Int. J. Advance Soft Compu. Appl, Vol. 5, No. 3, December 2013
ISSN 2074-8523; Copyright © SCRG Publication, 2013

On Test Case Generation Satisfying the MC/DC Criterion

Kamal Z. Zamli, AbdulRahman A. Al-Sewari, and Mohd Hafiz Mohd Hassin

Faculty of Computer Systems and Software Engineering
Universiti Malaysia Pahang
Pahang, Malaysia
e-mail:{kamalz, alsewari, hafizhassin}@ump.edu.my

Abstract

Given the large domain of inputs and possibly too many possible execution paths, the software is often tested using a sampled set of test cases. A variety of coverage criteria have been proposed to assess the effectiveness of the sampled set of test cases. As far as structural testing involving predicate evaluation is concerned, criteria exercising aspects of control flow, such as statement, branch and path coverage have been the most common. Although useful, these criteria are often susceptible to the problem of masking. Addressing this issue, this paper explores to adoption of MC/DC as the necessary criteria for structural testing. Additionally, this paper also highlights the current state-of-the-art and identifies the strengths and limitations of existing work. Complementing existing work and in line with the current trends, this paper justifies on the development of a Harmony Search based test generation strategy for satisfying the MC/DC criterion.

Keywords: *MC/DC test generation, structural testing, optimization algorithms, Harmony Search algorithm.*

1 Introduction

Software testing relates to the process of finding errors (i.e. sometimes involves executing the software of interest) and of validating the software/system against its specification [1]. Apart from reducing the risk of software failures, software testing gives a direct indication of quality (i.e. proving that the program is good or otherwise).

Given the large domain of inputs and possibly too many possible execution paths, the software is often tested using a sampled set of test cases. A variety of coverage criteria have been proposed to assess the effectiveness of the sampled set of test cases. As far as structural testing involving predicate evaluation is concerned, criteria exercising aspects of control flow, such as statement, branch and path coverage [2], have been the most common. At a glance, statement, decision and path coverage appear sufficiently effective exercising the various parts of the software implementation. Nonetheless, a closer look reveals otherwise. Statement, branch, and path coverage are often susceptible to the problem of masking. Here, the usage of AND and OR operations to form compound predicates as the control flow for statement, branch and paths can potentially be problematic. Consider two predicates – (A or B) and (A and B) respectively. The predicate (A or B) always evaluates to true when either A is true (regardless of B) and vice versa. Similarly, the predicate (A and B) is always false when B is false (regardless of A) and vice versa. In this case, A and B are said to have masked each other.

For small inputs, the problem of masking can be straightforwardly addressed by considering all exhaustive input combinations. Yet, for large inputs involving complex predicates, the number of exhaustive combinations can be prohibitively too many. Additionally, as the software is modified and new test cases are often added to the test suite, the test cases grow and the cost of regression testing kept increasing. To address the test-suite size problem within the context of structural testing, many researchers (e.g. [2],[3],[4], [5]) have started to advocate the usage of modified condition/decision coverage (MC/DC) criterion as a strategy to systematically minimize the number of test cases for testing.

In a nut shell, MC/DC is a white box testing criterion ensuring each condition within a predicate can independently influence the outcome of the decision - while the outcome of all other conditions remains constant [6]. In this manner, MC/DC criterion subsumes statements, decisions, and path coverage [2]. As the problem of test case generation fulfilling MC/DC criterion is NP complete, no single existing approach (see related work) can generate optimal set of test set, that is, with the minimum number of test cases for every predicate consideration especially involving large and complex expressions [2]. Furthermore, the process of finding a set of test cases to achieve MC/DC criterion is typically a labor-intensive activity requiring much automation support.

The rest of the paper is organized as follows. Section 2 illustrates an overview of Modified Condition/Decision Coverage. Section 3 elaborates the reflection on related works. Finally, section 4 provides some discussion and conclusion.

2 Overview of Modified Condition/Decision Coverage

As running example, consider the following if statements involving AND and OR operations (see Figure 1). For both AND and OR operations in Figure 1, decision

coverage is registered at 100% even without the need to change the value of y. Specifically, x is masking y and giving misleading coverage. To illustrate further, consider the equivalent if statements for both AND and OR operation as shown in Figure 2. Given the same inputs for x and y (i.e. x=15, y =3 and x=5, y=3), there are parts of the program which has not been covered (as in shaded regions in Figure 2).

