International Journal of Software Engineering & Computer Sciences (IJSECS)
ISSN: 2289-8522 ,Volume 1, pp. 15-27, February 2015
©Universiti Malaysia Pahang
DOI: http://dx.doi.org/10.15282/ijsecs.1.2015.2.0002

COMPARATIVE BENCHMARKING OF CONSTRAINTS T-WAY TEST GENERATION STRATEGY BASED ON LATE ACCEPTANCE HILL CLIMBING ALGORITHM

Kamal Z. Zamli[1], Abdul Rahman Alsewari[2] and Basem Al-Kazemi[3]

[1,2]Faculty of Computer Systems and Software Engineering,
Universiti Malaysia Pahang, 26300 Kuantan, Pahang, Malaysia
[3]College of Computer and Information Systems,
Umm Al-Qura University, Makkah, Kingdom of Saudi Arabia
Email: kamalz@ump.edu.my

ABSTRACT

This paper describes the new t-way strategy based the Late Acceptance based Hill Climbing algorithm, called LAHC, for constraints t-way test generation. Unlike earlier competing work, LAHC does not require significant tuning in order to have it working. In fact, LAHC merely requires minor adjustment of the common controlling parameters involving iteration and population size depending on the given system configuration. Our benchmarking results have been promising as LAHC gives competitive results in most constraints configurations considered.

Keywords: Optimization Algorithms, Software Testing, Artificial Intelligent

INTRODUCTION

Considering cost and resources constraints, exhaustive testing is practically infeasible. Many sampling based strategies have been proposed in the literature to help test engineers select a subset of test cases (i.e. from the exhaustive testing) and yet not sacrificing the fault detection capability of the testing process. In the field of combinatorial testing, researchers have turned into t-way strategies (Zamliet al., 2011) whereby t indicates the interaction strength. Here, all t-way strategies generate the t-way test suite with the aim to cover every possible combination produced by the interacting parameters (or also known as tuples).

In the last 20 years, many t-way strategies have been proposed in literature (including that of GTWay (Klaib 2009; Zamli, Klaib et al. 2011), MIPOG(Younis, Zamli et al. 2008; Younis, Zamli et al. 2008; Younis 2010; Younis and Zamli 2010), TConfig(Williams 2000; Williams 2010), TCG(Tung and Aldiwan 2000), Jenny(Pallas 2003), TVG (Yu-Wen and Aldiwan 2000; Arshem 2010), IRPS (Younis, Zamli et al. 2008), and TConfig(Williams 2000; Williams 2010)). Recently, researchers have also started to adopt meta-heuristics based strategies including that of Genetic Algorithm (GA) GA (Sthamer 1995; Shiba, Tsuchiya et al. 2004; Bryce and Colbourn 2007; Afzal, Torkar et al. 2009; Chen, Gu et al. 2009; McCaffrey 2010), Ant Colony (ACS) ACA (Harman and Jones 2001; Shiba, Tsuchiya et al. 2004; Wang, Xu et al. 2008; Afzal, Torkar et al. 2009; Chen, Gu et al. 2009), Particle Swarm Optimization (PSTG) PSTG (Ahmed and Zamli 2010; Ahmed and Zamli 2010; Ahmed and Zamli 2011; Ahmed, Zamli et al. 2012), and Harmony Search Strategy (HSS) (Alsewari and Zamli 2012). Although these aforementioned strategies are useful, many of them fall short in term of the support for constraints (i.e. exclusion of impossible combinations) rendering the generation of potentially unusable test cases.

Addressing these issues, this paper describes the new meta-heuristics t-way strategy based the Late Acceptance based Hill Climbing algorithm, called LAHC, for constraints t-way test generation. Like competing meta-heuristics based strategies, LAHC gives sufficiently optimal results as compared to general computational based strategies. Unlike competing meta-heuristics based strategies, LAHC does not require significant tuning in order to have it working. In fact, LAHC merely requires minor adjustment of the common controlling parameters involving iteration and population size depending on the given system configuration. The development of LAHC serves as our research vehicle to investigate the use of variant of Hill Climbing for constraints t-way test generation.

COVERING ARRAY NOTATIONS

Any system under test (termed SUT) consists of number of parameters (or factors) with their associated values (or levels). Typically, t-way strategy aims to generate the most minimum test cases according to the coverage interaction criteria.

Mathematically, t-way interaction test suite can be abstracted using the covering array (CA) notations. Normally, the CA has four parameters; *N*, *t*, *p*, and *v* (i.e., CA (N, t, v^p). Here, the symbols *p*, *v*, and *t* are used to refer to number of parameters, values, and interaction strength for the CA, respectively. For example, CA (9, 2, 3^4) represents a test suite consisting of 9×4 arrays (i.e., the rows represent the size of test cases (*N*), and the column represents the parameter (*p*)). In this case, the test suite also covers two-way interaction for a system with four 3-value parameters.

When the CA gives the most optimal result, the covering array can be rewritten as CAN (N, t, v^p). Often, there is no exact formula to estimate the most optimal result of CA (given the value of t,v, and p). However, the lower bound of a particular covering array can be estimated. Here, no strategy can produce test size lower than the lower bound (Daich 2003). Often, the lower bound for CA can be determined by the product of the values up to t, that is, v_1x v_2xv_1..v_t=v^tin descending order. Taking CA (9, 2, 3^4) as example, the lower bound is 3x3=9.

