A New Method of Generating Synchronizable Test Sequences that Detect Output-shifting Faults Based on Multiple UIO Sequences

Kai Chen
Department of Computer Science
University of Science and Technology of China
Hefei 230027, P.R.China
ckg@mail.ustc.edu.cn

Fan Jiang
Department of Computer Science
University of Science and Technology of China
Hefei 230027, P.R.China
fjiang@ustc.edu.cn

Chuan-dong Huang
Department of Computer Science
University of Science and Technology of China
Hefei 230027, P.R.China
yellow@mail.ustc.edu.cn

ABSTRACT
The objective of testing is to determine the conformance between a system and its specification. When testing distributed systems, the existence of multiple testers brings out the possibility of synchronization problems among remote testers and the possibility that output-shifting faults go undetected. This paper proposes a new method of generating minimal synchronizable test sequences that detect output-shifting faults based on multiple UIO sequences. The procedure of test generation involves two steps: constructing several auxiliary digraphs from a given specification and finding a rural Chinese post tour (RCPT) in the resultant digraph. When constructing the auxiliary digraphs, different from all the former methods, we use vertices to denote transitions and edges to represent two consecutive transitions. In terms of property and application, the proposed method can construct a relatively simple digraph which makes test generation easily. After applying it to practice, we got hold of better results than the existing methods.

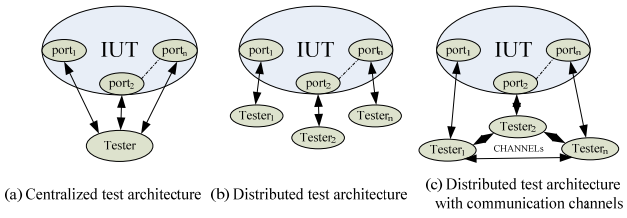
Categories and Subject Descriptors
B.4.5 [**I/O and Data Communications**]: Reliability, Testing, and Fault-Tolerance − *Error-checking*, *Test generation*; D.2.4 [**Software Engineering**]: Software/Program Verification – *Formal methods*, *Validation*; D.2.5 [**Software Engineering**]: Testing and Debugging – *Error handling and recovery*.

General Terms
Algorithms, Performance, Reliability, Verification

Keywords
distributed system, conformance testing, FSM, synchronization problems, output-shifting faults

SAC'06, April, 23-27, 2006, Dijon, France.

1. INTRODUCTION
The developing distributed systems have led to many issues in relevant researches. An important aspect involves testing such complex systems which requires the test techniques both effective and efficient. The objective of testing is to determine whether an *Implementation Under Test* (*IUT*) conforms to its specification. Testing is usually realized by generating test sequences from a specification and applying them to an *IUT* in a test architecture. When testing a distributed system, we may confront with several test architectures shown in Fig.1. This paper mainly considers the architecture in Fig.1c. In this architecture, the *IUT* contains a number of separate interfaces called ports and the test system consists of a local tester for each port of the *IUT*. Each local tester communicates with the *IUT* through its corresponding port, and they also can exchange coordination messages with each other through independent communication channels among them.



(a) Centralized test architecture (b) Distributed test architecture (c) Distributed test architecture with communication channels

Fig.1: Several test architectures for distributed systems.

During the application of a test sequence in a distributed test architecture, the existence of multiple testers brings out the possibility of coordination problems among testers known as synchronization problems and output-shifting faults. The synchronization problem occurs if a tester cannot determine when to apply a particular input to an *IUT*. It is therefore necessary to construct a synchronizable test sequence that the coordination among testers is achieved indirectly through their interactions with the *IUT* [1]. For some specifications, however, there is no synchronizable test sequence [2]. In this case, it is requisite for testers to exchange coordination messages directly through reliable communication channels which are independent of the *IUT*. Moreover, due to the lack of a global clock, it is difficult to determine which input triggered a particular output. This makes it challenging to detect output being shifted between adjacent transitions. In order to detect these output-shifting faults, the test

sequence needs to be augmented either by additional subsequences selected from the specification of the *IUT* [3] or by coordination message exchanges between testers [4]. Furthermore, for some specifications, there may not exist a test sequence where output-shifting faults can be detected without using coordination messages [5].

When considering the costs of coordination messages and I/O operations, it is desirable to construct minimal test sequences. Chen [6] and Wu [7] proposed a method to generate minimal synchronizable test sequences, yet the sequences generated cannot detect output-shifting faults. Cacciari [4] showed that how a minimal set of messages may be added to a given test sequence to produce a synchronizable test sequence that detects output-shifting faults. However, a short initial test sequence may require many additional messages. Hong [8] introduced a synchronization relation digraph to describe the communication channels between testers; however their method may bring out many unnecessary messages either. Although Hierons [9] presented a method to generate test sequences that detect output-shifting faults, we found that the auxiliary digraph is too complicated and only considers the single *UIO* sequence. Consequently, this paper will exhibit a new method of generating minimal test sequences based on multiple *UIO* sequences. What is more, the method executes in a relatively easy manner and the generated test sequences have the ability not only to solve the synchronization problems but also to detect output-shifting faults.

The paper is structured as follows: Section 2 introduces the preliminary knowledge. Section 3 presents the proposed method and briefly compares it with the existing methods. Section 4 gives the conclusions and future work.