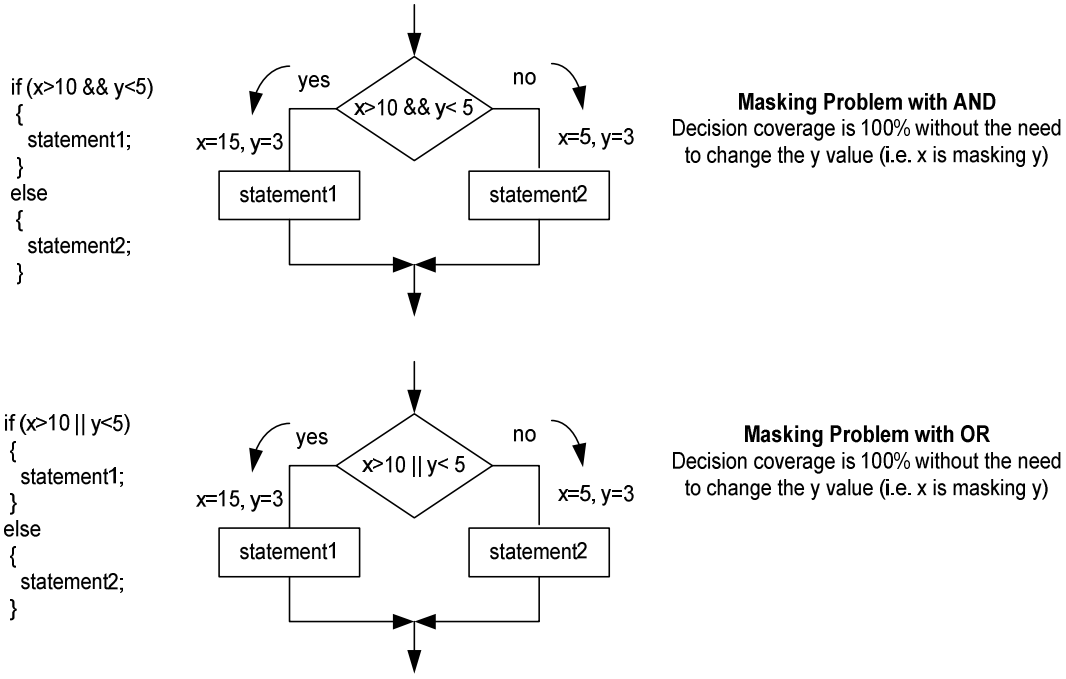

Fig. 1. Masking Problem

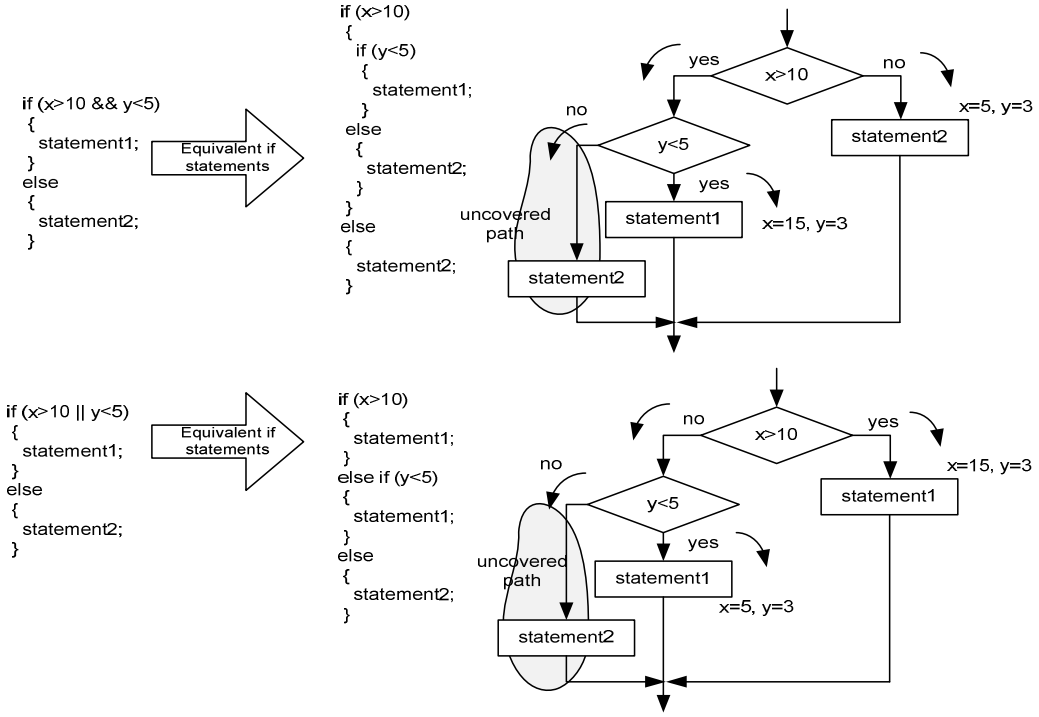

Fig. 2. Equivalent if statements

The main concerns here is on how to cover the uncovered path and hence eliminate masking problems for both AND and OR operations. Exhaustive combinations (often termed as multiple condition coverage (MCC)) are the most desirable alternatives. However, considering MCC is practically infeasible especially when the combinations are large. Here, the number of conditions grew with 2^n where n is the number of Boolean variables.

Condition coverage (CC) and Condition/Decision coverage(C/DC) are also possible. CC dictates that every condition in a decision has taken all possible outcomes at least once. C/DC requires CC and also dictates the TRUE and FALSE decision outcome at least once. Despite being useful, CC and C/DC does not consider independence as the criteria for selecting test cases.

Summing up in Table 1, it is clear that MC/DC is the most viable alternative but with significantly reduced test size as compared to exhaustive combination, MCC. Here, MC/DC dictates that each condition within a predicate can independently influence the outcome of the decision. MC/DC is a stricter form of decision coverage. For decision coverage, each decision statement must evaluate to TRUE on some execution of the program and must evaluate to FALSE on some execution of the program. MC/DC, however, requires execution coverage at the condition level.

