IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

Diversity Based Genetic Algorithm for Efficient Test Case Selection

A. Ramarajan[1], S. Usha[2]

P.G Scholar, Department of Computer Science and Engineering, University College of Engineering, Anna University (BIT CAMPUS), Tiruchirappalli, Tamilnadu, India[1]

Assistant Professor, Department of Computer Science and Engineering, University College of Engineering, Anna University (BIT CAMPUS), Tiruchirappalli, Tamilnadu, India[2]

Abstract: The testing is a very important innovate the event of the software system cycle. The test cases are generated multiple test suite. The regression testing is done to reduce the effort of testing by selecting a subset of test cases from the test suite with respect to some testing criteria. Combination of Greedy and multi-objective genetic algorithms (MOGAs) does not produce better results. The greedy algorithm which is used to find optimal solution reduces the time but increases cost. Genetic also find next generation of test case selection from the test suite. Hence by injecting the new diversity based genetic algorithm (DIV-GA) during the search process a better solution is provided for the detection of test cases. A way to reduce the cost of regression testing consists of selection or prioritizing subset of test cases. Therefore by selecting and prioritizing the subset of test cases and given as input to DIV-GA reduce the time & cost estimation of efficient test cases.

Keywords: Regression Testing; Greedy Algorithm; Multi-Objective genetic algorithm (MOGA); Diversity Based Genetic Algorithm (DIV-GA); Test case selection.

I. INTRODUCTION

The test case may be a set of conditions utilized by the software system tester to examine the correct operating conditions of the developed software system. The major classifications of the test cases are formal and informal test cases. The test suite contains the elaborated description about the test case of the particular program. Some test suites will take hours, even days; therefore developers cannot exercise the system instantly or in cheap time [15]. The problem is clearly applied by the expansion of the test suites because the system evolves. many methods are planned to reduce the hassle of regression testing by choosing a (possibly minimal) set of test cases from the test suite with relation to some testing criteria [1], [2], [4]. In general resolution these issues needs the tester selecting some testing criteria to be satisfied, and using an optimization technique (e.g., greedy or search-based algorithm) to select/order the test cases on the premise of the chosen criteria. Such an approach is wide used once finding multi-objective optimization problems; this might produce less optimal results compared to Pareto-efficient ways. Thus, Yoo and Harman [18], [19] treated the take a look at suite optimization issues mistreatment Pareto-efficient multi-objective genetic algorithms (MOGAs) to handle multiple and different objectives. This incontestable that these mechanisms don't perpetually facilitate in outperforming straightforward greedy algorithms, as a result of MOGAs converged untimely to some sub-optimal solutions. Previous study [4] it is steered adding a diversity-preserving objective perform (measured in keeping with a coverage criterion, like code coverage) to typical multi-objective for- mutations aimed toward minimizing the price and maximizing the coverage. They introduce 2 novel genetic operators to promote diversity between the candidate solutions (sub-sets of the take a look at suite) within the genotype house instead of within the composition house. Specifically, It is tend to introduce a generative algorithmic rule to create a diversified initial population, supported orthogonal style [11], associate degrees, an orthogonal exploration mechanism of the search area through Singular worth Decomposition (SVD) [16], aimed toward conserving the variety throughout the evolution of the population [5].It is tend to conduct an empirical study on eleven world open-source programs. They tend to additionally find that DIV-GA outperforms each traditional MOGAs and greedy algorithms.

A new MOGA known as DIV-GA (Diversity based Genetic Algorithm) that integrates orthogonal evolution and orthogonal design into MOGAs to resolve multi-criteria test case selection issues. DIV-GA addresses the problem of diversity within the genotype area thus severally of the quantity and also the reasonably take a look at criteria.

The analysis of DIV-GA on a collection of open-source programs from the Siemens suite the edu house Agency suite and antelope open-source distribution. The chosen programs were conjointly used in several previous work [3], [4], [9], [14], [15], [17], [18], [19], [20].The comparison of DIV-GA with previous techniques, namely greedy algorithms and also the island version of NSGA-II [17] (named vNSGA-II), antecedently employed by Yoo

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

and Harman for test suite optimization [4], [18], [19], [20]. The comparison issues each optimality and effectiveness.

