		SKBA	1.7234	358.40	106.63	176.63	0.0000
11	**Best**	**CBA**	1.7112	64	2.83	5.61	0.0000
		SKBA	1.7112	113	44.54	58.02	0.0000
	Av.	**CBA**	1.5917	642.80	27.85	41.85	0.1195
		SKBA	1.6137	538.40	203.58	372.42	0.0975
12	**Best**	**CBA**	1.5705	227	7.44	53.79	0.1482
		SKBA	1.5593	155	53.52	526.00	0.1593
	Av.	**CBA**	1.5043	686.40	35.22	69.73	0.2144
		SKBA	1.5364	588.00	308.85	539.15	0.1822

From above table, notice that the SKBA makes little differences in obtaining good results but it takes more time than CBA.

8. Successive Rules

8.1 Successive Rule Concept

Let $P(i,j)$ be the probability of digit i successes digit j in the key DK_m (denoted by $i\rightarrow j$) (or we say column i successes column j in the text M_m using key DK_m), where $1\leq m\leq n!$, and a **threshold (T_1)** for the acceptance of $P(i,j)$ s.t.

$$P(i,j) \geq T_1, \text{ where } 0<T_1\leq 1, \qquad (12)$$

In another word, if $P(i,j)$ satisfies condition (12) then its called **accepted probability** $AP(i,j)$.

Definition (1): In TCP, if digit i successes digit j in the key DK_m to decrypt the text M_m with good accepted probability $AP(i,j)$ then we say that DK_m (or M_m of TCP) is submitted to **successive rule** (SR).

In another words, we can define the SR's by the rules which are enforcing the obtained sequence (*DK*) to be arranged in some specific order.

We believe that the calculation of $P(i,j)$ is relevant to some iterative solving methods of TCP which we can generate the *DK* in somehow. The BAB and LS methods can be considered as kinds of these iterative solving methods. In this paper we are focus in generating *DK* which is submit to a good SR (SR with $AP(i,j)$) using BAB and LS (like BA).

Let's suppose that some SR's are explored, now how we can exploit these SR's to increase the performance of solving techniques of TCP?

Example (2): Let σ be a 5-sequence digits with the following SR: $2\rightarrow 4$ and $3\rightarrow 1$, s.t. the number of SR (*NSR*)=2, then σ will have the following arrangements: (2,4,5,3,1), (2,4,3,1,5), (3,1,5,2,4),…,etc. While if σ enforced by the following SR: $2\rightarrow 4$, $4\rightarrow 5$ (2-4-5) and $3\rightarrow 1$ (3-1), s.t. *NSR*=3 then σ will have the following arrangements: (2,4,5,3,1) and (3,1,2,4,5) only.

8.2 Generating Subsequences from SR

Let σ be an n-sequence digits (*DK*), s.t. $\sigma=(1,2,\ldots,n)$, if σ has NSR of SR's, if the digits $i\rightarrow j$ and $j\rightarrow \ell$, then we can obtain a subsequence $S_k=(i\text{-}j\text{-}\ell)$ with length 3. In general, we can generate $N\leq n$ number of strings (subsequences) S_k each with length SL_k, s.t. $1\leq k\leq N$, then the initial N-sequence is $\pi=(S_1,S_2,\ldots,S_N)$ obtained from σ of n-sequence after applying SR.

For example (2), σ=(1,2,3,4,5), *NSR*=3, S_1=(2,4,5) and S_2=(3,1) with SL_1=3 and SL_2=2 respectively, then *N*=2, s.t. π=(S_1,S_2)=(2-4-5,3-1). In general, notice $n=\sum_{k=1}^{N} SL_k$ and $NSR=\sum_{SL_k\geq 2}(SL_k-1)$.

Table 16 shows the SR subsequences (S_k), with lengths (SL_k) for *n*=11,...,20.

