International Journal of Pure and Applied Mathematics
Volume 118 No. 20 2018, 181-189
ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
Special Issue

ijpam.eu

Generation of Test Cases Using Combinatorial Methods Based Multi-Output Domain of an Embedded System through the Process of Optimal Selection

M. Lakshmi Prasad[1]
[1]Ph.D Scholar, Dept. of Computer Science & Engineering,
NBKR Institute of Science and Technology
Vidyanagar, Nellore, AP, India
prasad.hinduniv@gmail.com.

Dr. J. K. R. Sastry[2]
[2] Dept. of Electronics and Computer Science Engineering,
Koneru Lakshmaiah Educational Foundation
Vaddeswram, Guntur District, AP, India.
drsastry@klce.ac.in

***Abstract*—the most important issue in any of the black-box testing is production of required test cases from input domain of the system under test (SUT) on the beginning of user's specification than on a multi output domain of a system. However, for some of the systems including embedded systems, test cases can be derived from output domain more easily than from the input domain, especially, when the output domain is minimal. This methodology ensures better reliability of the SUT.**

In this paper, the authors present an Optimal Selection Procedure (OSP) to choose the output values from multiple domains to derive the minimal number of test cases from a multi output domain embedded systems. In this paper a pilot project "Temperature Monitoring and Controlling of Nuclear Reactor System" (TMCNRS) has been considered as Multi output domain embedded system for experimentation.

Newly proposed algorithm generates test cases which are useful to conduct pseudo- exhaustive testing to detect all possible faults in the system output. The generator considers all the combinations from multiple output domains, and finds the corresponding inputs while optimizing the number of test cases generated. The Experimental results show that this approach is very effective and is applicable for different an embedded system that deals with multi-output domains.

Keywords— Optimal selection procedure; Output domain testing; Embedded systems; Cleanroom software engineering; Pseudo-exhaustive testing.

I. Introduction

Testing is playing a significant role in the development of any system which is a systematized process to verify the reliability, behavior and performance of a system against considered stipulations. It enables a device or a system to be as defect-free as possible which act as a one of the detective measures, and verification is one of the corrective measures of quality.

Black-box testing inspects the functionality of an application without seeing into its internal structures or workings. It mainly concentrates on the functional requirements of the embedded system without considering the internal working of the system. The main aim of this testing is to select the acceptable test cases and detect as many faults based on requirements specification at least cost and time.

Testing embedded systems considering testing Software, hardware and both. Testing of Hardware and software however can be conducted independently and then the testing has to be undertaken after the software is migrated to the hardware.

Embedded systems are a mixture of various computing devices such as microcontrollers, application-specific integrated circuits, and digital signal processors. Some widely used systems in real world applications such as routers, power plant system, medical instrument system, home appliances, air traffic control station, and firewalls, telecommunication exchanges, robotics, industrial automation and smart cards etc. are example of embedded system. Falseness in hardware systems may be designated in terms of defect, error and faults.

Combinatorial testing is commonly utilized black-box practice that could dramatically diminish the number of test cases, as it is a highly competent technique to perceive software faults. This method originates test cases from input domain of the system under test. But, when the input domain is noteworthy and the output domain is much tiny, it is desirable to go for testing the output domain either exhaustively [3, 4] or as much as possible.

For a few safety critical embedded systems, building test cases drawn from output domain will be more suitable than from input domain as it guarantees that all or as many possible output combinations are comprehensively tested. Exhaustive testing [5, 6] of output domain is out of question when many input variables exist and they act in several combinations. Pseudo-Exhaustive testing aims at considering only those combinations that will most likely result in failure conditions [5].

In this paper, the authors have proposed a method namely Optimal Selection Procedure (OSP) that derives test cases by choosing certain combinations of outputs based on feasibility for multi-output domain embedded system i.e. Temperature Monitoring and controlling of Nuclear Reactor system (TMCNRS).

In the above mentioned multi output embedded system, the operation of pumps and the buzzers are independent. But both the status of the pumps and the buzzers depend on the input temperatures detected by the temperature sensors. Test cases should be generated to verify the functionality of all the pumps and the buzzers. Based on the test case, the temperatures are to be simulated at the temperature sensors to check the functionality of pumps and buzzers.

In [1], Genetic algorithm has been used to generate the test cases to cover all the possible Buzzer Status Codes (BSC) and Pump Status Codes (PSC). In the genetic algorithm, a random gene is selected and it will be varied till the expected fitness value is achieved and it consumes more execution time as the number of temperature sensors increase.

In our proposed optimal selection procedure, we choose the values from the multi output domains which can be covered at the same time using the feasibility table. Due to this approach, instead of searching over all the possible input values to find out the best fitness value, we do reverse approach of selecting a temperature based on reference temperature to satisfy the given OSC code and given BSC code which saves lot of time in execution.

