Maejo Int. J. Sci. Technol. **2012**, *6*(03), 415-429

Maejo International Journal of Science and Technology

ISSN 1905-7873
Available online at www.mijst.mju.ac.th

Full Paper

General variable strength t-way strategy supporting flexible interactions

Rozmie Razif Othman [1], Kamal Zuhairi Zamli [2,*] and Lukito Edi Nugroho [3]

[1] School of Computer and Communication, Universiti Malaysia Perlis (UniMAP), PO Box 77, d/a Pejabat Pos Besar, 01007 Kangar, Perlis, Malaysia

[2] Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, Lebuhraya Tun Razak, 26300 Kuantan, Pahang, Malaysia

[3] Electrical Engineering and Information Technology Department, Faculty of Engineering, Universitas Gadjah Mada, Yogyakarta, Indonesia

* Corresponding author, e-mail: kamalz@ump.edu.my

Received: 10 February 2012 / Accepted: 6 November 2012 / Published: 7 November 2012

Abstract: To ensure conformance and establish quality, software testing is an integral part in software engineering lifecycle. However, because of resource and time-to-market constraints, testing all exhaustive possibilities is impossible in nearly all practical testing problems. Considering the aforementioned constraints, much research now focuses on a sampling technique based on interaction testing (termed as t-way strategy). Although helpful, most t-way strategies (e.g. AETG, In-Parameter-Order General (IPOG), and GTWay) assume that all parameters have uniform interaction. In reality, the interaction among parameters is rarely uniform. Some parameters may not even interact, wasting the testing efforts. As a result, a number of newly developed t-way strategies that consider variable-strength interaction based on input–output relationships have been developed, e.g. Union, ParaOrder, and Density. Although useful, these strategies often suffer from lack of optimality in terms of the generated test size. Furthermore, no single strategy is dominant because the optimal generation of t-way interaction test suite is considered an Nondeterministic Polynomial (NP) hard problem. Motivated by the above-mentioned challenges, this paper proposes and implements a new strategy, called General Variable Strength (GVS). GVS has been demonstrated, in some cases, to produce better results than other competing strategies.

Keywords: interaction testing, t-way test generation, variable strength interaction, software testing

---

INTRODUCTION

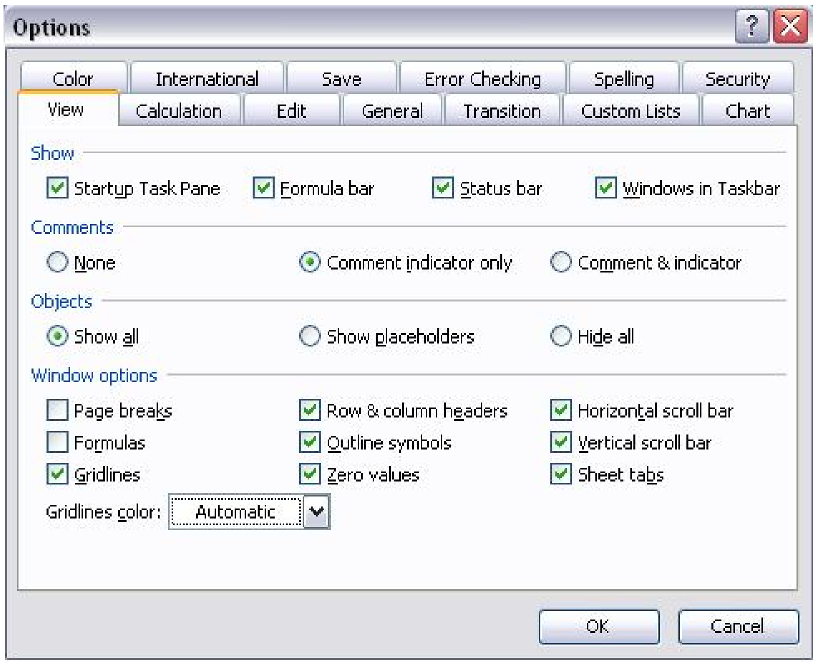
Nowadays, we are increasingly dependent on software to facilitate our daily chores, from mobile phone applications to sophisticated airplane control system. To ensure quality and reliability, we have to consider many combinations of possible input parameters, hardware/software environments, and system conditions, tested and verified for conformance. Because of resource and time-to-market constraints, testing all exhaustive possibilities is practically impossible. As a result, many t-way strategies (where *t* identifies the interaction strength) have been proposed in the literature for the past 20 years. All strategies help in searching, as well as minimising, the final test cases, i.e. to form a complete suite that deals with all required interactions.

Although helpful, most existing t-way strategies, e.g. GTWay [1, 2], In-Parameter-Order (IPO) General (IPOG) [3], IBM's Test Case Handler (ITCH) [4], and Jenny [5], assume that all parameters have uniform interaction. In reality, interaction among parameters is rarely uniform. Actually, some parameters may not even interact, wasting the testing efforts. To address the aforementioned issues, several newly developed t-way strategies have been developed, which consider variable-strength interaction based on input-output relationships. Schroeder proposed two strategies, called Union [6] and Greedy [7]. Meanwhile, Wang et al. proposed three strategies, namely ReqOrder [8], ParaOrder [8] and Density [9]. Finally, a public-domain tool available from SourceForge, called Test Vector Generator (TVG) [10], also supports variable-strength interaction based on the input-output relationships. Although useful, these newly proposed strategies suffer from lack of optimality (i.e. in terms of test size). Furthermore, no single strategy is dominant because the optimal generation of t-way interaction test suite is considered an Nondeterministic Polynomial (NP) hard problem [11, 12]. Motivated by the aforementioned challenges, this paper discusses the design and evaluation of a new strategy, called General Variable Strength (GVS).