Table 16. The SR subsequences (S_k), with lengths (SL_k) for *n*=11,...,20.

n	*NSR*	*N*	SL_k	π=(S_k), *k*=1,…,*N*
11	8	3	3,7,1	(6-3-8),(2-11-7-9-4-1-10),(5)
12	8	4	4,6,1,1	(3-8-4-11),(12-7-9-5-1-10),(6),(2)
13	8	5	3,5,3,1,1	(7-10-5),(3-9-4-12-8),(1-11-6),(2),(13)
14	8	6	4,3,2,3,1,1	(4-13-9-6),(2-14-8),(3-10),(11-5-1),(7),(12)
15	8	7	3,2,5,2,1,1,1	(9-11-6),(8-4),(10-5-14-12-2),(1-13),(3),(15),(7)
16	8	8	2,4,2,2,3,1,1,1	(12-6),(11-5-15-13),(3-16),(2-7),(14-8-4),(10),(1),(9)
17	8	9	2,3,3,4,1,1,1,1,1	(17-9),(5-16-14),(8-4-11),(13-6-1-15),(2),(7),(3),(10),(12)
18	8	10	2,2,2,3,2,3,1,1,1,1	(4-13),(16-12),(5-1),(2-11-8),(3-14),(10-7-6),(9),(15),(18),(17)
19	8	11	3,2,2,2,2,2,2,1,1,1,1	(14-17-13),(6-1),(2-12),(9-4),(15-11),(8-7),(10-16),(19),(18),(3),(5)
20	9	12	2,3,2,2,2,2,2,1,1,1,1,1	(5-15),(18-14-6),(1-2),(13-9),(16-12),(8-7),(10-17),(20),(19),(3),(11),(4)

9. Applying Exact and SKBA Methods with SR to Solve TCP

9.1 Applying CEM with SR to Solve TCP

Notice from table 16 that if *N*≤9 we can apply CEM to solve TCP with *n*=11,…,17, to obtain exact solution in reasonable time. To find *ADK* for each n mentioned in table 16 we have to apply CEM for *N*≤9. The CEM applied for σ of *n*-sequences consists of *N*-subsequence to obtain π of *N*-sequences where some subsequence is multi digits, then we called it **multi digits CEM** (**MDCEM**). Now we can propose a subalgorithm MDCEM:

```
Subalgorithm MDCEM
READ n, N, k=1,…,N, ($SL_k$,$S_k$).
MDCEM=CEM(N,$SL_k$,$S_k$).
```

Table 17 shows the results of applying MDCEM with SR using table 16 for *n*=11,…,17, RT(*N*) and ERT(*N*) the required and expected required time.

Table 17. The results of applying MDCEM with SR for *n*=11,…,17.

n	*N*	*N*!	*ADK*, SOF(*ADK*)≈1.72	MDCEM	
				RT(*N*)	ERT(*n*)
11	3	6	(2-11-7-9, 4-1-10 ,6-3-8-5)	0.02	10991≈3h
12	4	24	(2-12-7, 9-5, 1-10-6, 3-8-4-11)	0.04	34638≈10h
13	5	120	(2-13, 7-10-5-1, 11-6-3, 9-4-12, 8)	0.16	91940≈25h
14	6	720	(2-14-8, 11-5, 1-12, 7, 3-10-4, 13-9-6)	1.41	215228≈60h
15	7	5040	(3-15, 9, 11-6, 1-13-8, 4-10, 5-14, 12-2-7)	9.69	------
16	8	40320	(3, 16-9-12, 6-1, 14-8, 4, 11-5-15, 13-2, 7-10)	76.09	------
17	9	362880	(3-17-9, 13, 6-1,15-8, 4-11-5, 16, 14-2, 7,10-12)	658.8	------

9.2 Applying New BAB with SR to Solve TCP

As well known, each arc in classical search tree of BAB method represents by single digit of *n*-sequence, and then branching from a node. We can exploit the SR to decrease the number of levels in BAB's search tree and solve a TCP with *N*-1 levels instead of *n*-1 levels by obtaining sequences π of *N*-sequence. To make this happen we have to consider each arc as a string S_k of digits with length SL_k.

Now we want to exploit the SR to construct a new style of BAB search tree. Each arc of BAB search tree may represents a subsequence of the main sequence. In section 7.2 we propose a new BAB method we called it

MBAB, this method will be applied to find sequences π of N-sequence with elements S_k. We call the new BAB method by **multiple digits BAB** (MDBAB) method. The MDBAB algorithm is shown below.

