A Meta Heuristic Search based T-way Event Driven Input Sequence Test Case Generator

Mostafijur Rahman, *Rozmie Razif Othman, R Badlishah Ahmad, Md. Mijanur Rahman
School of Computer and Communication Engineering
Universiti Malaysia Perlis (UniMAP)
Pauh Putra, 02600 Arau, Perlis, Malaysia
Email: {mostafijur, *rozmie, badli, mijanur}@unimap.edu.my
*Corresponding Author

***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 a meta heuristic search called Simulated Annealing (SA) for T-way Event Driven Input Sequence Test Case (EDISTC) Generator and abbreviated as T-way EDISTC-SA generator. The T-way EDISTC-SA technique focuses on a heuristic analysis for generating feasible and near optimum test suite(s), where a cost function carefully initiates acceptable test input sequences and a fine-tuned cooling rate with temperature takes part as an iterative perspective. We corroborate on EDISTC-SA algorithm by doing a number of experiments to achieve optimum and/or near optimum test cases from a number of test input sequences. The experimental results are tested on a real application called Embedded Network Traffic Monitoring System (ENTM). Analysis on EDISTC-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 require at least 50% cost from 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 benefitted 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 to model event testing relating to CA, such as: algebraic, recursive, greedy, metaheuristics, etc [3].

The Combinatorial Input Interaction Testing (CIIT) [12] 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 input sequence and the accepted test input sequences called Test Case (TC); the final group of test cases known as test suite. Today's complex and multi-configurable software system looking at CIIT for test planning because of input event interaction cohesion. From the recent literature [4-11,13,17,19-22] 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 input sequence 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 are in sequence) [4,5,8,17,22]. All the T-way strategies (for sequence less and sequence based) are NP-hard problem; i.e., not single strategy can produce optimum number of test cases for all configurations. This paper presents design and implementation of T-way test strategy for Event Driven Input Sequence Testing by adopting Simulated Annealing (EDISTC-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 T-way EDISTC-SA test strategy is deployed. Section III discusses the proposed T-way EDISTC-SA generator. Section IV explains about the proposed algorithm based on 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 execution of the System Control Module. 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 starts to capture packets from a selected network segment then process, analyze and store all the data into files. The S2 process is stopped either for Network Interface Card (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.

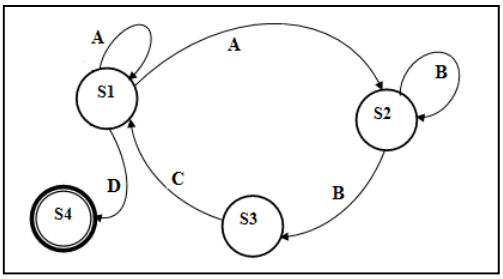

Figure 1. ENTM system state diagram.

TABLE I. ENTM System Events

Events	Description
A	Initiate network packet capture
B	Continue network packet capture
C	Reset network packet capture with notification
D	System control

For T-way sequence event testing the T-way events need to be tested in any sequence to detect fault on 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 10 input events based system and need 10! (or, 3628,800) possible test input sequence to test, which is not a best 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 T-way EDISTC-SA 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 T-way sequence tuple check from the test input sequences.

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 input sequences. Figure 2 show in details of the problem solution design using SCT. As shown in Figure 2, 24 exhaustive test input sequence need to test that could cover 3-way SCT. Note that, each test input sequence generating 4 independent T-way sequence tuples. The main focus of EDISTC-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 input sequences sequence from Table III to check SCT's T-way sequence tuple covered status.

TABLE II. Random Selection of Test Cases From The Exhaustive Test Input Sequences

No	Test Input Candidates	No	Test Input Candidates	No	Test Input Candidates
1	**A B C D**	9	B C A D	17	C D A B
2	A B D C	10	B C D A	18	C D B A
3	A C B D	11	B D A C	29	D A B C
4	A C D B	12	B D C A	20	**D A C B**
5	A D B C	13	C A B D	21	D B A C
6	A D C B	14	**C A D B**	22	**D B C A**
7	B A C D	15	C B A D	23	D C A B
8	**B A D C**	16	**C B D A**	24	D C B A