Problem Definition Model

Exhaustive testing is impossible because the number of test cases can be exorbitantly large, even for simple software and hardware products. Let us consider a hardware product with 20 on/off switches. Testing all possible combinations would require 2^{20} = 1,048,576 test cases. If the time required for one test case is 5 min, then the test would take nearly 10 years to complete.

The same argument is applicable in any software system. As an illustration, let us consider the option dialog in the Microsoft Excel software (Figure 1). Even if only the View tab option is considered, 20 possible configurations have to be tested. Except for the gridline colour that takes 56 possible values, each configuration can take two values, namely checked or unchecked. Here, we must evaluate $2^{20} \times 56$ or 58,720,256 combinations of test cases. Using the same calculation assumption, a complete test of the View tab option would require nearly 559 years.



Show

[✓] Startup Task Pane [✓] Formula bar [✓] Status bar [✓] Windows in Taskbar

Comments

○ None [● Comment indicator only] ○ Comment & indicator

Objects

[● Show all] ○ Show placeholders ○ Hide all

Window options

[ ] Page breaks	[✓] Row & column headers	[✓] Horizontal scroll bar
[ ] Formulas	[✓] Outline symbols	[✓] Vertical scroll bar
[✓] Gridlines	[✓] Zero values	[✓] Sheet tabs

Gridlines color: Automatic

Figure 1. Microsoft excel view tab options

The above-mentioned examples highlight the common combinatorial explosion problem in software testing. Given limited time and resources, the main research questions are as follows:

- What is the minimum number of (sample) tests to be considered?
- How can one decide (i.e. the strategy) which combination of values to choose over the large combinatorial data sets?

Background

Over the years, many sampling-based testing strategies, e.g. equivalence partitioning, cause and effect analysis, decision table, and boundary value analysis, have been developed [13]. Although helpful, many strategies were not sufficiently effective in dealing with the faults due to interaction. Thus, t-way strategies have been proposed to address this issue. Briefly, the t-way strategies offer four possible interactions to generate the test suite: uniform strength, variable strength, input-output based relationship, and mixed interactions. Figure 2 shows the features of the possibilities of each interaction using a software system with five parameter inputs (P_0, P_1, P_2, P_3 and P_4) and three outputs (f_0, f_1 and f_2).

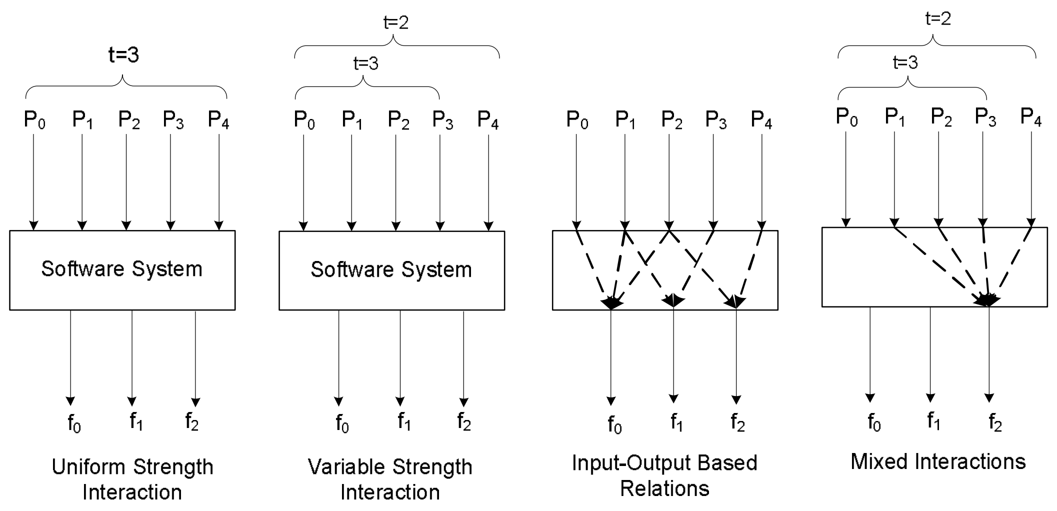

t=3
P0 P1 P2 P3 P4
Software System
f0 f1 f2
Uniform Strength Interaction
t=2
t=3
Software System
Variable Strength Interaction
Input-Output Based Relations
Mixed Interactions

Figure 2. Interaction possibilities within the software system

Uniform-strength interaction is the basis of interaction testing, where all input parameters are assumed to be uniformly interacting (i.e. with constant interaction strength (*t*) throughout). To test all interacting parameters, the test suite must cover all the t-way combinations at least once. In this manner, all possible uniform strength interactions can be tested and, hence verified for correctness. Mathematically, the uniform-strength test suite can be represented using the covering array notation as

$$F = CA(N,t,C) \tag{1}$$

where: *N* is the final test suite size,

t is the interaction strength,

C is the value configuration, which can be represented as $v_0^{p_0}, v_1^{p_1}, \ldots, v_n^{p_n}$, indicating p_0 parameters with v_0 values and p_1 parameters with v_1 values, and so on.