2. PRELIMINARIES

2.1 FSM and Its Graphic Representation

An n-port Finite State Machine (*np*-FSM) $M = (S, \Sigma, \Gamma, \delta, \lambda, s_0)$ where:

- S is a finite set of states of M and $s_0 \in S$ is the initial state of M;
- Σ is an *n-tuple* $(\Sigma_1, \Sigma_2, \ldots \Sigma_n)$, where Σ_k is the input alphabet of $port_k$, and $\Sigma_i \cap \Sigma_j = \varnothing$ for $i \neq j$ $(i, j, k = 1, 2, \ldots n)$. Let $I = \Sigma_1 \cup \Sigma_2 \cup \cdots \cup \Sigma_n \cup \{\varepsilon\}$, where ε represents null input;
- Γ is an *n-tuple* $(\Gamma_1, \Gamma_2, \ldots \Gamma_n)$, where Γ_k is the output alphabet of $port_k$, and $\Gamma_i \cap \Gamma_j = \varnothing$ for $i \neq j$ $(i, j, k = 1, 2, \ldots n)$. Let $O = (\Gamma_1 \cup \{\varepsilon\}) \times (\Gamma_2 \cup \{\varepsilon\}) \times \ldots \times (\Gamma_n \cup \{\varepsilon\})$, where ε represents null output;
- δ is the transition function: $D \rightarrow S$, and λ is the output function: $D \rightarrow O$, where $D \subseteq S \times I$.

An *np*-FSM M can be represented by a digraph $G = (V, E)$ where V represents the set S of states of M and E represents all specified transitions of M. Fig.2 shows two examples of *3p*-FSM with $\Sigma_1 = \{a\}$, $\Sigma_2 = \{b\}$, $\Sigma_3 = \{c\}$, $\Gamma_1 = \{\alpha, \zeta\}$, $\Gamma_2 = \{\beta\}$, $\Gamma_3 = \{\gamma\}$. A transition of an *np*-FSM is a triple $(s_i, s_j; x/y)$ where $s_i, s_j \in S$, $x \in I$, $y \in O$, such that $\delta(s_i, x) = s_j$ and $\lambda(s_i, x) = y$. For example in Fig.2a, the transition *t1* denotes that if s_0 is the current state and the input a is received, then the state moves to s_1 and leads to the output α to $port_1$ and γ to $port_3$ respectively.

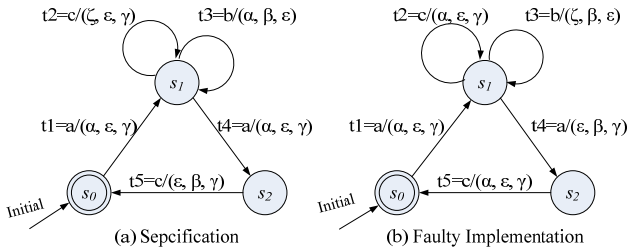


Fig.2: Two examples of *3p*-FSM.

An FSM M is deterministic if for each input there is at most one transition defined at each state of M. An FSM M is minimal if none of its states are equivalent. In a digraph $G(V, E)$, a path is defined by a finite sequence of adjacent but not necessarily distinct edges. G is strongly connected if for any pair of vertices v_i and v_j there is a path from v_i to v_j. G is weakly connected if its underlying undirected graph is connected. Let $d_v^+(G)$ denotes the number of edges leaving v, $d_v^-(G)$ denotes the number of edges entering v and $l_v(G)$ denotes the number of self-loop of v. G is symmetric if for each vertex $v \in V$ $d_v^+(G) = d_v^-(G)$. A digraph $G'(V', E')$ is a sub-graph of $G(V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. A sub-graph G' which contains all the vertices of G is called a spanning sub-graph of G. An edge-induced sub-graph $G'[E_c]$ of G is the sub-graph of G whose vertex set is the set of heads and tails of edges in E_c and whose edge set is E_c where $E_c \subseteq E$. An edge-induced sub-graph $G'[E_c]$ is said to be an edge-induced spanning sub-graph of G if its vertex set is equal to V.

A tour in digraph G is a path which begins and ends at the same vertex. When it contains each edge exactly once we call it an *Euler Tour*. It is known that G has an *Euler Tour* iff G is symmetric and strongly connected. Finding a shortest tour that contains each edge from a given edge set E_c at least once is called *rural Chinese postman problem* (*RCPP*) [10].

2.2 Testing from an FSM with UIO Sequences

Traditionally, there are four basic test generation methods: *T*-, *U*-, *D*-, and *W*-methods [11]. This paper mainly considers *U*-method for its better fault coverage than *T*-method as well as better computational complexity and flexibility than *D*- and *W*-methods.

Usually, there are two types of fault a transition may have: *output fault* and *state transfer fault*, in which the transition produces the wrong output and moves the system to the wrong state respectively. Thereafter, the *U*-method, which uses a *unique input/output* (*UIO*) sequence to verify the ending state of each transition besides testing each transition at least once, is widely used in protocol conformance testing. This method involves deriving *UIO* sequences for each state of M. A *UIO* sequence for a state of M is an I/O behavior that is not exhibited by any other state of M. In other words, the test sequence u is regarded to be a *UIO* sequence for state s_i $(1 \leq i \leq n)$ if and only if $\forall 1 \leq j \leq n,\ j \neq i \Rightarrow \lambda^*(s_i, u) \neq \lambda^*(s_j, u)$. An example of *UIO* sequence for each state of *3p*-FSM in Fig.2a is listed in Table.1.

Remark 1: The function λ^* here is defined as follows, for $x, x' \in I$, $\lambda^*(s, xx') = \lambda(s, x)\lambda^*(\delta(s, x), x')$, and $\lambda^*(s, \varepsilon) = \varepsilon$.

