CHAPTER FOUR

CRYPTANALYSIS OF TRANSPOSITION CIPHER PROBLEMS USING COMBINATORIAL OPTIMIZATION PROBLEMS TECHNIQUES

4.4 Solving TCP using Exact Methods

In this section, for TCP, we will apply the exact methods which represented by complete enumeration and branch and bound methods. These methods are chosen to solve the TCP using the proposed cryptanalysis tools.

4.4.1 Solving TCP using Complete Enumeration Method (CEM)

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 (4.6) for the TOF and SOF (3,σ).

Table (4.6): L=999, CT='etheu stins hibpu cliia toonr ft…';

	DK	Dd	Dt	Dq	SMHF	CDF	SOF	Text
1	3,2,1	654	151	169	0.9764	0.6284	1.3480	HTESUENITIHSUPBILCTA
2	3,1,2	642	102	116	0.8621	0.5288	1.3332	HETSEUNTIISHUBPICLTI
3	2,3,1	727	321	323	1.3747	0.3516	2.0232	THEUSEINTHISPUBLICAT
4	2,1,3	634	140	154	0.9303	0.6228	1.3076	TEHUESITNHSIPBULCIAI
5	1.3.2	647	154	112	0.9152	0.6008	1.3144	EHTESUTNISIHBUPCILIT
6	1,2,3	647	113	120	0.8821	0.4894	1.3927	ETHEUSTINSHIBPUCLIIA

The σ=ADK has different closed values s.t. 1.6886≤SOF(n,σ) ≤2.0232. So we can consider that σ=ADK if SOF(n,σ) ≥ 1.68 while if SOF(n,σ)≤1.32 then σ≠ADK.

4.4.2 Solving TCP using New 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 chapter, we will see how the BAB method is efficient in solving the TCP? First, since we want to maximize SOF(n,σ) for TCP so we have to find a suitable lower bound (LB), first set LB=SOF(n,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 for TCP. 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. To improve the value of LB in each level, we 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, which is shown in the next page.

4.5 Successive Rules of DK for TCP

4.5.1 Successive Rule (SR) Concept

Let P(i,j) be the probability of digit i precedes digit j in the key DK_r (denoted by $i\to j$) (or we say column i precedes column j in the text M_r using key DK_r), where $1\le r\le n!$, and a **threshold (T_1)** for the acceptance of P(i,j) s.t.

$$P(i,j) \ge T_1, \text{ where } 0<T_1\le 1, \qquad \ldots(4.11)$$

In another word, if P(i,j) satisfies condition (4.11) then its called **accepted probability** AP(i,j).

Definition (4.1): In TCP, if $i\to j$ in the key DK_r to decrypt the text M_r with good accepted probability AP(i,j) then we say that DK_r (or M_r of TCP) is submitted to **successive rule** (SR).

In other 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 local search methods can be considered as kinds of these iterative solving methods. In this thesis 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 (4.1): Let σ be a 5-sequence digits with the following SR: $2\to 4$ and $3\to 1$, 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: $2\to4$, $4\to5$ (2-4-5) and $3\to1$ (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.

4.5.2 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 $i\to j$ and $j\to\ell$, then we can obtain a subsequence $S_k=(i\text{-}j\text{-}\ell)$ with length 3. In general, we can generate $m\leq n$ number of strings (subsequences) S_k each with length SL_k, s.t. $1\leq k\leq m$, then the initial m-sequence is $\pi=(S_1,S_2,\ldots,S_m)$ obtained from σ of n-sequence after applying SR.

Example (4.2): Let $\sigma=(1,2,3,4,5)$, NSR=3, $S_1=(2,4,5)$ and $S_2=(3,1)$ with $SL_1=3$ and $SL_2=2$ respectively, then m=2, s.t. $\pi=(S_1,S_2)=(2\text{-}4\text{-}5,3\text{-}1)$. In general, notice $n=\sum_{k=1}^{m}SL_k$ and $NSR=\sum_{SL_k\geq2}(SL_k-1)$. Table (4.15) shows the SR generates subsequences (S_k), with lengths (SL_k) for n=11,...,20.

Table (4.15): The SR generates subsequences (S_k), with lengths (SL_k) for n=11,...,20.

n	**NSR**	**m**	**SL_k**	**$\pi=(S_k)$, k=1,…,m**
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	8	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)