II. Literature Survey

Genetic algorithms have been used by Dr. V. Chandra Prakash et al., [1, 2] for generating test cases in the criticality regions of embedded systems, Genetic algorithms have been used in conjunction with combinatorial methods, Output domain of an embedded system that controls temperatures within nuclear reactor has been considered and they have shown how executively test cases have been generated in the criticality regions of the embedded systems. They have employed pseudo-exhaustive test case generation method to generate test cases that can find many faults at different levels. The generator developed by them considered all combinations of outputs and their relationships with Inputs. The number of test cases generated through the method proposed by them is found to be quite less compared to other methods.

Neural network based methods are in use for solving several problems including its use for generating test cases or for conducting actual testing. Ruilian Zhao et.al [3] has combined genetic algorithms and neural networks for generating test cases. They have created a function model that substitutes software. The function model identifies several computing notes that represent hidden nodes in the neural network. Genetic algorithm is used to fins the corresponding inputs using the outputs generated through neural network model.

There is interdependence among various faults that can occur while software is in execution. It becomes difficult to test software when faults occur due to occurrence of many other faults. It has been presented by D. Richard Kuhn et al [4] that few of the conditions built into the software causes many of the faults considering many of the domains for which software is written. Selecting combinations of inputs that can cover all kinds of faults that can occur will lead to generation of test cases that can exhaustively test software. However, the behavior of the software involves complex event sequences it is difficult to find accurate input combinations.

Covering arrays is another that has been advanced quite rapidly. Two methods have been proposed by D. Richard et al [5] that include covering arrays that represents interactions among the input variables and the method model checking using which the test cases are generated. They have tested the method on a traffic collision avoidance system. They have considered 6 way interactions among the input variables. Several tools existed for generating combinatorial test cases. R. Kuhn et al. [6] have presented a commission of various methods using real-world applications. They have presented empirical analysis of the methods.

Breeding test cases through genetic algorithms have been presented by D. Berndt, J. Fisher et al. [7]. The test cases can be breaded based on severity, novelty and proximity. A fitness function has been presented that reveal the behavior of microorganisms. The have presented the technique through a case study and using several visualization techniques using which the behavior is analyzed using fossil records.

Branch and find fault method along with genetic algorithm has been used for generating the test cases by B. F. Jones et al. [8]. The test cases generated by that method are fewer in number when compared to ransom testing. The methods also cover entire software. However, the computation effort in this case is very high. The test cases generated are very close to the sub-domains of inputs. Genetic algorithms have been used for exploiting the faults that can be caused when branching takes place within the software execution.

Adequate testing has to be carried to ensure entire coverage of the software requiring more number of test cases. Automated tools are required to generate adequate number of test cases. One such tool has been presented by Kamal Zuhairi Zamli et al [9]. The tool presented them uses combinatorial approach for generating the test cases. The have considered many combinations of inputs, environments on which the hardware and software is used and many of the system conditions for generating test cases through combinatorial methods. The conformance of the test cases against the system specification has also been shown by the authors.

A testing complex system is a challenge. A kind of system is required for testing software systems. Once unit testing is carried, building on it generates test cases for carrying system level testing is possible by suing the genetic algorithms. The test cases used for unit testing can be used as test case generation seeds. Watkins et al [10] have presented vocabulary of attributes and its related framework for generating test cases. The have also presented some visualization techniques using which one can find the system failures right in the Initial stages.

Pair wise testing method is quite frequently used combinatorial method for generating test cases. Huge number of test pairs gets created when number input variables are quite high. Meta- heuristic search techniques are required to find the pairs that are minimum and lead optimum number of test cases. Xiang Chen et al., [11] have used PSO (Particle swarm optimization) algorithm to find optimum input pairs. They have presented many algorithms based on PSO to systematically generate all the test cases. They have used two strategies that include IPO-Like and the other one-at-a-time strategy. In both these algorithms PSO is used for generating the test cases. They have also presented a method using which the search space can be defined.

PSO also has been used by B.S. Ahmed and K.Z. Zamli [12] for generating test cases using a t-way strategy. They compared test size reduction achieved by using this method against other similar methods presented in the literature. Some improved algorithms in similar lines have been presented by Jiang Shouda [13].

K. Rabbi, et al., [14] had proposed an efficient particle swarm intelligence based strategy to generate optimum test data in t-way testing. They represent a swarm intelligent based searching strategy to generate near optimum test data. The performances are analyzed and compared to other well-known strategies. Empirical result shows that the proposed strategy is highly acceptable in terms of the test data size.