Table.1: Examples of *UIO* sequences in Fig.2a.

State	*UIO* sequence	Final State
s_0	a/(α, ε, γ), c/(ζ, ε, γ)	s_1
s_1	b/(α, β, ε)	s_1
s_2	c/(ε, β, γ)	s_0

Let u_j denote a *UIO* sequence for s_j, then a transition $t = (s_i, s_j; x/y)$ followed by *UIO* sequence u_j is defined as a test segment for t. The test sequence generation based on *U*-method is executed by concatenating the test segment for each transition. This can be expressed as an instance of *RCPP*. Computing a *rural Chinese postman tour* (*RCPT*) is known as NP-complete; however, the problem would have some effective algorithms under certain conditions [12].

Lemma 1: If the edge-induced sub-graph $G[E_c]$ of G is weakly connected, then finding an *RCPT* over all edges in E_c can be solved in polynomial-time and the algorithm has low order polynomial complexity.

Lemma 2: If the edge-induced sub-graph $G[E_c]$ of G is not weakly connected, then finding an *RCPT* over all edges in E_c can apply some heuristic algorithms and the results are sub-optimal and within a bound from the optimization.

Remark 2: Proofs of the above lemmas have been well established in many former works (e.g. [12, 13]), and the main work of this paper is not on the improvement of the algorithms of finding an *RCPT* but on the new method to be introduced. In the following, we just make use of the well-established algorithms.

2.3 Synchronization Problems and Output-shifting Faults

When testing a distributed system, the synchronization problem arises if a tester cannot determine when to apply a particular input to *IUT* because it is not involved in the previous transition, i.e. it does not send the input or receive the output in the previous transition. Given $x \in I$, let $port(x)$ denotes the port associated with input x. Given $y = (y_1, y_2, ...y_n) \in O$, let $ports(y)$ denotes the set of ports associated with values from y that are not null.

Definition 1: Two consecutive transitions $t = (s_i, s_j; x/y)$ and $t' = (s_j, s_k; x'/y')$ are synchronizable if $port(x') \in ports(y) \cup \{port(x)\}$. A given test sequence is said to be synchronizable if any two consecutive transitions of the sequence are synchronizable.

Example 1: Consider the transition sequences <a/(α, ε, γ), b/(α, β, ε), c/(ζ, ε, γ), a/(α, ε, γ), c/(ε, β, γ)> (i.e. <t_1, t_3, t_2, t_4, t_5>) of 3*p*-FSM in Fig.2a. It is required that when s_1 is reached, the *IUT* should receive b (sent by $Tester_2$) before receiving c (sent by $Tester_3$). Since it does not send the input or receive the output in transition t_3, $Tester_3$ does not know whether b has been received by the *IUT*. Therefore, $Tester_3$ has no means to determine the order of b and c, i.e. synchronization problem happens.

Due to the lack of a global clock, it is difficult to determine which input triggered a particular output. Even if the behaviors of all the ports are the same as expected, the output-shifting faults may stay in the *IUT*. Consider a transition $t = (s_i, s_j; x/Y)$. $Y = (y_1, y_2, \ldots y_n)$ is a vector that represents the output on each port of the *IUT*. Suppose that if $y_i = \varepsilon$ then we regard it as 0, otherwise we regard it as 1.

Definition 2: The sequence tt', for transitions $t = (s_i, s_j; x/Y)$ and $t' = (s_j, s_k; x'/Y')$ with $Y = (y_1, y_2, \ldots y_n)$ and $Y' = (y_1', y_2', \ldots y_n')$ respectively, has potential output-shifting faults iff $\bigcup_{i \in [1,n], i \neq port(x')} (y_i \oplus y_i') = 1$. The ports on which $y \oplus y' = 1$ have potential output-shifting faults.

The output-shifting faults may occur in two forms: 1) suppose that $y_k \neq \varepsilon$, $y'_k = \varepsilon$, $port(x') \neq k$, the actual outputs in the corresponding transitions in the *IUT* are $y_k = \varepsilon$, $y'_k \neq \varepsilon$ respectively; 2) suppose that $y_k = \varepsilon$, $y'_k \neq \varepsilon$, $port(x') \neq k$, the actual outputs in the corresponding transitions in the *IUT* are $y_k \neq \varepsilon$, $y'_k = \varepsilon$ respectively.

Example 2: Consider two consecutive transitions <a/(ε, β, γ), c/(α, ε, γ)> (i.e. <t_4, t_5>) of 3*p*-FSM in Fig.2b. Compared it with its specification in Fig.2a, the output α has been "shifted" from t_4 to t_5 while the output β has been "shifted" from t_5 to t_4. Within a distributed architecture, however, these output-shifting faults cannot be detected because all the testers receive their expected outputs in right order (Fig.3) although the *IUT* is actually faulty.

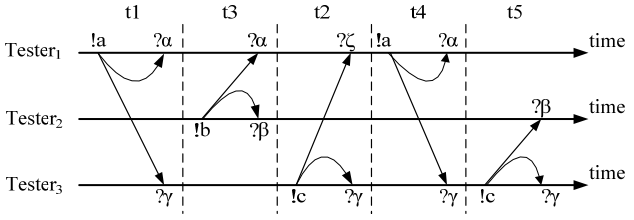


Fig.3: The expected behavior of each tester in Fig.2a.

Remark 3: The sending and receiving of message x are denoted by !x and ?x respectively.