Table 1: Types of Structural Coverage Adopted from Hayhurst et al [7]

Coverage Criteria	Decision Coverage	Condition Coverage	Condition/ Decision Coverage	MC/DC	Multiple Condition Coverage
Every point of entry and exit in the program has been invoked at least once.	√	√	√	√	√
Every decision in the program has taken all possible outcomes at least once.	√		√	√	√
Very condition in a decision in the program has taken all possible outcomes at least once.		√	√	√	√
Every condition in a decision has been shown to independently affect the decision's outcome.				√	√
Every combination of condition outcomes within a decision has been invoked at least. once					√

An MC/DC test predicates exist in pairs. Each one of the pair differs only by the Boolean value of one condition, but gives a different result for the decision statement. For AND operation, MC/DC pairs are {{F,T}, {T,T}}, {{T,F}, {T,T}}. As the entry {T,T} is redundant, the complete MC/DC compliant test predicate is reduced to {F,T}, {T,F} and {T,T}. In similar manner, for OR operation, the MC/DC pairs are compliant test predicates are {{F,F}, {T,F}}, {{F,T}, {T,F}}.

As the entry {T,F} is redundant, the complete MC/DC compliant test predicate is reduced to {F,F}, {F,T} and {T,F}.Converting the MC/DC compliant predicates into test cases values for AND and OR operation, Figure 3 revisits the masking problem in Figure 2. Here, the test cases fulfilling the MC/DC criterion are able to cover all the paths.