In contrast to the uniform-strength interaction counterpart, the variable-strength interaction considers more than one interaction strength for the test suite generation. Here, a particular subset of input parameters can have higher interaction dependence than the other parameters, which indicates that failure due to the interaction of that subset may have more significant effects on the overall system. Thus, stronger interaction strength can be assigned accordingly. Using the covering array notation, the variable-strength test suite *F* can be represented as

$$F = VCA(N,t,C,S) \tag{2}$$

where: *N* is the final size of the test suite,

t is the dominant interaction strength,

C is the value configuration, which can be represented as $v_0^{p_0}, v_1^{p_1}, \ldots, v_n^{p_n}$,

S is the multi-set of the disjoint covering array with strength larger than *t*, as given in Eq. (1).

Although the uniform- and variable-strength interactions assume that all input parameters interact with one another, the input-output-based relationships use the knowledge of the input-output relationships for the test suite generation. Using similar covering array notations as that in the uniform- and variable-strength interactions, the input-output-based relationship F can be represented as

$$F = IOR(N, C, Rel) \quad (3)$$

where: N is the final size of the test suite,

C is the value configuration, which can be represented as $v_0^{p_0}, v_1^{p_1}, \dots, v_n^{p_n}$,

Rel is the input-output relationship definition set based on the combination index $p_0 \dots p_n$,

$|Rel|$ is the number of input-output relationship definition sets

By combining the uniform strength, variable strength, and input-output-based relationships, the mixed interaction represents the amalgam of knowledge on the input and output behaviours of the system under test. Mathematically, the mixed interaction adopts the same covering array notation as the input-output-based relationships. The uniform and variable strengths can also be represented in the same manner. Table 1 summarises the input-output conversion for the earlier example shown in Figure 2.

Table 1. Input-output conversions for Figure 2

F= IOR (N,C, Rel)	**Input–output interaction representations**
Uniform-strength interaction ($\|Rel\|$ = 10) Rel = {{0,1,2}, {0,1,3}, {0,1,4}, {0,2,3}, {0,2,4}, {0,3,4}, {1,2,3}, {1,2,4}, {1,3,4}, {2,3,4}}	$\{P_0,P_1,P_2\}$, $\{P_0,P_1,P_3\}$, $\{P_0,P_1,P_4\}$, $\{P_0,P_2,P_3\}$, $\{P_0,P_2,P_4\}$, $\{P_0,P_3,P_4\}$, $\{P_1,P_2,P_3\}$, $\{P_1,P_2,P_4\}$, $\{P_1,P_3,P_4\}$, $\{P_2,P_3,P_4\}$
Variable-strength interaction ($\|Rel\|$ = 10) Rel = {{0,1,2}, {0,1}, {0,2}, {0,3}, {0,4}, {1,2}, {1,3}, {1,4}, {2,3}, {2,4}}	$\{P_0,P_1,P_2\}$, $\{P_0,P_1,P_3\}$, $\{P_0,P_2,P_3\}$, $\{P_1,P_2,P_3\}$, $\{P_0,P_1\}$, $\{P_0,P_2\}$, $\{P_0,P_3\}$, $\{P_0,P_4\}$, $\{P_1,P_2\}$, $\{P_1,P_3\}$, $\{P_1,P_4\}$, $\{P_2,P_3\}$, $\{P_2,P_4\}$
Input-output-based relationships ($\|Rel\|$ = 3) Rel = {{0,1,2}, {1,3}, {2,4}}	$\{P_0,P_1,P_2\}$, $\{P_1,P_3\}$, $\{P_2,P_4\}$
Mixed interactions ($\|Rel\|$ = 11) Rel = {{1,2,3,4}, {0,1,2}, {0,1}, {0,2}, {0,3}, {0,4}, {1,2}, {1,3}, {1,4}, {2,3}, {2,4}}	$\{P_1,P_2,P_3,P_4\}$, $\{P_0,P_1,P_2\}$, $\{P_0,P_1,P_3\}$, $\{P_0,P_2,P_3\}$, $\{P_1,P_2,P_3\}$, $\{P_0,P_1\}$, $\{P_0,P_2\}$, $\{P_0,P_3\}$, $\{P_0,P_4\}$, $\{P_1,P_2\}$, $\{P_1,P_3\}$, $\{P_1,P_4\}$,$\{P_2,P_3\}$, $\{P_2,P_4\}$

The most general representation of any form of parameter interactions is the input-output-based relationship. Thus, GVS is developed as a general strategy to integrate seamlessly and support all interaction possibilities.

Related Work

Many t-way strategies have been proposed for the past 20 years. In a nutshell, existing t-way strategies can be categorised either as a one-parameter-at-a-time (OPAT) or a one-test-at-a-time (OTAT) strategy.

The OPAT strategy initially generates an exhaustive test for a few selected parameters. Then it iteratively adds OPAT until all parameters are covered, i.e. horizontal extension. Upon completion, new test cases may be added to ensure complete interaction coverage, i.e. vertical extension.

In-Parameter-Order (IPO) is the precursor of the OPAT strategy developed by Lei and Tai [14]. Because IPO is limited to pairwise interaction [15], IPOG was developed as the general version of IPO to support higher order interactions. As far as interaction support is concerned, IPOG addresses the uniform-strength, as well as the variable-strength, interaction. No support is provided for the input-output-based relationships.

