Volume 7, No. 7, Nov-Dec 2016
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

A Comparative Study of Clustering Deviations for Black-Box Regression Test-Selection Techniques

Dr. R. Beena MCA.,M.Phil., Ph.D.,
Head, Department of Computer Science
Kongunadu Arts and Science College
Coimbatore, India

Ms. S. Gomathi M.Sc(CT).,
Research Scholar, Department of Computer Science
Kongunadu Arts and Science College
Coimbatore, India

Abstract: Regression test-selection techniques decrease the cost of regression testing by choosing a separation of an existing test suite to use in retesting a customized program. Over the history, similarity based regression test-selection techniques have been described in the literature. This paper aims to present a comparative study of present techniques of clustering deviations in black-box regression testing under the data mining clustering and classification techniques that are in use in today's software engineering of verification and validation tasks. Number of comparative study has been performed to evaluate the performance of predictive accuracy on the test cases and the outcome discloses that Hierarchical Clustering (HC) and decision Tree outperforms having better performance other predictive methods like Simple *K*-means, Randomized algorithms, are not performing well.

Keywords: Clustering, Regression testing, Black-box, Genetic algorithm.

I. INTRODUCTION

Clustering is a data mining technique of grouping set of data objects into multiple groups or clusters so that objects within the cluster have high similarity, but are very dissimilar to objects in the other clusters. Dissimilarities and similarities are assessed based on the attribute values describing the objects. Clustering algorithms are used to organize data, categorize data, for data compression and model construction, for detection of outliers etc. Common approach for all clustering techniques is to find clusters centre that will represent each cluster. Cluster centre will represent with input vector can tell which cluster this vector belong to by measuring a similarity metric between input vector and all cluster centre and determining which cluster is nearest or most similar one [3].

Testing software is a very important and challenging activity. Nearly half of the software production development cost is spent on testing. The main objective of software testing with clustering approach is to eliminate as many errors as possible to ensure that the tested software meets an acceptable level of quality.

Regression testing is a highly important but time consuming activity [1]. A great deal of work has been performed on devising and evaluating techniques for selecting, minimizing, and prioritizing regression test cases [2]. Such techniques are necessary, but unfortunately not sufficient to help scale regression testing to large, complex systems. Indeed, in practice, even with efficient prioritization or selection, numerous regression test deviations may need to be analyzed to determine if they are due to a regression fault or simply the effect of a change. A problem that has been largely ignored so far, but which is highly important in practice, is how to cope with the many discrepancies (deviations) that can be observed when running regression test cases on a new version of a system.

Regression testing is performed when changes are made to existing software; the purpose of regression testing is to provide confidence that the newly introduced changes do not obstruct the behaviors of the existing, unchanged part of the software. It is a complex procedure that is all the more challenging because of some of the recent trends in software development paradigms. For example, the component based software development method tends to result in use of many black-box components, often adopted from a third-party. Any change in the third-party components may interfere with the rest of the software system, yet it is hard to perform regression testing because the internals of the third-party components are not known to their users.

Software systems and their environments change continuously. They are enhanced, corrected, and ported to new platforms. These changes can affect a system adversely, thus software engineers perform regression testing to ensure quality of the modified systems. Because regression testing is responsible for a significant percentage of the costs for software maintenance and because the maintenance costs often dominate total lifecycle costs [13], regression testing is one of the largest contributors to the overall cost of software. To

improve the cost effectiveness of regression testing techniques, many researchers have proposed and empirically studied various regression testing techniques, such as regression test selection (e.g., [14]), test suite minimization (e.g., [11]), and test case prioritization (e.g., [12]).

Requirements-Tests linking resolution obtain the clusters of requirements, to utilize the requirement-test cases traceability matrix to collect test cases that are associated with each requirement cluster. Figure 1 reviews the process. The two ovals on the left side represent the clusters of requirements. For instance, cluster 1 contains requirements 1, 3, and 4. The figure represents the requirement-tests traceability matrix. There are two cases (TC1 and TC2) associated with requirement 1. The requirements-tests mapping resolution process obtains the clusters of test cases (the ovals on the right side of the figure) by reading the requirements in the clusters and identifying their corresponding test cases from the matrix. For instance, cluster 1 on the right side contains five test cases (1, 2, 6, 7, and 8) that are associated with requirements 1, 3, and 4.

