In the last 20 years, software has grown tremendously in terms of size (that is, line of codes (LOCs), and functionalities). With the increase in LOCs and functionalities, more and more unwanted interactions amongst software systems, hardware components, and operating systems are to be expected, rendering increased possibility of faults. While traditional static and dynamic testing strategies are useful for fault detection and prevention, they may not be sufficient to tackle bugs due to interaction. This paper describes the development of an integrated t-way test data generation (ITTDG) strategy for interaction testing (where t indicates the interaction strength). Unlike most existing works, ITTDG seamlessly integrates all forms of interaction possibilities including uniform, variable strength, and input output based relationship. Empirical evidence demonstrates that, ITTDG produces competitive test size as compared to existing strategies on a number of benchmark configurations.

Key words: Interaction testing, t-way test data generation, uniform strength, variable strength, input-output relationship.

INTRODUCTION

Our continuous dependencies on software often raise dependability issue. In the last 20 years, software has grown tremendously in terms of size (that is, line of codes (LOCs), and functionalities). With the increase in LOCs and functionalities, more and more unwanted interactions amongst software systems, hardware components, and operating systems are to be expected, rendering increased possibility of faults. While traditional static and dynamic testing strategies (for example, boundary value analysis, cause and effect analysis and equivalent partitioning) are useful for fault detection and prevention (Zamli et al., 2008), they may not be sufficient to tackle bugs due to interaction.

Addressing this issue, researchers are now focusing on a sampling method, termed t-way testing strategy (that is, where t indicates interaction strength). Despite its inherent benefit in terms of minimizing the test data for consideration, the applicability and adoption of existing t-way strategies in the industry appears to be lacking (Czerwonka, 2006). Rather than giving the test engineers (as domain experts) the flexibility to choose amongst all interaction possibilities, some strategies dictates only uniform t-way interactions (for example, GTWay (Klaib, 2009), in-parameter-order general (IPOG) (Lei et al., 2007), multi-core modified IPOG (MC-MIPOG) (Younis, 2010), TConfig (Williams, 2010), Jenny (Jenkins, 2010), and IBM’s test case handle (ITCH) (Hartman et al., 2010) while others impose on variable strength interaction (for example, simulated annealing (SA) (Cohen et al., 2003), ant colony system (ACS) (Chen et al., 2009) and pairwise independent combinatorial testing (PICT) (Czerwonka, 2006)). In fact, there are also strategies that prescribe interactions due to input-output based relationship (for example, ReqOrder (Wang et al., 2007), Union (Schroeder, 2001) and Greedy (Schroeder et al., 2002)). As such, there is a need for a strategy that is sufficiently flexible and be able to integrate all forms of interaction
possibilities. In this manner, test engineers can readily exercise their creativity depending on the testing problem at hand.

Not until recently, a number of such strategies have started to appear (for example, Density (Wang et al., 2008), ParaOrder (Wang et al., 2007) and TVG (Arshem, 2010). However, as t-way test data generation is NP hard problem (Klaib, 2009; Cohen et al., 1997), no single existing strategy can claim dominance as far optimality of test size is concerned. Motivated by the aforementioned challenges, this paper discusses the development of a new strategy, called integrated t-way test data generation (ITTDG) that seamlessly integrates all interaction possibilities including uniform, variable strength, and input output based relationship. Empirical evidence demonstrates that ITTDG produces competitive test size on a number of benchmark configurations as compared to existing strategies.

Related work

In general, existing t-way strategies can be categorized either as a one-test-at-a-time (OTAT) strategy or a one-parameter-at-a-time (OPAT) strategy. An OTAT strategy generates the final test suite by adding one complete test data into the final test iteratively. Based on the method of generating test data, OTAT strategies can be further characterize into three categories as follows.

The first category is an iterative based OTAT strategy. The strategies in this category executed certain process iteratively to produce the test data. GTWay (Klaib, 2009; Zamli et al., 2011a), ITCH (Hartman et al., 2010), Jenny (Jenkins, 2010), TVG (Arshem, 2010), PICT (Czerwonka, 2006), Union (Schroeder, 2001; Schroeder and Korel, 2000) and Greedy (Schroeder et al., 2002) are the example of iterative based OTAT strategy. GTWay is a t-way strategy that iteratively used the backtracking algorithm in generating a complete test data while ITCH, on the other hand, utilizes its exhaustive search algorithm in constructing the test data.

As for Jenny, TVG and PICT, these three tools are a public domain tools and available for download at the developer’s site. Jenny generates test data by constructing a test suite that covers 1-way interaction first. The strategy then extends the test suite to cover 2-way interaction and repeating the process until the test suite covers t-way interaction where t is the interaction requested by the user. Unlike Jenny, PICT generates test data by selecting one uncovered tuple (the parameter-value combinations produced by interaction) and iteratively fill the “do not care” (parameters that does not involve with current tuple) with the best found value (that covers the most uncovered tuples). Different from other strategies, TVG generate test data either using its T-Reduces, Plus-One or Random Sets algorithm. Due to limited description in the literature, it is not clear how each of the algorithms (for example, T-Reduced, Plus-One or Random Sets) is actually implemented. Based on our experience with TVG; T-Reduced often produces the most optimal test suite as compared to Plus-One and Random Sets. As for Union, the strategy generate a complete test data to satisfy all the input-output relation and remove any repeating test data (that is, finding the union of the test data) to find the optimal test suite. Unlike Union, Greedy select test data based on greedy algorithm as its final test suite candidates.