TABLE III. SELECTION OF TEST INPUT SEQUENCES FROM TABLE II THAT CAUSE OPTIMUM TEST SUITES

No	Test input sequences
1	[A C D B] [B A D C] [B C D A] [C A B D] [D A B C] [D C B A]
2	[A B D C] [A C D B] [B C A D] [C B D A] [D B A C] [D C A B]
3	[A B D C] [A C D B] [B C D A] [C B A D] [D B A C] [D C A B]
4	[A C B D][A D B C][B C A D] [B D A C] [C D A B] [D C B A]
5	[A C B D] [B A D C] [B C D A] [C A D B] [D A B C] [D C B A]
6	[A B C D] [A D C B] [B D A C] [C B A D] [C D A B] [D B C A]
7	[A B C D] [B A D C] [C A D B] [C B D A] [D A C B] [D B C A]
8	[A C B D] [A D B C] [B C A D] [B D A C] [C D B A] [D C A B]
9	[A D C B] [B A C D] [B D C A] [C A B D] [C D B A] [D A B C]

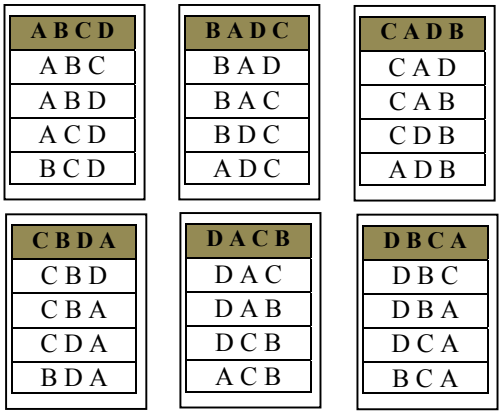

Figure 2. T-way sequence tuple generation for 4 events and strength 3.

IV. PROPOSED T-WAY EDISTC-SA 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. EDISTC-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 sequence 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 input sequence covered T-way sequence tuple with the new candidate test case covered T-way sequence tuple with a probability similar to the probability in the dynamics of the annealing process. Figure 3 shows the proposed EDISTC-SA algorithm.

In our algorithm, we set Cooling Rate (CR) to iterate the search till all the T-way sequence tuple covered in SCT. The main drawback of our algorithm is, large number of Event and strength takes time and requires more memory and processing power compare to deterministic T-way strategis. 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 T-way sequence tuple from new and current test input sequence. If *current test input sequence* number of uncovered T-way sequence tuple is less than the *new test input sequence* number of uncovered T-way sequence tuple then the current test input sequence acceptance probability is high. At worse case of the new solution, the acceptance probability is calculated from the following equation:

$$Z = e^{(\Delta UcovT / Temperature)}$$

Where, Z is a new value and compare with a random value range 0.000 to 1.000; ΔucovT is the difference between number of uncovered T-way sequence tuple for new test input sequence and current test input sequence and T is the current temperature. The greater value of Z compares with the random value make the current test input sequence to accepted test input sequence. This iteration will continue as long as the threshold is not met.

Here we set the threshold to test all T-way sequence tuples covered or the initial Temperature reduced to 0.0 units. The final test suite is selected once all T-way sequence tuples are covered. If temperature become 0.0 units but not all T-way sequence 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 on input test sequences acceptance found from the following equation:

$$Z = e^{(-(\Delta UcovT / (I*Temperature)))}$$

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 δ is a small value (δ < 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 T-way sequence tuples.

1. Initialize Temperature and Cooling Rate

2. Get an initial Test input sequence ***iTCase*** and select as a test case, and then initialize the covered branches in SCT

3. While not get required solution do

A. While ***not_all_branch_covered*** do the following:

a. Select a new test input sequence ***NTCand***

b. Get the ***numberOfT-way sequence tupleCovered*** in SCT

c. ***Find acceptance_probability*** of the ***NTCand***

i. If ***numberOfT-way sequence tupleCovered*** (***iTCase***) ≤ ***numberOfT-way sequence tupleCovered*** (***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 ***AcceptedTCase*** = ***NTCand*** ;

iTCase = ***NTCand*** ;

Initialize the covered branches in SCT

B. Reduce temperature T = T – δ

4. Return Feasible test Suite(s).

Figure 3. EDISTC-SA algorithm.