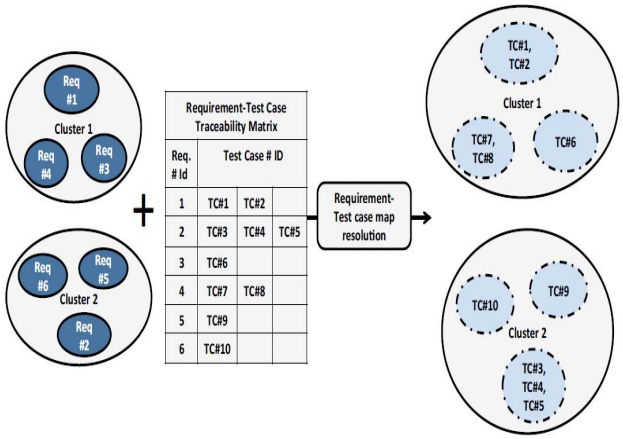

Figure 1: Requirement-tests mapping resolution

In this paper, we explore strategies for feature selection regression test cases based on class regression decision tree models. Such representations have been usually used to partition the input domain of the system being tested [3], which in turn is used to choose and create system test cases so as to attain certain strategies for partition coverage. Such models are widely applied for black-box system testing for database applications and is therefore a natural and practical choice in our context.

II. RELATED WORK

S. Yoo, M. HarmanIn [2] discussed a survey about Regression testing activity that is performed to provide confidence that changes do not harm the existing behavior of the software. Test suites tend to grow in size as software evolve, often making it too costly to execute entire test suites. A number of different approaches have been studied to maximize the value of the accrued test suite: minimization, selection and prioritization.

E. Rogstad, L. Briand, E. Arisholm, R. Dalberg, and M. Rynning [4] presented a practical approach and tool (DART) for functional black-box regression testing of complex legacy database applications. Such applications are important to many organizations, but are often difficult to change and consequently prone to regression faults during maintenance. They also tend to be built without particular considerations for testability and can be hard to control and observe. This approach is to fully integrate DART with the daily test operation of the project, and ideally as a continuous part of the development process, as a means for early fault detection.

E. Rogstad and L. Briand [5] proposed an approach for selecting regression test cases in the context of large-scale database applications. We focus on a black-box (specification-based) approach, relying on classification tree models to model the input domain of the system under test (SUT), in order to obtain a more practical and scalable solution. We perform an experiment in an industrial setting where the SUT is a large database application in Norway's tax department. The authors compared both fault detection rate and selection execution time. In general random selection is superior to similarity-based selection in terms of selection execution time. However, the difference for smaller sample sizes in the range of interest is less than a few minutes (i.e., 39 s when selecting 30% of the test suite when comparing similarity partition-based with random selection).

A. Arcuri and L. Briand [6] discussed features a systematic review regarding recent publications in 2009 and 2010 showing that, overall, empirical analyses involving randomized algorithms in software engineering tend to not properly account for the random nature of these algorithms. Many of the novel techniques presented clearly appear promising, but the lack of soundness in their empirical evaluations casts unfortunate doubts on their actual usefulness. In software engineering, though there are guidelines on how to carry out empirical analyses involving human subjects, those guidelines are not directly and fully applicable to randomized algorithms.

C. Zhang, Z. Chen, Z. Zhao, S. Yan, J. Zhang, and B. Xu [7] proposed a new regression test selection technique by clustering the execution profiles of modification traversing test cases. Cluster analysis can group program executions that have similar features, so that program behaviors can be well understood and test cases can be selected in a proper way to reduce the test suite effectively. This technique effectively deals with the trade-offs between test suite reduction and fault detection capability, performing better on large programs.

S. Chen, Z. Chen, Z. Zhao, B. Xu, and Y. Feng [8] discussed a semi-supervised clustering method, namely semi-supervised Kmeans (SSKM), is introduced to improve cluster test selection. SSKM uses limited supervision in the form of pairwise constraints: Must-link and Cannot-link. These pairwise constraints are derived from previous test results to improve clustering results as well as test selection results. The experiment results illustrate the effectiveness of cluster test selection methods with SSKM. Two useful observations are made by analysis. (1) Cluster test selection with SSKM has a better effectiveness when the failed tests are in a medium proportion. (2) A strict definition of pairwise constraint can improve the effectiveness of cluster test selection with SSKM. Although the authors found some observations on different definitions of Must-link and Cannot-link, it may be not sufficient in other applications.

P. G. Sapna and H. Mohanty [9] analyzed clustering is used to select a subset of scenarios for testing. First, a distance matrix is obtained by using Levenshtein distance to compare scenarios. This distance matrix is used as input for the Agglomerative Hierarchical Clustering (AHC) technique with the objective of selecting dissimilar test scenarios and at the same time achieving maximum coverage and rate of fault detection. Distance measure between scenarios obtained from UML activity diagrams, calculated using Levenshtein distance was used as the basis for clustering.