II. RELATED WORK

2.1 TEST SUITE

The goal of the test suite minimization (TSM) problem consists of reducing the size of the test suite by deleting test cases that are redundant with respect to some coverage criteria [5], such as code coverage, branch coverage, data flow, dynamic program invariants or call stacks [21]. Clearly, one issue of the test suite minimization is that removing some test cases from the test suite may potentially affect its ability to detect faults, since a smaller test suite might have a lower effectiveness [4], [5]. Finding the minimal subset of a test suite is NP-complete, as it can be reduced to the minimal hitting set problem in polynomial time. McMaster and Memon[] proposed a test suite minimization technique based on call-stack coverage. Black et al. [2] considered a bi-criteria approach that takes into account two testing criteria such as code Coverage and past fault detection history. They combined the two objectives by applying a weighted- sum approach, and used Integer Linear Programming (ILP) optimization to find subsets then reducing the multi-objective problem to a single-objective one. Greedy algorithms have also been used to solve a bi-criteria TCP problem by combating two objectives (coverage and cost) in only one function (coverage per unit cost) to be maximized by applying the weighted-sum approach [7], [10]. Test case selection (TCS) focuses on selecting a sub- set from an initial test suite to test software changes, i.e., to test whether un-muddied parts of a program still continue to work correctly after changes involve other parts [13]. Once the test cases covering the unmodified parts of programs are identified using a given technique, an optimization algorithm—e.g., additional greedy—can be used to select a minimal set of such test cases according to some testing criteria— e.g., statement coverage—with the purpose of reducing the cost of regression testing.

2.2 SEARCH BASED TEST SUITE

Test case selection, test suite minimization, and test case prioritization will be viewed as multi-objective problems, were the goal is to select a Pareto-efficient subset of the test suite, based on multiple test criteria [21]. Multi-objective algorithms like MOGAs will then be applied to resolve them. Selecting a subset $\Gamma 0 \subseteq \Gamma$ such $\Gamma 0$ is that the Pareto-optimal set with reference to the objective functions. The optimality of the solutions is measured through the ideas of Pareto| sociologist| economist| economic expert } optimality and Pareto dominance. It's necessary to note that this search-based formulation is named the test case selection drawback tackled during this paper that doesn't need test case ordering. Identifying a Pareto frontier is especially useful as a result of the programmer will use the frontier to create a intelligent decision that balances the trade-offs between the various objectives. As an example, the programmer will select the solution with lower execution cost or higher code coverage on the idea of the resources accessible for execution the selected test cases. They also evaluated totally different optimization algorithmic programs to find Pareto- best sub-sets of the test suite further greedy algorithms and a variant of the multi-objective genetic algorithm NSGA-II [2]. The additional greedy algorithmic programs were applied by using the normal weighted add approach to connate all the objectives in precisely one perform to be optimized, a cost cognizant version of the extra greedy algorithm was used for the two-objective formulation, whereas the weighted add of code coverage per unit of your time and fault coverage per unit of your time was considered for the three-objective formulation. The empirical comparison between MOGAs and greedy algorithms didn't reveal a transparent winner between them, and in some cases the MOGAs were not able to exceed the greedy algorithms. Moreover the combination between these 2 forms of algorithms wasn't al- ways in which helpful to achieve higher results [21].

2.3DIVERSITY BASED GENECTIC ALGORITHM (DIV-GA):

This section describes however we tend to use DIV-GA (Diversity based Genetic Algorithm) to solve the multi-objective test case selection drawback. Specifically, we tend to detail however we tend to inject diversity into the most loop of NSGA-II, that is that the sociologist efficient multi-objective genetic algorithm designed by deb et al. [2]. Whereas previous approaches to multi-objective test case selection [21], [22] used the island variant of NSGA-II (vNSGA- II), we tend to based DIV-GA on the quality version NSGA- II. a new population is generated using a selection operator, to pick folks and offspring in line with the values of the objective functions.

The process of choice is performed victimization the thought ofPareto optimality that leads the selection of non-dominated solutions within the current population. The situation distance is used so as to create a decision concerning that individual to select: the people that area unit remote from the remainder of the population has higher chance to be chosen. Moreover, NSGA-II uses the quick non-dominated sorting algorithm to preserve within the next generation the people forming this Pareto frontier (elitism). Once some generations, the algorithm converges to "stable" solutions, i.e., the Pareto-optimal set of the problem.