Similar to CA, mixed covering array (MCA) has three parameters; *N*, *t*, and Configuration (*C*) (i.e., MCA (*N, t, C*)). In addition to *N* and *t* that carry the same meaning as in CA, MCA adopts a new symbol, *C*. Consistent with earlier given notations, *C* represents the parameters and values of each configuration in the following format: $v_1{}^{p1}$ $v_2{}^{p2}$... $v_n{}^{pn}$ indicating that there are p1 parameters with $v_1$values, p_2 parameters with $v_2$values, and so on. For example, MCA (1265, 4, 10^2 4^1 3^2 2^7) indicates the test size of 1265 that covers four-way interaction. Here, the configuration takes 12 parameters: two 10-value parameters, one 4-value parameter, two 3-value parameters, and seven 2-value parameters.

Intuitively, lower bound estimate for MCA is similar to CA and can be determined by the product of the values up to t in descending order. Considering MCA (*N*, 4, 10^2 4^1 3^2 2^7), the estimated lower bound is 10x10x4x3=1200.

In the case of variable interaction strength covering array (VCA), the parameters consist of *N, t, C,* and Set (*CS*) (i.e., VCA (*N, t, C, CS*)). Similar to MCA, *N*, *t*, and *C* carry the same meaning. Set *CS* consists of a multiple set of disjoint (mixed) covering array with strength larger than t (i.e., as sub-strength from the main strength). For example, VCA (*N*, 2, 3^2 2^2, {CA (3, 3^1 2^2)}) indicates the test size of 12 for pairwise interaction (with two 3-value parameters and two 2-value parameters) as the main strength and three-way interaction (with one 3-value parameters and two 2-value parameters) as the sub-strength.

At a glance, the lower bound estimation of VCA is similar to that of MCA and CA. A closer look reveals a subtle difference. Here, the lower bound is dependent on the VCA involved. There are potentially 2 possible cases. In the first case, when the contribution of the main strength is larger than that of the sub-strength, the lower bound can be estimated by taking the product of the main strength values up to t in descending order. For example, the estimated lower bound for VCA (N, 2, $10^2 3^2\ 2^2$, {CA (3, $3^1\ 2^2$)}) is 10x10=100 instead of 3x2x2=12. In the second case, when the contribution of the sub-strength is larger than that of the main strength, the lower bound can be estimated by taking the product of the sub-strength values up to t in descending order. Considering the VCA (N, 2, $3^2\ 2^2$, {CA (3, $3^1\ 2^2$)}) as example, the estimated lower bound is 3x2x2= 12.

Finally, to cater for constraints, a new variable called constraints (F) interaction is introduced to represent the set of disallowed interactions (i.e. CCA (N, t, v^p, F) or MCCA (N, t, C, F) or VCCA (N, t, C, F)). Here, C takes the following format $\{c_{a,b}\}$ where a indicates the p_{th} parameter and b indicates the v_{th} value are within the list of constraints. For example, consider CCA (10, 2, 3^3, F) where $F = \{c_{1,1}, c_{3,1}\}$. In this case, the CCA indicates the test size of 10 for pairwise interaction of three 3-value parameters with constraints pair interaction elements from parameter 1 and value 1, as well as parameter 3 and value 1. It should be noted that the lower bound estimation may not hold true in the presence of constraints (i.e. as constraints forbid some combination, hence, allowing much less generated test size).

RELATED WORK

In general, existing t-way strategies can be categorized into two categories, that is, algebraic approaches or computational approaches respectively (Lei, Kacker et al. 2007).

Algebraic approaches construct test sets using pre-defined rules or mathematical function (Lei, Kacker et al. 2007). Often, the computations involved in algebraic approaches are typically lightweight, and in some cases, algebraic approaches can produce the most optimal test sets. However, the applicability of algebraic approaches is often restricted to small configurations (Yan and Zhang 2006; Lei, Kacker et al. 2007). Orthogonal Arrays (OA) (Hedayat, Sloane et al. 1999; Hartman and Raskin 2004), MOA (Mandl 1985) and TConfig (Williams 2002) are typical example of the strategies that are based on algebraic approach.

