Design of A Harmony Search Algorithm Based on Covering Array T-Way Testing Strategy

Abubakar Aminu Muazu[1]
Department of Mathematics & Computer Science.
Umaru Musa Yar'adua University
Katsina, Nigeria.
abubakar.muazu@umyu.edu.ng

Aminu Aminu Muazu[2]
Department of Mathematics & Computer Science.
Umaru Musa Yar'adua University
Katsina, Nigeria.
aminu.aminu@umyu.edu.ng

Abstract— *Software testing aims to minimize the recognized risk of software which is worthless to an admissible value. It is given a higher priority and lack of testing may lead to harmful ends which includes the loss of an important data, or even the lives of people. With increase in the advancement of hardware technology and demands for new functionalities and innovations, software applications grew tremendously in term of size over the last decade. This sudden increase in size has a profound impact as far as testing is concerned. The Covering Arrays (CA) are mathematical objects used in the functional testing of software components. They enable the testing of all interactions of a given size of input parameters in a procedure, function, or logical unit in general, using the minimum number of test cases. Building CA is a complex task that involves lengthy execution times and high computational loads. This paper proposes a design of an algorithm of a t-way testing strategy (t refers to the interaction strength), which is named Harmony Search based Covering Array t-way testing strategy (HCATS). HCATS is based on a meta-heuristic search algorithm as its high level. It is able to support a configuration with covering array notations only. Unlike other existing meta-heuristics, HCATS is adopting One Parameter at a time approach on the strength of Global Neighborhood Algorithm and Particle Swarm Optimization. Our results are promising as HCATS manages to outperform existing t-way strategies on many of the benchmarks.*

Keywords— *Combinatorial Testing, t-way techniques, Optimization, one parameter at a time approach, Covering Array notation.*

I. INTRODUCTION

Many application systems are comprised of many subsystems and each subsystem can provide some functionalities to the overall system by considering it number of parameter and their respective values. Testing the overall system is not attainable practically, since it might be a complex system having a different implementation for each subsystem. In this way, the quality assurances activities are very nervous particularly to the increasing of testing costs and time-to-market pressure. Therefore, testing each subsystem is required such as to minimize the testing time and cost [1].

Some or many of the subsystems are configuration with covering array (CA) notation representations. Unfortunately, testing the configurations with CA notation or other notations are not feasible, this is due to the combination of all parameters values (test data); thus, a sampling strategy to test all parameter-values interactions is required.

T-way combinatorial testing strategies are classified as either algebraic strategies or computational strategies [2] [3] [4].

Algebraic strategies are constructing test suite using the mathematical procedures [4]. Even though, the manipulations in algebraic approaches are typically lightweight which are not in to use by combinatorial explosion problem. This is the reason why all strategies that constructs test suite based on algebraic approach are very fast [4] [5]. On the other hand, algebraic approaches often impose restrictions on the system configurations to which they can be applied [2]. This significantly limits the applicability of algebraic approaches for software testing [2].

But the computational approaches rely on generation of all tuples (possible combinations) and then search the tuple space to generate the required test suite continuously until all tuples have been covered [4] [2]. Any time the number of tuples to be considered are very large, then adopting any computational approaches will make it expensive, especially in terms of the space required to store the tuples and the time required for explicit enumeration [2]. Furthermore, computational approaches can be applied to arbitrary system configurations.

In most recent research, it has shown that an optimization problems focuses on the adoption of meta-heuristic algorithms as the basis for combinatorial t-way testing strategies. The Search Based Software Engineering, is a field that proposed meta-heuristic based combinatorial t-way testing strategies [6]. Example of such strategies include: Genetic Algorithms (GA) [6], Particle Swarm

Optimization (PSO) [6] [7], Harmony Search Algorithm (HS) [7], Ant Colony Algorithm (ACO) [7], Simulated Annealing (SA) [7] and Cuckoo Search (CS) [7]. In view of the above example, it appears that the adoption of these meta-heuristic based strategies provided to be effective for obtaining good result as it's reported in their benchmarking experiments [1].

