Protocol Test Sequence Generation Using MUIOS Based on TSP Problem

G. Prem Kumar and P. Venkataram
Department of Electrical Communication Engineering
Indian Institute of Science; Bangalore - 560 012, INDIA
(Tel : (+91) (080) 3340855; Fax : (+91) (080) 3347991
e-mail : {prem, pallapa}@ece.iisc.ernet.in)

Abstract

In this paper, a method for optimal test sequence generation using Multiple Unique Input/Output Sequences (MUIOS) is presented. The test sequence generation problem using MUIOS is viewed as an asymmetric travelling sales person problem and is solved using *simulated annealing* methodology. This technique is applied to various test cases and the results are compared with the results obtained by using rural Chinese postman algorithm.

Keywords

Protocol conformance testing, formal methods for test sequence generation, multiple input/output sequences, travelling sales person problem, simulated annealing, rural Chinese postman algorithm.

1 INTRODUCTION

Conformance testing for communication protocols is to test whether a protocol under test, conforms to the specifications laid down during the design (Aho, 1991). The conformance testing of protocols is being done in two ways (Sarikaya, 1984) (Sarikaya, 1987) (Linn, 1989/1990). First approach is to build a test system, which simulates the protocol. These test systems designed by users and suppliers to their requirements, often have following disadvantages : (a) it is a costly effort to build individual testers, and (b) standardization of protocol products is difficult. Second approach deals with generation of efficient test cases for testing the implemented protocol product. The desirable features are short test sequences and less human intervention. Following the second way of conformance testing, this paper gives an efficient method of test sequence generation.

The formal methods used to generate conformance test sequences, that are modeled as Finite State Machines (FSMs) (Kohavi, 1978) are Transition Tours (T-method) (Bosik, 1991) (Naito, 1981) Distinguishing Sequence (D-method) (Gonenc, 1970) (Ural, 1992) (Chow, 1978), Characterizing Sequences (W-method) (Bosik, 1991) and Unique

Input/Output Sequences (U-method) (Aho, 1991) (Bosik, 1991) (Sidhu, 1989) (Ural, 1992) (Sabnani, 1988) (Miller, 1993).

We formulate the problem of test sequence generation (TSG) based on MUIOS as an asymmetric travelling sales person problem (we call this as TSP) where sub-sequences (defined later) corresponding to an edge of the FSM are considered as a node and bridge length between the nodes as elements of the TSP-matrix. We solve the TSP problem thus formulated by using Simulated Annealing (SA) methodology.

In Section 2, the notation used in this paper and an overview of some of the protocol test sequence generation methods are introduced. In Section 3, the asymmetric travelling sales person problem and its solution using Simulated Annealing methodology are described. Section 4 explains the method of test sequence generation. Section 5 illustrates the proposed method with an example. Section 6 presents the results and comparison of the proposed method with the one solved using Rural Chinese Postman (RCP) problem. Finally, some concluding remarks are given in Section 7.

2 NOTATION AND OVERVIEW OF FORMAL METHODS FOR TEST SEQUENCE GENERATION

Before giving an overview of formal methods, we introduce the notation related to finite state machines, graph theory and protocol conformance testing, that is followed in this paper.

2.1 Definitions

Definition 1: A protocol represented as a FSM consists of a set of states and a set of transitions. It can be modelled as a directed graph $G = (V, E)$, where V is a set of vertices corresponding to the states of the FSM and E is a set of edges corresponding to the transitions of the FSM. G is called as *FSM graph*.

Definition 2: A FSM M, is *deterministic* if there is at most one transition defined for each input at each state of M.

Definition 3: States s_i and s_j of M are *equivalent* if they have identical edges.

Definition 4: M is *minimal* if none of its states are equivalent.

Definition 5: M is *strongly connected* if there exists a path between every pair of states.

Definition 6: M is *fully specified*, if every state can accept all the possible inputs. Otherwise, M is *partially specified*.

Definition 7: A state s_i has a *self loop* if it has a transition that starts and ends in s_i.

Definition 8: A *tour* in a graph $G(V, E)$ is a sequence of consecutive edges which starts and ends at the same vertex in V.

Definition 9: A graph $G(V, E)$ is *symmetric* if for every vertex $v_i \in V$, the indegree of v_i equals its outdegree.

Definition 10: An *Euler tour* in G is a tour that contains every edge of G exactly once.