Abdul Rahman [15] had presented a survey on input output relation stands on combination test data generation strategies They reviewed the existing combinatorial test data generation strategies supporting the IOR features specifically taking the nature inspired algorithm as the main basis. Benchmarking results illustrate the comparative performance of existing nature inspired algorithm based strategies supporting IOR

Combinatorial methods can also be used for testing software that predominantly uses logical expressions based on Boolean or binary inputs. In the software used related to most of the safety critical applications, Boolean expressions are used extensively. S. Vilkomir [16] has steadied effectiveness combinatorial testing when binary inputs are used.

Deepa Gupta [17] had proposed a sequence generation of test case using pair wise approach. They presented an approach which uses the series origination approach for pair wise test case origination. This approach makes certain to disseminate the required intent of trial run cases which cover all available relations between all instructions pairs at least once. Trial run selection specification is this approach based on combinatorial testing.

Jose Torres-Jimenez et al., [18] had presented a two stage algorithm for combinatorial testing. They suggested a two-stage simulated annealing algorithm to construct covering arrays through ternary covering arrays of strength three.

S. Route [19] presents two case studies of CTD implementation in client engagements and focuses on the approach, process and challenges addressed to scale up the implementation and make CTD a mainstream activity. The IBM Focus tool was employed in both cases to execute combinatorial test design for optimization of tests and for reducing test effort while increasing test coverage.

P. S et al., [20] had built the combinatorial test input model from use case artifacts Building this input space model is field knowledge and experience intensive task. Their main objective of the paper is to help test designer in building this test model. A rule based semi-automatic approach is proposed to derive the input space model elements from Use case specifications and UML use case diagrams. A natural language processing based parser and an XMI based parser are implemented. The rules formulated are applied on synthetic case studies and the output model is evaluated using precision and recall metrics.

Y. Yao et al., [21] had designed and implemented the various combinatorial testing tools. They provide useful combinatorial testing tools to carry the application of combinatorial testing technique on industrial scenarios, as well as in the academic research for combinatorial testing technique. A suite of combinatorial testing tools has been developed, whose functions include test case generation, test case optimization, and etc. For the requirements from both industrial and academic scenarios, the tools should be configurable, scalable, modular, and etc.

III. Pilot Embedded system

A Pilot project has been developed which is meant for monitoring and controlling temperatures within a nuclear reactor system (TMCNRS), the hardware block diagram of which is shown in Figure 1.

Temperature sensors fixed in the nuclear reactor tubes measures the temperatures, and the analog signals representing the temperatures are amplified by signal conditioners and the same are fed as input to A/D converter. The firmware placed in the micro-controllers reads the temperatures into digital form and the same are placed in RAM.

Two pumps and a Buzzer are connected to the system. The pumps control the flow of coolant into nuclear reactor tubes. The operation of the pumps is controlled based on whether a sensed temperature is greater than a reference temperature or otherwise. The pumps are operated through relays which can be made to be open or closed by triggering a signal on the output PIN to which the relays are connected. A host connected to the Micro Controlled through RS232C interface is used for feeding the reference temperatures which are used for comparing with the sensed temperatures.

The TMCNRS is protected through password protection. Keyboard which is connected to the micro controller system is used for imputing the password. An LCD is also connected to the Micro controller system for displaying the temperatures that are sensed by the sensors and also for displaying alerts when huge variations are sensed in temperature gradients. The requirements that must met by TMCNRS are shown in TableI.

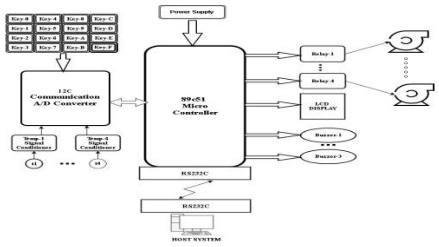


Fig. 1 System Integration Diagram of TMCNRS

TABLE I. REQUIREMENT CONDITION OF TMCNRS

S. No.	Functional requirements
Req. 1	Temperature 1 must be sensed once in 10 Milli Seconds and the same must be displayed on LCD and transmitted to a remote HOST. The Relay connected to PUMP 1 must be opened when the reference temperature 1 is less than the sensed temperature 1.
Req. 2	Temperature 2 must be sensed once in 10 Milli Seconds and the same must be displayed on LCD and transmitted to a remote HOST. The Relay connected to PUMP 2 must be opened when the reference temperature 2 is less than the sensed temperature 2
Req. 3	The buzzer must be activated if the temperature gradient between the temperature 1 and temperature 2 is beyond the expected regions.