Unlike algebraic approaches, computational approaches often rely on the generation of the all pair combinations. Based on all pair combinations, the computational approaches iteratively search the combinations space to generate the required test case until all pairs have been covered. In this manner, computational approaches can ideally be applicable even in large system configurations. However, in the case where the number of pairs to be considered is significantly large, adopting computational approaches can be expensive due to the need to consider explicit enumeration from all the combination space. Example strategies that adopt this approach includes An Automatic Efficient Test Generator (AETG) (Cohen, Dalal et al. 1996; Cohen, Dalal et al. 1997), its variant (mAETG) (Cohen 2004), PICT [8, 36], IPOG (Lei, Kacker et al. 2007), Jenny(Pallas 2003), TVG (Yu-Wen and Aldiwan 2000; Arshem 2010), IRPS (Younis, Zamli et al. 2008), GTWay(Klaib 2009; Zamli, Klaib et al. 2011), MIPOG(Younis, Zamli et al. 2008; Younis, Zamli et al. 2008; Younis 2010; Younis and Zamli 2010), TConfig(Williams 2000; Williams 2010), TCG(Tung and Aldiwan 2000)). Another variant of the computational based approach is based on meta-heuristics algorithm including that of GA (Shiba, Tsuchiya et al. 2004), HSS (Alsewari and Zamli 2012), PSTG (Ahmed and Zamli 2010; Ahmed and Zamli 2010; Ahmed and Zamli 2011; Ahmed, Zamli et al. 2012), SA(Stardom 2001; Cohen, Colbourn et al. 2008), and ACA (Shiba, Tsuchiya et al. 2004)). Although useful to help test engineers sample combinatorial test suite, the support for constraints within the aforementioned strategies have not been sufficiently considered. In fact, there has been only a handful of strategies address the constraints support including that of IPOG (Lei, Kacker et al. 2007), PICT [8, 36], TestCover(Sherwood. 2006), HSS (Alsewari and Zamli 2012), SA_SAT (Cohen, Dwyer et al. 2007), mAETG_SAT(Cohen, Dwyer et al. 2007) and GVS_CONST(R.R. Othman, N. Khamis et al.). In line with the focus of this paper, what follows is the survey of the constraints supporting strategies. A comprehensive recent survey for t-way strategies is beyond the scope of this paper and can be found in Cohen (Cohen 2004), Grindal(Grindal, Offutt et al. 2005), and Nie(Nie and Leung 2011).

IPOG (Lei, Kacker et al. 2007) starts the generation process by generating an exhaustive test suite for the first t parameters (in the case of variable interaction strength (t), the highest t will be chosen) as the initial test suite. Later, IPOG relies on two processes called horizontal extension and vertical extension. Horizontal extension is a process of adding one parameter to the initial test suite. This process is repeated until all parameters are covered by the test suite. Vertical extension is a complementary process for horizontal extension in order to ensure that all tuples are covered. During horizontal extension, there is a possibility that several tuples cannot be covered by the initial test suite. In this case, the initial test suite will be extended vertically by adding several new test cases to the initial test suite. In the case of constraints, IPOG defines a number of Boolean operators allowing the defined constraints to be specified as the rules for selecting the test cases during the vertical and horizontal extension.

PICT [8, 36] generates all specified interaction tuples and randomly selects their corresponding interaction combinations to form the test cases as part of the complete test suite. In case a particular test case matches a specified constraint, PICT randomly generates a new combination for covering the interaction tuples. Constraints are supported in PICT through its Boolean constraints definition that avoids forbidden tuple.

TestCover(Sherwood. 2006) is a commercial software tool. Not much information can be gathered regarding TestCover apart from its commercial availability and some benchmark configurations on constraints that can be obtained from its website.

HSS (Alsewari and Zamli 2012) is a meta-heuristic strategy based on the Harmony Search Algorithm. Intuitively, HSS mimics the musician trying to compose good music from improvisation form the best tune from his memory or from random. In doing so, HSS iteratively exploits the Harmony memory to store the best found solution through a number of defined improvisations within its local and global search process. In each improvisation, one test case will be selected to the final test suite (i.e. provided that no constraints are detected) until all the required interaction tuples are covered.

SA_SAT (Cohen, Dwyer et al. 2007) is a variant strategy based on earlier work based on Simulated Annealing, SA (Stardom 2001; Cohen, Colbourn et al. 2008). Using the metal re-heat and cooling methaphor, SA_SAT relies on a large random search space for generating a t-way test suite. Using probability-based transformation equations, SA_SAT adopts binary search algorithm to find the best test case per iteration to be added to the final test suite. Here, the selection of the best test case per iteration also takes into account the presence of constraints (i.e. upon finding one, the iteration will generated new candidate). In the reported work, SA merely addresses small values of interaction strength(i.e., t≤ 3). Another variant of SA, called CASA (Garvin, Cohen et al. 2011), has been developed by to address the support for constraints. Empirical evidences suggest its successful use for software product lines testing.

Similar to SA_SAT, mAETG_SAT(Cohen, Dwyer et al. 2007) is also a variant of an existing strategy called AETG (Cohen, Dalal et al. 1996; Cohen, Dalal et al. 1997). Similar to its predecessor, mAETG_SAT generates one final test case for every cycle of iteration. For each cycle, AETG generates a number of test case candidates, and from these candidates, one is greedily selected as the final test case (i.e., covering the most uncovered tuples). mAETG_SAT provides the support for constraints through its forbidden tuple implementation.

GVS_CONST (Othman and Zamli 2011) is perhaps the newest strategy that addresses t-way test generation and constraints. In general, GVS_CONST generates several test case candidates and selects the best candidates (one that covered the most uncovered tuples) as final test case. Basically, GVS_CONST adopts iterative and random heuristics for test case selection. GVS_CONST only generates new test case candidates when there is a tie situation (i.e. when more than one value can cover the most uncovered tuples or in the presence of constraints).

