Abstract—Exhaustive testing for event driven sequence input interaction is costly and not always practicable for all types of software testing. So, an alternative technique is crucial where optimum/near optimum test case generation is key concern. This paper presents a feasible test suite generation technique using Simulated Annealing (SA) algorithm for Event Driven Input Sequence Testing (EDIST) and called EDIST-SA T-way test strategy. The EDIST-SA technique on a heuristic analysis for generating feasible and near optimum test suite(s), where a cost function carefully initiates acceptable test candidates and a fine-tuned cooling rate with temperature takes part as an iterative perspective. We corroborate on EDIST-SA algorithm by doing a large number of experiments to achieve optimum and/or near optimum test cases from a large number of test candidates. The experimental results are tested on a real application called Embedded Network Traffic Monitoring System (ENTM). Analysis on EDIST-SA strategy shows that the optimum test suite is found from some of the iterated solution and there is possibility to have more feasible accepted test suites.

Keywords—Event Driven Sequence Testing; Feasible Test Suite; Simulated Annealing; T-Way Sequence Covering Tree; Software Planning.

I. BACKGROUND

Software test planning is a crucial topic for the hardware/software development companies. It is not only important but also expensive once we want to go through a system (combine with software and hardware) development process. Many sequences of possible inputs must be checked to verify the behavior a system. Investigation says that the software test planning and testing requires at least 50% cost of the total cost, but it may cost higher once the hardware and safety-critical services are included [1]. The National Institute of Standards and Technology (NIST) estimated that the annual cost of inadequate testing in US as much as $59 billion [2]. This estimation focuses the significance of effective methods for software test planning.

Software tests can plan based on either individual set of input event(s) or all possible interacted input events called exhaustive testing. But the exhaustive testing is not cost effective and not always benefited for all types of software testing. In this situation we focus on alternative strategy where optimum/near optimum test case generation is key concern. Covering Arrays (CA) provides the functionality to allow testing all the interactions within a given size to produce optimized/near optimized test cases. There are few reported methods [3] to model event testing relating to CA, such as: algebraic, recursive, greedy, metaheuristics, etc.

The Combinatorial Input Interaction Testing (CIIT) is one of the tests planning area of functional testing method. In this test planning, large input events are divided into small subset called test candidate and the accepted test candidates called test cases; the final group of test cases known as test suite. Today’s complex and multi-configurable software system looking at CIIT [12] for test planning because of input event interaction cohesion. From the recent literature [4-11,13,17,19-21] software test engineers interest converging towards the CIIT, because of its input interaction coverage to generate optimum/near optimum test cases.

In CIIT the test candidate selection plan is made based on specific input strength is called t-way; where t must be less than total input events. Beside the input data interaction (sequence less) [6,7,9-11,13,19-21] CIIT expands into sequence event testing (focuses on all input event interaction in sequence) [4,5,8,17]. This paper presents design and implementation of t-way test strategy for Event Driven Input Sequence Testing by adopting Simulated Annealing (EDIST-SA) algorithm which is able to produce feasible (optimum and/or near optimum) test cases.

This paper is organized as follows: Section II describes a real event driven application in which the EDIST-SA T-way test strategy is applied. Section III discusses the proposed T-way strategy called EDIST-SA T-way test strategy. Section IV explains details about the proposed algorithm relate to SA. Section V shows the results and Section VI conclude the proposed algorithm implementation and application.