II. PROBLEM DEFINITION MODEL

A simple configuration with CA notation is used here as a model to illustrate the uniform interaction testing. As illustrated in Figure 1, the subsystem of Techno mobile phone which has twelve (12) parameters with two (2) values (on/off). This will require 2^{12}=4,096 test cases. Therefore, if one (1) minute is required to run a single test case. Then the time required to complete the exhaustive testing is 4,096 minutes, which would be around three (3) days.

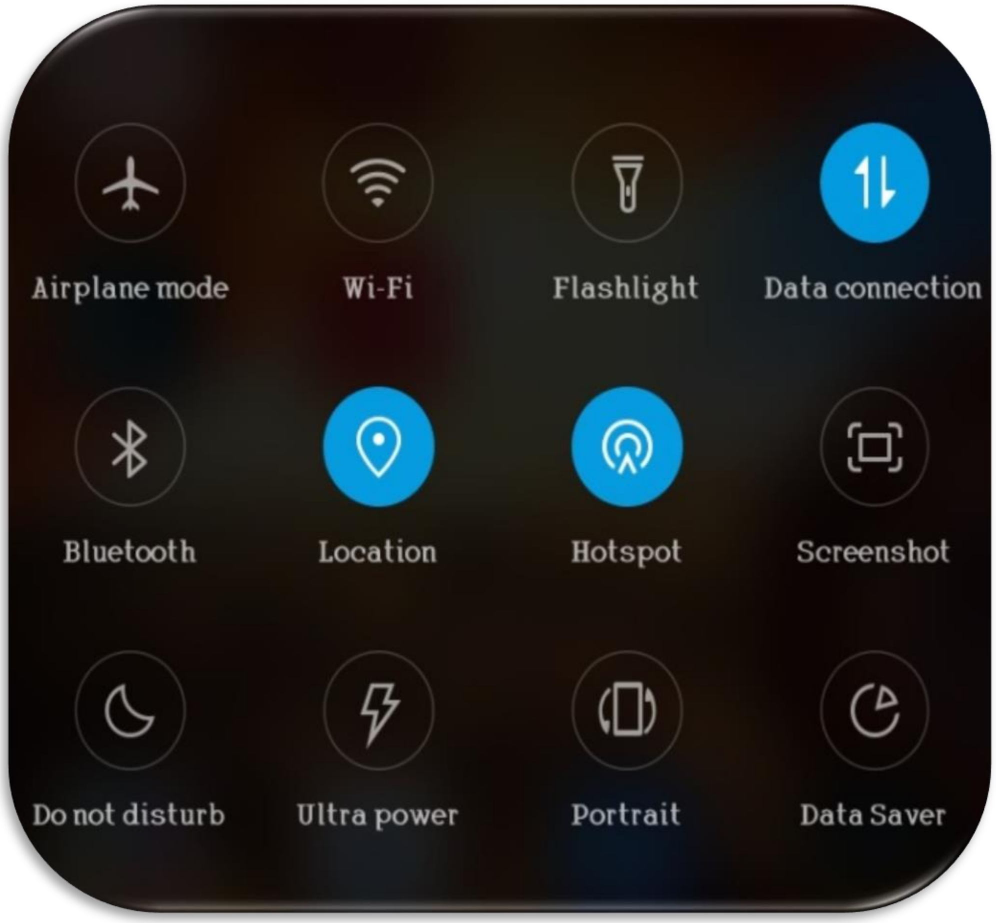

Fig. 1. Techno-K9 Notification-Panel-Shortcuts

This is just a subsystem of a mobile phone that has just (12) parameters, so what about a complete or a large application system which has hundreds or even thousands of parameters? Or maybe the number of values in each parameter are more than two, like an application that has ten (10) parameters; in which 5 parameters have 6 values,
.
3 parameters have 5 values, and the other parameters have 3 values, so the test cases which has to be tested for these 10 parameters is $6^5*5^5*3^2$ = 8,748,000 test cases, if every test case needs one (1) minute to do, so the time to complete the exhaustive testing is almost 17 years

TABLE 1: PARAMETER VALUES OF TECHNO K9 NOTIFICATION-PANEL-SHORTCUTS.