TABLE II RELATIONSHIP BETWEEN INPUT PARAMETERS AND STATUS OF PUMP AND BUZZER OUTPUTS

Temp. sensed by Sensor 1 (in ^oC)	Temp. sensed by Sensor 2 (in ^oC)	Expected Status of Pump1 Ref=30^oC	Expected Status of Pump2 Ref=32^oC	Expected Status of Buzzer1
30	31	0 (Pump OFF)	0 (Pump OFF)	0 (Buzzer OFF)
20	33	0 (Pump OFF)	1 (Pump ON)	1 (Buzzer ON)
33	30	1 (Pump ON)	0 (Pump OFF)	1 (Buzzer ON)
32	33	1 (Pump ON)	1 (Pump ON)	0 (Buzzer OFF)

IV. GENERATION OF TEST CASES FOR TMCNRS USING OPTIMAL SELECTION PROCEDURE

This test case can be derived from the output domain. In the current scenario, we have two output domains that have to be covered, one is Pump Status Code (PSC) i.e. in the above system OSC is pump and another is Buzzer Status Code (BSC).

As mentioned in [1], 2^8 test cases can be generated to cover all the BSC codes and 2^7 test cases can be generated to cover all the OSC codes for 8 sensors and 7 buzzer TMCNRS system. Authors in [1] suggested pseudo-exhaustive testing can be conducted, where first test cases are generated to cover all OSC codes and then test cases are generated to cover all BSC codes. So, total of 2^8+2^7 test cases are required to cover all possible outputs in multi output system.

In this paper, we have proposed Algorithm 1 to cover all possible BSC codes, and Algorithm 2 to cover all possible PSC codes.

Algorithm 1: This algorithm explains how to derive the test case from the given BSC code.

Assuming buzzer code will be 1 when absolute difference of consecutive temperature sensors is more than 2 (assumed Gradient value), otherwise 0.

Let BSC be a vector containing n-1 binary values each representing a set of bits (b_1, b2, b3$_{\dots}$b$_{n-1)}$ where b_i (i=1, 2… (n-1)), takes value either 0 or 1.

Let PSC be a vector containing n binary values, each represented by a set of bits showing the status of the pumps related to specific temperatures ti (i=1, 2 …n).

As the buzzer output depends on the absolute difference of adjacent temperatures, Let the first temperature value is selected randomly from 1 to 99 and second value t_i ($2\leq i\leq n-1$) is derived from the following equation.

First temperature is defined as t_1= random (1, 99). Then the temperatures t_i (i= 2 …n) is calculated as

$$t_i = \begin{cases} t_{i-1} + \text{Random } [-2, 2] \text{ if } b_i = 0 \\ \text{Random } [1, 99] \text{ such that } |\text{Random } [1, 99] - t_{i-1}| > 2 \\ \text{if } b_i = 1 \quad \text{for } 2\leq i\leq n \end{cases} \quad (1)$$

Pseudo code 1: Generate test cases based on BSC array

```
Initialize BSC array = {0, 1, 2, 3…, ($2^{n-1}$-1)} of a multi output
embedded system each value representing a binary value
While (No. of BSC elements) > 0
{
    Get a BSC code from BSC array.
    Generate the test case from the above BSC code using
    equation 1.
    Remove the selected BSC code from the BSC array.
    Decrement the number of BSC elements.
}
```

Algorithm-2: This algorithm tells how to derive the test case from the given PSC code based on the reference temperature inputs.

Let (Ref_1, Ref_2, and $Ref3_{...}Ref_n$) be the Reference temperature of the sensors used in the system.

Let (p_1, p_2, $p_{3...}p_{n)}$ be the given PSC code, where p_i (i=1, 2...n) takes value either 0 or 1.

(t_1, t_2, $t_{3...}t_{n)}$ be the test vector being generated from the given PSC code and reference temperatures.

$$t_i = \begin{cases} \text{Random } [1, Ref_i] \text{ if } p_i = 0 \\ \text{Random } [Ref_i+1, 99] \text{ if } p_i = \text{ for } 1<=i<=n \end{cases} \qquad (2)$$

Pseudo code 2: Generate test cases based on PSC array

```
Initialize OSC array = {0, 1, 2, 3..., ($2^n$-1)} of a multi output embedded system
While (no. of OSC elements >0)
{
    Get an OSC code from OSC array.
    Generate the test case from the above OSC code using equation 2.
    Remove the selected OSC code from the OSC array.
    Decrement the number of OSC elements.
}
```

In [1], authors proposed one more optimization technique which finds the union of two sets getting a resultant set whose size is smaller in most of the cases i.e. 256 <= size of the resultant set <= 2^8+2^7.