Definition 11: *Chinese Postman Problem* is defined as finding the minimum cost tour of a strongly connected directed graph G, that contains every edge of G at least once.

Definition 12: *Rural Chinese Postman (RCP) problem* is to find a tour in a strongly connected directed graph G such that it traverses some subset of edges in G at least once with minimum cost.

Definition 13: *Controllability* of a protocol is the ability to bring the implementation into a desired state.

Definition 14: *Observability* of a protocol is the ability to observe the current state of the protocol (this is generally done by looking at the output(s) generated).

In a protocol specification using FSM, testing an edge $(s_i, a_p/x_q, s_j)$ is done in three steps. In the first step, place the FSM in state s_i, second step is to apply the input a_p and observe the output x_q and the third step is to verify that the FSM entered into state s_j.

Definition 15: *Reset edges* are those edges that bring M from any state of the FSM to the initial state.

We discuss, briefly, four formal methods for test sequence generation and their implementation as automated software tools.

2.2 T-method

Transition Tour (Bosik, 1991) (Naito, 1981) is an input sequence of FSM M, called TT, that starts at some state s_i, visits all the edges of M at least once and comes back to s_i. One method of generating TT is to make M (i.e., its graph G) symmetric (call that obtained graph as $G*$) such that it replicates minimum number of edges of G, which is solved as a linear programming problem, and finding an Euler tour of $G*$.

2.3 D-method

In Distinguishing Sequence method (Gonenc, 1970) (Chow, 1978), there is a sequence of inputs I, called DS, when applied to all the states of M, results in different outputs. One method of generating DS is to find the minimal covering of @(i,DS-diagram) (Kohavi, 1978) (where $i \in I$ and the symbol '@' denotes the concatenation operation on strings). One of the necessary conditions for a FSM to have DS is that, the FSM be fully specified.

2.4 W-method

Characterizing Sequence method (Bosik, 1991) is used only when M does not have a DS. The idea here is to concatenate some more input transitions to form a *partial DS*, called CS, that distinguishes a state from a subset of those states which are responding with same output. The set of such input sequence of a FSM is called the characterizing set W of the FSM. Once W is obtained, the test sequence can be built using the method used in DS method. Like DS method, in this case also, the FSM needs to be fully specified.

2.5 U-method

In this method, a Unique Input/Output (UIO) sequence is obtained for each state of the FSM. For example, if UIO_j is $(a/x, b/y)$ then no state other than j will generate the output sequence x,y for the input sequence a,b. To each edge e_{ij}, UIO_j is concatenated (together is called a *sub-sequence*, defined in detail later) and all such sub-sequences are joined in one of the several ways (Aho, 1991) (Bosik, 1991) (Miller, 1993) by using *bridge sequences* to build a test sequence. In (Sabnani, 1988), it is shown that U-method can be used to generate test sequences if combined with the rural Chinese postman algorithm, an NP-complete problem, for the graphs with two sufficient but not necessary conditions : FSM has reset feature and each state of the FSM has a self loop.

In a case, when some states have more than one UIO sequences (of minimal length), they are termed Multiple Unique Input/Output (MUIO) sequences. Then, for each state, choose one of the MUIO sequences, that gives the shortest test sequence. Many refinements of MUIOS are reported all of which attempt to minimize the length of the test sequence and yet have the properties of controllability and observability (Aho, 1991) (Lombardi, 1992) (Shen, 1992).

3 THE TSP PROBLEM

Travelling Sales Person problem is an example of a computationally hard problem (Rosenkrantz, 1977). In this problem, a sales person starts at an initial city and visits all n cities exactly once so that the effective length of the tour is optimal (minimal). In asymmetric travelling sales person problem, where $d_{ij} \neq d_{ji}$, there are (n-1)! distinct tours with different lengths. Here, d_{ij} represents the distance between the cities i and j. Many solutions have been proposed to solve the TSP problem. Some of them are Nearest Neighbour approach, Branch and Bound algorithm, Linear Search, and Simulated Annealing. Since Simulated Annealing is used in this work, we briefly present about the first three methods and describe Simulated Annealing methodology in detail. In Nearest Neighbour approach, starting from a given city, the unvisited nearest neighboring city is visited next till the tour is complete (Rosenkrantz, 1977). In Branch and Bound algorithm (Gillet, 1979), an attempt is made to find a solution using Assignment Problem (AP). If AP fails to find a solution and results in sub-tours then Branch and Bound strategy is again applied to solve all the sub-problems using AP. The process is repeated till a tour with smallest bound is obtained. In Linear Search method (Kanellakis, 1980), the TSP problem is formulated as no-wait flowshop scheduling problem.