	PARAMETER											
	Airplane mode	Wi-Fi	Flashlight	Data Connection	Bluetooth	Location	Hotspot	Screenshot	Do not Disturb	Ultra-power	Portrait	Data Saver
VALUE	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON
	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Therefore, the exhaustive testing is unattainable and then the optimization of the test cases is crucial criterion to do the testing of the system and these selected test cases should be covered in all combination at least once.

III. COVERING ARRAY NOTATIONS

A covering array (CA) notations can be presented using four parameters; β, *t*, λ, and ℓ (i.e. CA (β, t, λ^{ℓ}). Where β represent the size of test cases, *t* the interaction strength of the array, λ the number of values of parameter and ℓ the number of the parameters [1] [8] [9] [10] [11]. Example, like a system that has thirteen parameters of which each parameter has two values and that can cover three-way interaction for the system. The CA notation for the above example is donated as: CA (8, 3, 2^{13}). Here, this three-way interaction will come up with a total number of 8 test cases.

Similar to CA, mixed covering array (MCA) is when the number of parameters' values varies. Therefore, this can be handled by Mixed Covering Array as: MCA (β, t, $\lambda_1^{\ell 1}$, $\lambda_2^{\ell 2}$, $\lambda_3^{\ell 3}$ … $\lambda_j^{\ell j}$). Where β represent the size of test cases, *t* the interaction strength of the array, λ_1 the number of values for the first parameter, λ_2 the number of values for the second parameter, λ_3 the number of values for the third parameter, λ_j the number of values for the last parameter, and ℓ_1, ℓ_2, ℓ_3, ℓ_j represent number of the parameters respectively [8] [10].

The variable-strength covering array (VCA) is somehow complex compare to CA and MCA. VCA is CA or MCA that contains another CA or MCA inside (i.e. VCA is a super set of either CA or MCA). Here, VCA can be presented using the following parameters β, *t*, λ, ℓ, μ, and can be denoted as VCA (β, t, $\lambda_1^{\ell 1}$, $\lambda_2^{\ell 2}$, $\lambda_3^{\ell 3}$ … $\lambda_j^{\ell j}$ {μ1…. μk}) [8].

The Input-Output relation IOR adopt VCA notation. The mathematical notation denoted as: IOR (N, {μ1... μk}), λ1ℓ1, λ2ℓ2, λ3ℓ3 … λjℓj). Where μ consists of more than one set of parameters relationship definition that will contribute toward the outputs. These set of parameter μ can be indexing starting from 0, 1, 2… n-1, reported in the lecture book [12].

IV. RELATED WORK

The existing strategies that support combinatorial *t*-way interaction can be categorized into two approaches. These are: one test at a time (OTAT) approach and one parameter at a time (OPAT) approach [8] [14].

The OTAT approach begins with an empty initial test suite, then a complete test cases will be constructed and added into a final test suite one after another, until all test cases are covered by a final test suite [1] [13]. Some examples of combinatorial t-wat testing strategy adopting OATA approach are AETG [13], HSS [1] [6].

The OPAT approach begins with an initial test suite which consists of several selected parameters, then it iteratively adds one parameter at time until all parameters are covered (i.e. horizontal extension). Upon the completion, some missing test cases may be added (vertical extension) to ensure maximum interaction coverage [2] [15]. Some examples of combinatorial t-wat testing strategy adopting OPAT approach are: IPO [15] [16], IPOG [2] [16], MIPOG [2] [18], MC_MIPOG [2], ReqOrder [17], ParaOrder [17]. Finally, in the paper work of Alsewari et al [1] demonstrates that HSS performs better than other t-ways testing tool that based on OTAT approach, in both terms of test generations time and the sizes of the generated test suites. Therefore, for these reasons we have adopted the OPAT approach as our basis design of this propose strategy which will only support the configuration with CA notations.