MBAB algorithm

```
STEP(1): INPUT CT, $L$, $N$;
            LB=1.0, $\ell=0$, $s\pi=(S_1,S_2,\ldots,S_N)$, $ND=N$, [FOR $k=1,\ldots,N$ $SEQ(k)=k$];
STEP(2): $\ell=\ell+1$, $m=0$;
            FOR $k=1,\ldots,ND$
                Branching from node last string $\ell$ in $SEQ$;
                $UNSEQ= s\pi$ without $SEQ$;
                $\pi$ = concatenate($SEQ$,$UNSEQ$);
                Calculate $UB_k$= SOF($\pi$)                    {in level –$\ell$ }
                IF $UB_k \geq LB$ THEN
                    $m = m + 1$;
                    $LIST(m , :) = \sigma$; $SUB(m) = UB_k$;
                END;
            END;
STEP(3): LB=mean {$SUB$};
            $BestFit = \max_{1\le i\le m}\{SUB\}$, BestDK= $LIST(i)$;
            $SEQ$=cut from $LIST$ first $\ell$ strings, $LIST=\Phi$, $SUB=\Phi$; $ND=m$;
            IF $\ell=N-1$ STOP ELSE GOTO STEP(2);
            IF $BestFit \geq 1.68$ STOP;
STEP(4): OUTPUT $BestFit$, $BestDK$;
```

Example (3): Let n=6, with σ of 6-sequence has SR with the following subsequencs: S_1=(1), S_2=(4), S_3=(3,5), S_4=(6,2), with lengths 1,1,2,2 respectively this mean N=4 and $\pi=(S_1,S_2,S_3,S_4)$=(1,4,3-5,6-2). First, set initial LB (ILB)=1.0.

For level 1: $UB_{\{1\}}$((1,4,3-5,6-2))=1.3513 (≥ILB), $UB_{\{4\}}$((4,1,3-5,6-2))=1.2717, $UB_{\{3\text{-}5\}}$ ((3-5,1,4,6-2))=1.2281, $UB_{\{6\text{-}2\}}$((6-2,1,4,3-5))=1.3302, so we branch from the nodes with good UB's, the new LB_1=mean($UB_{\{1\}}$)=$UB_{\{1\}}$=1.3513.

For level 2: from node with $UB_{\{1\}}$, $UB_{\{4\}}$((1,4,3-5,6-2))=1.3513 (≥LB_1), $UB_{\{3\text{-}5\}}$ ((1,3-5,4,6-2))=1.2312, $UB_{\{6\text{-}2\}}$((1,6-2,4,3-5))=1.7187 (≥LB_1), so we branch from the nodes with $UB_{\{4\}}$=1.3513 and $UB_{\{6\text{-}2\}}$=1.7178, the new LB_2=mean($UB_{\{1\}}$,$UB_{\{6\text{-}2\}}$)=1.5350.

For level 3: from the node with $UB_{\{4\}}$, $UB_{\{3\text{-}5\}}$((1,4,3-5,6-2))=1.3513 and $UB_{\{6\text{-}2\}}$((1,4,6-2,3-5))=1.3251. From node with $UB_{\{6\text{-}2\}}$, $UB_{\{4\}}$((1,6-2,4,3-5))=1.7187 (≥LB_2) and $UB_{\{3\text{-}5\}}$((1,6-2,3-5,4))=1.3547 so the only UB≥LB_2 is the node with $UB_{\{4\}}$ to obtain the best fitness = 1.7187 hence the sequence π=(1,6-2,4,3-5) is the *ADK* (see figure 5).