II. A REAL EVENT DRIVEN APPLICATION: ENTM SYSTEM

Figure 1 shows a State Transition (ST) diagram for a real application called Embedded Network Traffic Monitoring (ENTM) system [14], which represents the possible finite and deterministic states [15]. S1 state is the initialization state where the system boot, loading device drivers (such as, keypad and LCD panel), starting services (such as: Apache, FTP, SSH, Dynamic IP) and execute the System Control Module take place. In S1 state, a user choice menu waits for
an input from user through the keypad. User choice menu consists of “Process Control”, “System Information” and “Whole System Control” option. User is assisted by instructions displayed on the LCD panel and the inputs are made through the keypad. The selection of “Program Start” option moves to state S2. In state S2, depending on the inputs, system will start to capture packets, process, analyze and store all the data into files. The S2 process is stopped either for NIC setting error or from S1, and then move to state S3 (Process Management); then the system move back to S1 state. In state S1, the selection of Restart causes the system back to the State S1 and the selection of Shutdown causes the whole system move to state S4 (closed system). Table I shows the description of four events A, B, C and D.

![Figure 1. ENTM system state diagram.](image)

<table>
<thead>
<tr>
<th>Events</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Initiate network packet capture</td>
</tr>
<tr>
<td>B</td>
<td>Continue network packet capture</td>
</tr>
<tr>
<td>C</td>
<td>Reset network packet capture with notification</td>
</tr>
<tr>
<td>D</td>
<td>System control</td>
</tr>
</tbody>
</table>

For t-way sequence event testing the t-way events need to be tested in any sequence to detect any sequence of event interaction. The system monitors a network segment and notifies if there is any error or bottleneck. Event A might be triggered followed by B or C or D. All the events possible sequences must be tested to find sequence faults. Total possible sequences for this 4 events is 4!=24. Consider a 10 input events based system and need 10^4 (or, 3628,800) possible test candidate to test, which is not a good practice. To uncover the exhaustive sequence event testing, following mechanism is proposed for t-way sequence based testing to generate test cases and described in the next sections.

III. PROPOSED EDIST-SA T-WAY TEST STRATEGY

A. Sequence Covering Tree(SCT)

Sequence covering tree is defined as SCT(N; t, n), where N is the number of test cases, t refers to test strength (t = 2, 3, 4, ...k; k is a positive integer) and n indicates the number of nodes. The main advantage to use SCT is to reduce redundant tuple check from the test candidates.

B. 3-way Input Generation Example

Consider four Input Events from Figure 1, such as A, B, C, and D. The problem is to test all 4 events in sequence. For ease of computation, 3-way sequence needs only six test cases out of 4! = 24 test candidates. Figure 2 show in details the problem solution design using SCT. As shown in Figure 2, 24 exhaustive test candidate need to test that could cover 3-way SCT. Note that, each test candidate generating 4 independent tuples. The main focus of EDIST-SA is to find the optimum/near optimum test cases to reduce the exhaustive test cost. Table II show six test optimum candidates selected for test out of 24. This random choice can vary as shown in Table III, where 9 possible optimum test suites are shown. For our test we consider number 7 test candidates sequence from Table III to check SCT sequence tuple covering.

![Figure 2. Sequence Tuple Generation for 4 events and strength 3.](image)

<table>
<thead>
<tr>
<th>No</th>
<th>Test Input Candidates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A B C D</td>
</tr>
<tr>
<td>2</td>
<td>A B D C</td>
</tr>
<tr>
<td>3</td>
<td>A C B D</td>
</tr>
<tr>
<td>4</td>
<td>A C D B</td>
</tr>
<tr>
<td>5</td>
<td>A D B C</td>
</tr>
<tr>
<td>6</td>
<td>A D C B</td>
</tr>
<tr>
<td>7</td>
<td>B A C D</td>
</tr>
<tr>
<td>8</td>
<td>B A D C</td>
</tr>
<tr>
<td>9</td>
<td>B C A D</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No</th>
<th>Test Input Candidates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A C B D</td>
</tr>
<tr>
<td>2</td>
<td>A B D C</td>
</tr>
<tr>
<td>3</td>
<td>A B C D</td>
</tr>
<tr>
<td>4</td>
<td>A C B D</td>
</tr>
<tr>
<td>5</td>
<td>A B C D</td>
</tr>
<tr>
<td>6</td>
<td>A B D C</td>
</tr>
<tr>
<td>7</td>
<td>A C B D</td>
</tr>
<tr>
<td>8</td>
<td>A B D C</td>
</tr>
<tr>
<td>9</td>
<td>A B C D</td>
</tr>
</tbody>
</table>
IV. PROPOSED EDIST-SA T-WAY STRATEGY ALGORITHM