Yue Liu, Kang Wang, Wang Wei, Bofeng Zhang, Hailin Zhong [10] discussed web application test cases optimization based on clustering is researched, and a novel method named USCHC (User Sessions Clustering based on Hierarchical Clustering algorithm for test cases optimization) is proposed. This method firstly gives the function to calculate the distance between the user sessions, and then employs the bottom-up agglutinate hierarchical clustering algorithm to cluster the initial testing cases and produces different kinds of test suites. The work of testing web applications based on mining user sessions is a complex systematic project. It is not an easy thing to get an effective and practical tool.

J. Jones and M. Harrold [11] discussed the software testing is particularly expensive for developers of high-assurance software, such as software that is produced for commercial airborne systems. One reason for this expense is the Federal Aviation Administration's requirement that test suites be modified condition/decision coverage (MC/DC) adequate. Despite its cost, there is evidence that MC/DC is an effective verification technique, and can help to uncover safety faults. As the software is modified and new test cases are added to the test suite, the test suite grows, and the cost of regression testing increases. To address the test-suite size problem, researchers have investigated the use of test-suite reduction algorithms, which identify a reduced test suite that provides the same coverage of the software, according to some criterion, as the original test suite, and test-suite prioritization algorithms, which identify an ordering of the test cases in the test suite according to some criteria or goals.

G. Rothermel, R. Untch, C. Chu, and M. J. Harrold [12] illustrated the test case prioritization techniques schedule test cases for execution in an order that attempts to increase their effectiveness at meeting some performance goal. Various goals are possible; one involves rate of fault detection, a measure of how quickly faults are detected within the testing process. The authors described several techniques for using test execution information to prioritize test cases for regression testing, including: 1) techniques that order test cases based on their total coverage of code components; 2) techniques that order test cases based on their coverage of code components not previously covered; and 3) techniques that order test cases based on their estimated ability to reveal faults in the code components that they cover.

G. Rothermel and M. J. Harrold [14] proposed a regression testing is a necessary but expensive maintenance activity aimed at showing that code has not been adversely affected by changes. Regression test selection techniques reuse tests from an existing test suite to test a modified program. Many regression test selection techniques have been proposed, however, it is difficult to compare and evaluate these techniques because they have different goals. This paper outlines the issues relevant to regression test selection techniques, and uses these issues as the basis for a framework within which to evaluate the techniques.

III. COMPARISON ANALYSIS

This paper aims to collect and consider papers that deal with different regression testing techniques. Our objective is not to undertake a logical review, but quite to provide a broad state-of-the-art view on these related fields. Many different approaches have been projected to assist regression testing, which has mentioned in a body of literature that is spread over a wide variety of fields and periodical locations. The majority of comparison study has been available in the software engineering domain, and particularly in the software testing and software maintenance literature. However, the regression testing literature also overlaps with those of programming language analysis, empirical software engineering and software metrics.

Table 1: SUMMARY TABLE FOR COMPARISON OF CLUSTERING DEVIATIONS FOR BLACK BOX REGRESSION TECHNIQUES

Title	Algorithm	Key-Idea	Techniques	Results	Performance
Industrial Experiences with Automated Regression Testing of a Legacy Database Application [2]	Trigger generation in DART	Black-box regression testing	Classification tree models	Good Fault Detection in multiple Norwegian Tax Accounting System	To identified 60 % additional faults
Test case selection for black-box regression testing of database applications [5]	Similarity-based Selection algorithm	System under test (SUT) is a large database application in Norway's tax department.	classification tree models	To be preferred when faults are located in partitions containing a large number of test cases.	73% of the faults were located in small partitions whereas 27% of the faults were located in larger partitions.
A Hitchhiker's Guide to Statistical Tests for Assessing Randomized Algorithms in Software Engineering [6]	Randomized algorithms (e.g., Genetic Algorithms)	To focus on software verification and validation.	Genetic search optimization model	To predict a problems with a particular focus on software verification and validation.	In 27 cases out of 54 show at least 10 runs.
An Improved Regression Test Selection Technique by Clustering Execution Profiles [7]	Simple K-means algorithm.	To improve the efficiency of regression testing many test selection techniques.	Cluster Selection technique	Clustering processing step by 20 times for each modified version and analyzed the results statistically.	The technique could statistically select a large part of fault-revealing test cases, more than 80%.
Using Semi-Supervised Clustering to Improve Regression Test Selection Techniques [8]	Semi-supervised dimensionality reduction Algorithm (SSDR) Algorithm	To predict the pair-wise constraints: Must-link and Cannot-link	Regression Test Selection Technique	The semi-supervised K-means (SSKM) and improve the clustering results for Flex and Space.	The tailed test and set the scaling term is the value of F-measure with confidence level is 0.95.
Clustering Test Cases to achieve Effective Test Selection [9]	Agglomerative Hierarchical Clustering(AHC) algorithm	Levenshtein distance measure	Agglomerative Hierarchical Clustering(AHC) technique	The two test suites was done according to priority, random and AHC based selection of scenarios.	To assuming the constraint that 50% and 60% of test scenarios need to be selected for the test suites, respectively.