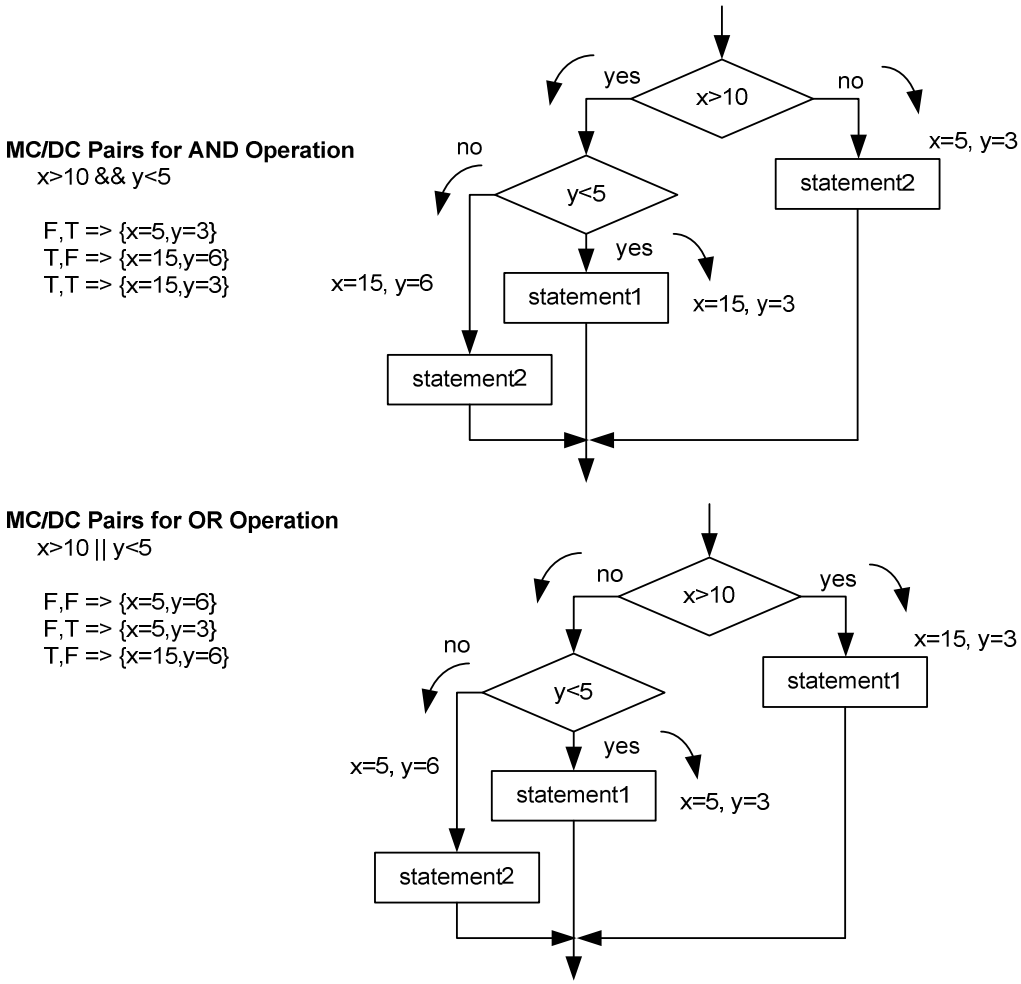

Fig.3. MC/DC Coverage

3 Reflection on Related Work

There is already a number of related works that deals with test case generation for MC/DC coverage. Jones and Harrold [3] introduce two strategies for generating MC/DC compliant test cases. The first strategy is based on the breakdown algorithm whilst the second strategy is based on the prioritization algorithm. At the start, both strategies generate the exhaustive MC/DC pairs as the basis for selection. For the first strategy, the selection of the test candidates is based on iterative generation of essential test cases. Here, essential test cases are established by summing up contribution of each test case towards MC/DC coverage. In each iteration, the least contributing test case is systematically removed leaving only available for selection. For the second strategy, the selection of test candidates is

also done iteratively. In this case, in each iteration, the contribution for each candidate test case is prioritized based on greedy ordering, that is, to cover the most pairs. The iteration stops when no more pairs are available for selection. Although helpful, both strategies appear unsuitable for handling large predicates owing to the need to generate all exhaustive MC/DC pairs.

In other work, Jun-Ru and Chin-Yu [4] usefully exploit n-cube graph in order to generate appropriate MC/DC compliant test data. In this case, the vertex of the cube represents the resultant boolean enumeration for predicates under evaluation. Each vertex is traversed and arranged and evaluated using Gray code sequence ordering until all the required sequences are covered. As the sequence of ordering for MC/DC pairs are non-unique (and not generalizabile to only Gray code sequence), this strategy appears not optimized as far as the number of test cases is concerned.

Ghani and Clark [5] are perhaps the pioneer researchers that adopt optimization algorithm based on Simulated Annealing (SA) for MCC and MC/DC test generation. SA works based on the process of maximizing material's crystal size via heating and slow cooling [9, 10]. The heating process excites the atom to move from its initial position (to avoid a local minima of internal energy) while the slow cooling process allows the atom to settle for lower internal energy configurations for better crystal size. Analogous to the physical process, SA based strategy starts with a randomly generated MC/DC pair of test cases (as initial state) and applies a series of transformations according to a pre-defined probability equation. Here, the probability equation depends heavily on parameter T (namely, the controlling temperature of the simulation) to simulate the heating and cooling process.