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 T-way EDISTC-SA test algorithm 10 times for every initial temperature where temperature decreased to 50 units/test. 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 then the test suite is considered as acceptable; 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.

T-way EDISTC-SA strategy uses non-deterministic (heuristic) search to generate test suites for different configurations. Multiple executions of T-way EDISTC-SA algorithm are produced different test suites size. Table IV and V show sample of achieved feasible test suites size for the configurations SCT(N;3,4), and SCT(N;3,5) respectively using T-way EDISTC-SA algorithm. In order to generate the optimum or near optimum results, there is a need to tune the relevant EDISTC-SA parameters. These parameters are cooling rate and temperature. Here the temperature value is used for iterative purpose and hence the cooling rate need to tune in order to obtain feasible (optimum/ near optimum) test suites.

For each configuration, a large number of iteration is needed to adjust to generate feasible test suites. Data in Table IV, and Table V are generated using the initial temperature 10000 to 50 and cooling rate 0.00001 to 0.1. Time-to-time the cooling rate needs to decrease to get tune cooling rate and temperature for minimum Test Suite (TS). It is found that most of the cases the minimum TS size are found at cooling rate 0.00001. For obtaining better TSs the temperature is reduced vary slowly. For example, in first step temperatures is initialized as 10000 unit and then reduce to 999.99 unit in second step and then reduced to 999.98 unit and so on. It is anticipated that the maximum number of iterations need to repeat 100,000,000 times for each temperature values. The condition is set such that, covering of all T-way tuples are caused exit from the iteration. This type of interruption returns a temperature (called accepted temperature) which can found in Table IV and Table V. Initially large value of temperature 10,000 unit is selected for the iterative purpose and cooling rate 0.00001 to ensure the appearance of the most optimum and feasible TS size in the search domain. Then the cooling rate keeps fix and temperature is reduced very slowly to ensure the appearance of the most optimum and feasible TS size in the same search domain. For more accuracy, each temperature value is iterated ten times while the best and average TSs sizes found are recorded. The same experiments are applied on others configurations while the results are found in Table VI (for strength 3).

Table IV and V show the results for the above experiments, where the average and best TS sizes are shown respectively. The darkness cells indicate the preferred feasible TS size. Noted that, the TSs sizes are decimal data, hence, for average TS sizes, the higher decimal value is counted from the real type data (e.g., average TS size 7.10 is counted as 8). It is found in Table IV that, the feasible and optimum TS size for configuration SCT(N;3,4) is 6. Table IV and

TABLE IV. SAMPLE OF FEASIBLE TEST SUITE GENERATION FOR THE CONFIGURATION SCT(N;3,4) USING T-WAY EDISTC-SA .

Initial Temperature		10000	9000	8000	7000	6000	5000	4000	3000	2000	1000
CR	Description										
0.00001	Avg. TS Size	8	8	8	8	8	8	8	8	8	8
	Best TS Size	6	7	7	7	7	7	7	6	6	6
	Acpt. Temp	9366.97	8788.56	7348.23	6843.63	5242.18	4797.31	3796.16	2699.16	1449.62	99.96
0.0001	Avg. TS Size	8	8	8	8	8	8	8	8	8	8
	Best TS Size	6	6	7	6	6	7	8	7	6	6
	Acpt. Temp	9465.85	8753.48	7921.97	6863.99	5006.01	4870.45	3005.19	2956.81	1741.61	595.39
0.001	Avg. TS Size	8	9	8	8	8	8	8	8	8	8
	Best TS Size	6	7	8	7	6	6	7	7	7	6
	Acpt. Temp	9547.28	8198.98	7038.92	6080.33	5148.43	4822.89	3164.29	2814.24	1896.13	641.60
0.01	Avg. TS Size	8	8	8	8	8	8	8	8	8	8
	Best TS Size	7	7	6	6	7	6	6	7	7	7
	Acpt. Temp	4611.74	7394.20	3995.19	756.56	3953.01	2876.77	1672.35	878.47	1438.96	479.64
0.1	Avg. TS Size	8	8	8	8	7	8	8	8	8	8
	Best TS Size	7	6	7	6	6	6	7	7	8	7
	Acpt. Temp	8.12	450.08	197.75	5.18	8.44	90.33	183.70	606.24	2.91	12.64

TABLE V. SAMPLE OF FEASIBLE TEST SUITE GENERATOR FOR THE CONFIGURATION SCT(N;3,5) USING T-WAY EDIST-SA .

