Sequence Generation of Test Case Using Pairwise Approach Methodology

Deepa Gupta, Ajay Rana and Sanjay Tyagi

Abstract There are various instruction specifications in the software system that interacts with each other. While designing software system, software engineers have to face a tedious task of ensuring the validation of every available software use case. If the instruction specifications are huge, the required number of use cases to validate becomes unmanageable within reasonable time budget and therefore designers have to choose one of the options of either delaying the projects or delivering without validating all available test case scenarios. Even for moderately sized software systems, comprehensive testing sometimes becomes impossible due to limited time availability for validation phase. Empirical studies have indicated that most of the faults in software systems often precipitate by interaction between smaller numbers of instruction specifications. If designers can cover all available interactions between all pairs of instructions specifications (instead of comprehensive testing), it can give reasonable confidence to designers about the software attribute within the limited time. This paper presents one such approach which uses the sequence origination approach for pairwise test case origination. This approach makes certain to propagate the required intent of trial run cases which cover all available interactions between all instructions pairs at least once. Trial run selection specification is this approach based on combinatorial testing. The paper also discusses the results with this new approach on one of the candidate software system to demonstrate its efficacy over already existing validation approaches.

Keywords Software testing · Trial run · Combinatorial Testing · Pairwise testing

D. Gupta (✉)
AIIT, Amity University, Noida, India
e-mail: deepa19july@gmail.com

A. Rana
Amity University, Noida, India
e-mail: ajay_rana@amity.edu

S. Tyagi
Kurukshetra University, Kurukshetra, India
e-mail: tyagikuk@gmail.com

S.K. Bhatia et al. (eds.), *Advances in Computer and Computational Sciences*,
Advances in Intelligent Systems and Computing 554,
https://doi.org/10.1007/978-981-10-3773-3_9

1 Introduction

Combinative testing (CT) is an exemplification approach which develop trial run that focuses on the comportment of interaction of orderliness components with their collaborators [1]. Inclination is established on the surveillance that utmost software flaws precipitate through orderliness of solely two aspects analogous instruction expenses. Its search group that is much smaller than that of comprehensive trial earlier was efficacious in findings flaws [2]. Pairwise testing is an efficacious, combinative testing approach that, for each amalgamation of instruction attribution to a software arrangement, tests all available amalgamation of this attribution [3].

In software engineering, software testing and removing flaws is still very labor-comprehensive and expensive. Roughly software testing is a paramount of project. Therefore, the aim is to search for a mechanized worth-efficacious software testing and unscramble scheme to check high attribute software release [4]. Forthwith analysis and inquisition on testing the software that targets on the test coverage criterion design, obvious error and test localization difficulty. Among these difficulties, test case origination difficulty is a valuable one and has come forward in producing error-free programs. Explain this, pairwise strategy, known for its productive test case reduction scheme and adaptability to notice from superlative per hundred of the faults, can be used. It is necessary to state that an efficacious way of finding any flawless solution is not established and the time required for finding a smallest instance of trial run grows very fast when the number of attribution and available expenses rises.

A genetic iterative is an approach that resembles the familiar rule of progression' [5]. The following algorithm was discovered for handling exploration and augmentation related disputes and is known to be efficacious for determining results to the disputes with the very massive search space and complex disputes. In genital algorithm, we tend to find a better answer from community of possible choice called entity to a complication which tends to find much better solutions.

In this paper, metaheuristic algorithms are applied and also the concept of genital algorithm concepts have been applied for generating pairwise test sets as a search dispute and this paper presents more details of sequence test origination using a pairwise approach. Here we have provided with the sequence origination method which combines the feature of NP—Complete dispute [3]; as well as sequence origination approach which reduces the number of trial run.

2 Metaheuristic Algorithms

A metaheuristic iterative process can be explained as a continual origination process that guides a dependent heuristic by combining smartly various conceptions to analyze and minimize the search space. To find nearly flawless solutions, information should be framed using informative ways [6]. The subsequent part explains the sequence generation algorithm:

2.1 N-IPO (Novel-IPO)

- An adjustment with many instruction particularization, the IPO hereditary develops an amalgamation test intent for the early dual attributes, stretches the developed test intent to develop a pairwise test intent being the early trio attributes and go on to do so for each descriptive attributes [3]. The N-IPO yields into attention the illustrative expenses of attribution authority instead of all the available expenses. During spanning the preceding test intent for over and above attribution dual steps supervenes:
- In the early step the current trial run are stretched inconsistently by adding the expense of recent attribution. The following is known as inconsistent extension.
- In the dual stepping the test intent is enhanced by summing more trial run, which may be mandatory due to inclusion of new attribution. This impression is the perpendicular extension.
  Antecedent partitions must be decided in the instruction domains and take out the illustrative expenses from each domain. To escape any kind of viewpoint we endorse the illustrative expense from each separation on irregular support.