The second category is the artificial life-based (AL) OTAT strategy. As the name suggests, the AL-based OTAT strategy adopts an artificial life technique for generating the test data. Genetic algorithm (GA) and ant colony algorithm (ACA) are amongst the most common AL technique that has been adopted for generating interaction test data. GA based strategy typically starts by randomly creating a number of test cases (m), as chromosomes, in a test candidate list. The chromosomes inside the candidate go through a series of mutation processes until the desirable interaction criteria are met. In the end, the best chromosomes inside the test candidate list will be taken in the final test suite. As for ACA, the candidate test cases are searched by colonies of ants on some possible paths. The paths qualities are evaluated in terms of the pheromones which signify convergence. Here, the optimum paths correspond to the best test candidate to be included in the final test suite. Concerning related work, GA (Shiba et al., 2004) and ACA (Shiba et al., 2004) are the Shibas's implementation of genetic algorithm and ant colony algorithm respectively, while GA-N (Nie et al., 2005) is the Nie’s version of implementation based on genetic algorithm. In other work, Chen (Chen et al., 2009) also introduces a variant ant colony based strategy, called ACS.

The last category of OTAT strategy is the heuristic based strategy. Heuristic based OTAT strategy uses a heuristic method in deciding the final test data. Density (Wang et al., 2008), simulated annealing (SA) (Cohen et al., 2003) and automatic efficient test generator (AETG) (Cohen et al., 1994) are the strategies that can be characterized as heuristic based OTAT strategy. Density used the global and local density calculation (extension of “density” concept introduced by Bryce and Colbourn (2009, 2007) for constructing the final test data. Formulae to calculate both global and local density can be found in Wang et al. (2008). As for SA, the strategy starts with constructing a feasible solution and the applying a series of transformation to the solution until the solution cover all generated pairs. A binary search algorithm is adopted in order to find the smallest possible solution. In the case of AETG, the strategy generates a number of test data candidates based on the parameter-value configuration that cover the most number of uncovered tuples. From the list of test data candidate, a test data that covered the most uncovered tuples will be selected in the final test suite.
OPAT strategy is essentially strategies that adopt the vertical and horizontal extension in Lei et al. (2007) order to generate test data. Unlike OTAT strategy, OPAT strategy generates an initial test suite (an exhaustive test suite for selected number of parameters only) and extends the test suite by adding one-parameter-at-a-time (horizontal extension). To ensure the interaction coverage, a completely new test data may be added into the test suite during the horizontal extension (vertical extension). In-parameter-order (IPO), a pairwise strategy that is, 2-way interaction) proposed by Lei and Tai (1998), is the pioneer of OPAT strategy. Later, Lei et al. (2007) introduces IPOG the general version of IPO to support higher order of interaction. Younis and Zamli (2010) introduce MC-MIPOG, a multi-core version of IPOG. In addition, Nie et al. (2005) and Williams (2010) also came out with their own version of IPOG called IPO-N and TConfig respectively. Apart from that, Wang et al. (2007) also introduces two new strategies, called ParaOrder and ReqOrder. ParaOrder and ReqOrder differ for its predecessor, IPOG, in terms of how the initial test case is generated. In IPOG, the initial test case is generated in-defined-order-of-parameter found whilst, in ParaOrder, the initial test case is generated based on the first defined input output relationships. In the case of ReqOrder, the selection of initial test case does not necessarily follow the first defined input output relationship.

**METHODOLOGY**

**Theoretical background**

To introduce the concept of interaction (or t-way) testing, consider an integrated home security system shown in Figure 1. The system consist of three inputs from various sensors (that is, to detect both fire and intrusion) and one input from security control panel which is used to activate or deactivate the intrusion detection system. The system operation is as follows. If fire is detected through smoke detector, the system will ring the security bell. If intrusion is detected (either from the glass break detector or door open sensor while security control panel is set to activate) the system will alert the house owner via short messaging services (SMS). The parameters configuration for the system is summarized in Table 1.

As far as testing the integrated home security system is concerned, 3 types of interaction can be considered (that is, uniform strength interaction, variable strength interaction and input-output based relations). Uniform strength interaction is the most common interaction type. In uniform strength interaction, it is assumed that every parameter in the system interacts uniformly with single value of interaction strength. For example, the integrated home security system given earlier can be tested using 2-way testing (with the interaction strength = 2). The final test suite, $F$, should cover any 2 combinations of parameter values by at least once (that is, (smoke detector, glass break Detector), (smoke detector, door open sensor), (smoke detector, security control panel), (glass break detector, door open sensor), (glass break detector, security control panel) and (door open sensor, security control panel)). Mathematically, the final test suite for uniform interaction can be expressed using covering array notation from Cohen (2004) and Zekaoui (2006) as in Equation 1:

$$F = CA(N, t, C)$$

Where, $N$ = the number of test data inside the final test suite, $t$ = the interaction strength, $C$ = value configuration can be represented as follows: $v_1^{p_1}, v_2^{p_2}, ..., v_t^{p_t}$ which indicate that there are $p_1$ parameters with $v_1$ values, $p_2$ parameters with $v_2$ values and so on.

Another variant of interaction is variable strength interaction. Variable strength interaction extends the notion of uniform strength interaction. Like uniform strength interaction, variable strength interaction consists of a dominant interaction, which involves all parameters of the system. However unlike uniform strength interaction, variable strength interaction allows the specification number of disjoint covering array to be incorporated within the dominant interaction. This allows the test engineer to specify higher interaction strength for highly interacting parameters. In the case of the integrated home security system, if it is known that the inputs from intrusion detection (that is, glass break detector, door open sensor and security control panel) are highly likely to produce error, higher interaction strength can be assigned to those inputs accordingly as shown in Figure 2.