3.1 Simulated Annealing

Simulated Annealing concept is derived from annealing of metals in metallurgy. In annealing, metals are heated to high temperatures and are cooled down slowly and gradually. At each decrement in the value of temperature, the metal is allowed to reach the stable state, so that when it is cooled completely, it reaches thermal equilibrium. This concept is used to solve many optimization problems (Aarts, 1990) (Kirkpatrik, 1983). We use this technique to solve the test sequence generation problem by formulating it as a TSP problem. We discuss, here, the Simulated Annealing procedure (TSP-SA), to solve TSP problem.

3.1.1 Functioning of *TSP-SA*

The procedure TSP-SA (see Figure 1) uses two basic functions : *Obj-fn()* and *Perturb()*. *Obj-fn(R)* returns length of the given tour R of a graph with n nodes. Initially tour R is either selected randomly with a fixed initial node or taken from the user. p is the number of current outer loop iterations. $Perturb(R^0, p)$ perturbs the route vector R^0 at pth iteration. Here two nodes, i, j ($1 \leq$ i,j $\leq$ n-1), of the route vector R^0 are randomly selected and swapped. (i,j $\neq 0$ since the initial city is fixed.) s is the cooling rate of the annealing process. $random(0, 1)$ generates a value between 0 and 1. The iterative values, T_i and T_f are analogous to the initial and final temperatures used in annealing of metals. T_i and T_f have to be selected properly based on the value, $(V_0 - V_1)$ (Aarts, 1990). We select T_i and T_f values as 1000 and 10 respectively for $(V_0 - V_1)$ of order 10. s should be less than and very close to 1. We set s = 0.975.

```
Procedure TSP-SA(input: T_i,T_f, Max-ite; output: R(1..n))
begin
    var p,k : int
        V_0, V_1 : real
    Initialize R to R^0,
    V_0 = Obj-fn(R^0), p = 0
    while (T_i > T_f)
    begin
        k = 0
        while (k < Max-ite)
        begin
            R^1 = Perturb(R^0 , p),
            V_1 = Obj-fn(R^1), /* TSP(n,n) is used here */
            If((V_1 < V_0) Or (exp(-(V_1 -V_0)/T_i) > random(0,1)))
            begin
                R^0 = R^1, V_0 = V_1,
                if(Obj-fn(R^0) < Obj-fn(R))
                begin
                    R = R^0
                end
            end
            k = k + 1
        end
        p = p + 1, T_i = s * T_i
    end
    return R
end
```

Figure 1 Procedure Simulated Annealing.

4 TEST SEQUENCE GENERATION USING MUIOS, THE PROPOSED METHOD

In this section, for a protocol given in the form of a deterministic, strongly connected, minimal FSM, we formulate the conformance test sequence generation problem as an

asymmetric travelling sales person problem and find a solution to this problem by using Simulated Annealing methodology as described in the previous section.

First, we introduce some terminology related to the proposed method and llustrate them using FSM *P1* (see Figure 2).

Definition 16 : *Sub-Sequence* (SS_{mk}) is a concatenation of an edge $e_m \in E$ of the FSM graph, G(V,E), and kth UIOS of its tail state. For example, the sub-sequences for the edge e_2(s_1,b/y,s_3) of FSM $P1$ are,

$$SS_{21}\text{: } (s_1,\text{b/y},s_3;s_3,\text{a/x},s_2;s_2,\text{a/x},s_1),$$
$$SS_{22}\text{: } (s_1,\text{b/y},s_3;s_3,\text{b/y},s_1;s_1,\text{a/y},s_2)$$

since there are two UIOS corresponding to the tail state s_3, of e_2. Where as, edge e_1(s_1,a/y,s_2) has only one sub-sequence,

$$SS_{11}\text{: } (s_1,\text{a/y},s_2;s_2,\text{a/x},s_1;s_1,\text{a/y},s_2).$$

Definition 17 : *Bridge length* (t_{ij}) is the distance between the tail state of sub-sequence SS_{ik} and the head state of sub-sequence SS_{jl}. The t_{ij} is an integer value which measures in terms of number of transitions *essential* to reach from tail state of SS_{ik} to head state of SS_{jl}. The sequence joining them is called a *bridge-sequence*, b_{ij}. For example, bridge length t_{21} between SS_{22} and SS_{11} is 1 and the corresponding bridge sequence b_{21} is 'a/x'. Similarly, the bridge length t_{21} between the sub-sequence SS_{21} and sub-sequence SS_{11} is zero and the corresponding bridge-sequence b_{21} is NULL.

