An Effective Approach to Build Optimal T-way Interaction Test Suites over Cloud Using Particle Swarm Optimization

Priyanka[1], Inderveer Chana[2], and Ajay Rana[1]

[1] CSED, Amity University, Noida, India
priyankamatrix@gmail.com, ajay_rana@amity.edu
[2] Thapar University, India
inderveer@thapar.edu

Abstract. Software testing is an expensive and time consuming activity that is often restricted by limited project budgets. There is a need for advanced software testing techniques that offer a solid cost-benefit ratio in identifying defects. Interaction testing is one such method that may offer a benefit. Combinatorial or Interaction Testing is a practical test generation technique that offers a benefit when used to complement current test generation techniques such as equivalence partitioning, boundary value analysis. There are many existing issues which have not been fully addressed. One of the key issues of Combinatorial Testing is Combinatorial Explosion problem which can be addressed through Parallelization. In this paper, we propose an effective approach to build optimal t-way interaction test suites over the cloud environment which could further reduce time and cost.

Keywords: T-way Testing, Pairwise Testing, Interaction test suites, Combinatorial testing.

1 Introduction

Software testing is an expensive and time consuming activity that is often restricted by limited project budgets. Accordingly, the National Institute for Standards and Technology (NIST) reports that software defects cost the U.S economy close to $60 billion a year [1]. They suggest that approximately $22 billion can be saved through more effective testing. There is a need for advanced software testing techniques that offer a solid cost-benefit ratio in identifying defects. Interaction testing is one such method that may offer a benefit. Interaction testing or Combinatorial Testing implements a model based testing approach using combinatorial design. In this approach, it creates test suites by selecting values for input parameters and by combining these parameter values This testing method has been applied in numerous examples like medical devices, browsers, servers etc. [2] [3].

Combinatorial Testing can detect hard to find software faults more efficiently than manual test case selection methods. It can be categorized into two types Pairwise Testing and T-way testing.

2 Related Work

T-way testing is very promising technique for generating test data in software quality assurance because it provides effective error detection at low cost. There are three main types of algorithms to construct combinatorial test suites: Algebraic, Computational and Heuristic search algorithms [4][5]. Comparison chart is shown in Table 1. Test data generation process for multi-way testing can be fully automated. Several tools that automate the production of complete test cases covering up to 6-way combinations are summarized in Table 2. Combinatorial Testing is a practical software testing approach, which could detect the faults that triggered by single factors in software and even interactions of them. Existing Combinatorial Testing Algorithms is summarized in Table 3.

Table 1. Comparison chart of Combinatorial Strategies

Algebraic Approach	**Computational Approach**	**Heuristic search**
1. Extensions of the mathematical methods.	It iteratively searches the combinations space to generate the required test case until all pairs have been covered.	These techniques start from a pre-existing test set and then apply a series of transformations to the test set until a test set is reached that covers all the combinations that need to be covered.
2. They do not enumerate any combinations, hence less expensive.	It is expensive due to the need to consider explicit enumeration from all the combination space.	Produce smaller test suites than Computational Approach.
3. Algebraic approaches can be extremely fast.	Time required is more than Algebraic Approach.	Time required is more than Computational Approach.
4. Impose serious restrictions on the system configurations to which they can be applied.	It can be applied to arbitrary system configurations.	It can be applied to arbitrary system configurations.
5. Test prioritization and Constraint handling can be more difficult for algebraic approaches.	It constructs tests in a locally optimized manner. Thus, the size of test sets generated may not be minimal	Easy to adapt computational approaches for test prioritization and constraint handling.
6. Deterministic Approach. For e.g. Covering Arrays and Orthogonal Arrays.	Can be either deterministic or Non-Deterministic. For e.g. AETG, IPO etc	Can be either deterministic or Non-Deterministic. For e.g. Simulated Annealing, Hill-Climbing.

