A Match-based Approach to Optimize Conformance Test Sequence Generation using Mp-method

Jitian Xiao
Edith Cowan University

2009 International Forum on Computer Science-Technology and Applications

A Match-based Approach to Optimize Conformance Test Sequence Generation using Mp-method

Jitian XIAO
School of Computer and Security Science, Edith Cowan University, 2 Bradford Street, Mt Lawley, WA 6050, Australia
j.xiao@ecu.edu.au

ABSTRACT: An important issue in protocol conformance testing is how to generate test sequences in an efficient and effective way that achieves the required fault detection coverage. We proposed an approach for finding shorter test sequences for protocol conformance testing based on the Wp method in our previous work. While the method generated good quality test sequences, an extra leading sequence may have to be added if the final test sequence generated was not started from the same starting state of the given FSM. A new approach is proposed in this paper to overcome this problem thus to improve the quality of the final test sequence. The new test sequence generated always starts from the same starting state of the given FSM. This effectively reduces the length of generated test sequence.

KEYWORDS: conformance testing, test sequence, Wp method, match, asymmetric traveling salesman problem (ATSP).

I. INTRODUCTION

Many systems such as communication protocols and control systems can be modelled using the finite state machines (FSMs). Protocol conformance testing generally involves checking whether the implementation under test (IUT) conforms to a given specification. A sequence of inputs, generated from the specification, is applied to the implementation and the outputs is verified as to whether the expected sequence of outputs is obtained [1, 3]. As the implementation is a black box from a testing perspective, the protocol conformance testing problem is difficult to tackle efficiently [1]. One of the most important issues in protocol conformance testing is how to generate the sequence of inputs, called *test sequence*, in an efficient and effective way to achieve the required fault detection coverage.

Many methods have been proposed to address the protocol conformance testing problem [1, 2, 3, 9]. Among these, the W-method [4] and the improved Wp method [6] have attracted much attention. Unlike other methods, the W-method and Wp method can be applied to all protocols, and can guarantee the detection of any output and transfer faults under certain conditions [4, 6]. However, these two methods assumed that the implementation under test has reliable reset functions which is difficult to be realized for some systems. If an implementation doesn't have a reliable reset function, alternative reset technique can be used to home the test sequence [6]. In addition, the length of the generated test sequences using these two methods are usually the longest amongst all methods.

In our previous work [10], we proposed a method to find shorter test sequences for protocol conformance testing based on the Wp method. The approach provided a transformation technique for converting the problem of the test sequence generation from a given FSM into finding the shortest path in an asymmetric travelling salesman problem (ATSP) using one of the many existing meta-heuristic algorithms for addressing ATSP. The approach consisted of four phases: the first phase generates test segments using the Wp-method; the second phase employs the concepts of overlap to reduce test sequence length; the third phase concatenates without linking cost into the test sequences generated and forms a set of so called *(test) segment sequences*; and, the last phase converts the problem of finding the shortest test sequence into a problem of finding the shortest tour in the ATSP over a graph built from the segment sequences generated from the previous phase. The approach also reduced the complexity of the resulting ATSP, ensuring that the computations involved with the test sequence generation are greatly reduced.

However, as mentioned in the discussion section of the paper [10], the test sequence generated may have to be changed by adding extra leading sequence if the final test sequence generated was not started from the same starting state of the given FSM. The latter case would generally increase the length, thus degrade the quality of the final test sequence generated.

To solve this problem, we now propose a new algorithm, in this paper, to improve the quality of the final test sequence generated by the approach developed in [10]. Instead of generating a shortest tour and then forming the shortest path from the tour generated, the new algorithm generates directly the shortest path using a match-based method in the last phase of the approach. By adding extra restriction on matching condition, the new method guarantees that the generated test sequence always starts from the same starting state of the given FSM. In this way, the new algorithm generates better test sequence in the sense that the length of generated test sequence is generally shorter than the one generated by the method proposed in [10].