Let $syn_{\Delta\Gamma}$ denotes a coordination message from $port_\Delta$ to $port_\Gamma$. To ensure the solution to synchronization problems and the detection of output-shifting faults, the coordination messages should be added to the test sequences where necessary. The following rules will guarantee the synchronization of multiple testers and the detection of output-shifting faults in consecutive transitions tt' ($t = (s_i, s_j; x/y)$ and $t' = (s_j, s_k; x'/y')$).

ALGORITHM 1:
Step1: $Tester_{port(x)}$ inputs x to the *IUT*,
Step2: If $port(x') \notin ports(y) \cup \{port(x)\}$ then $Tester_{port(x)}$ sends a synchronization message to $Tester_{port(x')}$. (This coordination message is necessary to resolve the synchronization problem between two testers.)
Step3: While receiving an output from the *IUT* or a synchronization message from $Tester_{port(x)}$, $Tester_{port(x')}$ sends coordination messages to all the testers having potential output-shifting faults. (These testers can be calculated by *Definition2* and output-shifting faults can be detected by inserting these messages.)
Step4: $Tester_{port(x')}$ inputs x' to the *IUT*.

3. THE PROPOSED METHOD

Formally, the proposed method exhibits a new solution of generating minimal test sequences that detect output-shifting

faults based on multiple *UIO* sequences. Firstly we design an auxiliary digraph G^* from the given specification M, then the digraph can be applied to generate test sequences. In addition, by taking into account the costs of both transitions and coordination messages, a minimal synchronizable test sequence that detects output-shifting faults for M will be constructed as an *RCPT* over all the test segments of M in G^*.

3.1 The Description of the Proposed Method

Consider a minimal and deterministic *np*-FSM *M0* represented by a strongly connected digraph $G = (V, E)$ (e.g. Fig.4a). In the first place, the proposed method constructs the first auxiliary digraph $G' = (V', E')$ (e.g. Fig.4b) from $G(V, E)$ according to the following steps:

ALGORITHM 2:

Step1: For each edge $t = (s_i, s_j; x/y) \in G$, create a vertex v_t in G',

Step2: For any two adjacent edges $t = (s_i, s_j; x/y) \in G$ and $t' = (s_j, s_k; x'/y') \in G$, create a fine edge $e_{tt'} = (v_t, v_{t'}; label_{tt'})$ in G'. The initial vertex of $e_{tt'}$ is v_t and the final vertex of $e_{tt'}$ is $v_{t'}$,

- ***2a***) $label_{tt'} = t$, if there is no synchronization problem or potential output-shifting faults in tt',
- ***2b***) $label_{tt'} = t@co\text{-}msgs$, if there exist synchronization problems or potential output-shifting faults in tt'.

Remark 4: @ represents concatenation and *co-msg*s denote all necessary coordination messages in tt'.

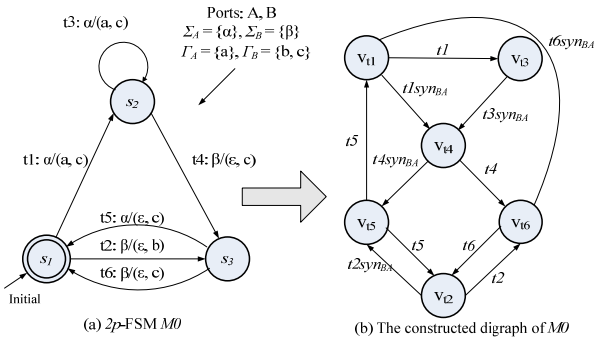


Fig.4: An example of *2p*-FSM *M0* and the first auxiliary digraph of *M0* constructed by *Algorithm2*.

With the aid of Fig.4b, multiple *UIO* sequences with necessary coordination messages for *2p*-FSM in Fig.4a is shown in Table.2.

Table.2: Multiple *UIO* sequences for *M0* in Fig.4a.

State	*UIO* sequences	Final state
s_1	$UIO_1(s_1) = [t2]$	s_3
s_2	$UIO_1(s_2) = [t4syn_{BA}t5]$	s_1
	$UIO_2(s_2) = [t4t6]$	s_1
s_3	$UIO_1(s_3) = [t5]$	s_1
	$UIO_2(s_3) = [t6syn_{BA}t1]$	s_2
	$UIO_3(s_3) = [t6t2]$	s_3

In the second place, the second auxiliary digraph $G^*(V^*, E^*)$ (e.g., Fig.6) for an *np*-FSM M (e.g., Fig.4a) is constructed by adding a set of bold and dashed edges and bold vertices to $G'(V', E')$ (e.g., Fig.4b) according to the following steps:

ALGORITHM 3:

Step1: For each transition $tk = (s_i, s_j; x/y) \in E$, add a bold vertex $V_{tk\text{-}sj}$ and a bold edge $(v_{tk}, V_{tk\text{-}sj}, tk)$,

Step2: For each bold vertex $V_{tk\text{-}sj}$ and each $UIO_i(s_j)$ for state s_j, a dashed edge $(V_{tk\text{-}sj}, v_{tl}, co\text{-}msgs@UIO_i'(s_j))$ is created with coordination messages *co-msg*s if necessary. (Here, tl is the last transition in $UIO_i(s_j)$ and $UIO_i(s_j) = UIO_i'(s_j)@tl$ and *co-msg*s are requisite when there exist synchronization problem or potential output-shifting faults between tk and the first transition in $UIO_i(s_j)$.)