V. OVERVIEW OF HCATS

The framework of HCATS strategy can be describe here that can always construct a minimum test suite. Under the circumstances that different test case construction algorithm that have been developed with an objective to

generate a near minimum test suite. HCATS is to be design based on Harmony Search algorithm which have been proposed some modifications to work on OPAT approach instead of OTAT approach used in HSS [1]. These modifications will enhance and improve the generation of final test suite to be a near optimum size.
The basics behind choosing the harmony search algorithm in HCATS it has the power to control the search between the local solutions and global solutions based on its parameters, and it also shows an efficient performance when generating the test suite based on OTAT approach implemented in the HSS [1].
The HCATS will be develop to support uniform interaction strength test suite. This strategy is comprising of three main algorithms as follows:

- Initial pairs algorithm,
- The test suite generation algorithm
- The mapping algorithm

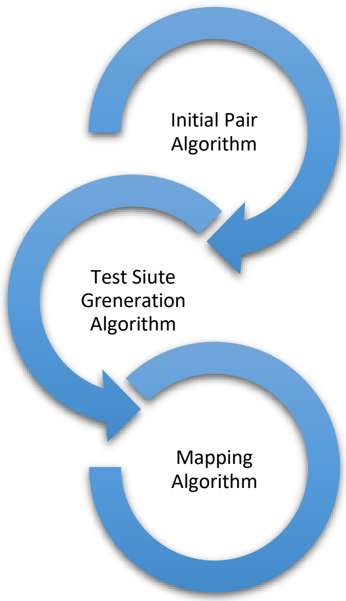

Fig. 2. HCATS Algorithms

The initial pairs algorithm does three things, it will start by reading the configuration data to be tested from a file, generate a pair based on the given interaction strength and then select the pairs with the highest value (longest pair).

The test suite generation algorithm is the main algorithm to optimize and also generate a near optimal final test suite. Here in this algorithm the concept and procedures in HSS is applied that would work on OPAT approach, while the HSS used the concept from HS algorithm but on OTAT approach [1]. These steps are as follow:

- ***Step 1 Initializing the Harmony Search Memory (HM)*:** HCATS will start by initializing the HM with a random test pair by considering the longest pair generated by initial algorithm. All test pairs from the initial pair algorithm will be selected to put a random value from the next parameter until all test pair finish. For every initialization of HM, a best test pair will be selected based on number of covered and uncovered interactions and then put it in the next pairs. The HM most have a specific size called HMS, it can be of any value.