Definition 18 : *TSG-graph* is a directed graph $TSG(N, B)$, where N is a set of nodes, $n_i \in N$, $n_i = \{ SS_{ik} | \ 1 \leq \text{k} \leq \text{U}$, U is the number of UIOS of the tail state of edge e_i and SS_{ik} are the sub-sequences corresponding to the edge e_i of the FSM $\}$ and B is a set of bridge-sequences, $b_{ij} \in B$ (the minimum bridge-sequence between the nodes n_i and n_j is represented as an edge). For example, the bridge-sequence b_{21} entered as the edge of the TSG-graph is NULL and not 'a/x', since former is the minimum possible bridge-sequence between the nodes n_2 and n_1 of the TSG-graph.

Definition 19 : A *tour in TSG-graph* always starts at the initial node, $INODE$ (the sub-sequences in the INODE must have the *initial state* of the FSM as the *head state*) and traverses the minimal path across all the nodes. For example, in FSM $P1$ of Figure 2, the INODE is n_1 or n_2, as both the nodes meet the requirement of having state s_1 as its head state.

Definition 20 : *TSG-matrix* is a 2-D matrix, where each element, $TSG(i, j)$ contains two fields, first field representing the sub-sequence, SS_{ik}, chosen at node n_i to traverse to n_j with minimum bridge-sequence b_{ij} and the second field representing the bridge-sequence b_{ij}. For example, the element, $TSG(2, 1)$ contains $SS_{21}|$ NULL.

Definition 21 : *TSP-matrix* is a 2-D matirx, where each element, $TSP(i, j)$ represents the bridge length, t_{ij}, between the nodes ni and nj. For example, $TSP(2, 1)$ represents the bridge length t_{21}, which is zero.

Definition 22: *Overlap edges* of two sub-sequences are those edges at the end of first sub-sequence, which overlap with same number of edges at the beginning of second sub-sequence.

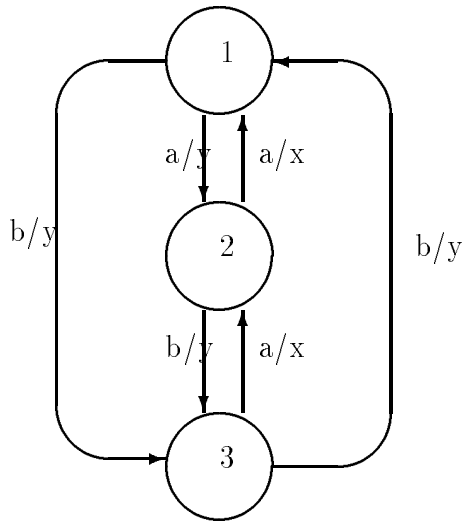

Figure 2 FSM P1.

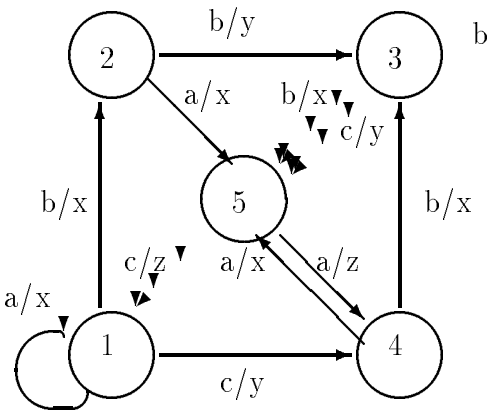

Figure 3 FSM P2.

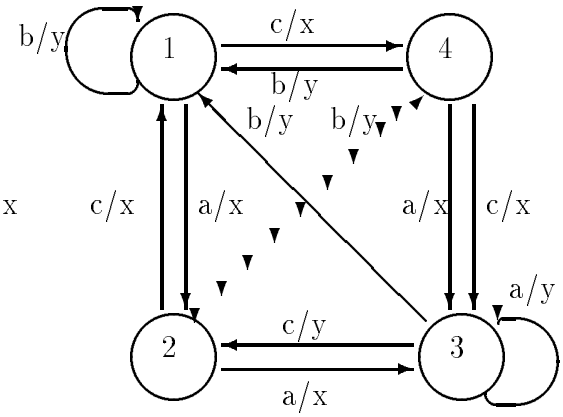

Figure 4 FSM P3.