Authors in [1,2] used genetic algorithm, in order to obtain the resultant set, ATCG first generates a test suite (set) for conducting exhaustive testing of buzzers. It, then, considers that suite as the initial population and generates the test suite for pumps. Thus, most of the test cases of buzzers (sometimes all) can be accommodated in the test suite of pumps.

In similar lines, we proposed yet another algorithm which derives test cases that covers all possible BSC and OSC codes together till all BSC codes are covered, by properly choosing the OSC code and BSC code based on Feasibility Table as shown in table 3. Test cases for the left over OSC codes are generated using Pseudo code 2.

Algorithm 3: This algorithm tells how to derive the test case from the given OSC code and BSC code based on the reference temperature inputs.

Step 1: Let (Ref_1, Ref_2, $Ref_{3...}Ref_n$) be the Reference temperature of the sensors used in the system.

Let (p_1, p_2, $p_{3...}p_n$) be the given PSC code, where p_i (i=1, 2...n) takes value either 0 or 1.

Assuming buzzer code will be 1 when absolute difference of consecutive temperature sensors is more than 2, otherwise 0.

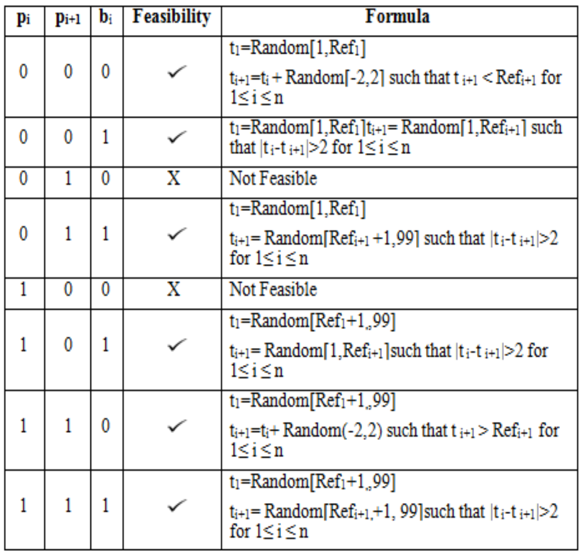
Let (b_1, b_2, $b_{3...}b_{n-1)}$ be the given BSC code, where b_i (i=1, 2, (n-1)) takes value either 0 or 1.

(t_1, t_2, $t_{3...}t_{n)}$ be the test vector being generated from the given PSC code and reference temperatures.

Before proceeding to generate the test case using BSC code and PSC code together, we should check the feasibility of the existence of these two codes first from table III.

Random [x, y] generates the random number between x and y. This is achieved in the program using rand () function as x+ rand () % (y-x+1).

TABLE III FEASIBILITY CHECK TABLE



p_i	p_{i+1}	b_i	Feasibility	Formula
0	0	0	✓	t_1=Random[1,Ref_1] t_{i+1}=t_i + Random[-2,2] such that t_{i+1} < Ref_{i+1} for $1 \le i \le n$
0	0	1	✓	t_1=Random[1,Ref_1]t_{i+1}= Random[1,Ref_{i+1}] such that $\|t_i - t_{i+1}\|$>2 for $1 \le i \le n$
0	1	0	X	Not Feasible
0	1	1	✓	t_1=Random[1,Ref_1] t_{i+1}= Random[Ref_{i+1} +1,99] such that $\|t_i - t_{i+1}\|$>2 for $1 \le i \le n$
1	0	0	X	Not Feasible
1	0	1	✓	t_1=Random[Ref_1+1,,99] t_{i+1}= Random[1,Ref_{i+1}]such that $\|t_i - t_{i+1}\|$>2 for $1 \le i \le n$
1	1	0	✓	t_1=Random[Ref_1+1,,99] t_{i+1}=t_i+ Random(-2,2) such that t_{i+1} > Ref_{i+1} for $1 \le i \le n$
1	1	1	✓	t_1=Random[Ref_1+1,,99] t_{i+1}= Random[$Ref_{i+1,}$+1, 99]such that $\|t_i - t_{i+1}\|$>2 for $1 \le i \le n$

Example for Feasibility check:

Let us consider an example of PSC code = (0,1,0,0) and BSC code = (0,0,0) and Reference temperatures as (30,32,34,36).

First find temperature t1 using the equation (2).

t_1= Random [1, Ref1]; say 30

Now let us calculate t_2.

From equation (1) to satisfy BSC code, $t_2= t_1+$ Random [-2, 2] i.e. 30±2

From equation (2) to satisfy OSC code, t_2 = Random [Ref_2+1, 99] i.e. Random [32+1, 99]

Now we are left with two solutions for t_2 which doesn't co-exist together.

This kind of selection of PSC code and BSC code together is not possible. i.e. $p_i =0; p_{i+1} = 1$ and $b_i=0$. Table 3 tells about the feasibility of the possible values of p_i and b_i.