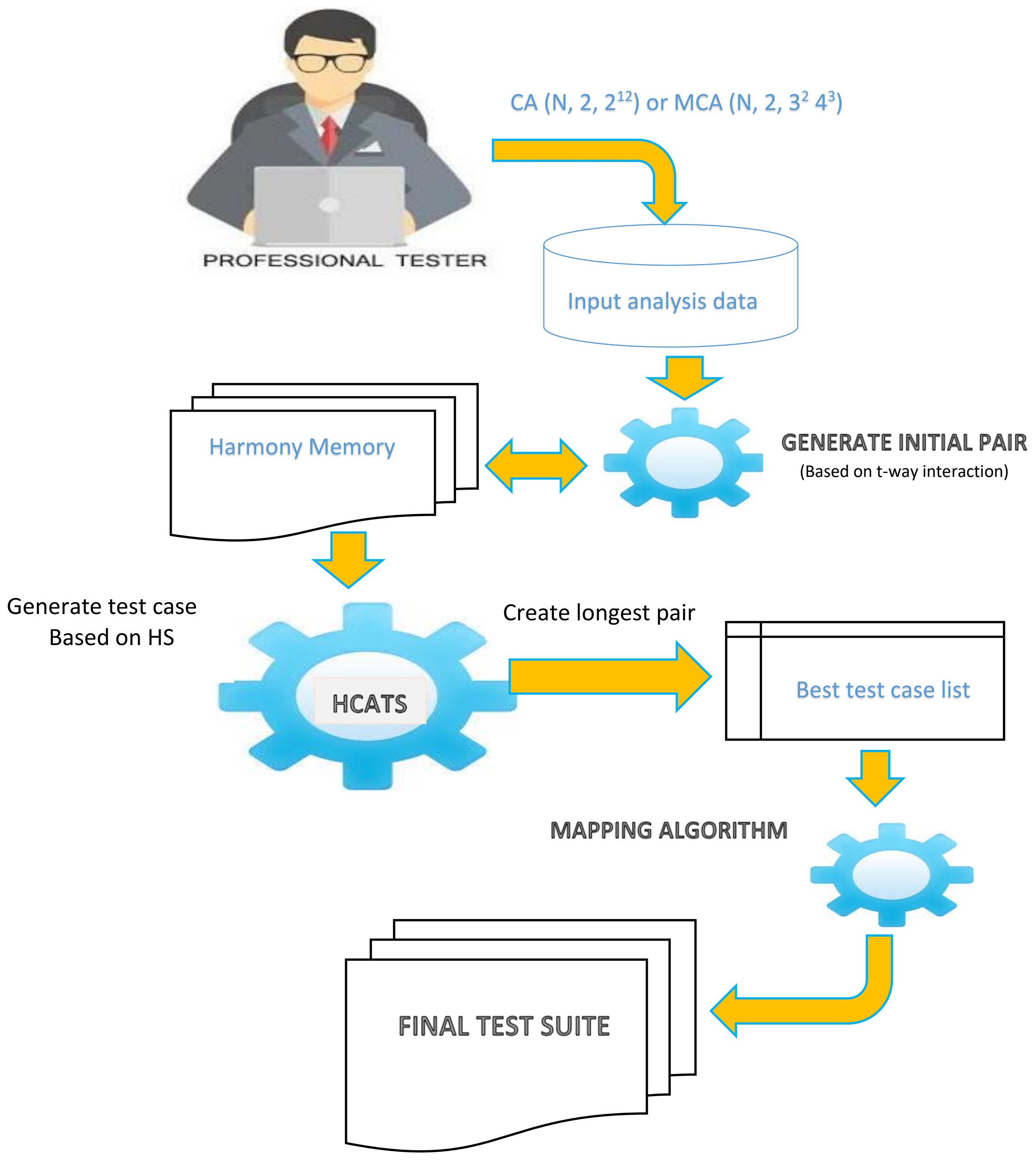

Fig. 3. Overview of HCATS

- ***Step 2 Improvise a new solution from the HM:*** The HCATS here will set a value of harmony memory considering rate (HMCR) to be a random value between 0-1 in order to improvise. This improvisation is based on the HMCR value, HMCR is set to 0.7 so that we can have 70% local improvisation while to have 30% for the global improvisation. If the HMCR< 0.7 it will improvise locally, else it will improvise globally.

For the local improvisation, a pitch adjustment rate (PAR) will be set to be a random value between 0–1 in order to make an adjustment for the selected test pair and it is set 0.9 so as to have 90% of adjusting the value of next parameter while the remaining 10% will adjust any value. It will adjust if PAR<0.9, otherwise no adjusting. Adjusting here will change the value of the next selected parameter to another value within that parameter. For the global improvisation, it will randomly select one test pair initialized in the HM and put it in the next pair.

- ***Step 3 Updating the HM*****:** From step 2 above, a new solution is evaluated (a new pair). So, if that solution is better than the worst in the HM, it will replace that one. Else, it will ignore. Better solution means to have less number of uncovered interaction while worst solution is have high number of uncovered interaction.
- ***Step 4 Iteration*****:** This step will repeat step 2 and step 3 until a maximal number of iterations is met and it generate the near optimal final test suite.