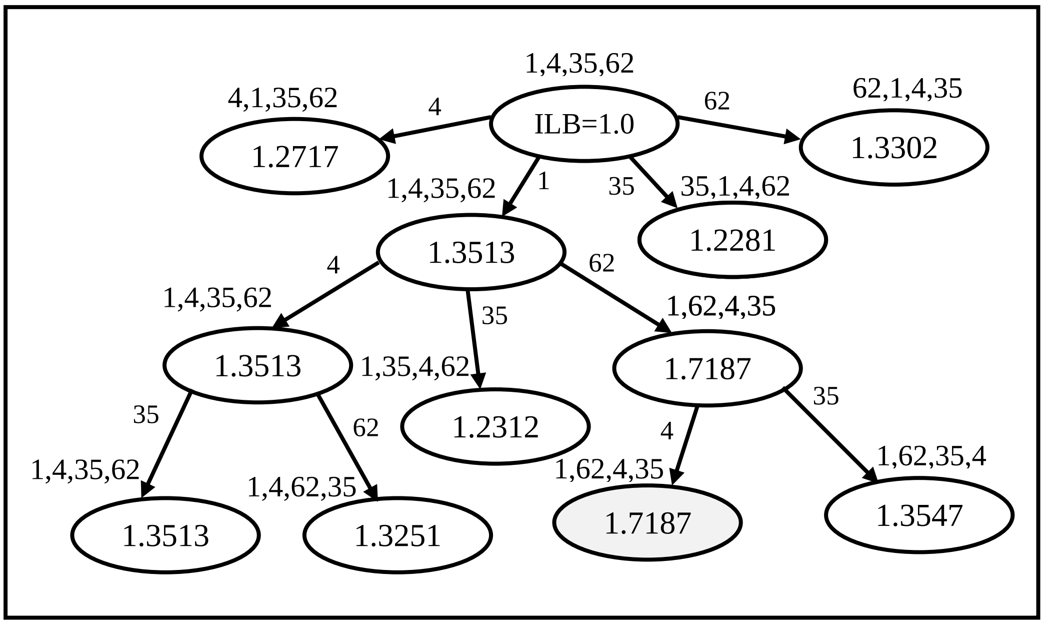

Figure 5. Applying of MDBAB for n=6.

From figure 5, the optimal solution is σ=(1,6,2,4,3,5), with SOF(σ)=1.7187. Since N=4, then the MDBAB search tree has 3 levels. The shaded node is the optimal solution.

Remark (3): For $N\leq 9$, if the value of the upper bound $UB_k(\pi)\approx 1.7$ (which is the fitness of text using *ADK*) is obtained in any level $k\leq N$ when applying MDBAB we can stop the process and no need for more branching.

We recall table 16 again, we will apply MDBAB to solve TCP with key size n=16,…,20, to obtain exact solution in reasonable time. Table 18 shows the results of applying MDBAB with SR using table 16 for n=16,…,20.

Table 18. The results of MDBAB with SR for n=16,…,20.

n	***N***	***N*!**	***π=ADK*, SOF(*π*)≈1.72**	**MDCEM**	**MDBAB**
				RT(*N*)	**RT(*N*)**
16	8	40320	(3,16,9,12,6,1,14,8,4,11,5,15,13,2,7,10)	76.09	0.35
17	9	362880	(3,17,9,13,6,1,15,8,4,11,5,16,14,2,7,10,12)	658.8	0.52
18	10	3628800	(4,13,16,12,5,1,2,11,8,3,14,10,7,6,9,15,18,17)	3306*	1.25
19	11	39916800	(5,14,17,13,6,1,2,12,9,4,15,11,8,7,10,16,19,18,3)	11659*	23.0
20	12	479001600	(5,15,18,14,6,1,2,13,9,4,16,12,8,7,10,17,20,19,3,11)	32774*	35.5

The RT(N) signed with * is the expected time which is interpolated by using Lagrange interpolation. Now we can propose a subalgorithm MDBAB:

Subalgorithm MDBAB

READ n,N,SL_k,S_k, k=1,…,N.

MDBAB=MBAB(N,SL_k,S_k)

9.3 Applying SKBA with SR to Solve TCP

Before we discuss the implementation of CBA using SR to Solve TCP, we have to focus more light about how we can generate the SR's? lets start with this example.

Example (4): First, let's apply CBA for n=10, we have population of NB=20 random σ 10-sequence, in chosen iterations, we choose a sequence (DK) with best fitness in the population. Table 19 shows the SR using NG=200, for n=10 and L=1000.

Table 19. The growth of fitness value for n=10 and L=1000.

Key#	**Iter.**	***σ=DK***	**Fitness**	***N***	***RT***
1	1	**8 3** 2 1 4 5 6 10 7 9	1.2234	9	0.04
2	2	5 7 6 4 8 **1 9** **2 10** 3	1.2679	8	0.11
3	3	**2 10 5** **3 4** **1 9** 7 6 8	1.3876	6	0.17
4	6	7 6 4 **1 9** 3 8 2 5 10	1.3955	9	0.32
5	8	**2 10 5** **3 4** **1 9** 7 6 8	1.4056	6	0.42
6	10	**7 1** 10 3 **9 8** **4 6 2** 5	1.4487	6	0.52
7	23	7 6 4 **1 9** 3 8 2 5 10	1.4623	9	1.15
8	112	**1 9** **3 4** 8 **6 2 10 5** 7	1.5752	5	6.48
9	151	**3 4 6 2** **7 1 9** **10 5** **8**	1.6132	4	13.51
ADK		7 1 9 8 3 4 6 2 10 5	1.7234	1	

where the underline subsequence represents the SR's.