Step3: Merge all the equivalent bold vertices. (Two bold vertices $V_{tk\text{-}sj}$ and $V_{tl\text{-}sj}$ are equivalent if the corresponding dashed edges associated with them have the same final vertices and labels. For example, in Fig.5, bold vertices $V_{t1\text{-}s2}$ and $V_{t3\text{-}s2}$ are equivalent since dashed edges associated with them have the same final vertices v_{t5}, v_{t6} and labels $syn_{BA}UIO_1'(s_2)$, $syn_{BA}UIO_2'(s_2)$, respectively. Then they are merged into V_{s2} in Fig.6.)

Step4: For each dashed edge, if it is the unique dashed edge that leaves the relating bold vertex, then change the dashed edge into a bold edge.

In *step1*, each bold edge $(v_{tk}, V_{tk\text{-}sj}, tk)$ represents the first transition in the test segment for tk. In *step2*, each dashed edge $(V_{tk\text{-}sj}, v_{tl}, co\text{-}msgs@UIO_i'(s_j))$ represents a synchronizable *UIO* sequence that can detect output-shifting faults. All the equivalent bold vertices are merged in *step3* to make the structure of digraph simpler and the purpose of *step4* is to increase the likelihood of obtaining a weakly connected sub-graph consisting of bold edges. Table.3 lists the bold and dashed edges to be added to Fig.4b, and Fig.5 and Fig.6 show the constructed digraphs after *step1*, *2* and *step3, 4* respectively.

Table.3 A set of edges for each transition of *M0* in Fig.4a.

tran	test segment	last tran	corresponding edge in G^*	
			bold edge	dashed edge
t1	$t1syn_{BA}UIO_1(s_2)$	*t5*	$(v_{t1}, V_{t1\text{-}s2}; t1)$	$(V_{t1\text{-}s2}, v_{t5}; syn_{BA}UIO_1'(s_2))$
	$t1syn_{BA}UIO_2(s_2)$	*t6*		$(V_{t1\text{-}s2}, v_{t6}; syn_{BA}UIO_2'(s_2))$
t2	$t2syn_{BA}UIO_1(s_3)$	*t5*	$(v_{t2}, V_{t2\text{-}s3}; t2)$	$(V_{t2\text{-}s3}, v_{t5}; syn_{BA}UIO_1'(s_3))$
	$t2UIO_2(s_3)$	*t1*		$(V_{t2\text{-}s3}, v_{t1}; UIO_2'(s_3))$
	$t2UIO_3(s_3)$	*t2*		$(V_{t2\text{-}s3}, v_{t2}; UIO_3'(s_3))$
t3	$t3syn_{BA}UIO_1(s_2)$	*t5*	$(v_{t3}, V_{t3\text{-}s2}; t3)$	$(V_{t3\text{-}s2}, v_{t5}; syn_{BA}UIO_1'(s_2))$
	$t3syn_{BA}UIO_2(s_2)$	*t6*		$(V_{t3\text{-}s2}, v_{t6}; syn_{BA}UIO_2'(s_2))$
t4	$t4syn_{BA}UIO_1(s_3)$	*t5*	$(v_{t4}, V_{t4\text{-}s3}; t4)$	$(V_{t4\text{-}s3}, v_{t5}; syn_{BA}UIO_1'(s_3))$
	$t4UIO_2(s_3)$	*t1*		$(V_{t4\text{-}s3}, v_{t1}; UIO_2'(s_3))$
	$t4UIO_3(s_3)$	*t2*		$(V_{t4\text{-}s3}, v_{t2}; UIO_3'(s_3))$
t5	$t5UIO_1(s_1)$	*t2*	$(v_{t5}, V_{t5\text{-}s1}; t5)$	$(V_{t5\text{-}s1}, v_{t2}; UIO_1'(s_1))$
t6	$t6UIO_1(s_1)$	*t2*	$(v_{t6}, V_{t6\text{-}s1}; t6)$	$(V_{t6\text{-}s1}, v_{t2}; UIO_1'(s_1))$

The final step of test generation based on multiple *UIO* sequences involves finding a minimal tour of G^* over each bold edge at least once. This is an instance of *RCPP*. As is shown in section 2.2, if the bold edges form a weakly connected spanning sub-graph, a polynomial-time algorithm can be used to obtain an *RCPT* over the bold edges in the digraph covering all the test segments at least once. The procedure of finding an *RCPT* is reduced to two phases: constructing a minimal rural symmetric