Table 1 MUIOS for the FSMs shown in Figures 2-4

FSM P1		FSM P2		FSM P3	
state	UIOS	state	UIOS	state	UIOS
1	a/y	1	a/x,a/x * b/x,a/x a/x,b/x c/y,b/x a/x,b/y	1	* a/x,c/x a/x,a/x
2	a/x,a/y			2	* b/y,a/x,c/y c/x,a/x,a/y b/y,a/x,a/y
3	* a/x,a/x b/y,a/y	2	b/y		
		3	* b/x,c/z b/x,a/z	3	* c/y a/y
		4	b/x,c/y	4	* c/x,a/y c/x,a/y
		5	* c/z a/z		

* UIOS selected in case of SUIOS.

Now, the solution to test sequence generation problem is to make a tour of the TSG-graph. Algorithm to generate protocol test sequences using the proposed method is given in Figure 5.

Algorithm TSG-TSP

```
begin
    var i,j,k : int
    1. Generate the MUIOS as given in (Sabnani, 1988).
    2. Construct TSG-graph TSG(N, B).
    3. Select an INODE in TSG(N,B).
    4. Construct TSG(n, n) and TSP(n, n) matrices, where n is
       number of nodes in the TSG-graph.
    5. Call TSP − SA with TSP(n,n) and obtain the route vector R.
    6. Calculate the test sequence, TS, using the route vector, R as follows :
            k = 0, TS = NULL
            repeat
            begin
                i = R(k),
                j = R(k+1),
                TS = @( TS, @(TSG(i,j).sub-sequence, TSG(i,j).bridge-sequence)),
                k = k+1
            end
            until(k ≥ (N-1))
            TS = @(TS, TSG(N-1,0).sub-sequence)
    7. Output TS.
end
```

Figure 5 Algorithm for test sequence generation using the proposed method.

5 AN EXAMPLE

We now illustrate the method of test sequence generation by the proposed *TSG-TSP* using FSM $P1$ as an input protocol (see Figure 2) for the case of MUIOS without reset and without overlap.

In step 1, we have the MUIOS of the FSM as shown in Table 1.

In step 2, construct the TSG-graph for the edges of the protocol, FSM P1,

$$E = \{s_1,a/y,s_2;s_1,b/y,s_3;s_2,a/x,s_1;s_2,b/y,s_3;s_3,b/y,s_1;\ s_3,a/x,s_2\}.$$

The nodes of the TSG-graph are, $n_i \in N$, where n_i represents a set of sub-sequences corresponding to e_i. They are :

n_1:
SS_{11}: $(s_1,a/y,s_2;s_2,a/x,s_1;s_1,a/y,s_2)$

n_2:
SS_{21}: $(s_1,b/y,s_3;s_3,a/x,s_2;s_2,a/x,s_1)$,
SS_{22}: $(s_1,b/y,s_3;s_3,b/y,s_1;s_1,a/y,s_2)$

n_3:
SS_{31}: (s_2,a/x,s_1;s_1,a/y,s_2)

n_4:
SS_{41}: (s_2,b/y,s_3;s_3,a/x,s_2;s_2,a/x,s_1),
SS_{42}: (s_2,b/y,s_3;s_3,b/y,s_1;s_1,a/y,s_2)

n_5:
SS_{51}: (s_3,b/y,s_1;s_1,a/y,s_2)

n_6:
SS_{61}: (s_3,a/y,s_2;s_2,a/x,s_1;s_1,a/y,s_2).

The edges of the TSG-graph are the minimum bridge-sequences between the nodes (given in TSG-matrix later). The TSG-graph for FSM P1 is shown in Figure 6.