PSC code (p_1, p2, p3, p4) and BSC code (b_1, b2, b3) is said to feasible:

If p_i, p_{i+1} and b_i satisfies the above table for each i = 1,...n-1 (3)

Step-2: Optimal Selection of test case using Feasibility Table

In this section, we will discuss about how to derive the test vector i.e. (t_1, t_2, $t_{3\ldots}t_{n)}$ satisfying both the OSC and BSC code.

Let us consider an example of PSC code = (1, 0, 0, 1) and BSC code = (1, 0, 1) which means pumps p_1, p_4 should be ON and p_2, $p_3$should be OFF, and the buzzers b_1, b_3 should be ON and $b_2$should be OFF.

We can observe that the selected PSC code and BSC code is feasible from equation (3). For the given example, p_1=1, p_2=0 and b_1=1 and from the feasibility table, we can get t_1 and t_2 as below.

t_1=Random [Ref_1+1, 99]

$t_2=t_1+$ Random [1, Ref_2] such that $|t_1-t_2| > 2$

t_1 and t_2 are derived now, move to next values in OSC code i.e. p_2 = 0, p_3=0 and b_2=0.

Now let's proceed to derive t_3.

We proceed to derive t3 directly from the table $t_3=t_2+$ Random [-2, 2] such that $t_3 \leq Ref_3$.

Next step is to derive t_4, from provided input of p_3=0, p_4=1 and b_3=1, t_4= Random [Ref_4+1, 99] such that $|t_4-t_3| > 2$.

In this same manner we derive the test case for any number of temperature sensors and for given PSC and BSC code as long as the feasibility is met.

Main Algorithm:

```
Initialize PSC array = {0, 1, 2, 3..., $(2^n-1)$}
Initialize BSC array = {0, 1, 2, 3..., $(2^{n-1}-1)$
While (no. of BSC elements>0)
{
    For each BSC code, find a PSC code which is feasible
    based on feasibility table.
    Use the above BSC code and PSC code generate the test
    case using Algorithm 3.
    Remove the BSC code from BSC array
    Decrement the no. of BSC elements by 1
    Remove the PSC code from PSC array
    Decrement the no. of PSC elements by 1
}
While (no. of PSC elements >0)
{
    Get a PSC code from PSC array.
    Generate the test case from the above PSC code using
    equation 2.
    Remove the selected PSC code from the PSC array.
    Decrement the no. of PSC elements.
}
```

With the first part of algorithm, we get the test cases that coverall the possible BSC codes (2^{n-1}) along with partial coverage of PSC codes (2^{n-1}). And we will be left with another 2^{n-1} PSC codes in the PSC array. Second part of the algorithm derives test case from the leftover PSC codes from the above step.

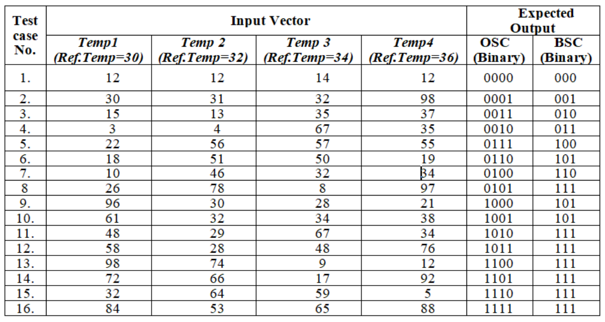
For simplicity, the PSC codes and BSC codes in the respective arrays are denoted in decimal form rather than in the binary form.

V. Experimental results

In this paper Optimal Selection Procedure (OSP) is implemented to derive the test cases for multi output domain embedded system. The optimized test suit derived from the multi output domain of TMCNRS with a configuration of 4-Sensors, 4-Pumps, 3-buzzers using optimal selection procedure is shown in Table IV. The optimized test suit derived for TMCNRS with a configuration of 8-Sensors, 8-Pumps, 7-buzzers using optimal selection procedure is shown in Table V.

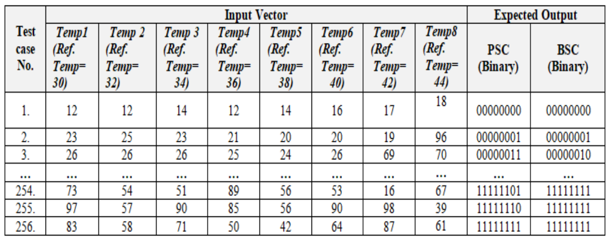
In order to calculate the effectiveness of this approach, the results have been compared with the results generated by the Genetic algorithm [1] and particle swarm optimization technique also. This approach yields to produce a less number of test cases with high efficiency in terms of time when compared to the previous existing technique i.e. genetic algorithms and particle swarm optimization as shown in Table VI. Execution time for various input parameters using 64 bit windows system is shown in Table VII.