3 Research Objectives

Based upon literature review, we found that one of the key issues of Combinatorial Testing is Combinatorial Explosion problem (i.e. too many data set to consider) which can be addressed through Parallelization. Many Combinatorial testing techniques have been proposed which mainly focus on minimization of the resulting test sets with balanced time and space requirements [6] [7] [16], removal of unwanted controls and data dependencies [8] [9], pairwise testing with efficient data structure for storing and searching pairs [10], but none of the techniques have been yet ported to cloud environment which could further reduce time and cost. Due to resource constraints, it is nearly always impossible to exhaustively test all of these combinations of parameter values. This problem can be addressed through Parallelization which can be an effective approach to manage the computational cost to find a good test set that covers all the combinations for a given interaction strength (t). In this paper we propose a strategy to build optimal t-way interaction test suites using artificial life techniques like particle swarm optimization that that can be executed in the cloud environment for further reduction in cost and time. Benefits of using Cloud as execution platform can be listed as: I) Computing clouds are huge aggregates of various grids (academic, commercial), computing clusters and supercomputers. They are used by a huge number of people either as users (300 million users of Microsoft's Live) or developers (330.000 application developers of Amazon EC2). II) Cloud computing has strength to tackle vast amounts of data coming not only from the web but also from a rising number of instruments and sensors as it draws on many existing technologies and architecture and integrates centralized, distributed and 'software as service' computing paradigms into an orchestrated whole [21] and III)The emergence of the computing cloud will invigorate academic research and will have strong potential to spawn innovative collaboration methods and new behaviors for eg. SETI@home and FOLDING@home.

4 Particle Swarm Optimization

Particle swarm optimization is a strategy that tries to manipulate a certain number of candidate solutions at once. The whole population is referred to as swarm whilst the solution is referred to as particle. Each solution is represented by a particle that works in the search space to enhance its position (i.e. in order to find solution of the problem at hand). Figure 1 illustrates the typical particles on the swarm search space. As compared with other artificial intelligent optimization methods, PSO has few parameters to regulate and can be easily merged with the environment that needs optimization. In addition, PSO does not need the calculation of derivatives that the knowledge of good solutions is kept by all particles and that particle share the information with others in the swarm [12].

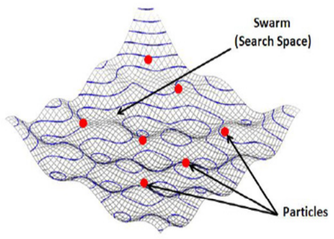

Fig. 1. Illustration of particle on the swarm search space [12][13]

The manipulation of the particles around the search space is restricted by a certain update and positions rule. The particles are manipulated according to the following equations [13]:

$$Vj,d\ (t) = w\ Vj,d(t\text{-}1) + c\ rj,d(pBestj,d\ (t\text{-}1) - Xj,d\ (t\text{-}1)) + c'r'j,d\ (lBestj,d\ (t\text{-}1)\text{-}Xj,d(t\text{-}1)) \quad (1)$$

$$Xj,d= Xj,d(t\text{-}1)+ Vj,d\ (t) \quad (2)$$

Where 't' is the iteration number or time, d is the dimension, j the particle index, w is the inertia weight, r and r' are two random factors, which are two random real numbers between 0 and 1, and c, c' are acceleration coefficients that are adjusting the weight between components.

Table 2. Summary of existing combinatorial testing tools

Tool Name	Origin	Citation	Tool Category
FireEye	NIST	[4]	Implements IPOG and IPOG-D; written in Java.
TVG	J.Arshem	[11]	Generates combinatorial Test Vectors based on the input-output relationship, N-way coverage or randomly;VSIT.
ITCH	IBM	[20]	Implements combination of several algebraic methods; written in Java.
TConfig	Williams	[16]	Pairwise Interaction Coverage.
ACTS	NIST	[1]	Implements IPOG, IPOG_D, IPOF1 and IPOF2.
			Implements Greedy Algorithm; uses Random Search Algorithm.
Jenny	Jencins	[17]	Implements Greedy Algorithm, written in C.
			Variable strength combinatorial test generation (VSIT).
PSTG	Bestoun S. Ahmed	[19]	Implements Particle Swarm Optimization.
MC-MIPOG	Mohammad Younis	[9]	Implements parallel strategy on multicore architecture.
PairTest	Yu Lei	[4]	Implements IPO; written in Java.