Initial Temperature		10000	9000	8000	7000	6000	5000	4000	3000	2000	1000
CR	Description										
0.00001	Avg. TS Size	12	11	11	11	11	11	12	12	11	11
	Best TS Size	9	9	9	9	10	9	10	10	10	10
	Acpt. Temp	9465.67	8585.31	7245.72	6948.26	5745.29	4141.96	3797.34	2898.52	1748.43	799.59
0.0001	Avg. TS Size	11	11	11	11	11	11	11	11	11	11
	Best TS Size	9	10	9	11	10	10	10	9	10	10
	Acpt. Temp	9837.82	8770.70	7516.85	6613.54	5476.40	4313.89	3435.79	2278.48	1861.82	845.59
0.001	Avg. TS Size	11	11	11	11	11	11	11	11	11	11
	Best TS Size	10	8	10	10	10	9	10	9	9	10
	Acpt. Temp	9462.77	8079.84	6478.04	3374.95	4682.12	3337.44	3551.29	1623.96	1720.76	851.85
0.01	Avg. TS Size	11	11	11	11	11	11	11	11	11	11
	Best TS Size	9	9	9	10	9	9	9	10	10	10
	Acpt. Temp	51.59	476.53	1499.03	1728.67	567.96	384.58	510.84	1251.36	5.03	491.74
0.1	Avg. TS Size	12	12	11	11	11	11	11	11	11	NA
	Best TS Size	11	12	10	11	11	10	11	10	10	NA
	Acpt. Temp	53.83	8.21	0.92	84.47	15.89	79.65	10.96	7.67	24.33	NA

Table V also report the average and minimum TS sizes with a specific setting of cooling rate and temperature. It can also be observed from the Tables IV and V, the temperature and cooling rate have impact on obtaining the TS. Consider Table V, at cooling rate 0.1, no result found because of small size of iteration. From the result sample, we can observe that, most of the TSs sizes are feasible but not optimum. Table VI shows a comparison with other T-way sequence input strategies.

Referring to Table IV and Table V, there is a trade-off between temperature value and cooling rates. Decreasing the cooling rate and increasing the temperature value improves the T-way EDISTC-SA performance. However, there are some limitations on memory allocation and processing time once go for higher events based configurations. In the case of SCT(N;3,4) configuration, Table IV shows that the minimum and optimum TS size is 6. On the other hand the SCT(N;3,5) configuration, Table V shows that the minimum TS size is 8.

Observation shows that in every cooling rate the optimum TS size can be achieved because of a large number of repetitions of search. In the case of SCT(N;3,5) configuration, the minimum TS size is found in the case of cooling rate 0.00001 and the size is 8. In this cooling rate huge numbers of repetitions are performed to find minimum TS size. The others feasible TSs size for the configuration SCT(N;3,5) are given in Table V. In such way, for the SCT(N;3,5) configuration, the minimum TS size is found at time of cooling rate 0.001 and the size is 8. In this cooling rate, a large number of repetitions search are performed to find minimum TS size. It should be noted that current T-way EDISTC-SA algorithm is designed for any number of strengths and events but empirical results are achieved only for strengths 3. This issue can be solved by improving the probability of acceptance function in EDISTC-SA algorithm and increasing the search iteration to achieve near optimum/optimum result.

Table VI shows the feasible test cases generated by EDISTC-SA, QnD and ASP at t=3. Figure 4 and Figure 5 show the EDISTC-SA experimented data graph for SCT(N;3,4) and SCT(N;3,5) configurations. In these figures the information about initial temperature, feasible temperature, feasible test suite, minimum number of test suite(s) and associate temperature are focused clearly. In the Figure 4 and Figure 5, the x- axis indicate 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. The sign (+) indicates the feasible test cases in y1-axis and associate temperature in y-axis.

