CURRENT TRENDS STRATEGIES IN THE TEST REDUNDANCY REDUCTION

[1]NorasyikinSafieny, [2] Kamal Z. Zamli, [3]Hasneeza L. Zakaria

[1,2]Faculty of Computer System and Software Engineering,
Universiti Malaysia Pahang (UMP),
Kampus Gambang, 26300 Kuantan, Pahang,
Malaysia
nsasyikins@gmail.com , kamalz@ump.edu.my

[3] School of Computer and Communication Engineering,
Universiti Malaysia Perlis (UniMAP),
Kampus Pauh Putra, 02600 Arau, Perlis,
Malaysia
hasneeza@unimap.edu.my

***Abstract* - Software testing is most important and expensive part in software development process. The failures of software can lead disastrous consequence, such as loss of data, fortune and lives. This process usually expensive, and the key of expensiveness of testing is that typically take a long time to execute the whole set of test cases. Test case minimization technique generates a representative set from the original test suite that satisfy all the requirements as an original test suite but contains less number of test cases. Redundant test cases are removed from the test suite. Many strategies in a greedy approach have been developed (including GE, GRE, and HGS), Formal Concept Analysis and non-greedy approach (tReductLAHC, tReductSA) using Metaheuristic Algorithm. The non-greedy approach is more effective compared to greedy approach. In this paper, a review of the strategies is provided to investigate the current trends in the test reduction research area. We can categorize these strategies as Greedy and Metaheuristic Approach. To enhance the performance by using a metaheuristic algorithm, we are proposing our work with adopts the sequence permutation and hybridization strategies.**

Keywords: Test suite redundancy reduction. Search based software engineering. Global Neighborhood Algorithm.Optimization. Simulated Annealing

I. INTRODUCTION

Test redundancy problem has been regarded as the NP complete (Yoo & Harman, 2012) that no single strategy can do well in all scenarios considered. Our work adopts sequence permutation and hybridization between two metaheuristics; a population-based and single-based solution with systematic merging rules technique. The growing complexity of the software makes the cost to test the software increase. Software is generally tested through test cases and it's defined in IEEE standard as "A set of test inputs, execution, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirements". A test suite consists of all the test cases that satisfy all the testing requirements.

In the literature, many strategies have been addressed to cater these issues. Non-greedy approaches perform well with existing works. The technique to cater the problem of selecting a representative set of test cases that provides the desired testing coverage focuses into two categorized as :

- Greedy approach
- Metaheuristic Algorithm

Greedy Approach including GE, GRE, and HGS. Strategies based on Requirement Cardinality, Essential, Weighted set covering and 1-to-1 redundant test concept. Formal concept analysis classifying objects based upon the overlap among their attributes. Metaheuristic Algorithm based approach uses Late Acceptance Hill Climbing and Simulated Annealing that come from Single Based solution. The objective value to find a minimal set of test requirements.

The rest paper organized as follows section II contains a review on existing technique. Section III discusses about greedy approach and metaheuristic algorithm trends. Finally section IV concluding remark.

II. RELATED WORKS

Software is tested with test cases. However, running all of the test cases in a test suite may take a great deal of effort. The test suite minimization problem (Zhong, Zhang, & Mei, 2008) can be formally stated as follows. Given:

i. A test suite T of test cases $\{t_1, t_2, t_3, .t_n\}$.

ii. A set of testing requirements $\{ r_1, r_2, r_3, ..r_n \}$ that must be satisfied to provide the desired testing coverage of the program.

iii. Subsets $\{ T_1, T_2, T_3, ..T_n \}$ of T, one associated with each of the r_iS, such that any one of the test cases t_js belonging to T_i satisfies r_i.

Table 2.1 Test cases in a test suite.