A number of IPOG variants exist in the literature, including TConfig [16], ParaOrder, and ReqOrder [8]. Similar to IPOG, TConfig adopts variation in the horizontal and vertical extensions as part of its algorithm. In contrast to IPOG, TConfig only addresses the uniform-strength interaction. ParaOrder and ReqOrder differ from their predecessor in terms of how the initial test case is generated [8]. In IPOG, the initial test case is generated in the defined order of parameters found, whereas in ParaOrder, the initial test case is generated based on the first defined input-output relationship. In ReqOrder, the selection of the initial test case does not necessarily follow the first defined input-output relationship. Additionally, in contrast to IPOG, ParaOrder and ReqOrder address uniform strength, variable strength, and input-output-based relationships.

In contrast to the OPAT strategy, the OTAT strategy greedily generates one complete test case into the final test suite per iteration until all tuples are covered. Based on the main approaches of each strategy, the OTAT strategy can be further characterised into three categories: artificial intelligence (AI)-based, iterative-based, and heuristic-based strategies.

The AI-based OTAT strategy adopts an AI technique. Simulated annealing (SA) [17], ant colony-based strategy (ACS) [18], and variable-strength particle swarm optimisation (VS-PSTG) [19] are some of the AI-based techniques adopted in generating interaction test suite.

Concerning SA, the strategy is based on the annealing process, i.e. maximising the crystal size of the material via heating and slow cooling. Heating excites the atom to move from its initial position to avoid a local minimum of internal energy, whereas slow cooling allows the atom to settle for lower internal energy configurations for better crystal size. Analogous to the physical process, the SA strategy starts with a randomly generated test suite, i.e. initial state, and applies a series of transformations according to a probability equation, which depends heavily on parameter T (the controlling temperature of the simulation to simulate heating and cooling).

In ACS, the candidate test cases are searched by colonies of ants for some possible paths. The path qualities are evaluated in terms of the pheromones which signify convergence. The optimum paths correspond to the best test candidate included in the final test suite. For the VS-PSTG, the search process is inspired by the behaviour of flocks of birds. Internally, the strategy iteratively combines local and global searches to find the best test cases that cover the given interaction tuples. We should note that SA, ACS and VS-PSTG address uniform- and variable-strength interactions.

Regarded as the most popular approach, the iterative-based OTAT strategy often performs systematic iterative search to generate the final test suite. GTWay [1], ITCH [20], Jenny [5], TVG [10], PICT [21], Union [6, 22] and Greedy [7] are few examples of the iterative-based OTAT strategy.

As far as implementation is concerned, GTWay starts by generating all the required interaction tuples using its tuple generation algorithm. Then the strategy iterates all tuples and tries to merge any ‘combinable’ tuples based on its backtracking algorithm. Although it adopts similar merger algorithm as GTWay does, ITCH relies heavily on its exhaustive search algorithm to find the best combinable tuples. Both GTWay and ITCH address uniform-strength interaction.

With regard to Jenny, PICT and TVG, their implementations can be downloaded from the developer's website. Jenny starts by constructing a test suite that covers one-way interaction first. The strategy then extends the test suite to cover two-way interaction, and the process is repeated until the test suite covers the required t-way interactions. In contrast to Jenny, PICT generates the test suite by selecting one uncovered tuple and iteratively fills the 'don't care' parameters (parameters that do not contribute to the current tuple of interest) with the best found value to cover the most uncovered tuples. TVG adopts three algorithms for test suite generation, namely T-Reduced, Plus-One, or Random Set algorithm. Because of limited literature, how each algorithm (T-Reduced, Plus-One, or Random Sets) works is yet unclear. Based on our experiences with TVG, T-Reduced often produces the most optimal test suite compared with Plus-One and Random Sets. Relative to interaction support, Jenny addresses uniform-strength interaction whereas PICT and TVG support uniform strength, variable strength and input-output based relationships.

Union [6, 22] and Greedy [7] are two related iterative-based OTAT strategies. In the case of Union, partial test cases are first generated based on the defined input-output relationships. Then random values are assigned to all parameters that do not contribute to the defined input-output relationships to complete the test cases. Upon completion, union operations are performed for all test cases to remove repetition. Based on the Union strategy, the Greedy strategy also works in the same manner. In contrast to the Union strategy, however, the Greedy strategy completes the partial test cases greedily to cover the most uncovered interactions. In this manner, the Greedy strategy often generates a more optimal test size than Union does. Both Union and Greedy address uniform-strength interaction, variable-strength interaction, and input-output-based relationships.

The last category of the OTAT strategy is the heuristic-based strategy. The heuristic-based OTAT strategy typically uses some form of heuristic models to decide on the test case selection. Bryce's Density strategy [23, 24] and Wang's Density strategy [9] are examples of heuristic-based OTAT strategy. Bryce's Density strategy pioneers the use of density calculation model [23, 24] in constructing the test suite. For each test case, the 'parameter density' of every unassigned parameter is calculated and the parameters with the highest value are selected. Then, the 'value density' that corresponds to the selected parameters is calculated, and the highest value density is fitted in accordingly. This process is repeated OTAT until all parameters have valid value assignments and the complete test suite is formed. In contrast to Bryce's strategy, which addresses only the uniform-strength interaction, the Wang's Density strategy extends its support to the input-output-based relationships. To enable the support, Wang introduced 'local density' and 'global density' calculations. During the test generation process, the Wang's strategy chooses one input-output relationship with the highest local density value. Then for each exhaustive combination of the selected input-output relationship, the strategy selects the combination with the highest global density value to fit into the current test case. The process continues until the complete test suite is formed.