LATE ACCEPTANCE HILL CLIMBING ALGORITHM

As the name suggest, Late Acceptance Hill Climbing (LAHC) algorithm is derived from Hill Climbing (HC). The main feature of LAHC is the fact that it is capable of avoiding the local optimum associated with HC. Unlike HC which decides on the best next move based on one-to-one comparison between the current and neighbour candidate, LAHC compares with all the randomly generated potential solutions captured in to the LAHC memory (in the form of list with fixed length). In fact, LAHC generates a current neighbour to be compared with all the corresponding value from the LAHC memory one-at-a-time. LAHC also maintains the previous cost function in the memory to allow selection of the best fit value. Ideally, the candidate cost is compared with the selected i_{th} cost from the memory. If the cost is not worse, the candidate will be accepted (as the current local best). Upon acceptance, the cost of the new current solution will be made to replace the original i_{th} cost from the memory. Here, the list keeps the fitness array F_a of length L_{fa} ($F_a = \{f_0, f_1, f_2..f_{Lfa-1}\}$). The position v, at the i_{th} iteration can be calculated via:

$$v = \mathrm{i} \bmod L_{fa} \tag{1}$$

where mod represents the remainder of the integer division

Assuming minimization problem, the final acceptance condition at i_{th} iteration can be expressed as:

$$C_i \leq \ C_{i-Lfa} \quad \text{or} \quad C_i \leq \ C_{i-1} \tag{2}$$

whereC_i = the candidate cost;C_{i-1} = the current cost;C_{i-Lfa}= the cost of the current L_{fa}iteration before

The main template for the LAHC tailored for constraints t-way generation is given in Figure 1.

Here, the internal M iteration loop will iteratively update L_{fa} with the local best value. Here, an index of the worst solution in L_{fa} is kept internally to facilitate the update of L_{fa}. Upon completion of the M iteration, the local best solution (with the best Cs) will be taken into the final test suite. Here, LAHC maintains the list of constraints as forbidden list in order to make sure that the local best solution does not contain the constraints tuple. If so, new test value will be generated accordingly. The main iteration loop will stop when all the interactions are covered.

```
        Define the constraints list, F
        Generate interactions IL (excluding constraints defined in F)
        Produce an initial solution s
        Calculate initial cost function C(s)
pecify $L_{fa}$, and iteration M
hile L is not empty
       begin
          ////////////////// Diversification //////////////////
       for all k ϵ {0...$L_{fa}$-1} do randomize $f_k$:=s,C(s)

        Assign the initial number of iteration I:= 0;
         do until I=M
        Construct a candidate solution s* based on at
            least 1 uncovered pair
            Calculate its cost function C(s*)
        v :=I mod $L_{fa}$
        if $C(s^*) \leq f_v$ or $C(s^*) \leq C(s)$
        then accept candidate (s :=s*)

          ////////////////// Intensification //////////////////
        s:=pertubate (s)
        if $C(s) > f_{worst}$
             Replace the worst solution in $L_{fa}$ , $f_{worst}$:= s,C(s)
         Increment the number of iteration I:= I+1
         end do
        Pick the best s  from $L_{fa}$ not in F
           If exist best s not violating F
              Add best s  to the final suite
           Reset $L_{fa}$for the next iteration
         End
```

Figure 1. LAHC Strategy

OBJECTIVE FUNCTION AND PARAMETER SETTING

The t-way optimization problem of concerned can be specified using Equation 3 and 4.

$$\text{Maximize } f(x) = \sum_{1}^{N} x_i \tag{3}$$

Subject to

$$x \in x_i \ , i = 1,2, \ldots , N \tag{4}$$

where $f(x)$ is an objective function capturing the weight of the test case in terms of the number of covered interactions; x is the set of each decision variable x_i ; x_i is the set of possible range of values for each decision variable, that is,$x_i = \{x_i(1), x_i(2), \ldots, x_i(K)\}$ for discrete decision variables $(x_i(1) < x_i(2) < \ldots < x_i(K))$; N is the number of decision parameters; and K is the number of possible values for the discrete variables.

Addressing the aforementioned optimization problem, the main consideration for our LAHC strategy is as follows.

A. PARAMETER INITIALIZATION

Firstly, the LAHC accepts the input parameters and their corresponding values. To do so, LAHC needs to initialize the size of L_{fa} as well as the number of iteration, M. As estimates, we use adopt Equation 4 and 5 to generate $L_{fa\ minimum}$ as well as $M_{minimum}$.

$$L_{fa\ minimum} = 2 \times \text{lower bound estimate at t} = 2 \qquad (4)$$

$$M_{minimum} = 10 \times \text{lower bound estimate at t} = 2 \qquad (5)$$

Using the same CA (N, 2, 4 ^{6}) for tuning as proposed by Stardom (Stardom 2001), we perform a number of experiments with our LAHC strategy. Specifically, we vary L and M and t to see the effects that they give on the generated test suite size. For all L and M values, we vary them as a multiple of $2^n x L_{fa\ minimum}$ and $2^n x M_{minimum}$for $n = 0\ till\ 5$ (i.e. based on the lower bound=16 at t=2) respectively. For t values, we vary them from t=2 till t=5 for the obvious reason (i.e. t=6 already gives exhaustive CA (4096; 6, 4 ^{6})). We run all the experiments 20 times to ensure statistical significance in terms of the average and the best test suite size.

Table 1. Effect of L and M on test suite size

CA (N, t, 4 ^{6})	$L_{fa\ min}$ =32, M_{min} =160		L_{fa}=64, M=320		L_{fa}=128, M =640		L_{fa}=256, M =1280		L_{fa}=512, M =2560		L_{fa}=1024, M =5120	
	Avg.	**Bst.**	**Avg.**	**Bst.**	**Avg.**	**Bst.**	**Avg.**	**Bst.**	**Avg.**	**Bst.**	**Avg.**	**Bst.**
t=2	24.15	22	24.1	23	23.6	22	23.7	23	23.75	22	23.75	22
t=3	106.25	104	105.85	102	104.25	102	104.5	98	104.8	101	104.4	101
t=4	418.55	413	416.65	411	416.8	410	415.8	410	416.25	412	416.4	409
t=5	1352.2	1339	1350.2	1329	1351.7	1333	1351.1	1334	1352.05	1337	1352.5	1337

From Table 1, we observe that L_{fa}=128 and M=640 give the best result as compared to other settings. As such, these values will be used throughout this paper.