Notice that as fitness increase (and iteration) the number of similarity digits with *DK*, the number and the length of strings S_k and the number of correct position-digits are increased.

From note (2) in remark (2), another improvement of CBA was suggested. These improvements represented by other modifications. These modifications summarized by applying shifting (by 1,…,n) and using some SR explored by implementation of CBA first, then we use these SR to generate keys in subalgorithm Initialization which are approximate the *ADK* with best fitness in order to start with good initial and to increase the probability to obtain the *ADK* in less iterations and time. We called the new modified BA with the SR keys BA (SRKBA). The proposed algorithm SRKBA included the SKBA. Now let:

$N(i,j)$: the number of frequency for i successes j in m sites.

Then the probability for i successes j in m sites for the total number of generations is:

$$P(i,j)=\frac{N(i,j)}{m*NG} \tag{11}$$

Table 20 shows the SR using NG=500, for n=10 and L=1000 and T_1=0.25.

Table 20. The SR using NG=500, for n=10 and L=1000.

SR	**1st (*i*)**	**2nd (*j*)**	***N(i,j)***	***P(i,j)***	**SR with *N*=4**		
1	1	9	690	0.276*	***k***	***SL$_k$***	***S$_k$***
2	2	10	290	0.116#			
3	3	4	1380	0.552*	1	4	3,4,6,2
4	4	1	312	0.125			
5	4	6	903	0.361*	2	3	7,1,9
6	6	2	903	0.361*			
7	6	8	90	0.036	3	2	10,5
8	7	1	903	0.361*			
9	7	6	21	0.008	4	1	8
10	10	5	1476	0.590*			

Then the SR with accepted probability (signed with *) are: S_1=3,4,6,2, S_2=7,1,9 and S_3=10,5, S_4=8 so the accepted number of SR (*ASR*) is N=4. The SR signed with (#) is correct but because that $P(2,10)<T_1$ it has been ignored.

Note that every obtained DK consists of number of subsequences S_k each has length SL_i, i=1,..,N, where N is the number of accepted subsequences for S_k, e.g. from table 20, the last *DK* has 4 subsequences each with length 4,3,2 and 1 respectively.

To obtain good or certain probability we can apply CBA with many runs (say w), each time $P_k(i,j)$, k=1,…,w may has new value, each $P_k(i,j)$ appear w_{ij} times $w_{ij}\in\{1,\dots,w\}$, $\forall i,j$ (if w_{ij}=1 it will be ignored unless $P_k(i,j)\geq 0.5$), then the arithmetic mean of probabilities:

$$T(i,j)=\frac{1}{w_{ij}}\sum_{k=1}^{w_{ij}}P_k(i,j) \tag{13}$$

When $T(i,j)$≥threshold (T_2) or $w_{ij}>2$ then the SR will be accepted. Table 21 shows the calculation of $T(i,j)$ for NG=2000, L=1000 and with 5 runs (w=5) for CBA.

Table 21. The calculation of $T(i,j)$ for NG=2000, n=10, L=1000 and w=5.

SR, *ADK*=(2,10,6,8,4,1,9,5,3,7)								
1-9			**2-10**			**3-7**		
$P_k(1,9)$	$w_{1,9}$	$T(1,9)$	$P_k(2,10)$	$w_{2,10}$	$T(2,10)$	$P_k(3,7)$	$w_{3,7}$	$T(3,7)$
0.261 0.278 0.603 0.009	4	0.288	0.769 0.052 0.598 0.781 0.586	5	0.557	0.350 0.575 0.534 0.003	4	0.366
4-1			**4-9**			**5-3**		
$P_k(4,1)$	$w_{4,1}$	$T(4,1)$	$P_k(4,9)$	$w_{4,9}$	$T(4,9)$	$P_k(5,3)$	$w_{5,3}$	$T(5,3)$
0.736 0.296 0.609 0.269	4	0.478	0.062 0.014	2	0.038	0.388 0.296 0.726 0.623 0.046	5	0.416
6-8			**8-4**			**8-6**		
$P_k(6,8)$	$w_{6,8}$	$T(6,8)$	$P_k(8,4)$	$w_{8,4}$	$T(8,4)$	$P_k(8,6)$	$w_{8,6}$	$T(8,6)$
0.431 0.306 0.295 0.502	4	0.384	0.227 0.697 0.074	3	0.332	0.001 0.164	2	0.083
9-5			**10-6**			Good SR		
$P_k(9,5)$	$w_{9,5}$	$T(9,5)$	$P_k(10,6)$	$w_{10,6}$	$T(10,6)$	NSR=9	1-9,2-10,3-7, 4-1,5-3,6-8, 8-4,9-5,10-6	
0.420 0.776 0.601 0.723	4	0.630	0.570 0.254 0.145 0.269 0.016	5	0.251	N=1		

The shaded cells are SR with accepted probability but they ignored because there exists better than them (e.g. 8-6 is ignored because 8-4 better).