TABLE IV Optimized Test suite Generated from Multi Output Domain of TMCRS Using OSP



Test case No.	Input Vector				Expected Output	
	Temp1 (Ref.Temp=30)	*Temp 2 (Ref.Temp=32)*	*Temp 3 (Ref.Temp=34)*	*Temp4 (Ref.Temp=36)*	OSC (Binary)	BSC (Binary)
1.	12	12	14	12	0000	000
2.	30	31	32	98	0001	001
3.	15	13	35	37	0011	010
4.	3	4	67	35	0010	011
5.	22	56	57	55	0111	100
6.	18	51	50	19	0110	101
7.	10	46	32	34	0100	110
8	26	78	8	97	0101	111
9.	96	30	28	21	1000	101
10.	61	32	34	38	1001	101
11.	48	29	67	34	1010	111
12.	58	28	48	76	1011	111
13.	98	74	9	12	1100	111
14.	72	66	17	92	1101	111
15.	32	64	59	5	1110	111
16.	84	53	65	88	1111	111

TABLE V OPTIMIZED TEST SUITE GENERATED FROM MULTI OUTPUT DOMAIN OF TMCRS



Test case No.	Input Vector								Expected Output	
	Temp1 (Ref. Temp= 30)	*Temp 2 (Ref. Temp= 32)*	*Temp 3 (Ref. Temp= 34)*	*Temp4 (Ref. Temp= 36)*	*Temp5 (Ref. Temp= 38)*	*Temp6 (Ref. Temp= 40)*	*Temp7 (Ref. Temp= 42)*	*Temp8 (Ref. Temp= 44)*	PSC (Binary)	BSC (Binary)
1.	12	12	14	12	14	16	17	18	00000000	00000000
2.	23	25	23	21	20	20	19	96	00000001	00000001
3.	26	26	26	25	24	26	69	70	00000011	00000010
...	...	...	...	...	...	...	...	...	...	...
254.	73	54	51	89	56	53	16	67	11111101	11111111
255.	97	57	90	85	56	90	98	39	11111110	11111111
256.	83	58	71	50	42	64	87	61	11111111	11111111

TABLE VI. ASSOCIATION BETWEEN PSEUDO EXHAUSTIVE TESTING USING GA,PSO AND OPTIMAL SELECTION PROCEDURE

S.No	Type of Testing	No of test case required			
		For 4-Pumps 3-Buzzers Config.	*For 6-Pumps 5-Buzzers Config.*	*For 8-Pumps 7-Buzzers Config.*	*For 10-Pumps 9-Buzzers Config.*
1.	Exhaustive Testing of input Domain	96,059,601	$9.4x10^{11}$	$9.2x10^{15}$	$9.0x10^{19}$
2.	Exhaustive Testing of Output Domain	128	2048	32,768	5,24,288
3.	Pseudo Exhaustive Testing using Genetic Algo. Ref[1]	16	64	259	1260
4.	Particle Swarm Optimization (PSO)	16	64	259	1026
5.	**Optimal Selection Procedure (OSP)**	**16**	**64**	**256**	**1024**

TABLE VII. EFFICIENCY OF OSP WITH RESPECT TO EXECUTION TIME (SECONDS)

S.No	Name of the Technique	*For 4-Pumps 3-Buzzers Config.*	*For 6-Pumps 5-Buzzers Config.*	*For 8-Pumps 7-Buzzers Config.*	*For 10-Pumps 9-Buzzers Config.*
1.	Genetic Algorithms	0.007	0.75	5.09	119.91
2.	Particle Swarm Optimization (PSO)	0.004	0.216	3.18	22.35
3.	**Optimal Selection Procedure (OSP)**	**0.004**	**0.013**	**0.076**	**0.476**

VI. CONCLUSION AND FUTURE ENHANCEMENTS

Input domain and output domain based combinatorial testing using combinations of inputs and outputs have been proved to be effective methods when combined with genetic algorithms or heuristic search methods. There are many applications that produce clusters of output called Multi-output domain. The output domain as such can be recognized as clusters of output vectors, each vector having common characteristics. In such situations Optimal Selection Procedure presented in this paper will generate most effective test cases that cover 100% of the source code. In the example used for showing the effectiveness of the procedure, two output vectors have been chosen. While one output vector shows the status of pumps, the second vector deals with the status of buzzers.