Like uniform strength interaction, variable strength interaction can be mathematically represented using variable strength covering array as shown in Equation 2:

$$F = VCA(N, t, C, S)$$

Where, $N$ = the number of test data inside the final test suite, $t$ = the dominant interaction strength, $C$ = value configuration can be represented as follows: $v_1^{p_1}, v_2^{p_2}, ..., v_t^{p_t}$ which indicate that there are $p_1$ parameters with $v_1$ values, $p_2$ parameters with $v_2$ values and so on, $S$ = the multi-set of disjoint covering array with strength larger than $t$ represented by the notation as given in Equation 1.

As for input-output based relations, a set of input-output relationship need to be obtained first before the specification of the covering array. Adopting the integrated home security system given earlier and assuming the system outputs have the following relationship:

(i) “Sound the security bell” depends on smoke detector inputs.
(ii) “Alert the house owner” depends on glass break detector, door open sensor and security control panel inputs.

The input-output relationships, Rel, of the system can be written as:

$$Rel = \{ \{\text{Smoke Detector}\}, \{\text{Glass Break Detector, Door Open Sensor, Security Control Panel}\} \}$$

As the final test suite, $F$ covers all parameter values combinations produces by each input-output relationships. Mathematically, the final test suite can be represented as input-output based relations covering array as shown in Equation 4:

$$F = IOCA(N, C, Rel)$$

Where, $N$ = the number of test data inside the final test suite, $C$ = value configuration can be represented as following: $v_1^{p_1}, v_2^{p_2}, ..., v_t^{p_t}$ which indicate that there are $p_1$ parameters with $v_1$ values, $p_2$ parameters with $v_2$ values and so on, Rel = input-output relation represented as equation shown in Equation 3.
Figure 1. Integrated home security system.

Table 1. Parameters configuration.

| Parameter               | Smoke detector | Glass break detector | Door open sensor | Security control panel |
|-------------------------|----------------|----------------------|------------------|------------------------|
| Values                  | High           | Active               | High             | Activate              |
|                         | Normal         | Inactive             | Low              | Deactivate             |

ITTDG strategy

Generally, ITTDG can be categorized as an OTAT strategy. ITTDG implements the greedy method in deciding which test data will be selected as the final test data. Similar to AETG, ITTDG uses a list called test data candidates in deciding the final test data. However, unlike AETG which generate a set of test data candidate (a fixed number of test data candidates) per iteration, ITTDG, on the other hand, creates a new test data candidate only in a “tie” situation. The ITTDG strategy is as follows.

ITTDG strategy iterates all uncovered tuples produced by every interaction or input-output relationship specify by the user. Iteratively, the strategy will push the visited tuples into a list referred as test data candidates list (Q). The test data candidates list then will be extended by adding one parameter at a time with value that covers the most uncovered tuples. In case of “tie” situation (that is, more than one value cover the most uncovered tuples), the corresponding test data will be duplicated with all the tie values and all duplicated test data will be pushed into Q. Once all test data in Q form a complete test data, the test data which has the highest weight (that is, covers the most uncovered tuples) among the test data candidates in Q will be selected as the final test data. In case of another tie situation (that is, more than one test data candidates have the highest weight), the first found candidate in Q will be selected as the final test data. The selected test data then will be pushed into the final test suite and the tuples covered by the test data are removed from uncovered tuples list. As the size of Q can potentially grow significantly during the parameter extension.
process, the number of test data candidates in Q subjected to a constant integer value (M) to avoid the possible out-of-memory error (that is, when dealing with large number of parameters and values).

As illustration, consider the parameters and values for the integrated home security system given in Table 1 with 2-way interaction as our running example. In order to simplify the discussion, all the values are converted into symbolic value as shown in Table 2.

Initially, the first uncovered interaction will be \(a_0, b_0, X, X\) (where \(X\) indicates do not care value). The ITTDG strategy will create a list of test data candidate (Q) and the first uncovered pair found is pushed into Q as the first test data candidate. Since no candidate in Q forms a complete test data as yet, the strategy will extend every candidate in Q one-parameter-at-a-time. As the first candidate in Q, the first found do not care will be Parameter C. Thus, the strategy will first extend the candidate with Parameter C. From Table 2, two values can be taken to replace do not care (that is, \(c_0\) and \(c_1\)). The strategy then will calculate the number of covered pairs for each value. Here, both \(c_0\) and \(c_1\) covers the maximum number of uncovered pairs, thus the candidate will be extended with \(c_0\) and a duplicate of candidate extended with value \(c_1\) will be pushed into Q.

Now, Q contains two candidates which are \(a_0, b_0, c_0, X\) and \(a_0, b_0, c_1, X\). Since there are incomplete candidates, the ITTDG strategy will iterate through all candidates in Q for its extension. Firstly, candidate \(a_0, b_0, c_0, X\) will be extended with the only Parameter D. As both \(d_0\) and \(d_1\) can cover the maximum number of uncovered pairs, the candidate will be extended with \(d_0\) and a duplicate of candidate extended with value \(d_1\) will be pushed into Q. The same process is repeated for candidate \(a_0, b_0, c_1, X\). Finally, Q

---

**Figure 2.** Variable strength interaction.

**Table 2.** Running example.

| Parameter | Smoke Detector | Glass Break Detector | Door Open Sensor | Security Control Panel |
|-----------|----------------|----------------------|------------------|-----------------------|
| Values    | High           | Active               | High             | Activate             |
|           | Normal         | Inactive             | Low              | Deactivate            |

**Table 2.** Interaction type supported by existing strategies.

| T-way strategies | Density | ParOder | TVG | PICT | AETG | GA | ACA | GA-N | IPO-N | TConfig | Jenny | IPOG | MCM | MIPOG | ITPCH | GTWay | SA | ACS | ReqOrder | Union | Greedy |
|------------------|---------|---------|-----|------|------|----|-----|------|-------|---------|-------|------|-----|------|-------|-------|----|-----|----------|-------|--------|
| Uniform strength | √       | √       | √   | √    | √    | √  | √   | √    | √     | √       | √     | X*   | X*  | X*   | X*    | X*    |    |     |          |       |        |
| Variable strength| √       | √       | √   | √    | X    | X  | X   | X    | X     | √       | X*   | X*   | X*  | X*   | X*    |       |    |     |          |       |        |
| Input-output based relation | √       | √       | √   | X    | X    | X  | X   | X    | X     | X       | X*   | X*   | X*  | X*   | X*    | X     |    |     |          |       |        |

Legend: Supported √, not supported X, no published results X*. 