II. THE IMPROVED APPROACH TO OPTIMISE TEST SEQUENCE GENERATION

In this section, we describe our new approach to improve the test sequence results generated by the approach in [10]. The main improvement is a match-based approach that not only builds a better approximation of shortest test sequence, but also guarantees that no extra leading sequence is needed to be added to the beginning part of the test sequence generated. To do this, some modifications to Phase 2 through to Phase 4 are required. However the main change will be in the last phase. As a comparison to the approach reported in [10], the phases after modification are described as below:

Phase 1 [Generating Test Segments]: Produce a set of test segments using the W-p method [3].
Phase 2 [Removing Overlapping Test Segments]: Remove a test segment from the test segment set if it is totally contained in another test segment (as a subsequence), unless it is the last test segment that shares same starting state with given FSM.
Phase 3 [Generating Segment Sequences]: (a) Construct segment graph G1; (b) Generate the set of segment sequences from G1 using the same technique from the approach in [10], however starting from one test segment that shares the same starting state with the given FSM.
Phase 4 [ATSP and Final Test Sequence]: (a) Using Dijkstra' shortest path algorithm, calculate the shortest paths between pairs of states from the given FSM and then construct a *test segment (TS) graph*; (b) Generate an approximation of the shortest path using the match-based algorithm (see Section 3) and use the result to construct the output test sequence.

Figure 1. Phases of the new approach

In the new approach, Phase 1, Phase 3 (a) and Phase 4 (a) are the same as that of the approach reported in [10]. Phase 2 removed overlapping test segments however it keeps at least one test segment that starts at the same starting state with the given FSM. Similarly, Phase 3 generate the set of segment sequences however it keeps at least one test segment that shares the same starting state with the given FSM. Phase 4 (b) guarantees that the approximation of the shortest path generated, using the match-based algorithm, starts from one of the starting states of the given FSM. Detailed algorithm of this last step will be presented in the next section.

III. THE MATCH-BAASED ALGORITHM

In this section, we describe our new algorithm to improve the test sequence results reported in [10].

A. *Match Related Concepts*

The following terminologies [8] are frequently used in the rest of this paper.

A *path graph* $G = (V, E)$ with n nodes is a graph in which nodes in V can be listed as a sequence v_1, v_2 , v_3,... v_{n-1}, v_n such that (v_1, v_2), (v_2, v_3), ... (v_{n-1}, v_n) are the only edges of E. A *match* of a graph is a set of edges, in which any two of them are not incident to the same node. A match is *maximal* if any edge in the graph that is not in the match has at least one of its endpoints matched, and the sum of the edge weights of the match is maximal among all matches of the graph. A *weighted matching* (WM) problem is, for a given (edge weighted) graph G, to find a match of G such that the sum of the edge weights of the match is maximal. The WM problem was solved by Edmonds [10] and the complexity of his algorithm is $O(n^2m)$, where n and m denote the number of nodes and edges in the graph respectively. Since then Edmonds' algorithm has been studied by a number of researchers. The fastest implementation of the Edmonds's algorithm is due to Cook and Role [5] with a time complexity of $O(nm \log n)$. For any graph, these algorithms output a *maximal match* of the graph.

Suppose that a maximal edge weight of a graph G=(V, E, w) is w_{max}. We construct a new graph G'=(V, E, w') which has the same node V and same edge set E of G, and the edge weight w' is defined as, for any $(u, v) \in E$, $w'(u, v) = w_{max} - w(u, v) + 1$. It is easy to prove that a match M is a maximal match of G if and only if it is a minimum match of G'. Based on this, to use Cook's maximal matching algorithm [5] for discussion, we will first introduced the main idea of the algorithm based on maximal matching, and then covert the algorithm to find out minimum matching for shortest test sequence generation.

Our new algorithm employs Cook and Role's maximal match implementation [5] as a component.

B. *The Algorithm*

The basic idea of our new algorithm is (1) to divide the TS graph into sets of disjoint path graphs such that the sum of the edge weights of the longest paths in the path graphs reaches the maximum, and (2) to link these paths using maximal match among the endpoints of the paths.

For easy understanding, in the following we describe only the case of undirected graphs. The principle applies to both undirected graphs and digraphs. For the case of digraphs, when choosing a weighted edge between a pair of nodes, not only the directions of edges between the nodes, but also the greater weights of the edges between the pair of nodes need to be considered.