A. Standard Simulated Annealing (SSA)

Simulated Annealing (SA) is a popular meta-heuristic algorithm to solve many combinatorial optimization problems and may possibly used to find the optimum or near optimum solution from a number of possible solutions. The main concern of combinatorial optimization problems solution using SA is based on cost functions. In this research our target to use the SA to find minimum cost solution. In Standard SA algorithm, initial setting of Cooling Rate and Temperature bring the system into finite state. The acceptance of neighborhood depends on the probability of acceptance function [16].

B. EDIST-SA Algorithm Design

Sequence Event testing can be exercised using linear and non linear programming techniques. This type of testing can be explained generically as follows: Given a number of events (E) with strength (t), an optimum test suite has to be found so that all the t-way sequenced tuple must covered. Increasing the number of events and strength the achievement of optimum solution becomes more complicated. There are several techniques [4,5,8,17] already applied for getting the near optimum solution. Our experiment efforts focused to acquire optimum/near optimum test suite using SA technique. Though this technique is not a classical optimization technique but based on heuristics from annealing process. The basic motivation of SA inherited from how crystalline structures are formed from annealing process. SA considers a lot of feasible test cases (local optima) to minimize the problem. In this paper the algorithm is set to break out from the problem of stuck into the current test candidate covered t-way tuple with the new candidate test case covered t-way tuple with a probability similar to the probability in the dynamics of the annealing process. Figure 3 shows the proposed EDIST-SA algorithm.

In our algorithm there is one more factor we use is Cooling Rate (CR) to iterate the search till all the t-way tuple covered in SCT. The main drawback of our algorithm is a large number of Event and strength takes a lot of time and requires a lot of memory and processing. Our algorithm considers five arguments such as: Events, Strength, Temperature, Cooling Rate, and Threshold (or, number of iteration). In the iteration decrease of temperature decrease the swapping event to get the new event. The acceptance probability is based on number of uncovered tuple from new and current test candidate. If current test candidate number of uncovered tuple is less than the new test candidate number of uncovered tuple then the current test candidate acceptance probability is high. At worse case of the new solution the acceptance probability is calculated from the following equation:

\[
Z = e^{\left(\frac{\Delta \text{UcovT}}{\text{Temperature}}\right)}
\]

Where, Z is a new value and compare with a random value range 0.000 to 1.000; \( \Delta \text{UcovT} \) is the difference between number of uncovered tuple for new test candidate and current test candidate and T is the current temperature. The greater value of Z compares with the random value make the current test candidate to accepted test candidate. This iteration will continue as long as the threshold is not met.

1. Initialize Temperature and Cooling Rate
2. Get an initial Test Candidate iTCase and select as a test case and initialize the covered branches in SCT
3. While not get required solution do
   a. Select a new test candidate NTCand
   b. Get the numberOfTupleCovered in SCT
   c. Find acceptance probability of the NTCand
      i. If numberOfTupleCovered (iTCase) ≤ numberOfTupleCovered (NTCand) , Return
         AcceptanceProbability = 1.
      ii. Else Return NTCand accepted probability with the probability \( e^{-\Delta T} \)
   d. If the return value of acceptance_probability ≥ RandomEstimatedValue
      Set AcceptediTCase = NTCand ;
      iTCase = NTCand ;
      Initialize the covered branches in SCT
4. Return Feasible test Suite(s).

Figure 3. EDIST-SA algorithm.