The mapping algorithm will only map the original values of each parameter with their respective integer value used when constructing the initial pairs/generating final test suit. Here in this algorithm, it will replace back the integer value used in constructing final test suit to their original values respectively.

VI. CONCLUSION

In the present paper, we have proposed and discussed a harmony-search-based covering array t-way strategy that support one parameter at a time approach, called HCATS. HCATS was designed based on Harmony Search algorithm which have been proposed some modifications to work on one parameter at a time approach instead of one test at a time approach used in published paper HSS. HCATS also has the potential to address constrains covering arrays for any interaction strength. We are currently investigating a first prototype implementation of HCATS that can be utilized within the FRAMEWORK environment.

REFERENCE

[1] A. A. Alsewari, and K. Z. Zamli, "Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support," Journals on Information and Software Technology, 54, 553–568, 2012.

[2] M. I. Younis, and K. Z. Zamli, "MC-MIPOG: A parallel t-way test generation strategy for multicore systems," ETRI Journal, 32(1), 73-83, 2010.

[3] M.B. Cohen et al., "Constructing Test Suites for Interaction Testing," *Proc. 25th IEEE Int. Conf. Software Engineering*, May 3-10, pp. 38-48. 2003

[4] M. Grindal, J. Offutt, and S.F. Andler, "Combination Testing Strategies: A Survey," *J. Software Testing, Verification, and Reliability*, vol. 5, no. 3, pp. 167-199. 2004.

[5] M. B. Cohen, M. B. Dwyer, J. Shi, "Interaction Testing of Highly-Configurable Systems in the Presence of Constraints". ACM 978-1-59593-734. 2007.

[6] K. Z. Zamli, Y. A. Basem, and K. Graham, "A Tabu Search hyper-heuristic strategy for t-way test suite generation," Journal of Applied Soft Computing, 44, 57-74, 2016

[7] A. B. Nasser, A. A. Alsewari, A. A. Mu'azu, K. Z. Zamli. "Comparative Performance Analysis of Flower Pollination Algorithm and Harmony Search based strategies: A Case Study of Applying Interaction Testing in the Real World". 2nd International Conference on New Directions in Multidisciplinary Research & Practice. ISBN: 978-969-9948-47-3. 2016.

[8] Ahmed, S. and Zamli, K. Z. (2011). A review of covering arrays and their application to software testing. *Journal of Computer Science*. 7(9), 1375-1385.

[9] Jose T. and Eduardo R. (2012). New bounds for binary covering arrays using simulated annealing. Journal of Information Sciences. 185, page 137–152

[10] Alsewari, A. A., Zamli, K. Z. and AL-Kazemi, B. (2015). Generating t-way test suite in the presence of constraints. *Journal of Engineering and Technology*. 6(2), 2180-3811.

[11] Mahmouda T. and Bestoun S. A. (2015). An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use. Journal of Expert Systems with Applications. 42, page 8753–8765

[12] Zamli, K. Z. and Alkazemi, Y. (2015). *Combinatorial T-Way Testing*. Universiti Malaysia Pahang. Malaysia.

[13] D. M. Cohen, S. R. Dalal, A. Kajla, and G. C. Patton, "The automatic efficient test generator (AETG) system," Journals on International Symposium on Software Reliability Engineering, 303-309, 1994.

[14] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG: A general strategy for t-way software testing," 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, 549-556, 2007

[15] K. Tai, and Y. Lie, "Test generation strategy using pairwise," IEEE transaction on software engineering, 28(1), 109-111, 2002.

[16] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG: A general strategy for t-way software testing," 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, 549-556, 2007.

[17] Z. Wang, C. Nie, and B. Xu, "Generating Combinatorial Test Suite for Interaction Relationship," Journal of Software ACM, ISBN 978-1-59593-724, 2007.

[18] M. I. Younis, K. Z. Zamli, N. Ashidi. "MIPOG - Modification of the IPOG Strategy for T-Way Software Testing" international conference. 2008.