augmentation digraph of *G** and finding an *Euler tour* on the digraph. Otherwise, if the bold edges cannot form a weakly connected spanning sub-graph, some heuristic algorithms for obtaining an *RCPT* over the bold edges of the digraph yields a test sequence whose cost is sub-optimal. They can all be well solved using any existing classical algorithm as we introduce in section 2.2. We just need to apply our constructed digraph to the algorithms. For example, an *RCPT* over all the bold edges in Fig.6 (For the minimal rural symmetric augmentation digraph see appendix A.): [*t1*, syn_{BA}, $UIO_2'(s_2)$, *t6*, $UIO_1'(s_1)$, *t2*, $UIO_2'(s_3)$, *t1*, *t3*, syn_{BA}, $UIO_1'(s_2)$, *t5*, $UIO_1'(s_1)$, *t2*, *t6*, syn_{BA}, *t1*, syn_{BA}, *t4*, syn_{BA}, $UIO_1'(s_3)$, *t5*] yields the test sequence [$\alpha/(a, c)$, syn_{BA}, $\beta/(\varepsilon, c)$, $\beta/(\varepsilon, c)$, $\beta/(\varepsilon, b)$, $\beta/(\varepsilon, c)$, syn_{BA}, $\alpha/(a, c)$, $\alpha/(a, c)$, syn_{BA}, $\beta/(\varepsilon, c)$, syn_{BA}, $\alpha/(\varepsilon, c)$, $\beta/(\varepsilon, b)$, $\beta/(\varepsilon, c)$, syn_{BA}, $\alpha/(a, c)$, syn_{BA}, $\beta/(\varepsilon, c)$, syn_{BA}, $\alpha/(\varepsilon, c)$] with total costs of 14 transitions and 7 coordination messages. It is a synchronizable test sequence that detects output-shifting faults. In this test sequence, the test segment for *t1* is [$\alpha/(a, c)$, syn_{BA}, $\beta/(\varepsilon, c)$, $\beta/(\varepsilon, c)$], for *t6* is [$\beta/(\varepsilon, c)$, $\beta/(\varepsilon, b)$], for *t2* is [$\beta/(\varepsilon, b)$, $\beta/(\varepsilon, c)$, syn_{BA}, $\alpha/(a, c)$], for *t3* is [$\alpha/(a, c)$, syn_{BA}, $\beta/(\varepsilon, c)$, syn_{BA}, $\alpha/(\varepsilon, c)$], for *t5* is [$\alpha/(\varepsilon, c)$, $\beta/(\varepsilon, b)$], for *t4* is [$\beta/(\varepsilon, c)$, syn_{BA}, $\alpha/(\varepsilon, c)$].

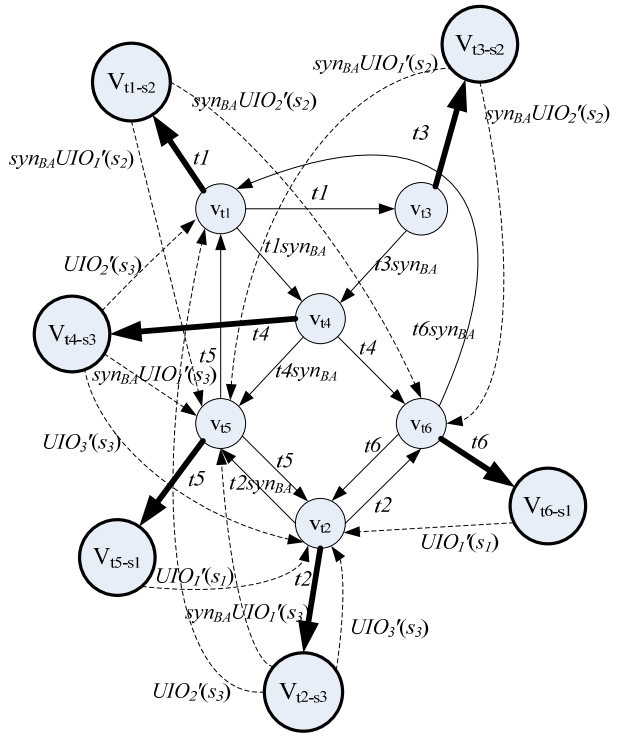


Fig.5: The digraph *G*(V*, E*)* obtained from the digraph *G'(V', E')* in Fig.4b after *step1* and *step2*. (The newly added vertices and edges are shown in Table.3. Each bold edge (v_{tk}, $V_{tk\text{-}sj}$, *tk*) represents the first transition in the test segment for *tk*. And each dashed edge, together with one of its subsequent fine or bold edges, forms a *UIO* sequence for the corresponding state. For example, $UIO_1'(s_1)$ and its subsequent *t2* form $UIO_1(s_1)$ that is a *UIO* sequence for state s_1.)

3.2 Properties of the Proposed Method

Given the digraph $G = (V, E)$ of *np*-FSM $M = (S, \Sigma, \Gamma, \delta, \lambda, s_0)$, let $|V| = |S| = r$, $|E| = m$, and $|I| = |\Sigma_1 \cup \Sigma_2 \cup \ldots \cup \Sigma_n| = p$. In *Algorithm2*, each transition of *M* leads to only 1 vertex and any two consecutive transitions lead to 1 edge in *G'*. Thus the first auxiliary digraph $G' = (V', E')$ has *m* vertices and $\sum_{v \in V}(d_v^+(G) \times d_v^-(G) - l_v(G))$ edges. Since there are at most $|I|$ edges that leave each vertex in *G*, so $0 \le d_v^+(G) \le |I| = p$ and $0 \le d_v^-(G)$, $l_v(G) \le |E| = m$, and then *G'* has $O(pm)$ edges. In *Algorithm3*, for each vertex in *G'* at most 1 corresponding bold vertex will be added in *G**. Given the number of *UIO* sequences for each state is less than *q*, then each newly added vertex leads to at most *q* edges in *G**. As a whole, the ultimate auxiliary digraph $G^*(V^*, E^*)$ has at most $2m$ vertices and $O((p + q)m)$ edges.