Here we set the threshold to test all t-way tuple covered or the initial Temperature reduced to 0.0 units. The final test suite is selected once all t-way tuples are covered. If temperature becomes 0.0 units but not all t-way tuples are covered, in this situation the test suite is called unexpected test suite and revoke. At worse case of new candidate test case for large number of Events and Strength the improvement acceptance found from the following equation:

\[
Z = e^{\left(\frac{\Delta \text{UcovT}}{\text{Temperature}}\right)}
\]

Where, I is the number of iteration. The temperature is controlled very slowly and changing after each complete iteration [16,19]. In this research the slow rate is calculated by the following equation:

\[
T = T - \delta
\]

Where, \( \delta \) is a small value (\( \delta < 1 \)) called cooling rare and determined based on the problem. The cooling rate is used here to continue the iteration by reducing the temperature. For optimum solution a given number of iterations need to carry out before reducing the temperature. Longer iteration not only depends on very small reduction of temperature but also covered tuple.
V. Experimental Results

A number of experiments were taken to achieve the optimum and/or near optimum results. For achieving optimum/near optimum test cases we set the starting temperature and cooling rate to 10000 units and 0.00001 units respectively. We executed EDIST-SA T-way test algorithm 10 times for every initial temperature where temperature reduced to 50 units. We kept track the entire possible test suite (called feasible test suite) into an ArrayList (in Java Programming Language) for further analysis. Acceptance or rejection of test suite depends on the SCT covering status. If all branches in SCT are covered means the test suite is accepted otherwise rejected. We consider lower strength $t$ as 3, where number of input events is always higher than the strengths. In this sense, for testing the algorithm we consider input sequence events 4 for $t=3$ configuration. Table IV shows the feasible test cases generated by EDIST-SA, QnD and ASP at $t=3$. Figure 4 and shows the EDIST-SA experimented data graph for 3-way configurations; information about initial temperature, feasible temperature, feasible test suite, minimum number of test suite(s) and associate temperature can be explored clearly. In Figure 4, x- axis indicating the number of tests were taken (around 140), y-axis referring the initial temperature (black colored line) and the exact temperature (green colored line) where the feasible test cases found. The y1-axis (blue colored line) indicating the number of feasible test cases. In Figure 4 the sign (+) indicating the feasible test cases in y1-axis and associate temperature in y-axis. The T-way EDIST-SA Test Suites (optimum and feasible number of test cases) are shown in Table IV.

![Figure 4. Experimental data graphs presentation for EDIST-SA test suites associated with temperature (initial and feasible) at T=4.](image)

<table>
<thead>
<tr>
<th>Events</th>
<th>Exhaustive TCs</th>
<th>EDIST-SA (Feasible Test Cases)</th>
<th>QnD [8]</th>
<th>ASP [17]</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>24</td>
<td>6, 7, 8, 9 (Optimum is 6)</td>
<td>6</td>
<td>NA</td>
</tr>
</tbody>
</table>

VI. Conclusion

EDIST-SA t-way test results are feasible and accepted compared to exhaustive test cases. Note that, experiment time we didn’t consider the estimation of time to achieve accepted test suites; because there is no fixed time to get an accepted test suite. For better outcomes we control the cooling rate very slowly and for this reason in some cases the computation time increase significantly. From Figure 4 we can clearly observe a random rise/down of temperature at accepted test suite at the moment of increasing input events. It is found that the source of rise/down of temperature is because of computation of cost function and probability of acceptance as well as the cooling rate. We can achieve the optimum test suite by approximating the cost function with simplified evaluation of cost routine. A test candidate acceptance depends on the probability function. In the simulation we found that, some cases, high temperature and steady cost function value causes acceptance probability very high and cause the number of test cases high. The cooling rate is a crucial part to find optimum test suits in different input event sequence testing. For each event testing our expectation is to find a unique cooling schedule to get the optimum test suite. We took around 10 test simulation by reducing the temperature by 50 units (initialized as 10000 units). In this simulation we found that most of the time the fast cooling schedule cannot cover all the t-way tuple. On the other hand, a very slow cooling schedule has near optimum solution and requires a lot of computation. This testing strategy can be expanded for higher strength t-way test generation.

REFERENCES