As illustration, consider the parameters and values for the integrated home security system given in Table 1 with 2-way interaction as our running example. In order to simplify the discussion, all the values are converted into symbolic value as shown in Table 2.

Initially, the first uncovered interaction will be \(a_0, b_0, X, X\) (where \(X\) indicates do not care value). The ITTDG strategy will create a list of test data candidate (Q) and the first uncovered pair found is pushed into Q as the first test data candidate. Since no candidate in Q forms a complete test data as yet, the strategy will extend every candidate in Q one-parameter-at-a-time. As the first candidate in Q, the first found do not care will be Parameter C. Thus, the strategy will first extend the candidate with Parameter C. From Table 2, two values can be taken to replace do not care (that is, \(c_0\) and \(c_1\)). The strategy then will calculate the number of covered pairs for each value. Here, both \(c_0\) and \(c_1\) covers the maximum number of uncovered pairs, thus the candidate will be extended with \(c_0\) and a duplicate of candidate extended with value \(c_1\) will be pushed into Q.

Now, Q contains two candidates which are \(a_0, b_0, c_0, X\) and \(a_0, b_0, c_1, X\). Since there are incomplete candidates, the ITTDG strategy will iterate through all candidates in Q for its extension. Firstly, candidate \(a_0, b_0, c_0, X\) will be extended with the only Parameter D. As both \(d_0\) and \(d_1\) can cover the maximum number of uncovered pairs, the candidate will be extended with \(d_0\) and a duplicate of candidate extended with value \(d_1\) will be pushed into Q. The same process is repeated for candidate \(a_0, b_0, c_1, X\). Finally, Q
Input:
\[ \mu = \{p_1, p_2, \ldots, p_n\} \] // list of uncovered tuples
M= maximum number of test data candidates
Output:
F = final test suite

Begin:
while \( \mu \) is not empty
\[ Q = \text{list of test data candidates} \]
\[ \text{take first tuple} (p_i) \text{ from } \mu \text{ and push into } Q \]
while none entry in Q form a complete test data
\[ \text{for each} \text{ candidate in } Q; \text{ } k = \text{ candidate} \]
\[ \text{find the best parameter-value configuration(s) for } k \]
\[ \text{expand } k \text{ with the first found best value} \]
for each tie values : \( s = \text{tie value} \)
\[ \text{if the size of } Q \text{ less than } M \]
\[ \text{temp = duplicate } k \]
\[ \text{expand temp with } s \text{ and push into } G \]
end if
end for each
end for each
end while
\[ \text{sort } Q \text{ in descending order of number of covered tuples} \]
\[ \text{push the first candidate of } Q \text{ into } F \]
remove pairs in \( \mu \) that cover by the first candidate of \( Q \)
end while
End

Figure 3. ITTDG strategy.

will contains four candidates which are \( \{a_0, b_0, c_0, d_0\} \), \( \{a_0, b_0, c_0, d_1\} \), \( \{a_0, b_0, c_1, d_0\} \) and \( \{a_0, b_0, c_1, d_1\} \). Since no candidate forms an incomplete candidate, the process of parameter extension is stopped and one candidate in \( Q \) will be selected as the final test data. It should be noted that if all four candidates cover the same number of uncovered tuples, the first found candidate will be selected (that is, \( a_0, b_0, c_0, d_0 \)). The tuples corresponding to the selected candidate are removed and the selected candidate is pushed into final test suite list. The whole process is repeated again until all the uncovered tuples are covered.

There were only slight variations as far as ITTDG’s support for variable strength and input output relationship is concerned. In the former case, instead of storing all uniform tuples, the list of uncovered tuples (\( \mu \)) merely stores all tuples produced by dominant interaction and all disjoint covering arrays. In the latter case, \( \mu \) will holds all exhaustive tuples produced by each requirement or input-output relationship. In both cases (that is, variable strength interaction and input-output based relations), all tuples in \( \mu \) is sorted based on the number of “do not care” inside the tuple (that is, tuple with less number of “don’t care” will be listed first). It should be noted that the iterative and selection process remains the same as described earlier. The summary of ITTDG strategy is shown in Figure 3.

RESULTS

The main goals of our evaluation are twofold. Firstly, we wish to demonstrate the capability of ITTDG to integrate all interaction possibilities. Secondly, we benchmark the performance (that is, in terms of generated test data size) of ITTDG against its competing strategies. As different t-way strategy supports different types of interaction, Table 2 summarizes the type of interaction supported by existing t-way strategies.

Our evaluation can be further explained thus. Subsequently, we compared ITTDG with other uniform strength strategies, after which we compared ITTDG with other existing variable strength strategies before demonstrating the performance of ITTDG as an input-output based t-way strategy. It should be noted that in all experiments, the shaded cells represent the most optimum result.