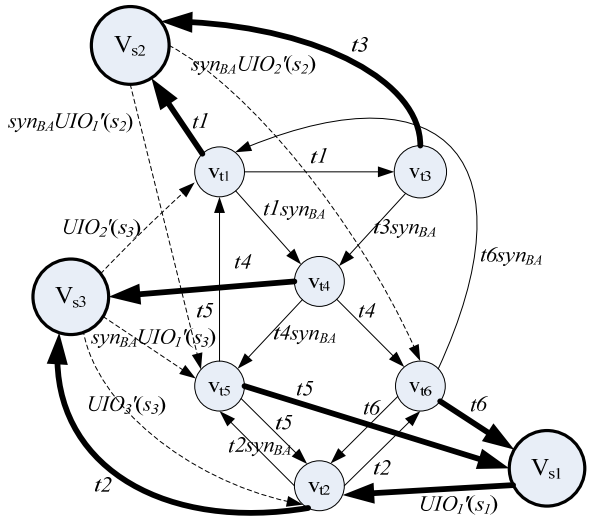


Fig.6: The ultimate digraph *G*(V*, E*)* after *step3* and *step4*. ($V_{t1\text{-}s2}$ and $V_{t3\text{-}s2}$ are merged into V_{s2}, $V_{t2\text{-}s3}$ and $V_{t4\text{-}s3}$ are merged into V_{s3}, and $V_{t5\text{-}s1}$ and $V_{t6\text{-}s1}$ are merged into V_{s1} respectively. Dashed edge (V_{s1}, v_{t2}, $UIO_1'(s_1)$) is changed into a bold edge because it is the unique dashed edge leaving bold vertex V_{s1}. Test generation based on multiple *UIO* sequences involves finding a minimal tour of this digraph over each bold edge at least once.)

3.3 Comparison with the Existing Methods

In the field of distributed systems testing, there are many well-done literatures concerning the test sequences generation. For example, Chen [6], Hierons[9] and Shen[13] consider similar topic as we do in this paper, however, our proposed method is quite different from all these papers. In addition, all these methods have their own deficiencies respectively. Table.4 shows a brief comparison of functions of these methods.

The proposed method can also deal with single *UIO* sequence. The situation of single *UIO* sequence has been well considered in [9]. However, for the same *np*-FSM (i.e. Fig.4a) used in [9], we will show that our method can yield a better result. When applying our method to the condition in [9], i.e. the *UIO* sequence

for s_1 is [β/(ε, b)], for s_2 is [β/(ε, c), β/(ε, c)] and for s_3 is [α/(ε, c)], the digraph $G^{\wedge}(V^{\wedge}, E^{\wedge})$ (Fig.7) is constructed.

Table.4: Functions based comparison

Method	Synchro nization	Output shifting	Single *UIO*	Multiple *UIO*
Ours	√	√	√	√
Chen[6]	√	−	−	√
Hierons[9]	√	√	√	−
Shen[13]	−	−	−	√

Remark 5: "√" denotes that the method covers the corresponding situation, while "−" denotes the method does not consider the situation.

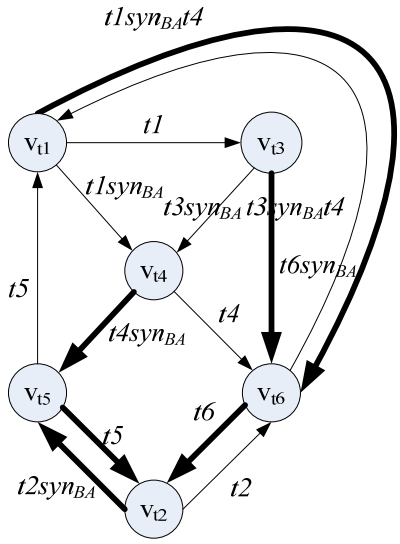


Fig.7: The digraph $G^{\wedge}(V^{\wedge}, E^{\wedge})$ under the condition that *UIO* sequence for each state has been given beforehand. (Since the *UIO* sequence for each state has been given, there is only one dashed edge leaving each bold vertex and these bold vertices can be eliminated by making a combination of edges associated with them. The resultant digraph has a more likelihood to be a weakly connected digraph. For the same *2p*-FSM, the resultant digraph in [9] is not weakly connected.)

Obviously, the bold edges in $G^{\wedge}$ form a weakly connected spanning sub-graph. Then a polynomial-time algorithm can be used to obtain an *RCPT* (For the minimal rural symmetric augmentation digraph see appendix B.): [t_1, syn_{BA}, t_4, t_6, t_2, syn_{BA}, t_5, t_1, t_3, syn_{BA}, t_4, t_6, syn_{BA}, t_1, syn_{BA}, t_4, syn_{BA}, t_5, t_2, t_6, syn_{BA}]. This yields a minimal synchronizable test sequence [α, syn_{BA}, β, β, β, syn_{BA}, α, α, α, syn_{BA}, β, β, syn_{BA}, α, syn_{BA}, β, syn_{BA}, α, β, β] that detects output-shifting faults. It has 14 transitions and 7 coordination messages. For the same question, the method in [9] generates a minimal synchronizable test sequence [sy_{BA}, α, $post_{AB}$, β, β, β, sy_{BA}, α, α, α, $post_{AB}$, β, β, sy_{BA}, α, sy_{BA}, β, sy_{BA}, α, sy_{AB}, β, β, β, sy_{BA}, α, β, β] that detects output-shifting faults with 18 transitions and 9 coordination messages. It has 4 more transitions and 2 more coordination messages than the test sequence of our method.

4. CONCLUSIONS AND FUTURE WORK

Synchronization problems and output-shifting faults arise in distributed testing since there is no global clock available. This paper has introduced a new approach that generates synchronizable test sequences that detect output-shifting faults based on multiple *UIO* sequences. The procedure of test generation can be phrased as an instance of the *rural Chinese postman problem* (*RCPP*). The resultant optimization problem may be resolved using standard algorithms.

When constructing auxiliary digraph from a given FSM, we use vertices to denote transitions and edges to represent two consecutive transitions. The application of the proposed method illustrates that it constructs a relatively simpler digraph and the resultant test sequences have a better characteristic than the existing methods.