REFERENCES

[1]. C. P. Vudatha, S. Nalliboena, S. K. Jammalamadaka, B. K. K. Duvvuri and L. S. S. Reddy, "Automated generation of test cases from output domain of an embedded system using Genetic algorithms," 2011 3rd International Conference on Electronics Computer Technology, Kanyakumari, pp. 216-220, 2011.

[2]. C. P. Vudatha, S. Nalliboena, S. K. Jammalamadaka, B. K. K. Duvvuri and L. S. S. Reddy, "Automated generation of test cases from output domain and critical regions of embedded systems using genetic algorithms," 2nd National Conference on Emerging Trends and Applications in Computer Science, Shillong, pp. 1-6, 2011.

[3]. Ruilian Zhao et al, "Neural-Network Based Test Cases Generation Using Genetic Algorithm,"13th IEEE International Symposium on Pacific Rim Dependable Computing, pp.97-100, 2008.

[4]. D. Richard Kuhn et al, "Software Fault Interactions and Implications for Software Testing," IEEE transactions on software engineering, Vol.30, No.6, June 2004.

[5]. D. Richard et al, "Pseudo-Exhaustive Testing for Software", 30th Annual IEEE/NASA Software Engineering WorkshopSEW-30 (SEW'06), 2006.

[6]. R. Kuhn et al. "Practical Combinatorial Testing: beyond Pair wise", IEEE Computer Society - IT Professional, Vol. 10, No. 3, May/June 2008.

[7]. D. Berndt, J. Fisher et al, "Breeding Software Test Cases with Genetic Algorithms", IEEE Proceedings of the 36th Hawaii International Conference on System Sciences, 2003.

[8]. B. F. Jones et al, “A Strategy for using Genetic Algorithms to Automate Branch and Fault-based Testing”, The Computer Journal, Vol. 41, No. 2, 1998.

[9]. Kamal Zuhairi Zamli et al, “A Tool for Automated Test Data Generation (and Execution) Based on Combinatorial Approach”, International Journal of Software Engineering and Its Applications Vol. 1, No. 1, July, 2007.

[10]. Watkins et al, “Breeding Software Test Cases for Complex Systems”, IEEE Proceedings of the 37th Hawaii International Conference on System Sciences – 2004.

[11]. Xiang Chen, Qing Gu, Jingxian Qi and Daoxu Chen, “Applying Particle Swarm Optimization to Pairwise Testing,” COMPSAC, 2010 IEEE 34th Annual Computer Software and Applications Conference, pp. no. 107-116, 19-23 July 2010.

[12]. B.S. Ahmed and K.Z. Zamli, "PSTG: A T-Way Strategy Adopting Particle Swarm Optimization," AMS, 2010 Fourth Asia International Conference on Mathematical/Analytical Modeling and Computer Simulation, pp.no.1-5, 26-28 May 2010

[13]. Wang Jian Feng and Jiang Shouda, "An Improved Algorithm for Test Data Generation Based on Particle Swarm Optimization," In Proceedings of the 2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control (IMCCC '11). IEEE Computer Society, Washington, DC, USA, pp.no.404-407, 2011.

[14]. K. Rabbi, Q. Mamun and M. R. Islam, "An efficient particle swarm intelligence based strategy to generate optimum test data in t-way testing," IEEE 10th Conference on Industrial Electronics and Applications (ICIEA), Auckland, 2015, pp. 123-128, 2015.

[15]. AbdulRahman A. Alsewari, Nasser M. Tairan and Kamal Z. Zamli, “Survey on input output relation based combination test data generation strategies," ARPN Journal of Engineering and Applied Sciences vol. 10, no.18, October 2015.

[16]. S. Vilkomir, "Combinatorial Testing of Software with Binary Inputs: A State-of-the-Art Review," IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Vienna, 2016, pp. 55-60, 2016.

[17]. Deepa Gupta, Ajay Rana, Sanjay Tyagi, "Sequence Generation of Test Case Using Pairwise Approach Methodology," Advances in Computer and Computational Sciences, pp 79-85, 2016.

[18]. Jose Torres-Jimenez, Himer Avila-George, Idelfonso Izquierdo-Marquez, "A two stage algorithm for combinatorial testing," OptimizationLetters, Volume 11, Issue 3, pp 457–469, March 2017.

[19]. S. Route, "Test Optimization Using Combinatorial Test Design: Real-World Experience in Deployment of Combinatorial Testing at Scale," IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Tokyo, pp. 278-279, 2017.

[20]. P. S., M. B., M. S. Narayan and K. Rangarajan, "Building Combinatorial Test Input Model from Use Case Artefacts," IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 220-228, 2017.

[21]. Y. Yao, Y. Yan, Z. Wang and C. Liu, "Design and Implementation of Combinatorial Testing Tools," 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Prague, pp. 320-325, 2017.