The algorithm is a recursive one containing three main steps: In the first step, a maximal match M of G is produced using Cook's maximum matching algorithm [5]. Each edge, together with its connected nodes, in M is taken as an initial graph path. That is, G is conceptually divided into sets of path graphs, each consists of a pair of matched nodes and the edge connecting them (each unmatched node of G forms a special path graph, i.e., one without an edge). In the second step, the graph G is coarsened by collapsing the matching nodes (or, endpoints of the longest paths in path graphs). At this step, each pair of matching nodes (or, endpoints of the path in individual path graphs) are combined to form a single node of the next level coarser graph $G' = (V', E', w')$. Nodes in V' are all in the form of either $v = \{v_i, v_j\}$, where $v_i, v_j \in V$ are matched in M (i.e., $(v_i, v_j) \in M$), or $v = \{v_i\}$, where v_i is a unmatched node of M.

Intuitively, each node $v=\{v_i, v_j\} \in V$ represents a path graph in G where v_i and v_j are the endpoints of its longest path. A node v of form $\{v_i, v_j\}$ in V' is referred to as a *t-node*, and a node v of form $\{v_i\}$ is referred to an *s-node*. A *multinode* can be either a t-node or an s-node.

E' and w' are then defined such that the edge between any pair of multinodes v' and v'' corresponds to an edge in E whose two endpoints are in v' and v'', respectively, and whose weight is maximal among all edges connecting nodes in between the multinodes v' and v'' (i.e., the endpoints of the path graph represented by $\{v_i, v_j\}$), if such an edge exists.

After graph G' is built, Cook's maximal matching algorithm is applied to G' again to produce a maximal match M'. By this point, the next level of coarser graph $G'' = (V'', E'', w'')$ can be built following the same procedure (as described above). The above matching-and-collapsing process continues until no further matching can be found.

At this point, each t-node $\{v_x, v_y\}$ in the last coarser graph can be stretched to a path graph, with the two endpoints as v_x and v_y, respectively. The shortest path is produced by printing nodes in all path graphs (i.e., the nodes in each path graph are listed in the order in its longest path), one after another.

Intuitively, we conceptually take a pair of matching nodes and the edge between them as a path graph at the end of first round of matching-and-collapsing process (i.e., each with a path of length 1 or 0), then from the second round of matching-and-collapsing process on, the longest paths in these path graphs are concatenated pairwisely using edges of maximal weight between the endpoints of the paths. With the matching-and-collapsing process going on, paths are linked using the maximal matching on levels of coarser graphs until a set of disconnected path graphs is reached. At this stage, a sequence of nodes of the longest path for each path graph is output.

Two points about the above algorithm need to be addressed. Firstly, to allow one test segment that starts from the starting state of the given FSM to be kept as the starting node of one path graph, we need to mark that test segment as a *special node*. Each time we apply the Cook's algorithm to generate a maximal match, we restrict that this special node be matched in the specific way that either it forms the starting endpoint of a match, or leaves it unmatched. Secondly, as the test segment graph is a completed graph [10], with the above restriction, the above match-and-collapsing process will always result in only one path graph at the end, with the special node (i.e., the test segment that starts at the starting state of the given FSM) as the starting endpoint of the path graph. At this stage, the final test sequence can be generated by printing all test segments along the path graph (from the starting endpoint).

Suppose that the tasks of adjusting edge weights of G and marking a special node as staring point has been done. Then the algorithm can be informally described using pseudocode:

```
Algorithm MaxMatchBasedASP(G)
Input: G = (V, E, w);   // A TS graph with V = {V_1, V_2, ... V_n}.
Output: V_{i_1}, V_{i_2}, ..., V_{i_n};   // a permutation of nodes in V.
{
  Find a maximal match M of G using Cook's algorithm,
      starting match from the special node;  //see Cook &Role [3]
  if no matching was found
    {For each isolated multimode
         output its nodes in the order in its longest path;
             //output the node sequence of one path graph
      Return};
  Coarsen G by collapsing matching nodes of M to produce a
      coarser graph G';
  MaxMatchBasedASP (G');
return;}
```

Suppose that the TS graph have *n'* nodes and *m'* edges. Let $n = \max\{n', m'\}$. The complexity of the algorithm is $O(n^2 \log n)$.