METHODS

The GVS search algorithm is inspired by earlier work in GTWay by Klaib and Zamli et al. [1, 2]. The GVS search algorithm works as follows: in contrast to the GTWay's search algorithm, which generates all tuples before iterating them, GVS generates one tuple at a time before the start of the iteration to minimise memory requirements. After requesting one tuple from the tuple generator, the search algorithm selects one 'don't care' parameter at a time (indicated by *X*) to establish a value that can produce another uncovered tuples.

Let us consider four 2-valued parameter systems, and the interaction strength required is three. The first uncovered tuples generated are "0", "0", "0" and *X*. Here, the first 'don't care' parameter is parameter X_3, which consists of two values (i.e. "0" and "1"). The search algorithm attempts to fit "0" first into the incomplete test case and check whether uncovered tuples produced by "0" exist. In this case, because the algorithm has just started its search, no tuple is covered yet, i.e. the covered tuple list is empty. Thus, "0" is selected; following the selection of "0" for X_3, {"0", "0", "0", *X*} is produced as one uncovered tuple (for the first input-output relationship), and {"0", *X*, *X*, "0"} is produced as another uncovered tuple (for the second input-output relationship). Then the same process is repeated for the other parameter selections in case more 'don't care' parameters are found. After completing all parameters, the search algorithm checks the generated test case and determines whether the generated test case has covered the most uncovered tuples. If it does, the generated test case is selected in the final test suite list, and the covered tuples are added to the covered tuple list. This process is repeated until all tuples are covered by test cases in the final test suite. The GVS search algorithm is further summarised in Figure 3.

```
Output: Final Test Suite, T

Begin:
Initialise k as the total tuple involve
Initialise Ct as covered tuple list

While (no of tuples in Ct != k)
  P = get next tuple
  if(P not in Ct)
    for every don't care in P
      select value that can produce uncovered tuples
    if(P has the most uncovered tuples)
      store the tuples covered by P in Ct
      store P in T
    if(P still consist don't care)
      replace don't care with the first value of the parameter
    store the tuples covered by P in Ct
    store P in T
End
```

Figure 3. GVS search algorithm

RESULTS AND DISCUSSION

The GVS evaluation was divided into three parts. In the first part, the performance of GVS (in terms of generated test suite size) against the other competing uniform-strength strategies was compared based on the experimental results [2]. In the second part, the performance of GVS against existing variable-strength strategies was evaluated based on the experimental results [18, 19]. Finally, the experimental results obtained by Wang [8, 9] were adopted to benchmark GVS against the existing input-output-based strategies. In all parts, the generated test suite size, rather than the execution time for the test generation, was compared because access to all the strategy implementations was not available. Comparing the execution time of each strategy is impossible, even from published results, which provided different running environments. Attempting to do so is counterproductive because the execution time is directly affected by the computer hardware performance, operating system and data structure, as well as language implementation.

For each part, the best test suite size for GVS on a single run was reported. GVS is a deterministic strategy, viz. multiple runs always produce identical test suite. Hence, no change occurred as far as the test size is concerned. The running environment consisted of a desktop PC with Windows XP, 2.8 GHz Core 2 Duo CPU, and 1 GB RAM. The GVS strategy was coded and implemented in Java (JDK 1.6). The results are presented in Tables 2-8. The darkened cells indicate the best obtained result for the configuration of interests. Cells marked as NA indicate that the results are not available in the publications.

GVS as Uniform Strength t-Way Strategy

Based on the benchmarking experiments [2], four groups of experiments were conducted and each group has the following system configurations:

i. Group 1: The number of parameters (P) and the value (V) were constant (10 and 5 respectively), but the interaction strength (*t*) varied from two to six.
ii. Group 2: The interaction strength (*t*) and the value (V) were constant (4 and 5 respectively), but the number of parameters (P) varied from 5 to 15.
iii. Group 3: The number of parameters (P) and the interaction strength (*t*) were constant (10 and 4 respectively), whereas the value (V) varied from 2 to 10.
iv. Group 4: The common traffic and collision avoidance system (TCAS), which consisted of 12 multi-valued parameters (two 10-valued parameters, one 4-valued parameter, two 3-valued parameters, and seven 2-valued parameters) and interaction strength (*t*) varied from 2 to the exhaustive testing (i.e. 12-way testing).

The results for Groups 1-4 are shown in Tables 2-5 respectively. The results for the other strategies are obtained from Zamli et al. [2].

Table 2. Generated Test Size For $CA(N, t, 5^{10})$

t	***N***						
	IPOG	**ITCH**	**Jenny**	**TConfig**	**TVG**	**GTWay**	**GVS**
2	48	45	45	48	50	46	44
3	308	225	290	312	342	293	288
4	1843	1750	1719	1878	1971	1714	1701
5	10119	NA	9437	NA	NA	9487	9237
6	50920	NA	NA	NA	NA	44884	45732