Now to apply MDCEM, we check if N less or equal to a reasonable number can be manipulated by MDCEM (N≤8). While if (8<N≤12) we can applied MDBAB. From example (4), for key#9, N=4, so TCP can be solved by MDCEM in 4! (=24) states. Otherwise for (N>13), we reapplied SRKBA to solve TCP or to obtain more new ASR. These procedures are repeated until the TCP is solved.

We introduce subalgorithm **FIND_SR** to obtain the SR when applying CBA.

Subalgorithm FIND_SR

```
FOR $i$=1 : $m$
   FOR $j$=1:$n$-1
        $n_1=Key_{i,j}$;  $n_2=Key_{i,j+1}$; $N(n_1,n_2)+1$;
END {$i,j$};
Calculate $P(n_1,n_2)= N(n_1,n_2)/(m*NG)$;
IF $P(n_1,n_2) \geq T_1$ THEN FIND $(N,S_k)$, $k$=1,…,$N$;
```

The proposed SRKBA is as shown below.

Algorithm SRKBA

```
Step (1): READ cipher text with length ($L$),
               READ $n, NB, m, e, nep, nsp, NG$.            {CBA parameters}
Step (2): CALL Sh_Initialization (1). (keys of TC)          {population of Bees}
Step (3): REPEAT
Step (4):      CALL CAL-Fitness.
Step (5):      CHOOSE Best $m$ fitness Keys.
Step (6):      CALL Improve-Keys ($nep,e$);  CALL Improve-Keys ($nsp,m-e$).
Step (7):      CALL FIND_SR.
Step (8)       IF $N \le 8$ THEN CALL MDCEM($N$) GOTO Step (10).
Step (8)       IF $8 < N \le 12$ THEN CALL MDBAB($N$) GOTO Step (10).
Step (9):      CALL Initialization ($m$+1,$S_k$).
          UNTIL (Reach $NG$) OR ($Fit_i$=Fitness of Plain).
Step (10): OUTPUT: Best $Key_i$.
```

Remark (4): As well as using high *NG* when applying CBA or SRKBA this means obtain high number of SR (this decrease *N*) and that will helpful in obtaining the *ADK* in less number of iterations and time when applying SRKBA. Table 22 illustrates the implementation of SKBA to solve TCP for *n*=13,…,20 using *NG*=3000 and *L*=1000 in number of solving stages.

Table 22. Using SRKBA to solve TCP for *n*=13,…,20, *NG*=3000 and *L*=1000.

n	Ex.	Solving Stages		*n*	Ex.	Solving Stages			
		1	2			1	2	3	4
13	1	AS	CEM(5)	17	1	AS	BAB(12)	-----	-----
	2	AS	CEM(5)		2	AS	AS	CEM(5)	-----
	3	AS	CEM(6)		3	AS	BAB(12)	-----	-----
	4	AS	BAB(9)		4	AS	AS	CEM(5)	-----
	5	AS	CEM(7)		5	AS	BAB(11)	-----	-----
14	1	AS	CEM(6)	18	1	AS	AS	CEM(5)	-----
	2	AS	BAB(8)		2	AS	AS	CEM(2)	-----
	3	AS	CEM(4)		3	AS	AS	CEM(6)	-----
	4	AS	CEM(6)		4	AS	AS	CEM(6)	-----
	5	AS	BAB(9)		5	AS	AS	CEM(7)	-----
15	1	AS	CEM(7)	19	1	AS	AS	BAB(9)	-----
	2	AS	BAB(8)		2	AS	AS	AS	BAB(10)
	3	AS	BAB(8)		3	AS	AS	CEM(5)	-----
	4	AS	BAB(9)		4	AS	AS	BAB(8)	-----
	5	AS	BAB(10)		5	AS	AS	BAB(8)	-----
16	1	AS	BAB(8)	20	1	AS	AS	AS	CEM(4)
	2	AS	CEM(7)		2	AS	AS	AS	CEM(5)
	3	AS	BAB(12)		3	AS	AS	AS	BAB(8)
	4	AS	BAB(10)		4	AS	AS	AS	CEM(2)
	5	AS	BAB(8)		5	AS	AS	AS	BAB(8)

Where AS is an approximation solution. The CEM(*N*) and BAB(*N*) mean applied CEM and BAB for *N* respectively. Notice that as *n* increases we need more solving stages in order to obtain *ADK*.