C. An Example

Now we reconsider the example from [10]. The FSM represented by the transition digraph M_f was from [3] as shown in Figure 2. All the assumptions used in [3] are also adopted here. Note that it was declared in [3] that only transitions T1, T2, …, T6 are required to be checked as these transitions represented the main behaviour of the protocol. For comparison, we have also adopted this declaration.

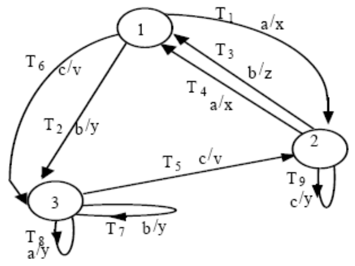

Figure 2. A transition digraph of an FSM M_f (from [3])

A set of test segments for M_f, as shown in Table I, is obtained in [3]. It is reported in [3] that an optimal test sequence for M_f in Figure 2 has a total length of 23.

We now illustrate our proposed method of test sequence generation by going through our new algorithm. We use the FSM derived digraph M_f in Figure 2 as the specification FSM. T_6 is taken as the special node.

In Phase 1, a set of test segments is generated based on the Wp method. The generated test segments are listed in Table I (This phase uses Chen's method [3]).

TABLE I. A SET OF TEST SEGMENTS FOR THE FSM M_F (FROM [3])

Starting State	Test Segments	Ending State
1	Check(T1)=[T1, T3]	1
1	Check(T2)=[T2, T8]	3
2	Check(T3)^1=[T3, T1]	2
2	Check(T3)^2=[T3, T2]	3
2	Check(T4)^1=[T4, T1]	2
2	Check(T4)^2=[T4, T2]	3
3	Check(T5)=[T5, T3]	1
1	Check(T6)=[T6, T8]	3

Since no single test segment listed in Table I is totally contained in any other test segment, Phase 2 does no change the test segment set.

In Phase 3 (a), a segment graph G_1 for this example is constructed as shown in Figure 3. In Phase 3 (b), we produce the segment sequences that cover every edge of G_1 exactly once. The segment sequences generated are listed in Table II. P_3 is the segment sequence that starts from the same starting state of M_f.

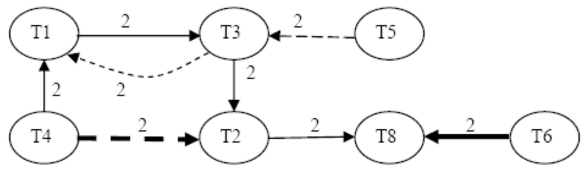

Figure 3. A segment graph G_1 and its segment sequences marked in different line types

TABLE II. SEGMENT SEQUENCES FOR TEST SEGMENTS IN TABLE 1

Segment Sequence	Appearing in Figure 3
n_1 : (T4, T1, T3, T2, T8)	in thin solid arrowed lines
n_2 :(T5, T3, T1)	in thin dotted arrowed lines
n_3 :(T6, T8)	in thick solid arrowed lines
n_4 :(T4, T2)	thick dotted arrowed lines

In Phase 4 (a), the execution of Dijkstra's algorithm generates the shortest paths between pairs of nodes in the original FSM derived digraph M_f. The TS graph G_2 is constructed as below: there are four nodes, each corresponding to one segment sequence generated in Phase 3 (b). The nodes are n_1, n_2, n_3 and n_4, as in Table II. By adjusting the weights, we get a TS graph, as in Figure 4. The first round of matching produces a match, which is shown in thick dotted lines in Figure 4.

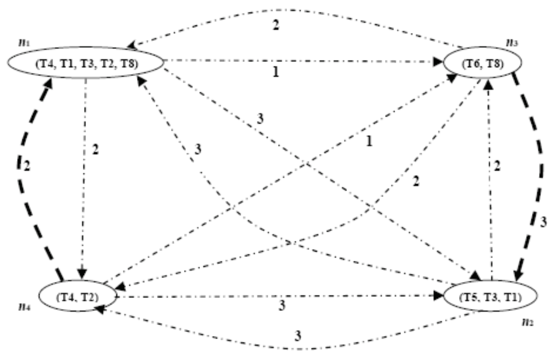

Figure 4. The TS graph G_2 with its edge weights adjusted (by function $w' = w_{max} - w + 1$). One of the maximal match is shown in thicker dotted lines.