Table 3. Generated Test Size For *CA(N, 4, 5^P)*

P	*N*						
	IPOG	**ITCH**	**Jenny**	**TConfig**	**TVG**	**GTWay**	**GVS**
5	784	625	837	773	849	731	733
6	1064	625	1074	1092	1128	1027	1012
7	1290	1750	1248	1320	1384	1216	1215
8	1491	1750	1424	1532	1595	1443	1398
9	1677	1750	1578	1724	1795	1579	1556
10	1843	1750	1719	1878	1971	1714	1701
11	1990	1750	1839	2038	2122	1852	1837
12	2132	1750	1964	NA	2268	2022	1955
13	2254	NA	2072	NA	2398	2116	2088
14	2378	NA	2169	NA	NA	2222	2193
15	2497	NA	2277	NA	NA	2332	2294

Table 4. Generated Test Size For CA(N, 4, V^{10})

V	*N*						
	IPOG	**ITCH**	**Jenny**	**TConfig**	**TVG**	**GTWay**	**GVS**
2	46	58	39	45	40	46	45
3	229	336	221	235	228	224	217
4	649	704	703	718	782	621	688
5	1843	1750	1719	1878	1971	1714	1701
6	3808	NA	3519	NA	4159	3514	3502
7	7061	NA	6482	NA	7854	6459	6405
8	11993	NA	11021	NA	NA	10850	8263
9	19098	NA	17527	NA	NA	17272	17188
10	28985	NA	26624	NA	NA	26121	25927

Table 5. Generated Test Size For TCAS Module, $CA(N, t, 10^2 4^1 3^2 2^7)$

t	N						
	IPOG	**ITCH**	**Jenny**	**TConfig**	**TVG**	**GTWay**	**GVS**
2	100	120	108	108	101	100	100
3	400	2388	412	472	434	402	404
4	1361	1484	1536	1476	1599	1429	1302
5	4219	NA	4580	NA	4773	4286	4255
6	10919	NA	11625	NA	NA	11727	10530
7	NA	NA	27630	NA	NA	27119	28760
8	NA	NA	58865	NA	NA	58584	59477
9	NA	NA	NA	NA	NA	114411	119040
10	NA	NA	NA	NA	NA	201728	206000
11	NA	NA	NA	NA	NA	230400	230400
12	NA	NA	NA	NA	NA	460800	460800

GVS produces the best test size for the 2-, 4- and 5-way interactions. ITCH produces the best test size for the 3-way interaction, and GTWay produces the best test size for the 6-way interaction (Table 2). GVS also produces the best test size for the 7-, 8-, 9- and 10-parameter systems. ITCH produces the best test size for the 5-, 6-, 11- and 12-parameter systems, whereas Jenny produces the best test size for the rest of the cases (Table 3). GVS outperforms all other strategies in most cases except for the system with two and four values, where Jenny and GTWay outperform all other strategies (Table 4). For the TCAS system in Table 5, GTWay outperforms the other strategies in almost all cases, i.e. for 2-, 7-, 8-, 9-, 10-, 11- and 12-way interactions. GVS and IPOG obtain the same test size as that of GTWay for the 2-way interaction. In the 4-, 6- and 11-way interactions, GVS outperforms all other strategies. Similarly, IPOG outperforms all other strategies for the 3- and 5-way interactions.

As far as the algorithmic complexity analysis of GVS is concerned, the test size grows exponentially with the interaction strength (*t*) (Tables 2 and 5). Additionally, the test suite grows logarithmically with the number of parameters (P) and quadratically with the number of values (Tables 5 and 6). Theoretically, these results are consistent with those in the existing literature with O (v^t log p) [25].

GVS as Variable Strength t-Way Strategy

The benchmark experiments using the test size results were adopted from Chen et al. [18] and Ahmed and Zamli [19]. Three basic system configurations are defined as follows:

i. Fifteen 3-valued parameter systems: $VCA(N, 2,3^{15},\{C\})$,
ii. Three 4-valued parameter, three 5-valued parameter, and two 6-valued parameter systems: $VCA(N, 2,4^3 5^3 6^2,\{C\})$,
iii. Twenty 3-valued parameter and two 10-valued parameter systems: $VCA(N, 2,3^{20}10^2,\{C\})$

The generated test size for GVS is shown in Table 6, along with the other existing variable-strength strategies.