2.2 Pairwise Testing

The above-mentioned approach uses the combinative testing approach in which one by one dual set of instruction attribution of software is being tried [1, 7]. It is observed as a feasible adjustment amidst combinative testing method. It can be implemented much rapidly than comprehensive testing that test an amalgamation for all instruction attribution is more efficacious than less comprehensive plans that fail to act on all available pairs of instruction attribution. The interpretation after this type of testing is the larger part of the software flaws which precipitate by the single instruction attribution. Pairwise testing thus depends upon that one by one dual instruction attribution conscience be occupied at slightly one time.

3 Combination Sequence Generation Algorithm (Proposed Algorithm)

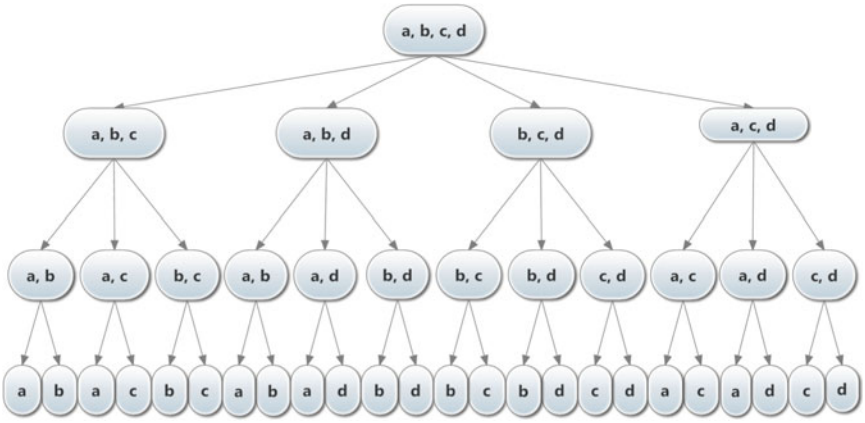
To begin with first we need to know how this optimized test sequence can be developed. For that we will use combination sequence origination algorithm. Then we will use this sequence generator in our test case origination algorithm to get the final outputs.

Combination Sequence Generation Algorithm

1. Let C be the array of elements of which Combinations are to be made.
2. Let M be the count of elements in C.
   Create a Multi-Way Tree with root node containing elements of C.
   Create all possible combination groups, when M − 1 elements are chosen from C.
   Make them the child nodes.
   Do steps 3–4 recursively in such a way that each parent has all possible combinations of child nodes with length one less than to itself, until each child node contains only one element.
3. Now Traverse this tree with depth first approach and generate the sequence.
4. Remove the duplicates from bottom to top, from the sequence.
5. Display sequence.

Example: C = [a, b, c, d].

M = 4 Tree Origination:



Traverse this tree with **depth first** approach and remove the duplicates from bottom to top.

3.1 Test Case Origination Algorithm

- This algorithm will develop the output containing test cases and their test results.
- The data structures to be maintained for this process:

• Name	• Description
• X • V • TV • xCount • S • N	• Set of parameters to be tested • Set of parameter's value domain for each item • Set of tested values • Number of items to be tested • Stack of parameter groups • Counter
• Output	• A table with three columns Group, Values, Result • Group = parameter group that has been tested • Values = values used for the test • Result = test result, Pass or Fail

Test Case Origination Algorithm

1. Input **X** with parameter names to be tested
2. Input **Y** with parameter value domains to be tested
3. **xCount** = count items in **X**
4. n = 0
5. n = n + 1
6. Select **Group** as first **n** parameters from **X**
7. Use **Combination Sequence Origination Algorithm** to generate combinations sequence of **Group** parameters and push in **S**, where combination is not present in **Output** [Group].
8. Select **testGroup** as pop(**S**).
9. Select **testParameterGroup** as **n** random values from **V** domain of each parameter in **testGroup**. If **V** has empty domains for any parameter then fetch random value from **TV** for that parameter.
10. Test **testGroup** with **testParameterGroup** and store in **Output**.
11. Add **testParameterGroup** to **TV**
12. Remove **testParameterGroup** values from V
13. If **S** is not empty go to step 8.
14. If **n** < **xCount** go to step 5.
15. Display Output.