After collapsed the graph, the next level of coarser graph is formed as shown in Figure 5, with its only match shown in thicker dotted line.

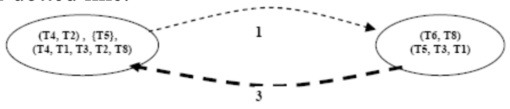

Figure 5. The clasped graph of the TS graph G_2 with its maximal match shown in thicker dotted line.

After another round of collapsing, a final multimode is formed, with the test sequence printed as ((T6, T8), (T5, T3, T1), (T4, T2), (T4, T1, T3, T2, T8)) with total traveling weight 1, which corresponds to only one transition, T_5, as shown beside the edges of the paths in Figure 5 (i.e., inside curly bracket in Figure 5). Mapping back to G_2, we get the following test sequence

T6,T8,T5,T3,T1,T4,T2,$\underline{\text{T5}}$,T4,T1,T3,T2,T8

where underlined transitions (e.g., T5 in this example) are those connecting consecutive segment sequences. It is evident that the generated test sequence starts from the same starting state of the given FSM M_f.

As a comparison, the total length of the final test sequence generated using our approach is 13 (as described above), while it was 23 using Chen's approach (see [3] for detailed test sequence generation).

IV. CONCLUSION

This paper has described a match-based approach for finding shorter test sequences for protocol conformance testing based on the Wp method. When compared with the approach reported in [10], the new approach generates shortest test sequences that always start from the same starting state of the given FSM. This overcome the problem that an extra leading sequenced may have to be added in the case that the test sequence generated started from a state different from the starting state of the given FSM. Therefore, the new algorithm generates better test sequence in the sense that the length of generated test sequence is generally shorter than the one generated by the method proposed in [10]. An example has been used to showcase the quality of the test sequence generated by the new approach. It has illustrated that the length of the test sequence generated by the new approach is shorter than that generated by some existing methods.

REFERENCES

[1] A. V. Aho, A. T. Dahbura, D. Lee, and M. U. Uyar, "An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours", *IEEE Transactions on Communications,* 39(11): 1604-1615, 1991.

[2] J. Chen, R. M. Hierons, H. Ural and H. Yenigun, "Eliminating Redundant Tests in a Checking Sequence", Proceedings of TESTCOM 2005, LNCS 3502, 2005, pp. 146-158.

[3] W. H. Chen, "An Optimization Technique for Protocol Conformance Testing Based on the Wp Method", International Journal of Applied Science and Engineering, 1(1): 2003, pp. 45-54.

[4] T. S. Chow, "Testing Design Modelled by Finite-State Machines", IEEE Transactions on Software Engineering 4 (3), 1978.

[5] W. J. Cook and A. Rohe, "Computing Minimum-Weight Perfect Matchings". INFORMS journal on computing, 11(2), 138, 1999.

[6] S. Fujiwara, G. V. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi, "Test selection based on finite state models", IEEE Transactions on Software Engineering, 17: 1991, pp. 591-603,.

[7] C. P. Lam, J. Xiao, and H. Li, "Ant Colony Optimisation for Generation of Conformance Testing Sequences using Characterising Sequences", Proceedings of The 3rd IASTED International Conference on Advances in Computer Science and Technology (ACS2007), Phuket, Thailand. April 2007, ACTA Press. PP140-146.

[8] E. L. Lawler, Combinatorial Optimization: Net-works and Matroids. Holt, Rinehart and Winston, New York, 1976.

[9] D. Lee and M. Yannakakis, "Principles and methods of testing finite state machines", Proceedings of IEEE, 84 (8): 1996, 1090-1123.

[10] J. Xiao, C. P. Lam, H. Li, and J. Wang, "Reformulation of the Generation of Conformance Testing Sequences to the Asymmetric Travelling Salesman Problem", Proceedings of International Genetic and Evolutionary Computation Conference 2006, Seattle, Washington, USA, July 8-12, 2006, ACM Press, pp.1933-1940.