Test Case	**r_1**	**r_2**	**r_3**	**r_4**	**r_5**	**r_6**	**r_7**	**r_8**
t_1:	X		X			X		
t_2:		X		X	X			X
t_3:		X	X			X		
t_4:		X	X		X		X	
t_5:		X		X	X		X	

Problem: find a minimal cardinality subset of T that is capable of exercising all r_is exercised by the minimized test suite T.

Table 1 is an example that shows the test cases in a test suite $\{t_1, t_2, t_3, t_4, t_5\}$. The symbol X means satisfaction of a requirement by a test case. Here we find that a subset $\{t_1, t_2, t_4\}$ of test suite is enough to cover all the requirements $\{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8\}$ while test cases t_2 and t_5 become redundant since the requirements covered by them are satisfied by the other three test cases. The statement shows the basic situation happened in test redundancy reduction.

The earliest work on the Greedy Heuristic strategy is highlighted by Chavatal (Chvatal, 1979). The greedy heuristic will first select the test case t_1, and throw out the requirements r_1, r_3 and r_6 from further consideration. Next, either of t_2, t_3, t_4 or t_5 could be picked since each of these cover one yet uncovered requirement. The redundant test case t_1 was selected because the decision to select t_1 was made too early. The choice of picking t_1 before picking any of the other test cases seemed a good decision at the time when t_1 was selected; however, it turned out to be not the best choice for computing the overall minimal test suite. Consider example in Table 2.1.

Complementing Chavatal's work, Harrold et al. develops a similar strategy, called HGS (Harrold, Gupta, & Soffa, 1993; Ho & Su, 2012). Harrold propose the heuristic algorithm H to reduce the size of the test cases uses the essentialness and first group the test requirement then repeatedly reduces the test cases and finally remove the redundant test case in the test requirement. HGS greedily ranks the cardinality of each requirement with the corresponding test case (from low to high) as the main basis for reduction. HGS works as follows. For each requirement that is exercised by one test case that is cardinality of 1, and HGS adds the test case into the minimized test suite and covered the requirements. Let's consider in Table 2.2 shows the requirements exercised by that test case table by Tallam (Tallam, 2005). $T_1=\{t_1,t_2\}$ cardinality one, $T_2=\{t_1,t_3\}$cardinality two, $T_3=\{t_2,t_3,t_4\}$cardinality three, $T_4=\{t_3,t_4,t_5\}$cardinality three,$T_5=\{t_2,t_6, t_7\}$cardinality three. There is no T_i of cardinality one, the HGS considers $T_1=\{t_1,t_2\}$ and selects the test case t_1. Next T_3,T_4 and T_5 are considered. The tie between T_3, is broken arbitrarily say in favour of selecting the test case t_2. Now only T_4 in $r_4$is still remains to be exercised because T_4 is still unmarked. So any $T_4=\{t_3,t_4,t_5\}$ can be selected at this stage. If we select t_3, thus the reduced test suite selected by the HGS heuristic is $\{t_1,t_2,t_3\}$. However, redundant test case still happen in requirements exercise by t_1,t_2, and t_3. This redundant test case was selected because the decision to select t_1 was made early.

Table 2.2 HGS study case

Test Case	**r_1**	**r_2**	**r_3**	**r_4**	**r_5**
t_1:	X	X			
t_2:	X		X		X
t_3:		X	X	X	
t_4:			X	X	
t_5:				X	
t_6:					X
t_7:					X

Although, the main strength of HGS is the fact that it creates a subtle (and stable) prioritization of test cases during its selection process (i.e. based on cardinality). Here, hard to cover requirement with low cardinality are considered first and followed by other requirements in order of increasing cardinality. The main limitation of this approach is the fact that, in real testing endeavour, prioritization is not solely a function of cardinality. In fact, prioritization can also be a function of likelihood of faults as well as their impacts.