ITTDG as uniform strength t-way strategy

Since there are varying supports as far as interaction strength is concerned (that is, some strategies deal with only small interaction strength), we have decided to perform two different experiments based on the benchmark configurations in Wang et al. (2008) and Klaib (2009). The first experiment adopts lower interaction strength (that is, \( t = 3 \)) while the second experiment used
higher order of interaction (that is, $t \geq 4$). For the first experiment, final test suite is constructed based on the following system configuration with interaction strength set to 3:

- S1: 6 3-valued parameters system, CA ($N, 3, 3^6$)
- S2: 6 4-valued parameters system, CA ($N, 3, 4^6$)
- S3: 6 5-valued parameters system, CA ($N, 3, 5^6$)
- S4: 6 6-valued parameters system, CA ($N, 3, 6^6$)
- S5: 10 6-valued parameters system, CA ($N, 3, 10^6$)
- S6: 7 5-value parameters system, CA ($N, 3, 5^7$)
- S7: 2 5-valued, 2 4-valued and 2 3-valued parameters system, CA ($N, 3, 5^24^23^2$)
- S8: 1 10-valued, 2 6-valued, 3 4-valued, and 1 3-valued parameters system, CA ($N, 3, 10^16^24^33^3$)

The final test suite size is reported in Table 3. It should be noted that the result from other strategies is taken from Wang et al. (2008).

For the second experiment, we adopted the benchmark experiments from Klaib (2009). Four groups have been formed and each group has the following system configuration:

- Group 1: Number of parameter ($P$) and the value ($V$) are constant (that is, 10 and 5 respectively) but the interaction strength ($t$) is varied from 2 to 6.
- Group 2: The interaction strength ($t$) and the value ($V$) are constant (that is, 4 and 5 respectively) but the number of parameter ($P$) is varied from 5 to 15.
- Group 3: The number of parameter ($P$) and the interaction strength ($t$) are constant (that is, 10 and 4 respectively) while the value ($V$) is varied from 2 to 10.
- Group 4: The common traffic and collision avoidance system (TCAS) which consist of 12 multi-valued parameters (that is, 2 10-valued parameter, 1 4-valued parameter, 2 3-valued parameters and 7 2-valued parameters) and interaction strength ($t$) varied from 2 to exhaustive testing (that is, 12-way testing).

The results for Group 1 until 4 are depicted in Tables 4 to 7 respectively. Results for MC-MIPOG are taken from Younis and Zamli (2010) while other strategies taken from Klaib (2009). It should be noted that not all strategies are available for both experiments due to limited published results in literature.

**ITTDG as Variable Strength T-Way Strategy**

To demonstrate ITTG as variable strength t-way strategy and benchmark performance (in terms of generated test data size), we adopted an experiment from Chen et al. (2009) and Cohen et al. (2003). The experiment consists of three system configurations which are:

- (i) 15 3-valued parameters system
- (ii) 3 4-valued parameters, 3 5-valued parameters and 2 6-valued parameters system
- (iii) 20 3-valued parameters and 2 10-value parameters system

The sizes of test data generated are shown in Table 8. Results from other strategies are obtained from Chen et
Table 5. Generated test data size for CA (N, 4, 5).

| P  | IPOG | ITCH | Jenny | TConfig | TVG | GTWay | ITTDG |
|----|------|------|-------|---------|-----|-------|-------|
| 5  | 784  | 625  | 837   | 773     | 849 | 731   | 730   |
| 6  | 1064 | 625  | 1074  | 1092    | 1128| 1027  | 1006  |
| 7  | 1290 | 1750 | 1248  | 1320    | 1384| 1216  | 1213  |
| 8  | 1491 | 1750 | 1424  | 1532    | 1595| 1443  | 1395  |
| 9  | 1677 | 1750 | 1578  | 1724    | 1795| 1579  | 1550  |
| 10 | 1843 | 1750 | 1719  | 1878    | 1971| 1714  | 1697  |
| 11 | 1990 | 1750 | 1839  | 2038    | 2122| 1852  | 1834  |
| 12 | 2132 | 1750 | 1964  | NA      | 2268| 2022  | 1956  |
| 13 | 2254 | NA   | 2072  | NA      | 2398| 2116  | 2078  |
| 14 | 2378 | NA   | 2169  | NA      | 2398| 2116  | 2185  |
| 15 | 2497 | NA   | 2277  | NA      | 2332| 2232  | 2295  |

al. (2009).

ITTDG as input-output based t-way strategy

To demonstrate ITTDG as input-output based t-way strategy, we adopt two different experiments based on benchmark system configurations defined in Wang et al. (2007). The first experiment assumes the system configuration consisting of 10 3-valued parameters. The second experiment assumes a non-uniform system consisting of 3 2-valued parameters, 3 3-valued parameters, 3 4-valued parameters and 1 5-valued parameter. For both experiments, the SUT input parameters are labeled from 0 to 9. Sixty input-output relationships, R, have been determined for both experiments which are R={1, 2, 7, 8}, {0, 1, 2, 9}, {4, 5, 7, 8}, {0, 1, 3, 9}, {0, 3, 8}, {6, 7, 8}, {4, 9}, {1, 3, 4}, {0, 2, 6, 7}, {4, 6}, {2, 3, 4, 8}, {2, 3, 5}, {5, 6}, {0, 6, 8}, {8, 9}, {0, 5}, {1, 3, 5, 9}, {1, 6, 7, 9}, {0, 4}, {0, 2, 3}, {1, 3, 6, 9}, {2, 4, 7, 8}, {0, 2, 6, 9}, {0, 1, 7, 8}, {0, 3, 7, 9}, {3, 4, 7, 8}, {1, 5, 7, 9}, {1, 3, 6, 8}, {1, 2, 5}, {3, 4, 5, 7}, {0, 2, 7, 9}, {1, 2, 3}, {1, 2, 6}, {2, 5, 9}, {3, 6, 7}, {1, 2, 4, 7}, {2, 5, 8}, {0, 1, 6, 7}, {3, 5, 8}, {0, 1, 2, 8}, {2, 3, 9}, {1, 5, 8}, {1, 3, 5, 7}, {0, 1, 2, 7}, {2, 4, 5, 7}, {1, 4, 5}, {0, 1, 7, 9}, {0, 1, 3, 6}, {1, 4, 8}, {3, 5, 7, 9}, {0, 6, 7, 9}, {2, 6, 7, 9}, {2, 6, 8}, {2, 3, 6}, {1, 3, 7, 9}, {2, 3, 7}, {0, 2, 7, 8}, {0, 1, 6, 9}, {1, 3, 7, 8}, {0, 1, 3, 7}.