Remark (5): We may increase the performance of SRKBA, and decreasing the solving stages to find *ADK* to TCP by running SRKBA for w times and obtain the good probabilities to pick the good SR. Table 23 shows the results of applying SRKBA for *w*=5, to solve TCP for *n*=13,…,20, *NG*=3000 and *L*=1000.

Table 23. Using SRKBA to solve TCP for n=13,…,20, NG=3000, L=1000, for w=5.

n	**Solving Stages**	***n***	**Solving Stages**	
	1		**1**	**2**
13	CEM(2)	**17**	CEM(6)	-----
14	CEM(5)	**18**	BAB(9)	-----
15	CEM(5)	**19**	BAB(10)	-----
16	CEM(6)	**20**	AS	CEM(4)

Compare the results of number of solving stages of solving TCP of table 23 with results of table 22.

10. Conclusions

1. We don't takes in the consideration in this study the space character "-" between words, which is considered a good weak point can be exploited by cryptanalyst, therefore the designers exclude it and that is the actual status.
2. Although the BA is an approximate algorithm, in this paper we are interest in using BA to find exact solution for TCP more than finding an approximation solution.
3. We notice the role of SR in solving the TCP, in general we convince that this role will be extended to all COP.
4. There are two sources to obtain SR, first is the BAB and the second is the LS.
5. As the number of generations in applying BA is increased then the number of obtained SR will be increased this implies that (N) the number of strings (subsequences) S_k is decreased this means the ADK will be obtained in less possible time.
6. In order to solve TCP in less possible time, we must find as high as possible number of SR, this implies that the number of subsequence (N) will be decreased with SL_k approach n.
7. A study can be made to show the role of changing the parameters of BA in solving TCP in order to tune these parameters in suitable form.
8. As future work, we suggest using another LS with SR to solve TCP or any COP.
9. According to what we discuss, we may suggest using a general solving system to solve any COP under the condition that the COP submits to "good" SR's which can be used to approach the actual solution in reasonable time.

References

Ahmed T., Laith A., and Hashim K. (2014), "Attacking Transposition Cipher Using Improved Cuckoo Search", Journal of Advanced Computer Science and Technology Research, Vol.4 No.1, pp.22-32.

Ali I. K. (2009), "Intelligent Cryptanalysis Tools Using Particle Swarm Optimization", Ph. D., Dept. of Computer Science, University of Technology.

Ashraf A., Michael P. and Marco C. (2009), "Bees Algorithm", *Manufacturing Engineering Center*, Cardiff University, Wales,UK.

Clark A. J., "Optimization Heuristics for Cryptology", Ph.D. Thesis, Information Security Research Centre, Faculty of Information Technology, Queensland University of Technology, 1998.

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

Matthews R. A. J., "The Use of Genetic Algorithms in Cryptanalysis", *Cryptologia*, 17(2):187–201, April 1993.

Pfleeger C. P. (1998), "Security in Computing Algorithms", Abroad Book.

Pham D. T., Ghanbarzadeh A., Koc E., Otri S., and Zaidi M. (2006), "The Bee's Algorithm – a Novel Tool for Complex Optimization Problems". In: Pham D.T., Eldukhri E., Soroka A. J. ed(s) *2nd Virtual International Conference on Intelligence Production Machines and Systems* (IPROMS 2006). Elsevier, Oxford, 2006, pp 454-459.

Russell M. D., Clark J.A., and Stepney S. (2003), "Making the Most of Two Heuristics: Breaking Transposition Ciphers with Ants", *The Congress on Evolutionary Computation* CEC 03, Vol. 4, pp.2653-2658.

Xiaodong L., Gao L. and Gao H. (2003), "Swarm Intelligence and its Applications", *Congress on Evolutionary Computation*, Canberra, Australia, 8th- 12th December.