Lau and Chen introduce another variant of greedy strategy, called GE (T. Chen & Lau, 1995). GE proposed a concept of essential for greedy selection. GE works based on essential and the best test cases that cover the most requirements. Firstly, identify the essential test cases. From the Table 2.3 we can see that r_6 is $t_{essential}$ that uncovered by any test case. Keep track on the uncovered requirements. Secondly, pick the best test cases that cover the most requirements. Only t_2 and t_4 that cover r_5 and t_1,t_2 and t_4 cover r_1. The best choice between t_2 and t_4 where is, the test cases cover r_1 and r_5. Based on second step, pick the best test case. The best test case is t_2, because t_2 cover = 3 reqs and t_4 only cover = 2 reqs. Iterate the step until all requirements covered. The reduced test suite selected by the GE heuristic is $\{t_2, t_3\}$.

Table 2.3 GE study case

Test Case	**r_1**	**r_2**	**r_3**	**r_4**	**r_5**	**r_6**
t_1:	X	X		X		
t_2:	X	X			X	
t_3:		X	X	X		X
t_4:	X				X	

Implementation wise, GE is straight forward to implement as compared to HGS. Furthermore, as GE considers $t_{essential}$ before greedily selecting candidate test case, the test suite size offered by GE is at least the same of better than that of Chavatal. The same argument cannot be applicable when comparing HGS and GE. On the negative note, GE does not address prioritization issue.

As enhancement of GE, Chen and Lau later introduce the GRE strategy (T. Y. Chen & Lau, 1998). GRE exploits the idea of redundant test case. It's based on three strategies; the essential strategy (the strategy of selecting all essential test cases), redundancy strategy (the strategy of removing 1-to-1 redundant test cases) , and greedy strategy (the strategy of selecting test cases that meet a maximum number of requirements that are not yet satisfied). In this case, if a test case satisfies only a subset of test -case requirements satisfied by another test case, then that particular test case is redundant. GRE starts by first removing redundant test cases from the test suite. In the process, GRE reduces the test suite and may make some test cases essential. Then, GRE applies the same algorithm as GE in order to choose the test cases that cover all the requirements. GRE inherit many advantages of GE. In fact, in the absence of redundant test case, GRE behaves much like GE. Interestingly, due to NP completeness of the test redundancy reduction problem, the performance of GE can still be better than GRE or even HGS in terms of test reduction. Similar to GE, GRE does not address the prioritization issue.

Shengwei adopts a strategy similar to GE (Xu, Miao, & Gao, 2012). Unlike GE, they exploits weighted set covering (for requirements) in order to eliminate test redundancy and prioritize the test suite according to cost order. The general performance of the algorithm appear the same to that of GE. On the negative note, although important, prioritization need not be considered merely on cost but on how effective of the

tests being prioritized. As highlighted earlier, prioritization can also be a function of likelihood of faults as well as their impacts.

Galeebathullah and Indumathi develop a strategy that combines the set theory and Greedy heuristics (B.Galeebathullah & C.P.Indumathi, 2010) . Initially, the strategy finds the intersection of each requirement with other requirements. If exist any intersection exist, the test cases are greedily combined and added to the final test suite. The process is repeated until all requirements are covered by the test case. In the work, prioritization issues are not reported. Additionally, no benchmarking result against other existing strategies is published.

Apart from the greedy heuristic approach, a number of researchers (Tallam, 2005)have started to adopt the Formal Concept Analysis (FCA). Basically, FCA is a technique for classifying objects based upon the overlap among their attributes. For reduction, test cases are considered as objects and requirements as attributes. Relationship between objects and attributes corresponds to the coverage information of test case. Using concept analysis, maximum grouping of objects and attributes can be deduced (termed context) in a table. Here, facilitated by graphical concept lattice and based on the object and attribute reduction rules, objects (i.e. Test cases) can be systematically reduced. Although helpful, FCA suffers from the problem of scale when the formal objects and their attributes grew, it is almost impossible to construct and manipulate the concept lattice graphically. Hence, the applications of FCA for large scale test reduction (and prioritization) can be problematic and difficult.