5 Proposed Strategy

Figure 2 shows the main classes of the tool that we have proposed in order to generate an optimal t-way test suites using particle swarm optimization. Class Starter contains the main program and will start the execution of operations for building the optimized test suite. Class PSuiteOptimizer will be called by class Starter that contains the algorithm based upon particle swarm optimization which generates the optimized test suite. Class Repository contains the version of the test suite that is considered optimal. In order to obtain real parallelism between threads, the instances of PSuiteOptimizer will be executed on different machines. For this, we used the framework MapReduce [14] that gives support for automatically distributing an application. MapReduce is a programming model and an associated implementation for processing and generating large datasets that is amenable to a broad variety of real-world tasks. Generated minimal t-way test suites executes simultaneously on different machines in cloud environment. A Cloud will be set up to implement and validate the above proposed model.

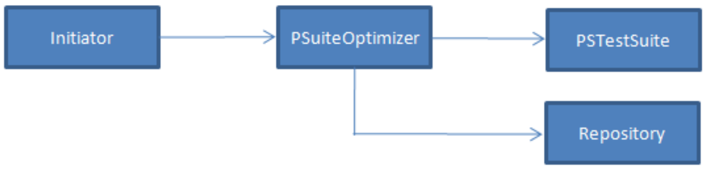

Fig. 2. Class diagram of proposed strategy

Table 3. Summary of Existing Combinatorial Testing Techniques

Technique	**Year of Origin**	**Approach**	**Description**
AETG	1997	Computational	Employs greedy algorithm uses random search algorithm; non-deterministic;.
SA	2004	Heuristic Search Technique	t-way test set is constructed from the initial test set by repeating modifications; Stochastic greedy algorithm; can produce smaller test sets than AETG and IPO; non-deterministic; time consuming.
GA	2004	Heuristic Search Technique	Genetic Algorithms (GAs) are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetic; Modification of AETG; non-deterministic; optimal test generation.
IPOG	2007	Computational	Generalization of IPO; Deterministic.
IPOG-D	2007	Computational	It combines the IPOG strategy with an algebraic recursive construction called D-construction: faster than IPOG; larger test set; Deterministic
MIPOG	2007	Computational	Variant algorithm of IPOG to address the issue of dependency : Deterministic
IRPS	2008	Computational	Efficient pairwise data generation strategy and data structure implementation to generate optimal pairwise test set.
IPOF	2008	Computational	Refinement of IPOG; Non-Deterministic.
G2Way	2008	Computational	Depends on two algorithms: the pair generation algorithm and the backtracking algorithm.
PITS	2009	Computational	Prioritized interaction test suite based on user importance
VSIT	2009	Computational	Interactions have variable strengths; greedy heuristic and proposed two concrete test generation strategies that were based on "one-test-at-a-time" and in-parameter-order strategy.
MC_MIPOG	2010	Computational	MC_MIPOG is a parallel t-way test generation strategy for multicore systems; Deterministic.

6 Conclusions and Future Work

In this paper, we propose and illustrate our effective approach to build optimal t-way interaction test suites over cloud using a novel approach particle swarm optimization technique with the software test case generation to gain near optimal solution. Our approach supports (t) equals up to 6-way consistent with the requirements as described by Kuhn et al [15]. Concerning future work, we are now looking to compare the performance of our approach particularly in terms of the test size with other strategies like IPOG with its tool FireEye, WHITCH, Jenny, TConfig, and TVG etc.