Table 6. Generated test size for different variable strength t-way strategies

{C}	*N*							
	SA	**Density**	**Para Order**	**PICT**	**TVG**	**ACS**	**VS-PSTG**	**GVS**
	VCA(*N*, $2,3^{15}$,{*C*})							
ϕ	16	21	33	35	22	19	19	19
$CA(3,3^3)$	27	28	27	81	27	27	27	27
$CA(3,3^3)^2$	27	28	33	729	30	27	27	27
$CA(3,3^3)^3$	27	28	33	785	30	27	27	27
$CA(3,3^4)$	27	32	27	105	35	27	30	28
$CA(3,3^5)$	33	40	45	121	41	38	38	39
$CA(4,3^4)$	NA	NA	NA	245	81	NA	81	81
$CA(4,3^5)$	NA	NA	NA	301	103	NA	97	99
$CA(4,3^7)$	NA	NA	NA	505	168	NA	158	157
$CA(5,3^5)$	NA	NA	NA	730	243	NA	243	243
$CA(5,3^7)$	NA	NA	NA	1356	462	NA	441	445
$CA(6,3^6)$	NA	NA	NA	2187	729	NA	729	729
$CA(6,3^7)$	NA	NA	NA	3045	1028	NA	966	947
$CA(3,3^4)$ $CA(3,3^5)$ $CA(3,3^6)$	34	46	44	1376	53	40	45	42
$CA(3,3^6)$	34	46	49	146	48	45	45	45
$CA(3,3^7)$	41	53	54	154	54	48	49	48
$CA(3,3^9)$	50	60	62	177	62	57	57	58
$CA(3,3^{15})$	67	70	82	83	81	76	74	75
	VCA(*N*, $2,4^3 5^3 6^2$,{*C*})							
ϕ	36	41	49	43	44	41	42	40
$CA(3,4^3)$	64	64	64	384	67	64	64	64
$CA(3, 4^3 5^2)$	100	131	141	781	132	104	124	127
$CA(3,5^3)$	125	125	126	750	125	125	125	125
$CA(4, 4^3 5^1)$	NA	NA	NA	1920	320	NA	320	320
$CA(5, 4^3 5^2)$	NA	NA	NA	9600	1600	NA	1600	1600
$CA(3,4^3)$ $CA(3,5^3)$	125	125	129	8000	125	125	125	125
$CA(4, 4^3 5^1)$ $CA(4, 5^2 6^2)$	NA	NA	NA	288000	900	NA	900	900
$CA(3,4^3)$ $CA(4, 5^3 6^1)$	NA	NA	NA	48000	750	NA	750	750
$CA(3,4^3)$ $CA(5, 5^3 6^2)$	NA	NA	NA	288000	4500	NA	4500	4500
$CA(4, 4^3 5^2)$	NA	NA	NA	2874	496	NA	472	463
$CA(5, 4^3 5^3)$	NA	NA	NA	15048	2592	NA	2430	2380
$CA(3, 4^3 5^3 6^1)$	171	207	247	1266	237	201	206	202
$CA(3, 5^1 6^2)$	180	180	180	900	180	180	180	180
$CA(3, 4^3 5^3 6^2)$	214	256	307	261	302	255	260	237
	VCA(*N*, $2,3^{20} 10^2$,{*C*})							
Φ	100	100	100	100	101	100	102	100
$CA(3,3^{20})$	100	100	103	940	103	100	105	102
$CA(3,3^{20} 10^2)$	304	401	442	423	423	396	481	413
$CA(4,3^3 10^1)$	NA	NA	NA	810	270	NA	270	270
$CA(5,3^3 10^2)$	NA	NA	NA	NA	2700	NA	2700	2700
$CA(6,3^4 10^2)$	NA	NA	NA	NA	8100	NA	8100	8100

Table 6 shows that SA produces the best test size in all system configurations with low interaction strength ($t \leq 3$). For high interaction strength ($3 < t \leq 6$), GVS, VS-PSTG and TVG regularly outperform all other strategies in most configurations. ACS, Density, and ParaOrder also display competitive results: in some cases, some of their results also match the best test size. PICT produces the worst overall results.

GVS as Input-Output Based Relations Strategy

Two experiments were adopted, which involved 60 input-output relationships for the 10-parameter system taken from Wang et al. [8, 9]. The input-output relationship definitions for both experiments are *Rel* = {{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}}.

For the first experiment, a system with ten 3-valued parameters was adopted. The experiment started with |*Rel*| = 10, viz. only the first ten relationships in *Rel* were used in generating the test suite. Subsequently, the first 20 relationships in *Rel* were used, until all 60 relationships were finally used. The test size obtained from the first experiment is shown in Table 7.

The second experiment involved the same relationships but using a system with multi-value parameters consisting of three 2-valued parameters, three 3-valued parameters, three 4-valued parameters, and one 5-value parameter. The result obtained from the second experiment is shown in Table 8.

Table 7. Generated Test suite Size for *IOR*{*N*, 3^{10}, |*Rel*|}

|*Rel*|	***N***						
	Density	**ReqOrder**	**ParaOrder**	**Union**	**Greedy**	**TVG**	**GVS**
10	86	153	105	503	104	86	104
20	95	148	103	858	110	105	98
30	116	151	117	1599	122	125	116
40	126	160	120	2057	134	135	117
50	135	169	148	2635	138	139	127
60	144	176	142	3257	143	150	140

Table 8. Generated Test suite Size for *IOR*{*N*, $2^3 3^3 4^3 5^1$, |*Rel*|}

|*Rel*|	***N***						
	Density	**ReqOrder**	**ParaOrder**	**Union**	**Greedy**	**TVG**	**GVS**
10	144	154	144	505	137	144	144
20	160	187	161	929	158	161	162
30	165	207	179	1861	181	179	169
40	165	203	183	2244	183	181	170
50	182	251	200	2820	198	194	200
60	197	250	204	3587	207	209	200

For |*Rel*| = 10, Density and TVG produce the best test size. For |*Rel*|= 20, Density produces the best result. For |*Rel*| = 30, both Density and GVS produce the best test size. Concerning |*Rel*| = 40 until |*Rel*| = 60, GVS produces the most optimum result (Table 7). We note that in all cases, Union produces the worst result.

In Table 8, Greedy produces the best test size for |*Rel*| = 10 and |*Rel*| = 20, whereas Density produces the best test size for all other configurations. Although it does not produce the best result, GVS nevertheless produces acceptable results in all cases, i.e. second to Density in almost all values of *R*. Union produces the worst result for all configurations.

CONCLUSIONS

A new variable strength t-way test suite-generation strategy called GVS, which is based on input-output relationships, has been proposed and evaluated. The evaluation was encouraging because GVS produces good results for uniform number of parameter values and system with high interaction strength (i.e. $t > 3$). As an area for further research, we are investigating new searching algorithm for integration into GVS to produce better test size, especially where the parameter values are non-uniform. Additionally, we are also considering automating the process of determining the input-output relationships among system parameters.