INTERACTION GENERATION (IL) AND HANDLING OF CONSTRAINTS

LAHC generates the interactions list *IL* containing all interactions tuple combinations for each pair which later forms the objective function described earlier. For the example given earlier, the 2-way parameter interaction has six possible combinations. For combination 1001, whereby P_1 and P_4 are available, there are 3×2 possible interaction elements between P_1 and P_4. For each parameter in the combination number (i.e. with two ones), the value of the corresponding parameter is included in the interaction elements. Here, the excluded values are marked as "don't care". This process is iteratively repeated for the other five interactions, i.e., (P_1, P_2), (P_1, P_3), (P_2, P_3), (P_2, P_4), and (P_3, P_4). Here, if the interaction elements capture the constraints specified in F, then they will be removed individually. The complete algorithm is illustrated in Figure 2.

A. DIVERSIFICATION AND INTENSIFICATION

To achieve optimal solution, there is a need for sufficiently elaborate local and global search via exploiting the diversification and intensification property of the algorithm of interest.

Within the general purpose LAHC algorithm, diversification for global search isappropriately addressed by the generation of random initial solution within the L_{fa} list. However, the intensification element within the local search is missing.

```
        Let ps = t-interaction elements
        Let m= max number of defined parameters
        Let p = {p0 ..pj}, where p represents the sets of values defined for each parameter
  for index=0  to 2^m - 1
egin
       Let b = binary number
        b = convert to binary
        if (the no of  '1's in b = t)
        Add the representitave interaction value for p[index] in Ps
        else
    Add don't care value for p[index] in Ps

        /////////////////// Remove Constraints/////////////////////
    if (Ps in the constraint list F)
        Remove Ps
       End
```

Figure 2. Interaction Element Generation and Constraints Removal

Addressing this intensification issue, there is a need for a good perturbation function which can "slightly" modify the current local best solution to get better solution (see Figure 3). For instance, consider a solution candidate, $S(x^{new})$:

$$S(x^{new}) = (\, x_{1,}^{new} x_2^{new}, \dots x_i^{new}, \dots . x_N^{new}), (Cs\,) \tag{6}$$

Here, only one value of x_i to be pertubated is randomly selected. If x_i range values is {0, 1, 2, 3, 4, 5}, and the new x_i^{new}in the L_{fa} has the value of {3} then this value can be moved to the neighbouring value from -5 to 5.

```
function Perturbate (solution s)

    begin
    Let old_s =s;
      Let i = random {0...length (s)}
   move_size = random between (-max value range of $x_i$ , max value range of  $x_i$)
       $x_i$:= $x_i$+ move_size;

   if   $x_i$>max value range
             $x_i$:= 0

     if $x_i$< 0 max value range
             $x_i$:=max_value_range

     update s($x_i$)
   if (s contain constraints in F)
      S = old S;

  return (s);
    end
```

Figure 3. Function Perturbation

Although IL containing constraints have been dealt with during interaction elements generation, constraint may reappear due to the diversified solution generated by perturbation function. In this case, the perturbation function re-takes the old solution value.

EVALUATION EXPERIMENTS

Our goal is to evaluate the performance of LAHC against other competing strategies. Here, we have adopted two main experiments as conducted in (Alsewari and Zamli 2012) and (R.R. Othman, N. Khamis et al.) respectively. In the first experiment, we are considering 9 CCAs to be benchmarked with LAHC (i.e. involving HSS, SA_SAT, mAETG_SAT, PICT and Test Cover). For the second experiment, we are considering additional 7 CCAs to be benchmarked with LAHC (i.e. involving GVS_CONST, IPOG, and PICT). For our experiments, we have used L_{fa} = 128, and M iteration =640 for all the experiments. Here, we report the best results after 20 runs for statistical significance. Table . : II and III summarize the results. Here, the best generated results are highlighted in bold font.

From the result in Table . : I, LAHC give competitive results overall. Specifically, HSS performs the best with the most numbers of optimal cases (i.e. 8 cases). SA_SAT comes in second with 7 cases. LAHC comes in third overall with 6 cases. PICT gives the poorest results overall (i.e. no single optimal test case size).