4 Experimental Outcomes

A sequence origination algorithm which is based on this hereditary has been replaced in fraternization with intranet portal Nico minds Inc. [8]. The algorithm was used to propagate the pairwise intent of trial run during different development phases. The trial run was delivered to validation team which ran through trial run in parallel with comprehensive mode trial run to check their efficaciousness. The results indicated that pairwise trial run developed with this hereditary were able to figure paramount of the bugs found by comprehensive intent of trial run whereas the validation effort put in for validating pairwise trial run was much less as compared to comprehensive testing. The general findings were that error occurred due to interaction between **a** and **b** attributions. The proposed sequence propagation hereditary was quite successful in downsizing the number of developed trial run Here is a sample table which depicts the results for one of the small module of the software:

Group	Values	Result
A	False	Pass
a, b	**True, 4**	**Fail**
B	2	Pass
a, b, c	**True, 1, 0**	**Fail**
a, c	False, 5	Pass
C	8	Pass
b, c	3, 5	Pass

5 Discussion and Conclusion

The paper presented a sequence propagation hereditary for generating the pairwise trial run based upon the pairwise testing of the software arrangements having the instruction attributions considering enormous spheres. In summary, sequence origination algorithm

- embraces gains of IPO hereditary, i.e., uniform and perpendicular extension;
- embraces (Novel-IPO) boundary value analysis;
- embraces benediction of Fibonacci series, i.e., resembling ascending–descending path;
- embraces the benediction of tree traversal.

The presented approach was evaluated on a software project of intranet portal and results indicated that paramount faults were discovered with the intent of pairwise trial run developed in this case and number of trial run developed and tried were much less than comprehensive intent of trial run. This approach obtains the

advantage of segregating all the spheres with reasonable budget. Also the dispute was formulated using depth first searching traversing approach and applied the test case propagation algorithm to find pairwise test sets which helped the trial run intent to cover the most of the instructions attributions interactions. With the approach of testing, the tradeoff for the carefulness of trial run coverage can be optimized by a tester whereas we have finite basis of time and liability that are available.

Going forward, the proposed need to be evaluated on multiple trial run and need to be enhanced and customized for domain specific disputes. A multilayered model is also available where this approach is first used to develop the top level of trial run to develop very few set of trial run to check the overall sanity of software. When these small number of trial run are validated successfully and all the bugs which are found are fixed, this approach is again used to cover more instruction attributions and develop more trial run to given better coverage.

References

1. Cohen, D. M., Dalal, S. R., Fredman, M. L., and Patton, G. C., The AETG system: An Approach to Testing Based on Combinatorial Design, IEEE Transaction on Software Engineering, in 1997, 23 (7), 437–443.
2. Tai, K. C., Lei, Y.: A Test Generation Strategy for Pairwise Testing. IEEE Trans. On Software Engineering, in Jan 2002, 28(1) 3.
3. Lei, Y and Tai, K. C., In-Parameter-Order: A Test Generating Strategy for Pairwise Testing, IEEE Trans. on Software Engineering, in 2002, 28 (1), 1–3.
4. Dr. Mohammed Abdul Kashem, Mohammad Naderuzzaman: On An Enhanced Pairwise Search Approach for Generating Optimum Number of Test Data and Reduce Execution Time. Computer Engineering and Intelligent Systems http://www.iiste.org ISSN 2222–1719 (Paper) ISSN 2222–2863 (Online) Vol. 4, No.1, 2013.
5. M. Mitchell, On an Introduction to Hereditary Algorithm. The MIT Press, in 1999.
6. I. H. Osman and G. Laporte, On “Metaheuristic: A bibliography”, Ann. Oper. Res. vol. 63, (1996), pp. 513–623.
7. J. Czerwonka. Pairwise Testing, combinatorial test case generation in (2010, Dec.) [online]. Available: http://www.pairwise.org.
8. http://www.nicominds.com.