Many existing works on variants of Greedy approach and some based on Formal Concept Analysis, but it's not necessarily the best. The next strategies adopt sequence permutation and optimization algorithm based on SA with systematic merging technique (K. Z. Zamli, Mohd Hassin, Al-Kazemi, & Naseer, 2014a) and also based on Late Acceptance Hill Climbing (LAHC) (Kamal Zuhairi Zamli, 2014). Both of them based on single-based solution algorithms. Late Acceptamce Hill Climbing useful in terms of systematically sampling of the appropriate test case, existing strategies have not sufficiently dealt with test prioritization. Addressing that issues, this work a novel approach od adopting Late Acceptance Hill Climbing based Strategy for test redundancy reduction and prioritization. When dealing with large line of codes (LOCs), there are potentially issues of redundancies, Simulated Annealing based strategy are build for counter that issues. This works adopts the random sequence permutation and merging rules technique based on single based solution metaheuristic called Simulated Annealing. Simulated Annealing is an optimization algorithm motivated by the metal annealing process. The metal heated slowly cooled into the uniform structure. It's start with an initial configuration obtained by random search and the annealing makes a sequence of small random perturbation. All the possible improve solution is always accepted.

In our work, we proposed a strategy by used sequence permutation with hybridization between single-based solution and population-based metaheuristic algorithm.

III. GREEDY APPROACH AND METAHEURISTIC ALGORITHM: TRENDS

One system of about 20,000 line of code requires seven weeks to run all its test cases (Rothermel, Untch, Chu, & Harrold, 2001). Tester engineers are under pressure to test more and more codes. The highest percentage of test cases can save cost, time, and resources. Most related works on Greedy Heuristic, Formal Concept approach and Table 3. 1 shows the different ship with existing work including non-greedy approaches.

Although many technique have been addressed in the literature based on greedy and non-greedy approach, As the test redundancy problem has been regarded as NP complete problem (Yoo & Harman, 2012), no single strategies can do well in all scenarios considered.

i. Greedy approach more based on the earlier selection of test cases
ii. Sequence Permutation and Merging Rules with Metaheuristic Algorithm provide a new diversified solution in test redundancy reduction area.

As part of our research, we try to enhance work from Zamli 2014 (K. Z. Zamli, Mohd Hassin, Al-Kazemi, & Naseer, 2014b) with is we use Metaheuristic Algorithm based on hybridization between single based and population based solution to see the variation of diversified solution for test redundancy reduction.

According to Talbi 2013, the best result found for many real life of classical optimization problem are obtained by hybrid algorithm (Talbi, 2013). Hybrid actually combinations of algorithm such as metaheuristic, mathematical programming, constraint programming and machine learning technique that provides a powerful search algorithm. Two competing goals govern the design of a metaheuristic there is exploitation and exploration. The proposed technique will work to find the optimal value among these local optima by switching between exploration and exploitation. Single based solution is powerful in the exploitation of the solution found and weak in the exploration of the search space.

Table 3.1 Quick Review

Name of Strategies / The strategies work	**Greedy Heuristic** (Chvatal, 1979)	**HGS** (Harrold et al., 1993)	**GE** (T. Chen & Lau, 1995)	**GRE** (Selvakumar, Dinesh, Dhineshkumar, & Ramaraj, 2010)	**Set Theory and Greedy** (Galeebathullah & Indumathi, 2010)	**WSC** (Xu et al., 2012)	**FCA** (Tallam, 2005)	**tReductLAHC** (Kamal Zuhairi Zamli, 2014)	**tReductSA** (K. Z. Zamli, Mohd Hassin, Al-Kazemi, & Naseer, 2014a)
Greedy Approach									
Greedy selection	✓	✓	✓	✓	✓	✓			
Requirement Cardinality		✓							
Essential			✓	✓		✓			
Weighted set covering (priority)						✓			
1-to-1 redundant test concept				✓					
Intersection of the one requirements to another requirements					✓				
							Formal Concept Analysis		
Classifying objects based upon the overlap among their attributes							✓		
								Metaheuristic Algorithm	
Sequence Permutation								✓	✓
Merging Rules								✓	✓
Single-based solution								✓	✓