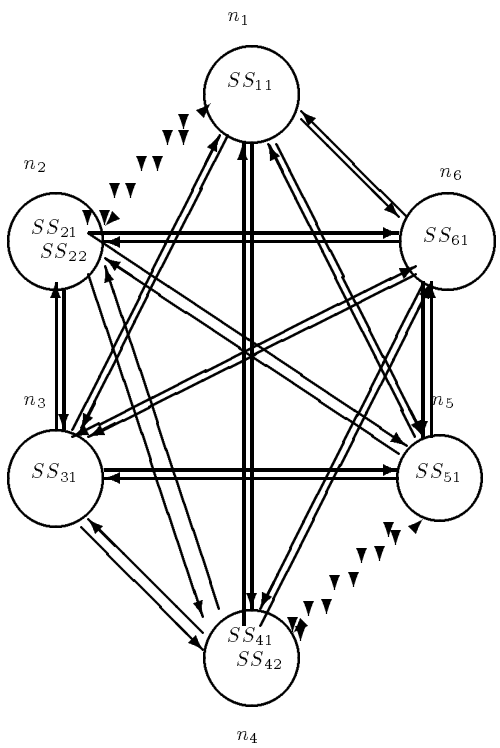

Figure 6 TSG-graph for the FSM P1 shown in Figure 2.

In step 3, select the INODE n_1 from the TSG-graph.

In step 4, the TSG-matrix is :

$$\begin{bmatrix} \text{NULL|NULL} & \text{SS}_{11}\text{|a/x} & \text{SS}_{11}\text{|NULL} & \text{SS}_{11}\text{|NULL} & \text{SS}_{11}\text{|b/y} & \text{SS}_{11}\text{|b/y} \\ \text{SS}_{21}\text{|NULL} & \text{NULL|NULL} & \text{SS}_{22}\text{|NULL} & \text{SS}_{22}\text{|NULL} & \text{SS}_{21}\text{|b/y} & \text{SS}_{21}\text{|b/y} \\ \text{SS}_{31}\text{|a/x} & \text{SS}_{31}\text{|a/x} & \text{NULL|NULL} & \text{SS}_{31}\text{|NULL} & \text{SS}_{31}\text{|b/y} & \text{SS}_{31}\text{|b/y} \\ \text{SS}_{41}\text{|NULL} & \text{SS}_{41}\text{|NULL} & \text{SS}_{42}\text{|NULL} & \text{NULL|NULL} & \text{SS}_{41}\text{|b/y} & \text{SS}_{41}\text{|b/y} \\ \text{SS}_{51}\text{|a/x} & \text{SS}_{51}\text{|a/x} & \text{SS}_{51}\text{|NULL} & \text{SS}_{51}\text{|NULL} & \text{NULL|NULL} & \text{SS}_{51}\text{|b/y} \\ \text{SS}_{61}\text{|a/x} & \text{SS}_{61}\text{|a/x} & \text{SS}_{61}\text{|NULL} & \text{SS}_{61}\text{|NULL} & \text{SS}_{61}\text{|b/y} & \text{NULL|NULL} \end{bmatrix}$$

and the TSP-matrix is :

$$\begin{bmatrix} 999 & 1 & 0 & 0 & 1 & 1 \\ 0 & 999 & 0 & 0 & 1 & 1 \\ 1 & 1 & 999 & 0 & 1 & 1 \\ 0 & 0 & 0 & 999 & 1 & 1 \\ 1 & 1 & 0 & 0 & 999 & 1 \\ 1 & 1 & 0 & 0 & 1 & 999 \end{bmatrix}.$$

In step 5, the procedure *TSP-SA* is called with $TSP(n, n)$, which returns the route vector $R = (1\ 3\ 5\ 4\ 2\ 6)$.

In step 6, the test sequence is :

a/y,a/x,a/y,a/x,a/y,b/y,b/y,a/y,b/y,a/x, a/x,b/y,a/x,a/x,b/y,a/x,a/x,a/y.

The length of the test sequence is 18.

6 RESULTS

We also present the results for other two FSMs shown in Figures 3 and 4. For each test case, we consider SUIOS without reset edges, with and without overlaps; MUIOS without overlaps, with and without reset edges. The results obtained by the proposed method are compared with results obtained using rural Chinese postman algorithm (see Table 2).

Table 2 Comparison of length of the test sequences obtained by TSG-TSP and RCP

	Test Sequence Length				
Testcase	SUIO w/o reset edges		MUIO w/o overlap edges		RCP with reset and w/o overlap using MUIOS
	overlap	w/o overlap	w/o reset	reset	
P1	13	18	18	25	26
P2	24	34	29	43	43
P3	31	40	37	46	54

7 CONCLUSION

In this paper, a technique for test sequence generation using MUIOS is described. The test sequence generation problem is considered as a TSP problem and is solved by using Simulated Annealing methodology. The results presented demonstrate that the method is *at least as efficient* as the method of test sequence generation using rural Chinese postman algorithm.