References

1. National Institute of Standards and Technology. The Economic Impacts of Inadequate Infrastructure for Software Testing. U.S. Department of Commerce (May 2002)
2. Berling, T., Runeson, P.: Efficient Evaluation of Multifactor Dependent System Performance Using Fractional Factorial Design. IEEE Transactions on Software Engineering 29(9), 769–781 (2003)

3. Kuhn, R., Reilly, M.: An investigation of the applicability of design of experiments to software testing. In: NASA Goddard/IEEE Software Engineering Workshop 2002, pp. 91–95 (2002)
4. Lei, Y., et al.: IPOG/IPOG-D: Efficient Test Generation for Multiway Combinatorial Testing. Software Testing, Verification and Reliability 18(3), 125–148 (2007)
5. Younis, M.I., Zamli, K.Z., Isa, N.A.M.: IRPS – an efficient test data generation strategy for pairwise testing. In: 12th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES), Zagreb, Croatia, pp. 493–500 (September 2008b)
6. Forbes, M., et al.: Refining the In-Parameter-Order Strategy for Constructing Covering Arrays. Journal of Reseach of the National Institute of Standards and Technology 113(5), 287–297 (2008)
7. Lei, Y., Carver, R.H., Kacker, R., Kung, D.: A combinatorial testing strategy for concurrent programs. Software Testing Verification and Reliability 17(4), 207–225 (2007)
8. Younis, M.I., Zamli, K.Z., Isa, N.A.M.: A strategy for grid based T-Way test data generation. In: Proceedings the 1st IEEE International Conference on Distributed Frameworks and Application (DFmA 2008), Penang, Malaysia, pp. 73–78 (October 2008)
9. Younis, M.I., Zamli, K.Z.: MC-MIPOG: A Parallel t-Way Test Generation Strategy for Multicore Systems. ETRI Journal 32, 73–82 (2010)
10. Younis, M., et al.: Assessing IRPS as an Efficient Pairwise Test Data Generation Strategy. Int. J. Advanced Intelligence Paradigms 2(1), 90–104 (2010)
11. Arshem, J.: TVG (September 23, 2011), `http://sourceforge.net/projects/tvg/`
12. Windisch, A., Wappler, S., Wegener, J.: Applying Particle Swarm Optimization to Software Testing. In: Proc. of the 2007 Conf. on Genetic and Evolutionary Computation GECCO 2007, London, England, United Kingdom, pp. 527–532 (2007)
13. Ganjali, A.: A Requirements-Based Partition Testing Framework Using Particle Swarm Optimization Technique. Master Thesis, University of Waterloo, Canada (2008)
14. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM 51(1), 107–113 (2008), `http://doi.acm.org/10.1145/1327452.1327492`, doi:10.1145/1327452.1327492
15. Kuhn, D.R., Wallace, D., Gallo, A.: Software fault interactions and implications for software testing. IEEE Transactions on Software Engineering 30(6), 418–421 (2004)
16. Williams, A.: TConfig (March 23, 2011), `http://www.site.uottawa.ca/'awilliam/`
17. Jenny, J.B. (March 23, 2011), `http://www.burtleburtle.net/bob/math/jenny.html`
18. Czerwonka, J.: Pairwise testing in real world: Practical extensions to test case generator. In: Proceedings of 24th Pacific Northwest Software Quality Conference, Portland, Oregon, USA, pp. 419–430 (October 2006)
19. Ahmed, B.S., Zamli, K.Z.: PSTG: A T-Way Strategy Adopting Particle Swarm Optimization, ams. In: 2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation, pp. 1–5 (2010)
20. Hartman, A., Klinger, T., Raskin, L.: IBM intelligent test case handler (March 23, 2011), `http://www.alphaworks.ibm.com/tech/whitch`
21. Weiss, A.: Computing in The Clouds. ACM Net. Worker, 16–25 (December 2007)