Table 2. Benchmarking Results Based On T-Way As In (Alsewari And Zamli 2012)

System Configuration	***Forbidden Constraints***	**LAHC**	**HSS**	**SA_SAT**	**mATEG_SAT**	**PICT**	**Test Cover**
CCA(N, 2,3^3,F)	*F={($C_{2,3}$,$C_{3,1}$),($C_{2,2}$,$C_{3,1}$),($C_{1,1}$,$C_{3,2}$), ($C_{1,3}$,$C_{2,4}$)($C_{1,3}$,$C_{3,3}$),($C_{1,3}$,$C_{2,3}$,$C_{3,3}$)}*	**9**	**9**	**9**	**9**	10	**9**
CCA(N, 2,4^3,F)	*F={($C_{1,1}$,$C_{2,2}$),($C_{1,3}$,$C_{3,4}$), ($C_{1,4}$,$C_{2,4}$,$C_{3,1}$)($C_{1,3}$,$C_{2,2}$)}*	**10**	**10**	**10**	**10**	10	**10**
CCA(N, 2,5^3,F)	*F={($C_{1,2}$,$C_{2,2}$),($C_{1,5}$,$C_{3,3}$),($C_{1,5}$,$C_{3,5}$), ($C_{1,5}$,$C_{2,4}$,$C_{3,2}$),($C_{1,5}$,$C_{2,3}$),($C_{1,2}$,$C_{2,4}$)}*	**16**	**16**	**16**	**16**	17	**16**
CCA(N, 2,6^3,F)	*F={($C_{1,4}$,$C_{2,6}$),($C_{2,4}$,$C_{3,5}$),($C_{1,3}$,$C_{2,1}$), ($C_{2,2}$,$C_{3,3}$),($C_{1,4}$,$C_{3,2}$),($C_{2,4}$,$C_{3,2}$), ($C_{1,6}$,$C_{2,5}$,$C_{3,5}$)}*	17	**16**	17	17	19	17
CCA(N, 2,7^3,F)	*F={($C_{2,1}$,$C_{3,6}$),($C_{1,6}$,$C_{2,6}$,$C_{3,4}$),($C_{1,5}$,$C_{3,1}$), ($C_{1,7}$,$C_{2,5}$),($C_{1,2}$,$C_{2,5}$),($C_{1,7}$,$C_{2,4}$)}*	**25**	**25**	**25**	**25**	26	**25**
CCA(N, 3,5^4,F)	*F={($C_{1,4}$,$C_{3,3}$,$C_{4,2}$),($C_{2,2}$,$C_{4,4}$), ($C_{1,3}$,$C_{2,4}$),($C_{1,2}$,$C_{3,4}$)}*	**25**	26	26	26	27	30
CCA(N, 3,6^4,F)	*F={($C_{1,5}$,$C_{4,3}$),($C_{3,4}$,$C_{4,2}$), ($C_{2,3}$,$C_{4,3}$),($C_{2,2}$,$C_{3,3}$)}*	38	**36**	**36**	37	39	**36**
CCA(N, 3,7^4,F)	*F={($C_{2,3}$,$C_{3,7}$),($C_{2,6}$,$C_{3,7}$),($C_{2,5}$,$C_{3,3}$), ($C_{4,2}$,$C_{4,6}$)($C_{3,3}$,$C_{4,5}$),($C_{1,3}$,$C_{3,7}$)}*	**36**	**36**	**36**	37	39	38
CCA(N, 4,3^5,F)	*F={($C_{1,2}$,$C_{2,2}$,$C_{3,2}$,$C_{4,2}$),($C_{2,1}$,$C_{3,1}$,$C_{4,1}$,$C_{5,1}$)}*	51	**49**	**49**	52	55	**49**

Table 3. Benchmarking Results Based on Variable Strength T-Way As in

System Configuration	***Forbidden Constraints***	***LAHC***	***GVS_CONST***	***IPOG***	***PICT***
VCCA$(N,3,3^7,CA(4,3^5),F)$	$F=(\{C_{1,2},C_{4,1},C_{7,1}\},\{C_{2,1},C_{6,2},C_{7,2}\},\{C_{3,1},C_{4,1},C_{5,3}\},\{C_{2,3},C_{4,2},C_{6,3}\},\{C_{2,2},C_{4,1},C_{7,3}\},\{C_{1,1},C_{2,2},C_{5,3},C_{6,2}\})$	***105***	*112*	*116*	*765*
VCCA$(N,3,4^7,CA(4,4^5),F)$	$F=(\{C_{3,2},C_{5,3}\},\{C_{2,2},C_{6,4},C_{7,4}\},\{C_{1,3},C_{5,3},C_{6,2},C_{7,1}\},\{C_{5,2},C_{6,2},C_{7,3}\},\{C_{1,4},C_{2,4},C_{3,1},C_{4,1}\})$	***325***	*344*	*355*	*4958*
VCCA$(N,3,5^7,CA(4,5^5),F)$	$F=(\{C_{1,5},C_{2,2},C_{3,1},C_{7,5}\},\{C_{1,2},C_{2,3},C_{3,4}\},\{C_{2,1},C_{4,1},C_{5,3},C_{6,4}\},\{C_{2,2},C_{4,2},C_{6,4},C_{7,1}\},\{C_{3,1},C_{6,5},C_{7,2}\},\{C_{4,2},C_{6,1},C_{7,3}\})$	***811***	*819*	*935*	*19994*
VCCA$(N,4,3^9,CA(5,3^6),F)$	$F=(\{C_{1,3},C_{4,1},C_{7,2},C_{9,1}\},\{C_{2,2},C_{3,3},C_{6,1}\},\{C_{4,1},C_{5,1},C_{6,3},C_{8,2}\},\{C_{5,2},C_{6,2},C_{8,2}\})$	***348***	*357*	*378*	*8101*
VCCA$(N,4,4^9,CA(5,4^6),F)$	$F=(\{C_{1,1},C_{5,3},C_{6,4},C_{8,2}\},\{C_{2,2},C_{3,4},C_{4,1},C_{7,2},C_{9,1}\},\{C_{1,2},C_{2,1},C_{8,4},C_{9,4}\},\{C_{3,2},C_{4,3},C_{6,1},C_{7,4}\},\{C_{1,3},C_{2,4},C_{3,1},C_{4,3},C_{5,2}\})$	***1434***	*1500*	*1471*	*87886*
VCCA$(N,4,5^9,CA(5,5^6),F)$	$F=(\{C_{2,2},C_{3,4},C_{8,5},C_{9,5}\},\{C_{3,3},C_{4,3},C_{7,4},C_{8,2}\},\{C_{2,1},C_{5,3},C_{6,5},C_{7,4},C_{8,1}\},\{C_{6,2},C_{7,1},C_{8,4},C_{9,1}\})$	***4403***	*4413*	*5186*	*534168*
VCCA$(N, 4,2^3 3^3 4^3, CA(5,2^2 3^2 4^2),F)$	$F=(\{C_{1,2},C_{2,1},C_{3,2},C_{9,4}\},\{C_{4,2},C_{5,3},C_{7,4},C_{8,1}\},\{C_{2,2},C_{7,2},C_{8,3},C_{9,2}\},\{C_{1,1},C_{2,1},C_{3,2},C_{5,3},C_{7,3}\},\{C_{6,2},C_{7,1},C_{8,2},C_{9,3}\})$	***329***	*345*	*387*	*6938*