ACKNOWLEDGMENT

The authors would like to thank Prof. Sidhu and Dr. Motteler, University of Maryland, USA, for their helpful suggestions.

REFERENCES

Aarts E. and Korst J. (1990) Simulated Annealing and Boltzmann machines - A stochasic approach to combinatorial optimization and Neural computing, John Wiley & Sons.

Aho A. V., Dahbura A. T., Lee D. and Uyar U. (1991), An optimization technique for protocol conformance test generation based on UIO subsequences and rural Chinese postman tours, *IEEE trans. on Comm.*, 39(11), 1604−1615.

Bosik B. S. and Uyar M. U. (1991), Finite state machine based formal methods in protocol conformance testing : from theory to implementation, *Comp. Networks and ISDN Systems*, 22, 7−33.

Chow T. S. (1978) Testing software design modeled by finite-state machines, *IEEE trans. on Software Engineering*, SE-4(3), 178−187.

Gillett B. E. (1979) Introduction to Operations Re search, A computer oriented algorithmic approach, Tata Mc Graw Hill Edition, New Delhi.

Gonenc G. (1970) A method for the design of fault detection experiments, *IEEE trans. on Comp.*, 19, 551–558.

Kanellakis P. C. and Papadimitriou C. H. (1980) Linear search for the asymmetric travelling salesman problem, Operations Research, 28(5), 1086–1099.

Kirkpatrick S., Gelatt J. C. D. and Vecchi M. P. (1983) Optimization by Simulated Annealing, *Science*, 13 , 220(4598), 671–680.

Kohavi Z. (1978) Switching and Finite Automata Theory, Mc Graw Hill, New York.

Linn J. R. J. (1989/1990) Conformance testing for OSI protocols, *Computer Networks and ISDN Systems*, 18, 203–219.

Lombardi F. and Shen Y. N. (1992) Evaluation and improvement of fault coverage of conformance testing by UIO sequences, *IEEE trans. on Comm.*, 40(8), 1288–1293.

Miller R. E. and Paul S. (1993) On the Generation of Minimal-Length Conformance Tests for Communication Protocols, *IEEE/ACM trans. on Networking*, 1(1), 116–129.

Naito S. and Tsunoyama M. (1981) Fault detection for sequential machines by transition-tours, in: *Proc. of* 11th *IEEE Fault Tolerant Computing Symp.*, 238–243.

Rosenkrantz D. J., Stearns R. E., and Lewis P. M. (1977) An analysis of several heursistics for the Travelling Salesman Problem, *SIAM Jl. of Computing*, 6, 563–581.

Sabnani K. and Dahbura A. T. (1988) A protocol testing procedure, *Comp. Networks and ISDN Systems*, 15, 285–297.

Sarikaya B. (1987) Conformance testing: Architectures and test sequences, *Comp. Networks and ISDN Systems*, 17, 111–126.

Sarikaya B. and Bochmann G. v. (1984) Synchronization and Specification issues in protocol testing, *IEEE Software Engineering*, Com-32(4), 389–395.

Shen Y. N., Lombardi F. and Dahbura A. T. (1992) Protocol conformance testing using Multiple UIO sequences, *IEEE trans. on Comm.*, 40(8), 1282–1287.

Sidhu D. P. and Leung T. K. (1989) Formal methods for protocol testing : A detailed study, *IEEE trans. of Software Engineering*, 15(4), 413–426.

Ural H. (1992) Formal methods for test sequence generation, *Comp. Comm.*, 15(5), 311–325.

Prem Kumar Gadey received his B.Tech. (Electronics & Communication Engineering) from Sri Venkateswara University in 1990 and M.Tech. (Artificial Intelligence & Robotics) from University of Hyderabad in 1992. Since then he is a Ph.D. student in department of Electrical Communicaiton Engineering, Indian Institute of Science, Bangalore. His major research interests include Communication Networks, Internetworking, Distributed Computing, Expert Systems and Artificial Neural Networks. Currently he is focussing on applying Artificial Intelligence techniques to the area of Network Management. He is a student member of IEEE Communication Society.

Pallapa Venkataram received his Ph. D. degree from The University of Sheffield, England, in 1986. He is currently an Associate Professor in department of Electrical Communication Engineering, Indian Institute of Science, Bangalore, India. He worked in the areas of Distributed Databases, Communiation Protocols and AI applicaitons in Communication Networks and has published many papers in these areas.