The proposed method exhibits a new method of generating synchronizable test sequences that detect output-shifting faults utilizing multiple *UIO* sequences. However, the paper does not consider the overlapping problems. How to address the overlapping problems effectively and how to choose *UIO* sequences more properly will be considered in the future work. Additionally, for some *np*-FSMs, the output-shifting faults can be detected utilizing test subsequences instead of exchanging coordination message. Producing the sufficient and necessary conditions for the existence of such test subsequences and finding minimal-length subsequences are included in our future work.

5. REFERENCES

[1] B. Sarikaya, G.v. Bochmann, Synchronization and specification issues in protocol testing, IEEE Trans. Comm. 32(1984) 389-395.

[2] S. Boyd, H.Ural, The synchronization problem in protocol testing and its complexity, Inform. Process. Lett. 49(1991) 131-136.

[3] G. Luo, R. Dssouli, G.v. Bochmann, P. Venkataram, A. Ghedamsi, Test generation with respect to distributed interfaces, Comput. Standards Interfa- ces 16(1994) 119-132.

[4] L. Cacciari, O.Rafiq, Controllability and observability in distributed testing, Inform. Software Technol. 41(1999) 767-780.

[5] J. Chen, R.M. Hierons, H. Ural, Conditions for Resolving Observability Problems in Distributed testing, FORTE 2004, LNCS 3235, 229-242.

[6] W. H. Chen, H. Urd, Synchronizable test sequence based on multiple UIO sequence, IEEE/ACM Transactions on Networking, 1995, 3(2):152-157

[7] W. I. Wu, W. H. Chen, C. Y. Tang, Synchronizable test sequence for multi-party protocol conformance testing, Computer Communications, 19- 98,21: 1177-1183.

[8] Hong Liu, Jian-Ping Wu, Xia Yin, Generating external synchronizable test sequences that detect output-shifting faults, ConTEL 2003, 565-571.

[9] R.M. Hierons, Testing a distributed system: generating minimal synchronised test sequences that detect output-shifting faults, Inform. Software Technol. 43(9)(2001) 551-560.

[10] A. Gibbons, Algorithmic Graph Theory, Cambridge University Press, 1985.

[11] D. P. Sidhu and T. K. Leung， Formal methods for protocol testing: A detailed study, IEEE Transactions on Software Engineering, 15(1989) 413- 426.

[12] A. Aho, A. Dahbura, D. Lee, M. Uyar, An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours, IEEE Trans. Comm.39 (11)(1991) 1604-1615.

[13] Y.-N. Shen, F. Lombardi, A. T. Dabbura, Protocol conformance testing using multiple UIO sequences, IEEE Trans. Commun, vol. 40, pp. 1282-1287, 1992.

APPENDIX

A: Rural Symmetric Digraph for Fig.6

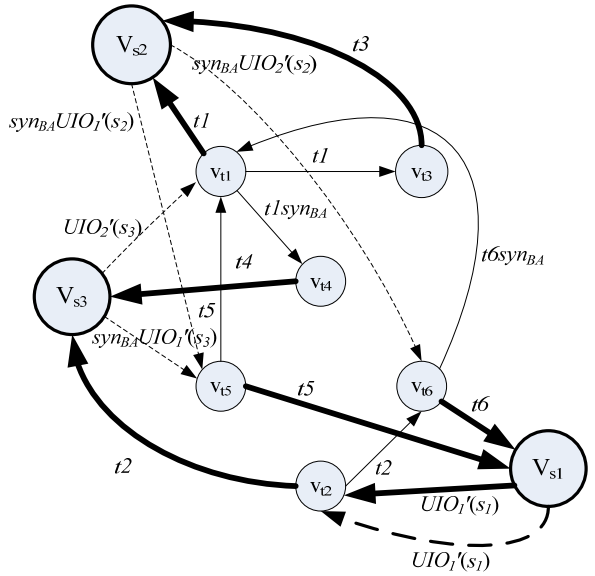


Fig.8: The minimal rural symmetric augmentation digraph for the digraph *G*(V*, E*)*, and test generation is reduced to find an *Euler Tour* in this symmetric and strongly connected digraph.

B: Rural Symmetric Digraph for Fig.7

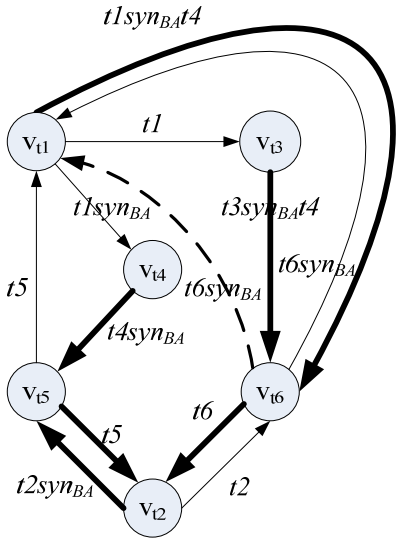


Fig.9: The minimal rural symmetric augmentation digraph for the digraph *G^(V^, E^)*, and test generation is reduced to find an *Euler Tour* in this symmetric and strongly connected digraph.

BIOGRAPHY

Mr. Chen Kai is now a graduate student in University of Science and Technology of China. Main research areas include Wireless Communication Networks, Protocol Conformance Testing and Parallel & Distributed Systems. Mr. Jiang Fan is now a Professor in USTC. Mr. Huang Chuan-dong is now a PhD in USTC.