As for Table 3, LAHC outperforms all against GVS_CONST, IPOG, and PICT. Putting LAHC aside, GVS_CONST appear to outperform IPOG and PICT. Finally, similar to earlier experiments in Table 2, PICT gives the poorest result overall.

CONCLUSIONS

In short, this paper has elaborated a new strategy, called LAHC, based on Late Acceptance Hill Climbing Algorithm for constraints covering array construction. Our experience with LAHC has been promising. As the scope for future work, we are looking to extend the capability of LAHC in terms of supporting high parameters (i.e. p>500) to be used for software product line testing.

ACKNOWLEDGMENT

This research work involves collaborative efforts between Universiti Malaysia Pahang and Umm Al-Qura University. The work is funded by grant number 11-INF1674-10 from the Long-Term National Plan for Science, Technology and Innovation (LT-NPSTI), the King Abdul-Aziz City for Science and Technology (KACST), Kingdom of Saudi Arabia. We thank the Innovation Office, Universiti Malaysia Pahang and the Science and Technology Unit at Umm Al-Qura University for their continued logistics support.

REFERENCES

Afzal, W., R. Torkar, et al. (2009). "A Systematic Review of Search-Based Testing for Non-Functional System Properties." Information and Software Technology 51 (6): 957-976.

Ahmed, B. S. and K. Z. Zamli (2010). PSTG: A T-Way Strategy Adopting Particle Swarm Optimization. Proceedings of the 4th Asia International Conference on Mathematical /Analytical Modelling and Computer Simulation, IEEE Computer Society.

Ahmed, B. S. and K. Z. Zamli (2010). T-Way Test Data Generation Strategy Based on Particle Swarm Optimization. Proceedings of the 2nd International Conference on Computer Research and Development, IEEE Computer Society.

Ahmed, B. S. and K. Z. Zamli (2011). "The Development of a Particle Swarm Based Optimization Strategy for Pairwise Testing." Journal of Artificial Intelligence 4 (2): 156-165.

Ahmed, B. S., K. Z. Zamli, et al. (2012). "Constructing a T-Way Interaction Test Suite Using the Particle Swarm Optimization Approach." International Journal of Innovative Computing, Information and Control 8 (1): 431-452.

Alsewari, A. R. A. and K. Z. Zamli (2012). "Design and Implementation of a Harmony-Search-based Variable-Strength t-way Testing Strategy with Constraints Support." Information and Software Technology 54 : 553-568.

Arshem, J. (2010). "TVG." Retrieved 16 June, 2010, from http://sourceforge.net/projects/tvg.

Bryce, R. and C. Colbourn (2007). One-Test-at-a-Time Heuristic Search for Interaction Test Suites. Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, London, England.

Chen, X., Q. Gu, et al. (2009). Variable Strength Interaction Testing with an Ant Colony System Approach. Proceedings of the 16th Asia-Pacific Software Engineering Conference, IEEE Computer Society.

Cohen, D. M., S. R. Dalal, et al. (1997). "The AETG System: An Approach to Testing Based on Combinatorial Design." IEEE Transactions on Software Engineering 23 (7): 437-444.

Cohen, D. M., S. R. Dalal, et al. (1996). "The Combinatorial Design Approach to Automatic Test Generation." IEEE software 13 (5): 83-88.

Cohen, M. B. (2004). Designing Test Suites For Software Interaction Testing. PhD, University of Auckland.

Cohen, M. B. (2004). Designing Test Suites for Software Interaction Testing. Ph.D, University of Auckland.

Cohen, M. B., C. J. Colbourn, et al. (2008). "Constructing Strength Three Covering Arrays with Augmented Annealing." Discrete Mathematics 308 (13): 2709-2722.

Cohen, M. B., M. B. Dwyer, et al. (2007). Interaction Testing of Highly-Configurable Systems in the Presence of Constraints. Proceeding of International Symposium on Software Testing and Analysis, London, UK, ACM.