TABLE VI. THE FEASIBLE TEST CASES GENERATED BY EDISTC-SA, QND[8] AND ASP[17] AT EVENT = 4 & 5, AND STRENGTH = 3.

Events	Exhaustive TCs	EDIST-SA (Feasible Test Cases)	QnD[8]	ASP[17]
4	24	6, 7, 8, 9 (Optimum is 6)	6	NA
5	120	{ 8, 9, 10, 11, 12, 13 }	8	8

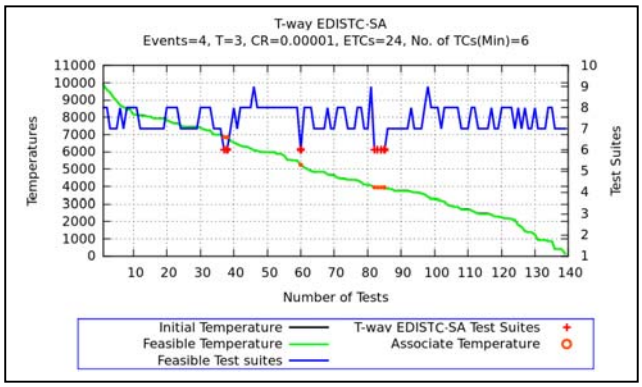

Figure 4. Experimental data graphs presentation for EDISTC-SA test suites associated with temperature (initial and feasible) at Events=4 and Strength=3.

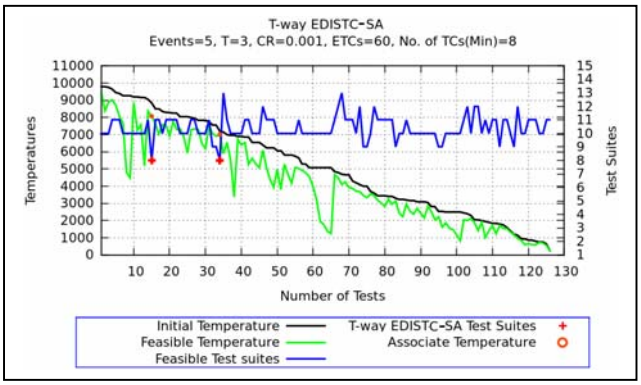

Figure 5. Experimental data graphs presentation for EDISTC-SA test suites associated with temperature (initial and feasible) at Events=5 and Strength=3.

VI. CONCLUSION

T-way EDISTC-SA test results are feasible and accepted compared to exhaustive test cases. Note that, experiment time is not considered with 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 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 input sequence 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 of 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 sequence tuple (Table V, CR=0.1, temperature=1000 units). 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

[1] A. Hartman, Graph Theory - Combinatorics and Algorithms : Software and Hardware Testing Using Combinatorial Covering Suites, Operations Research/Computer Science Interfaces Series, Vol. 34, Springer, 2005.

[2] G. Greg, “NIST Report Takes a Step toward Better Testing”, Software, IEEE, Volume:19 Issue:6, 2002.

[3] A. H. George, "Constructing Covering Arrays (using Parallel and Grid Computing)", PhD Thesis, Universitata Politecnica de Valencia, Spain, 2012.

[4] M. H. M. Zabil, K. Z. Zamli, & R. R. Othman, “Sequence-Based Interaction Testing Implementation Using Bees Algorithm”, Proc. IEEE Symposium on Computers & Informatics (ISCI’12), pp. 81-85, 2012.

[5] L. White, & H. Almezen, “Generating Test Cases for GUI Responsibilities Using Complete Interaction Sequences”, Proc. International Symposium on Software Reliability Engineering, Piscataway, NJ, pp. 110-121, 2000.

[6] K. Z. Zamli, M. F. J. Khalib, Younis, M. I., N. A. M. Isa, & R. Abdullah, “Design And Implementation Of A T-way Test Data Generation Strategy With Automated Execution Tool Support”, Information Sciences, vol. 181, issue 9, pp.1741-1758, 2011.

[7] Y. Lei, R. Kacker, R. Kuhn, V. Okun, & J. Lawrence, “IPOG/IPOGD: Efficient Test Generation for Multi-Way Combinatorial Testing”, Journal of Software Testing, Verification and Reliability, vol. 18, issue 3, pp. 125-148, 2008.