IV. CONCLUSION

In this paper presents an comparative study the various clustering techniques deviations for Regression testing discussed with the different categories in which algorithms can be classified (i.e., SSDR, AHC, Simple K-means, Genetic algorithm, similarity selection algorithm based). We concluded the discussion on clustering algorithms with regression testing by a comparative study with black-box regression category. We have also discussed the concept of Similarity measures which proves to be the most important criteria for regression clustering.

The further work enhanced and expanded for the automation of clustering deviations for black-box regression testing using *Hierarchical logical classification* (HLC) algorithm.

REFERENCES

[1] M. Harrold and A. Orso, "Retesting software during development and maintenance," in Proc. Frontiers of Software Maintenance, 2008 (FoSM 2008), 2008, pp. 99–108.

[2] S. Yoo and M. Harman, "Regression testing minimisation, selection and prioritisation: A survey," Softw. Test., Verif., Rel., vol. 22, no. 2, pp. 67–120, Mar. 2012.

[3] Manish Verma, Mauly Srivastava, Neha Chack, Atul Kumar Diswar, Nidhi Gupta," A Comparative Study of Various Clustering Algorithms in Data Mining," International Journal of Engineering Reserch and Applications (IJERA),Vol. 2, Issue 3, pp.1379-1384, 2012.

[4] E. Rogstad, L. Briand, E. Arisholm, R. Dalberg, and M. Rynning, "Industrial experiences with automated regression testing of a legacy database application," in Proc. 27th IEEE Int. Conf. Software Maintenance (ICSM), Sep. 2011, pp. 362–371.

[5] E. Rogstad and L. Briand, "Test case selection for black-box regression testing of database applications," Inf. Softw. Technol. (IST), vol. 31, no. 6, pp. 676–686, Jun. 2013.

[6] A. Arcuri and L. Briand, "A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering," in Proc. 33rd Int. Conf. Software Engineering (ICSE), May 2011, pp. 1–10.

[7] C. Zhang, Z. Chen, Z. Zhao, S. Yan, J. Zhang, and B. Xu, "An improved regression test selection technique by clustering execution profiles," in Proc. 2010 10th Int. Conf. Quality Software (QSIC), 2010, pp. 171–179.

[8] S. Chen, Z. Chen, Z. Zhao, B. Xu, and Y. Feng, "Using semi-supervised clustering to improve regression test selection techniques," in Proc. 2011 IEEE 4th Int. Conf. Software Testing, Verification and Validation (ICST), 2011, pp. 1–10.

[9] P. G. Sapna and H. Mohanty, "Clustering test cases to achieve effective test selection," in Proc. 1st Amrita ACM-W Celebration on Women in Computing in India Ser. A2CWiC'10., New York, NY, USA, 2010, pp. 15:1–15:8.

[10] Y. Liu, K. Wang, W. Wei, B. Zhang, and H. Zhong, "User-session-based test cases optimization method based on agglutinate hierarchy clustering," in Proc. 2011 Int. Conf. Internet of Things and 4th Int. Conf. Cyber, Physical and Social Computing, Ser. ITHINGSCPSCOM'11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 413–418.

[11] J. Jones and M. Harrold, "Test suite reduction and prioritization for modified condition/decision coverage," IEEE TSE, vol. 29, no. 3, pp. 193–209, 2003.

[12] G. Rothermel, R. Untch, C. Chu, and M. J. Harrold, "Prioritizing test cases for regression testing," IEEE TSE, vol. 27, no. 10, pp. 929–948, Oct. 2001.

[13] B. Beizer, Black-Box Testing. New York, NY: John Wiley and Sons, 1995.

[14] G. Rothermel and M. J. Harrold, "Analyzing regression test selection techniques," IEEE TSE, vol. 22, no. 8, pp. 529–551, Aug. 1996.