IV. CONCLUDING REMARK

This paper presents the brief summary of techniques that has been proposed in literature for test redundancy reduction. Most of the studies in the literature review are based on Greedy, Formal Concept Analysis and Metaheuristic Algorithm. Many of them generate significant reduction and each technique has strength to another in some aspect, but it is harder to tell which one performs the best. Metaheuristic Algorithm produced more diversified solutions. It's applied the concepts of single based solution and sequence permutation with merging rule technique. Evolution area of metaheuristic show the hybridization produce better results in optimal ways. Hybridization techniques in metaheuristic algorithm were shown in this paper to enhance the performance of metaheuristic algorithms in test redundancy reduction area.

REFERENCES

B.Galeebathullah, & C.P.Indumathi. (2010). A Novel Approach for Controlling a Size of a Test Suite with Simple Technique. *IJCSE) International Journal on Computer Science and Engineering*

Vol. 02, No. 03, 2010, 614-618.

Chen, T., & Lau, M. (1995). *Heuristics towards the optimization of the size of a test suite.* Paper presented at the Proceedings of the 3rd international conference on software quality management.

Chen, T. Y., & Lau, M. F. (1998). A new heuristic for test suite reduction. *Information and Software Technology, 40*(5), 347-354.

Chvatal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of operations research, 4*(3), 233-235.

Harrold, M. J., Gupta, R., & Soffa, M. L. (1993). A methodology for controlling the size of a test suite. *ACM Transactions on Software Engineering and Methodology (TOSEM), 2*(3), 270-285.

Ho, Y. C., & Su, C. C. (2012). A 0.1-0.3 V 40-123 fJ/bit/ch On-Chip Data Link With ISI-Suppressed Bootstrapped Repeaters. *Ieee Journal of Solid-State Circuits, 47*(5), 1242-1251. doi: 10.1109/Jssc.2012.2186722

Talbi, E.-G. (2013). Combining metaheuristics with mathematical programming, constraint programming and machine learning. *4OR, 11*(2), 101-150.

Tallam, S., Gupta, N.:. (2005). A Concept Analysis Inspired Greedy Algorithm for Test Suite Minimization. *6th ACM SIGPLAN-SINGSOFT Workshop on Program Analysis for Software Tools and Engineering, Lisbon, pORTUGAL (2005), ACM1-59593-239-9/05/0009*.

Xu, S., Miao, H., & Gao, H. (2012). Test Suite Reduction Using Weighted Set Covering Techniques. 307-312. doi: 10.1109/snpd.2012.87

Yoo, S., & Harman, M. (2012). Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability, 22*(2), 67-120.

Zamli, K. Z. (2014). *Late Acceptance Hill Climbing Based Strategy for Test Redundancy Reduction and Prioritization.* Paper presented at the Malaysia University Conference Engineering Technology.

Zamli, K. Z., Mohd Hassin, M. H., Al-Kazemi, B., & Naseer, A. (2014a). Simulated Annealing Based Strategy for Test Redundancy Reduction. In H. Fujita, A. Selamat, & H. Haron (Eds.), *New Trends in Software Methodologies, Tools and Techniques* (Vol. 265, pp. 818-832).

Zamli, K. Z., Mohd Hassin, M. H., Al-Kazemi, B., & Naseer, A. (2014b). Simulated Annealing Based Strategy for Test Redundancy Reduction. *New Trends in Software Methodologies, Tools and Techniques, 265*, 818-832. doi: 10.3233/978-1-61499-434-3-818

Zhong, H., Zhang, L., & Mei, H. (2008). An experimental study of four typical test suite reduction techniques. *Information and Software Technology, 50*(6), 534-546. doi: 10.1016/j.infsof.2007.06.003