Daich, G. T. (2003). Testing Combinations of Parameters Made Easy. Proceedings of the IEEE Systems Readiness Technology Conference (AUTOTESTCON 2003), CA, USA, IEEE Computer Society.

Garvin, B. J., M. B. Cohen, et al. (2011). "Evaluating Improvements to a Meta-Heuristic Search for Constrained Interaction Testing." Empirical Software Engineering(16): 61-102.

Grindal, M., J. Offutt, et al. (2005). "Combination Testing Strategies: A Survey." Journal of Software Testing, Verification and Reliability 15 (3): 167-199.

Harman, M. and B. F. Jones (2001). "Search-Based Software Engineering." Information and Software Technology 43v(14): 833-839.

Hartman, A. and L. Raskin (2004). "Problems and Algorithms for Covering Arrays." Discrete Mathematics 284 (1-3): 149-156.

Hedayat, A. S., N. J. A. Sloane, et al. (1999). Orthogonal Arrays: Theory and Applications. New York, Springer Verlag.

Klaib, M. F. J. (2009). Development Of An Automated Test Data Generation And Execution Strategy Using Combinatorial Approach. PhD, Universiti Sains Malaysia.

Lei, Y., R. Kacker, et al. (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 U.S.A, IEEE Computer Society.

Mandl, R. (1985). "Orthogonal Latin Squares: An Application of Experiment Design to Compiler Testing." Communications of the ACM 28 (10): 1054-1058.

McCaffrey, J. (2010). An Empirical Study of Pairwise Test Set Generation Using a Genetic Algorithm. Proceedings of the 7th International Conference on Information Technology, IEEE Computer Society.

Nie, C. and H. Leung (2011). "A Survey of Combinatorial Testing." ACM Computing Surveys 43 (2).

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

Pallas, D. (2003). "Jenny." Retrieved 16 June 2010, 2010, from http://www.burtleburtle.net/bob/math.

R.R. Othman, N. Khamis, et al. "Variable Strength T-Way Test Suite Generator with Constraints Support." Malaysia Journal of Computer Science 27 (3): 204-217.

Sherwood., G. (2006). "TestCover." Retrieved 15 June, 2011, from http://testcover.com/pub/constex.php.

Shiba, T., T. Tsuchiya, et al. (2004). Using Artificial Life Techniques to Generate Test Cases for Combinatorial Testing. Proceedings of the 28th Annual International Computer Software and Applications Conference, IEEE Computer Society.

Stardom, J. (2001). Metaheuristics and The Search for Covering and Packing Array Master of Scienc Master thesis, Simon Fraser University.

Sthamer, H. (1995). The Automatic Generation of Software Test Data Using Genetic Algorithms. PhD thesis, Universityof Glamorgan,Pontyprid, Wales.

Tung, Y. W. and W. S. Aldiwan (2000). Automatic Test Case Generation For The New Generation Mission Software System. Proceedings of IEEE Aerospace Conference, Big Sky, MT, USA.

Wang, Z. Y., B. W. Xu, et al. (2008). Greedy Heuristic Algorithms to Generate Variable Strength Combinatorial Test Suite. Proceedings of the 8th International Conference on Quality Software, IEEE Computer Society.

Williams, A. W. (2000). Determination of Test Configurations for Pair-wise Interaction Coverage. Proceedings of the 13th International Conference on Testing of Communicating System, Ottawa, Canada.

Williams, A. W. (2002). "TConfigure" Retrieved 16 June 2010, 2010, from http://www.site.uottawa.ca/~awilliam.

Williams, A. W. (2010). "TConfigure" Retrieved February, 2012, from http://www.site.uottawa.ca/~awilliam/.

Yan, J. and J. Zhang (2006). Backtracking Algorithms And Search Heuristics To Generate Test Suites For Combinatorial Testing. Proceeding of the 30th Annual International Computer Software and Applications Conference, IEEE Computer Society.

Younis, M. I. (2010). MIPOG: A Parallel T-Way Minimization Strategy For Combinatorial Testing. PhD, Universiti Sains Malaysia.

Younis, M. I. and K. Z. Zamli (2010). "MC-MIPOG: A Parallel T-Way Test Generation Strategy for Multicore Systems." ETRI Journal 32 (1): 73-83.

Younis, M. I., K. Z. Zamli, et al. (2008). IRPS --- An Efficient Test Data Generation Strategy for Pairwise Testing. Proceedings of the 12th international conference on Knowledge-Based Intelligent Information and Engineering Systems, Part I, Zagreb, Croatia, Springer-Verlag.

Younis, M. I., K. Z. Zamli, et al. (2008). MIPOG - Modification Of The IPOG Strategy For T-Way Software Testing. Proceeding of The Distributed Frameworks and Applications (DFmA), Penang, Malaysia.

Younis, M. I., K. Z. Zamli, et al. (2008). A Strategy For Grid Based T-Way Test Data Generation. Proceedings the 1st IEEE International Conference on Distributed Frameworks and Application (DFmA '08), Penang, Malaysia.

Yu-Wen, T. and W. S. Aldiwan (2000). Automating Test Case Generation for the New Generation Mission Software System. Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA, IEEE Computer Society.

Zamli, K. Z., M. F. J. Klaib, et al. (2011). "Design And Implementation Of A T-Way Test Data Generation Strategy With Automated Execution Tool Support." Information Sciences 181(9): 1741-1758