Both experiments start with |R| =10. Then, 20 relationships in R are used until all the 60 relationships in R are used. The sizes of final test data for both experiments are shown in Tables 9 and 10 respectively.

DISCUSSION

ITTDG as uniform strength t-way strategy

Based on the result shown in Table 3, in terms of generated test data size, ITTDG outperforms TVG, PICT, TConfig and Jenny in almost all system configurations. Compared to Density, ParaOrder and GA-N, ITTDG produces best results in all system configurations except for S8 (that is, where Density and GA-N outperforms ITTDG) and S7 (that is, where ParaOrder outperforms ITTDG).

As for AETG, ITTDG outperform AETG in S2, S7 and S8 while AETG outperform ITTDG in S1, S4, S5 and S6. Lastly, for GA, ACA and IPO-N, these strategies outperform ITTDG in all system configurations except for S2 where ITTDG equals to the three strategies and S7 where ITTDG outperforms IPO-N.

Based on results shown in Tables 4 to 7, one clear observation that can be made is that no single strategy can produce the most optimum result for every system configuration. Overall, ITTDG produces the most optimum result for more than half system configurations (that is, 18 out of 35 configurations). For other system configurations, ITTDG produces competitive test size.

ITTDG as variable strength t-way strategy

Referring to Table 8, we note that Simulated Annealing always produces the most optimum result for all system configurations. ACS comes second in most cases with the exception for VCA(2,4,5,6, (ф)), VCA(2,4,5,6, (ф), CA(3,4,5,6, (ф))) where ITTDG comes the runner up to Simulated Annealing. Putting ACS and Simulated Annealing aside, ITTDG produces the most optimal results in most cases compared to the rest (except for VCA(2,3,15, (ф), CA(3,3,15, (ф))) where ParaOrder produces the most optimal results, and VCA(2,3,15, (ф), CA(3,3,15, (ф))), VCA(2,3,20,10, (ф), CA(3,3,20,10, (ф)))) where Density produces the most optimal results). Overall, PICT in most configurations produces the worst result.

ITTDG as input-output based t-way strategy

From result of the first experiment (Table 9), ITTDG
Table 6. Generated test data size for CA(N, 4, V^10).

| V | IPOG | ITCH | Jenny | TConfig | TVG | GTWay | ITTDG |
|---|------|------|-------|---------|-----|-------|-------|
| 2 | 46   | 58   | 39    | 45      | 40  | 46    | 45    |
| 3 | 229  | 336  | 221   | 235     | 228 | 224   | 218   |
| 4 | 649  | 704  | 703   | 718     | 782 | 621   | 687   |
| 5 | 1843 | 1750 | 1719  | 1878    | 1971| 1714  | 1697  |
| 6 | 3808 | NA   | 3519  | NA      | 4159| 3514  | 3491  |
| 7 | 7061 | NA   | 6482  | NA      | 7854| 6459  | 6399  |
| 8 | 11993| NA   | 11021 | NA      | NA  | 10850 | 8258  |
| 9 | 19098| NA   | 17527 | NA      | NA  | 17272 | 17179 |
| 10| 28985| NA   | 26624 | NA      | NA  | 26121 | 25917 |

Table 7. Generated test data size for TCAS module, CA (N, t, 10^2413^22^2).

| t | MC-MIPOG | IPOG | ITCH | Jenny | TConfig | TVG | GTWay | ITTDG |
|---|-----------|------|------|-------|---------|-----|-------|-------|
| 2 | 100       | 100  | 120  | 108   | 108     | 101 | 100   | 100   |
| 3 | 400       | 400  | 2388 | 412   | 472     | 434 | 402   | 411   |
| 4 | 1265      | 1361 | 1484 | 1536   | 1476    | 1599| 1429  | 1309  |
| 5 | 4196      | 4219 | NA   | 4580   | NA      | 4773| 4286  | 4271  |
| 6 | 10851     | 10919| NA   | 11625  | NA      | NA  | 11727 | 10529 |
| 7 | 26061     | NA   | NA   | 27630  | NA      | NA  | 27119 | 28759 |
| 8 | 56742     | NA   | NA   | 58865  | NA      | NA  | 58584 | 59473 |
| 9 | 120361    | NA   | NA   | NA     | NA      | NA  | 114411| 119042|
| 10| 201601    | NA   | NA   | NA     | NA      | NA  | 201728| 206000|
| 11| 230400    | NA   | NA   | NA     | NA      | NA  | 230400| 230400|
| 12| 460800    | NA   | NA   | NA     | NA      | NA  | 460800| 460800|

Table 8. Generated test data size for different variable strength t-way strategies.