[8] D. R. Kuhn, J. M. Higdon, J. F. Lawrence, R. N. Kacker, and Y. Lei, “Combinatorial Methods for Event Sequence Testing”, Proc. IEEE 5th International Conference on Software Testing, Verification and Validation, pp. 601-609, 2012.

[9] A. A. Alsewari, and K. Z. Zamli, “Interaction Test Data Generation Using Harmony Search Algorithm”, In: IEEE Symposium on Industrial Electronics and Applications (ISIEA’11), Langkawi, Malaysia, pp. 559 – 564, 2011.

[10] B. S. Ahmed and K. Z. Zamli, “A Variable Strength Interaction Test Suites Generation Strategy Using Particle Swarm Optimization”, Journal of Systems and Software, vol. 84, issue 12, pp. 2171-2185, 2011.

[11] R. R. Othman, and K. Z. Zamli , “ITTDG: Integrated T-way Test Data Generation Strategy for Interaction Testing”, Scientific Research and Essays, 6(17), pp. 3638-3648, 2011.

[12] C. Nie, and H. Leung, “A survey of Combinatorial Testing”, ACM Computing Surveys, Vol.43, No. 2, Article 11, 2011.

[13] B. M. Cohen, P. B. Gibbons, W. B. Mugridge, C. J. Colbourn, J. S. Collofello , “Variable Strength Interaction Testing of Components”, Computer Software and Applications Conference, COMPSAC’03, pp. 413 – 418, 2003.

[14] M. Rahman, R. B. Ahmad, Z. I. A. Khalib, A. Yahya, M. M. Rahman, M. Ahmed, and N. Alee, "An Embedded Network Traffic Monitoring System for Portable Applications", Journal of Engineering and Applied Sciences, vol. 8, issue 1, pp. 1-8, 2013.

[15] J. Hopcroft, and J. Ullman, “Introduction to Automata Theory, Languages, and Computation”, Massachusetts: Addison-Wesley, 1979.

[16] D. R. Thompson, “Sample-Sort Simulated Annealing”, IEEE Transactions on Systems, MAN, and Cybernetics–Part B: Cybernetics, vol. 35, no. 3, 2005.

[17] M. Brain, E. Erdem, K. Inoue, J. Oetsch, J. Puhrer, H. Tompits, and C. Yilmaz, “ Event-Sequence Testing using Answer-Set Programming”, International Journal on Advances in Software, volume 5, number 3 and 4, pp. 237-251, 2012.

[18] M. B. Cohen, C. J. Colbourn and A. C. H. Ling, “Augmenting simulated annealing to build interaction test suites”. Proc. of the 14th International Symposium on Software Reliability Engineering. IEEE Computer Society, pp. 394–405, 2003.

[19] R. R. Othman, K. Z. Zamli, L. E. Nugroho, "General variable strength T-way strategy supporting flexible interactions", Maejo International Journal of Science and Technology, 6 (3), pp. 415-429, 2012.

[20] M. I. Younis, K. Z. Zamli, R. R. Othman, "Effectiveness of the cumulative vs. normal mode of operation for combinatorial testing", IEEE Symposium on Industrial Electronics and Applications (ISIEA'10), pp. 350-354, 2010.

[21] K. Z. Zamli, R. R. Othman, M. I. Younis, M. H. Mohamed Zabil, "Practical adoptions of T-way strategies for interaction testing", Communications in Computer and Information Science, 181 CCIS (PART 3), pp. 1-14, 2011.

[22] A. A-S. A. Rahman, K. Z. Zamli, "An Orchestrated Survey on T-Way Test Case Generation Strategies Based on Optimization Algorithms", The 8th International Conference on Robotic, Vision, Signal Processing & Power Applications,Lecture Notes in Electrical Engineering Volume 291, pp 255-263, 2014.

[23] R. R. Othman, N. Khamis, K. Z. Zamli, “Variable Strength T-Way Test Suite Generator with Constraints Support”, Malaysian Journal of Computer Science, 2014, Vol. 27(3), Pp204-217.