Complementing the work from Ghani and Clark, Awedikian et al [2] adopt two optimization algorithms based on Hill Climbing (HC) and Genetic Algorithm (GA) respectively to generate MC/DC compliant test cases. For HC, the algorithm starts by choosing a random test case as an initial solution. The quality of the test case is evaluated based on the defined fitness function. HC attempts to improve the current test case by moving to better points in a neighborhood of the current solution. This iterative process continues until a termination criterion. There are two termination conditions. First, for the given major clause, HC terminates if test case satisfying the MC/DC clause assignment are found. If after a fixed number of attempts, the algorithm is not able to satisfy the MC/DC major clause constraints, the search is stopped and another set of possible MC/DC assignments is selected. Concerning GA, the algorithm starts by creating an initial population of n test cases chosen randomly. Each chromosome represents a test case; genes are values of the input variables. In an iterative process, GA tries to improve the population from one generation to another. Test cases in a generation are selected according to their fitness in order to perform reproduction, that is, through crossover and/or mutation. Then, a new generation is constituted by the fittest test cases of the previous generation and the offspring obtained from

crossover and mutation. The iterative process continues until a stopping criterion is met. Here, two stopping criteria are defined. First, for the given major clause, GA terminates if test input data satisfying the MC/DC clause assignment are found. GA is also stopped when an upper limit in computation is reached.

4 Analysis

Based on the analysis in the previous sections, this section summarizes the description of existing MC/DC test generation strategies. Specifically, Table 2 depicts the merits and limitations of these strategies.

Table 2: Analysis of the existing MC/DC coverage test case strategies

Work	**Merits**	**Limitations**	**Other Observations**
Jones and Harrold [3]	Developed two strategies: the first strategy is based on the breakdown algorithm whilst the second strategy is based on the prioritization algorithm. Both strategies can generate MC/DC compliance test cases number for small systems.	Difficult to handle large predicates owing to the need to generate all exhaustive MC/DC pairs.	Deterministic output
Jun-Ru and Chin-Yu[4]	Exploits the novel property of n-cube graph for MC/DC test generation.	Complex implementation, hence, difficult to accommodate large predicates	Deterministic output

Ghani and Clark [5]	Adopts optimization algorithm based on Simulated Annealing (SA)	Depending on the choices of initial value, SA can get stuck into local minimum	Non-deterministic output based on local search
Awedikian et al [2]	Adopts two optimization algorithms based on Hill Climbing (HC) and Genetic Algorithm (GA) respectively to generate MC/DC compliant test cases	HC tends to get stuck into local minimum as it always uses the current best as its basis for the generation of its new neighbor.	Non-deterministic output based on local search.
		GA requires computationally intensive structure in terms of the need to frequently interact with peers and the environment in order to update and exchange information [11, 12].	Non-deterministic output based on both local and global search although without memories of the previous search.

Indeed, strategies based on optimization algorithms appear to be the trend for the future. However, there are still rooms for improvements particularly on improving the simplicity and scalability of the adopted strategies as well as reducing the proneness to local optima problem. Additionally, adopting other competing optimization algorithm other than HC, SA, and GA might give rise to new perspectives on MC/DC test generation.

Addressing the aforementioned issues and building from existing work, we are investigating the use of Harmony Search (HS) algorithm for our new MC/DC test generation strategy. Among the advantages of HS which justifies our choice include:

- The musical improvisation analogy in HS is relatively appealing and straightforward compared to the existing AI-based algorithms behaviors.
- HS requires only lightweight computation with small number of parameters that require tuning [12]

- HS offers good balance as far diversification (i.e., global investigation of solution space) and intensification (i.e., fine search around local optimal) are concerned [13].
- HS performs well as compared to other AI-based algorithms for various engineering applications (refer to [14 - 18]).
- HS is free from divergence [17].

5 Discussion and Conclusion

To ensure the quality software that conforms to specifications, the software needs to be thoroughly tested. One key aspect to be tested is on structural testing. Here, like most testing endeavor, exhaustive structural testing is not always feasible as it consumes significant resources in term of costing and man power.
Existing work on MC/DC for structural testing has been useful, but they are not without limitations. Our work strives to address the aforementioned limitations (as elaborated in Table 2). Summing up, our research questions are as follows:

- How can the HS strategy decide on which combination of data values to be chosen over large combinatorial data sets?
- How well does HS perform over other existing counter parts?
- How effective is the MC/DC coverage for finding faults?
- How can the MC/DC generation process be fully automated?