| (C) | N | SA | Density | Para Order | PICT | TVG | ACS | ITTDG |
|-----|---|----|---------|------------|------|-----|-----|-------|
| VCA(N, 2,3^{15},(C)) |   | 16 | 21       | 33         | 35   | 22  | 19  | 20    |
| φ   |   | 27 | 27       | 27         | 81   | 27  | 27  | 27    |
| CA(3,3) |   | 27 | 28       | 33         | 729  | 30  | 27  | 27    |
| CA(3,3) |   | 27 | 28       | 33         | 785  | 30  | 27  | 27    |
| CA(3,3) |   | 27 | 32       | 27         | 105  | 35  | 27  | 28    |
| CA(3,3) |   | 33 | 40       | 45         | 121  | 41  | 38  | 40    |
| CA(3,3),CA(3,3),CA(3,3) | | 34 | 46       | 44         | 1376 | 53  | 40  | 43    |
| CA(3,3) |   | 34 | 46       | 49         | 146  | 48  | 45  | 46    |
| CA(3,3) |   | 41 | 53       | 54         | 154  | 54  | 48  | 51    |
| CA(3,3) |   | 50 | 60       | 62         | 177  | 62  | 57  | 60    |
| CA(3,3^{15}) |   | 67 | 70       | 82         | 83   | 81  | 76  | 79    |

| VCA(N, 2,4^{5}3^{6}2^{2},(C)) |   | 36 | 41       | 49         | 43   | 44  | 41  | 40    |
| φ   |   | 64 | 64       | 64         | 384  | 67  | 64  | 64    |
| CA(3,4) |   | 64 | 64       | 64         | 384  | 67  | 64  | 64    |
Table 8. Contd.

| CA(3, 4\textsuperscript{3}5\textsuperscript{3}) | 100 | 131 | 141 | 781 | 132 | 104 | 127 |
| CA(3, 5\textsuperscript{3}) | 125 | 125 | 126 | 750 | 125 | 125 | 125 |
| CA(3, 4\textsuperscript{3})CA(3, 5\textsuperscript{3}) | 125 | 125 | 129 | 8000 | 125 | 125 | 125 |
| CA(3, 4\textsuperscript{3}5\textsuperscript{3}6\textsuperscript{3}) | 171 | 207 | 247 | 1266 | 237 | 201 | 205 |
| CA(3, 5\textsuperscript{3}6\textsuperscript{3}) | 180 | 180 | 180 | 900 | 180 | 180 | 180 |
| CA(3, 4\textsuperscript{3}5\textsuperscript{3}6\textsuperscript{3}) | 214 | 256 | 307 | 261 | 302 | 255 | 239 |

\textbf{VCA(N, 2, 3\textsuperscript{10} \times 1\textsuperscript{2} (C))}

| \(\phi\) | 100 | 100 | 100 | 100 | 101 | 100 | 100 |
| CA(3, 3\textsuperscript{20}) | 100 | 100 | 103 | 940 | 103 | 101 | 102 |
| CA(3, 3\textsuperscript{20}10\textsuperscript{2}) | 304 | 401 | 442 | 423 | 423 | 396 | 416 |

Table 9. Comparison of size generated by different strategies for IOR (N, 3\textsuperscript{10}, R).

| | Density | TVG | ReqOrder | ParaOrder | Union | Greedy | ITTDG |
|---|---|---|---|---|---|---|---|
| 10 | 86 | 86 | 153 | 105 | 503 | 104 | 81 |
| 20 | 95 | 105 | 148 | 103 | 858 | 110 | 94 |
| 30 | 116 | 125 | 151 | 117 | 1599 | 122 | 114 |
| 40 | 126 | 135 | 160 | 120 | 2057 | 134 | 122 |
| 50 | 135 | 139 | 169 | 148 | 2635 | 138 | 131 |
| 60 | 144 | 150 | 176 | 142 | 3257 | 143 | 141 |

Table 10. Comparison of size generated by different strategies for IOR(N, 3\textsuperscript{10}, R).

| | Density | TVG | ReqOrder | ParaOrder | Union | Greedy | ITTDG |
|---|---|---|---|---|---|---|---|
| 10 | 144 | 144 | 154 | 144 | 505 | 137 | 144 |
| 20 | 160 | 161 | 187 | 161 | 929 | 158 | 160 |
| 30 | 165 | 179 | 207 | 179 | 1861 | 181 | 169 |
| 40 | 165 | 181 | 203 | 183 | 2244 | 183 | 173 |
| 50 | 182 | 194 | 251 | 200 | 2820 | 198 | 183 |
| 60 | 197 | 209 | 250 | 204 | 3587 | 207 | 199 |

outperforms most other strategies except when \(|R| = 40\). In the case of \(|R| = 40\), ParaOrder produces the most optimal test size (that is, 120 test data). Overall, UNION produces the worst result.

In the second experiment (Table 10), Greedy produces the most optimum test data for \(|R| = 10\) and \(|R| = 20\) while Density produces the most optimum result for the rest of configurations. Although not producing optimal test size, ITTDG produces competitive test size (that is, second to the most optimum strategy) for all configurations. For \(|R| = 10\) and \(|R| = 20\), ITTDG produces the same number of test data as Density. For the other configurations, ITTDG comes second to Density. Similar to the first experiment, UNION produces the worst result for all configurations.

CONCLUSIONS AND RECOMMENDATION

In this paper, we have described the development of a new t-way strategy, called ITTDG, which provides seamless integration of all interaction possibilities. Empirical evidence demonstrates a sound performance as far as the generated test size is concerned especially involving uniform number of parameter values. As a scope for future work, we are now investigating sequence based interaction where the sequence of input (or the order of input arrivals) is first checked for interaction considerations (Zamli et al., 2011b). Moreover, we also investigate a statistical approach based on Design of Experiments in order to automatically determine the
required interaction strength as well as input-output relationship of a system of interest so as to further simplify the test generation process.