REFERENCES

1. M. F. J. Klaib, "Development of an automated test data generation and execution strategy using combinatorial approach", *PhD Thesis*, **2009**, Universiti Sains Malaysia, Malaysia.
2. K. Z. Zamli, M. F. J. Klaib, M. I. Younis, N. A. M. Isa and R. Abdullah, "Design and implementation of a t-way test data generation strategy with automated execution tool support", *Inform. Sci.*, **2011**, *181*, 1741-1758.
3. Y. Lei, R. Kacker, D. R. Kuhn, V. Okun and J. Lawrence, "IPOG/IPOG-D: Efficient test generation for multi-way combinatorial testing", *Softw. Test. Verif. Reliab.,* **2008**, *18*, 125-148.
4. A. Hartman, T. Klinger and L. Raski, "IBM intelligent test case handler", http://www.alphaworks.ibm.com/tech/whitch, **2005,** (Accessed: March 2011).
5. B. Jenkins, "Jenny test tool", http://www.burtleburtle.net/bob/math/jenny.html, **2010**, (Accessed: April 2011).
6. P. J. Schroeder and B. Korel, "Black-box test reduction using input-output analysis", Proceedings of 2000 ACM SIGSOFT International Symposium on Software Testing and Analysis*,* **2000**, Portland, USA., pp. 173-177.
7. P. J. Schroeder, P. Faherty and B. Korel, "Generating expected results for automated black-box testing", Proceedings of 17th IEEE International Conference on Automated Software Engineering, **2002**, Edinburgh, UK, pp. 139-148.
8. Z. Wang, C. Nie and B. Xu, "Generating combinatorial test suite for interaction relationship", Proceedings of 4th International Workshop on Software Quality Assurance, **2007**, Dubrovnik, Croatia, pp. 55-61.
9. Z. Wang, B. Xu and C. Nie, "Greedy heuristic algorithms to generate variable strength combinatorial test suite", Proceedings of 8th International Conference on Quality Software, **2008**, Oxford, UK, pp. 155-160.
10. J. Arshem, "Test vector generator (TVG)", http://sourceforge.net/projects/tvg, **2010**, (Accessed: March 2011).
11. D. M. Cohen, S. R. Dalal, M. L. Fredman and G. C. Patton, "The AETG System: An approach to testing based on combinatorial design", *IEEE Trans. Software Eng.,* **1997**, *23*, 437-444.

12. C. Nie and H. Leung, "A survey of combinatorial testing", *ACM Comput. Surv.,* **2011**, *43*, Article 11.
13. K. Z. Zamli, M. I. Younis, S. A. C. Abdullah and Z. H. C. Soh, "Software Testing", 1st Ed., Open University Malaysia, Kuala Lumpur, **2008**.
14. Y. Lei and K. C. Tai, "In-parameter-order: A test generation strategy for pairwise testing", Proceedings of 3rd IEEE International Conference on High Assurance Systems Engineering Symposium, **1998**, Washington DC, USA, pp. 254-261.
15. Y. Lei, R. Kacker, D. R. Kuhn, V. Okun and J. Lawrence, "IPOG: a general strategy for t-way software testing", Proceedings of 14th Annual IEEE International Conference and Workshops on The Engineering of Computer-Based Systems, **2007**, Tucson, USA, pp. 549-556.
16. A. W. Williams, "*TConfig*", http://www.site.uottawa.ca/~awilliam/, **2010**, (Accessed: Macrch 2011).
17. M. B. Cohen, P. B. Gibbons, W. B. Mugridge, C. J. Colbourn and J. S. Collofello, "Variable strength interaction testing of components", Proceedings of 27th Annual International Computer Software and Applications Conference, **2003**, Dallas, USA, pp. 413-418.
18. X. Chen, Q. Gu, A. Li and D. Chen, "Variable strength interaction testing with an ant colony system approach", Proceedings of 16th Asia-Pacific Software Engineering Conference, **2009**, Penang, Malaysia, pp. 160-167.
19. B. S. Ahmed and K. Z. Zamli, "A variable strength interaction test suites generation strategy using particle swarm optimization", *J. Syst. Softw.,* **2011**, *84*, 2171-2185.
20. A. Hartman, T. Klinger and L. Raski, "IBM Intelligent Test Case Handler", http://www.alphaworks.ibm.com/tech/whitch, **2010**, (Accessed: January 2011).
21. J. Czerwonka, "Pairwise testing in real world", Proceedings of 24th Pacific Northwest Software Quality Conference, **2006**, Portland, USA, pp. 419-430.
22. P. J. Schroeder, "Black-box test reduction using input-output analysis", *PhD Thesis*, **2001**, Illinois Institute of Technology, Chicago, USA.
23. R. C. Bryce and C. J. Colbourn, "A density-based greedy algorithm for higher strength covering arrays", *Softw. Test. Verif. Reliab.*, **2009**, *19*, 37-53.
24. R. C. Bryce and C. J. Colbourn, "The density algorithm for pairwise interaction testing", *Softw. Test. Verif. Reliab.*, **2007**, *17*, 159-182.
25. M. B. Cohen, "Designing test suites for software interaction testing", *PhD Thesis*, **2004**, University of Auckland, New Zealand.