Finally, to conclude, this paper has highlighted the need to support MC/DC in software testing. Additionally, this paper has also highlighted the current state-of-the-art on existing work involving MC/DC and identifies novel areas for further research.

ACKNOWLEDGEMENTS

This research is partially funded by ERGS Grant: CSTWay: A Computational Strategy for Sequence Based T-Way Testing, UMP RDU Short Term Grants: Development of a Pairwise Interaction Testing Strategy with Check-Pointing Recovery Support and .myGrants: A New Design of An Artifact-Attribute Social Research Networking Eco-System for Malaysian Greater Research Network.

References

[1] Zamli, K. Z., Klaib, M. F. J., M. I. Younis, Isa, N. A. M., and Abdullah, R. 2011. Design and Implementation of a T-Way Test Data Generation

Strategy with Automated Execution Tool Support. *Information Sciences*, Elsevier, Vol. 181, No. 9, 1741-1758.

[2] Awedikian, Z., Ayari, K., and Antoniol, G. 2009. MCDC Automatic Test Input Generation, *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO '09)*, 1657-1664.

[3] Jones, J.A., and Harrold, M.J. 2003. Test-suite Reduction and Prioritization for Modified Condition/Decision Coverage. *IEEE Transactions on Software Engineering*, Vol. 29, No. 3, 195-209.

[4] Jun-Ru, C., and Chin-Yu, H. 2007. A Study of Enhanced MC/DC Coverage Criterion for Software Testing, *Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, 457-464.

[5] Ghani, K., and Clark, J.A. 2009. Automatic Test Data Generation for Multiple Condition and MCDC Coverage, *Proceedings of the 4th International Conference on Software Engineering Advances*, Porto, Portugal, 152-157.

[6] Johnson, L. A. 1998. DO-178B, Software Considerations in Airborne Systems and Equipment Certification. *Crosstalk, October*.

[7] Hayhurst, K. J., Veerhusen, D. S., Chilenski, J. J., & Rierson, L. K. 2001. A Practical Tutorial on Modified Condition. *Decision Coverage*. NASA Technical Memorandum TM-2001-210876, NASA Langley Research Center.

[8] Quadri, S. M. K., & Farooq, S. U. 2010. Software Testing–Goals, Principles, and Limitations. *International Journal of Computer Applications*, Vol. *6*, No. 9, 7-10.

[9] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. 1953. Equation of State Calculations by Fast Computing machines. *The Journal of Chemical Physics*, Vol 21, 1087.

[10] Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. 1983. Optimization by Simmulated Annealing. *Science*, Vol. 220, No. 4598, 671-680.

[11] Ahmed, B. S., & Zamli, K. Z. 2011. A Variable Strength Interaction Test Suites Generation Strategy Using Particle Swarm Optimization. *Journal of Systems and Software*, Vol. 84, No. 12, 2171-2185.

[12] Alsewari, A. R. A., & Zamli, K. Z., 2012. Design and Implementation of a Harmony-Search-Based Variable-Strength T-way Testing Strategy with Constraints Support. *Information and Software Technology*, Vol. 54, No. 6, 553-568.

[13] Yang, X. S. 2009. Harmony Search as a Metaheuristic Algorithm. In *Music-Inspired Harmony Search Algorithm*, 1-14. Springer Berlin Heidelberg.

[14] Geem, Z. W. 2007. Optimal Scheduling of Multiple Dam System Using Harmony Search Algorithm. *Computational and Ambient Intelligence*, 316-323. Springer Berlin Heidelberg.

[15] Geem, Z. W. 2009. Particle-Swarm Harmony Search for Water Network Design. *Engineering Optimization*, Vol. 41, No. 4, 297-311.

[16] Geem, Z. W., & Hwangbo, H. 2006. Application of Harmony Search to Multi-Objective Optimization for Satellite Heat Pipe Design, *Proceedings of US-Korea Conference on Science, Technology, and Entrepreneurship* (UKC 2006), Teaneck, NJ, USA, (CD-ROM). 1-3).

[17] Geem, Z. W., Kim, J. H., & Loganathan, G. V. 2001. A New Heuristic Optimization Algorithm: Harmony Search. *Simulation*, Vol. 76, No. 2, 60-68.

[18] Geem, Z. W., & Park, Y. 2006. Harmony Search for Layout of Rectilinear Branched Networks. *WSEAS Transactions on Systems*, Vol. 5, No. 6, 1349-1354.