ACKNOWLEDGEMENT

This research is partially funded by the generous fundamental grants – “investigating t-way test data reduction strategy using particle swarm optimization technique” from Ministry of Higher Education (MOHE), the USM research university grants – “development of variable strength interaction testing strategy for t-way test data generation”, and the USM short term grant – “development of a pairwise test data generation tool with seeding and constraints support”.

REFERENCES

Arshem J (2010). TVG Available: http://sourceforge.net/projects/tvg [Last Accessed: December 2010].
Bryce RC, Colbourn CJ (2007). The Density Algorithm for Pairwise Interaction Testing. Softw. Test. Verif. Reliab., 17(3): 159-182.
Bryce RC, Colbourn CJ (2009). A Density-Based Greedy Algorithm For Higher Strength Covering Arrays. Softw. Test. Verif. Reliab., 19(1): 37-53.
Chen X, Gu Q, Li A, Chen D (2009). Variable Strength Interaction Testing With An Ant Colony System Approach. Proceedings of 16th Asia-Pacific Software Engineering Conference, Penang, Malaysia: pp. 160-167.
Cohen DM, Dalal SR, Fredman ML, Patton GC (1997). The AETG System: An Approach to Testing Based On Combinatorial Design. IEEE Trans. Softw. Eng., 23(7): 437-444.
Cohen DM, Dalal SR, Kajla A, Patton GC (1994). The Automatic Efficient Test Generator (AETG) System. Proceedings of the 5th International Symposium on Software Reliability Engineering, Monterey, CA, USA, pp. 303-309.
Cohen MB (2004). Designing Test Suites For Software Interaction Testing. PhD Thesis, University of Auckland.
Cohen MB, Gibbons PB, Mugridge WB, Colbourn CJ, Collofello JS (2003). Variable Strength Interaction Testing Of Components. Proceedings of 27th Annual International Computer Software and Applications Conference, Dallas, USA, pp. 413-418.
Czerwonka J (2006). Pairwise Testing In Real World. Proceedings of 24th Pacific Northwest Software Quality Conference, Portland, Oregon, USA, pp. 419-430.
Czerwonka J (2010). PICT Tool [Online]. Available: http://www.pairwise.org/tools.asp [Last Accessed: December 2010].
Hartman A, Klinger T, Raski L (2010). IBM Intelligent Test Case Handler [Online]. Available: http://www.alphaworks.ibm.com/tech/whitch [Last Accessed: February 2009].
Jenkins B (2010). Jenny Test Tool [Online]. Available: http://www.burtleburtle.net/bob/math/jenny.html [Last Accessed: December 2010].
Klab MFJ (2009). Development Of An Automated Test Data Generation And Execution Strategy Using Combinatorial Approach. PhD. Thesis, Universiti Sains Malaysia.
Lei Y, Kacker R, Kuhn DR, Okun V, Lawrence J (2007). IPOG: A General Strategy For T-Way Software Testing. Proceedings of the 14th Annual IEEE International Conference and Workshops on The Engineering of Computer-Based Systems, Tucson, AZ. pp. 549-556.
Lei Y, Tai KC (1998). In-Parameter-Order: A Test Generation Strategy For Pairwise Testing. Proceedings of 3rd IEEE International Conference on High Assurance Systems Engineering Symposium, Washington DC, USA, pp. 254-261.
Nie C, Xu B, Shi L, Dong G (2005). Automatic Test Generation For N-Way Combinatorial Testing. In: Reussner R, Mayer J, Stafford JA, Overhage S, Becker S , Schroeder PJ (eds.) Quality of Software Architectures and Software Quality. Berlin / Heidelberg: Springer.
Schroeder PJ (2001). Black-Box Test Reduction Using Input-Output Analysis. PhD Thesis, Illinois Institute of Technology.
Schroeder PJ, Faherty P, Korel B (2002). Generating Expected Results For Automated Black-Box Testing. Proceedings of 17th IEEE International Conference on Automated Software Engineering (ASE’02), Edinburgh, Scotland, UK. pp. 139-148.
Sheh Z, Nie C, Xu B (2007). Generating Combinatorial Test Suite For Interaction Relationship. Proceedings of 4th International Workshop On Software Quality Assurance (SOQUA2007), Dubrovnik, Croatia. pp. 55-61.
Williams AW (2010). TConfig [Online]. Available: http://www.site.uottawa.ca/~awilliam/ [Last Accessed: December 2010].

Younis MI (2010). MIPOG: A Parallel T-Way Minimization Strategy For Combinatorial Testing. PhD. Thesis, Universiti Sains Malaysia.
Younis MI, Zamli KZ (2010). MC-MIPOG: A Parallel T-Way Test Generation Strategy For Multicore Systems. ETRI J., 32(1): 73-83.
Zami KZ, Klab MFJ, Younis MI, Isam NAM, Abdullah R (2011a). Design And Implementation Of A T-Way Test Data Generation Strategy With Automated Execution Tool Support. Inf. Sci., 181(9): 1741-1758.
Zami KZ, Othman RR, Zabil MHM (2011b). On Sequence Based Interaction Testing. IEEE Symposium on Computers & Informatics 2011 (ISCI 2011), Kuala Lumpur, Malaysia. pp. 645-650.
Zami KZ, Younis MI, Suhaily SAC, Soh ZHC (2008). Software Testing. First ed. KL Malaysia: Open University Malaysia.
Zekau L (2006). Mixed Covering Arrays On Graphs And Tabu Search Algorithms. Msc Thesis, University of Ottawa.