III. EVALUATED DIV-GA ALGORITHM

- Population size: Since the search area of the test case selection problem is larger for programs with a bigger test suite, we use totally different population sizes per the scale of the test suites to be optimized.
- Initial population: The initial population is at random generated inside the solution area. For DIV-GA, the initial population consists of the orthogonal arrays [0, 1].
- Range of generations: The most variety of generations varies per the scale of the test suites to be optimized.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

- Crossover function: It is use a multi-point crossover, known as scattered crossover with chance computer = zero.50.
- Mutation operation: It is use a bit-flip mutation function with chance pm = 1/n, wherever n is the size of the test suite (or equivalently n is that the size of the chromosomes).
- Stopping criterion: The average change of the sociologist frontiers is lower than five-hitter for fifty sub sequent generations, or the maximum number of generations is reached.

3.1. INITIAL POPULATION

Generating an initial population plays an important role on the performance of GAs [23] since it performs an initial sampling of the search space.

A well-distributed and well-diversified initial population makes the exploration more effective and favors GA convergence toward global optima [23].

This issue becomes particularly critical for problems where the length of the chromosome (number of test cases in our case) is larger than the size of the population [24]. This is especially true for the test case selection problem.

A generic solution of the test suite minimization problem is an array of binary digits X = {x1,...,xn} where xi is equal to 1 if the 1-th test case is selected, 0 otherwise. Since test case selection is a multi-objective problem whose solutions are binary arrays?

3.2. POPULATION EVOLUTION

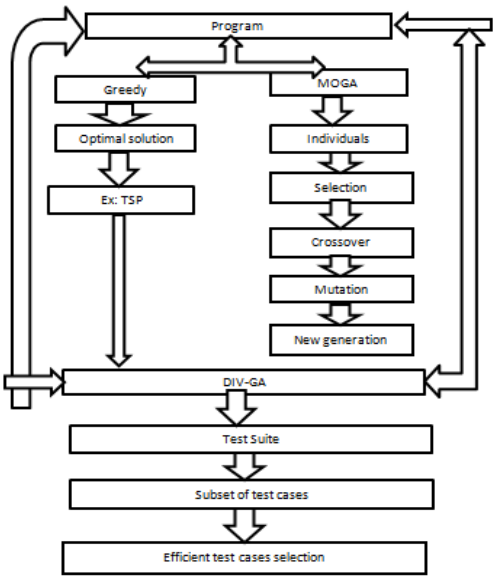
GAs is that a population tends to evolve search area with better fitness as a result of at every generation a selection operator is used to select the (best) individuals that need to survive within the next generation.

Multi- objective problems the selection operator selects for replica the individuals that are non-dominated by the other solution within the current population,(which represents an approximation of the best Pareto set).

As new generations are made the best individuals can tend to converge towards some locally- or globally-optimal Pareto regions.

They propose to adapt the approach planned by de Lucia et al. [15] has been adapted at 3 main points, like encryption of solution (binary-coded chromosomes rather than real-coded ones), selection of best individuals that is performed using the thought of Pareto optimality (while within the work by de Lucia et al. [15] and by Kifetew et al. [9] the best people were selected consistent with the dentition of attest individuals in single-objective paradigm), and once the new individual was generated through SVD, these customizations for the test case selection problem are provided within the following once describing the most steps of the planned algorithm of DIV-GA.

3.3 PROPOSED



IV. RESULT AND DISCUSION

SYSTEM	POPULATION SIZE	NO OF GENERATIONS
Calculator	300	400
Calendar	400	600
Clock	200	500
Flex	400	1000
Grep	200	1000
Gzip	300	500

Table: 4.1Example of open source program

Program	Loc	Test Case	Description
Calculator	255	250	Lexical analyzer
Calendar	234	200	Code optimization
Clock	220	175	Code generation
Flex	6,560	300	Fast lexical analyzer
Grep	4,343	350	Regular expression utility
Gzip	2,220	75	Data compression program

Table: 4.2 Find lines of code and test case selection

2-OBJECTIVE

Two-objective test case selection is finding a set of optimal solutions X which max cost,max coverage.
Max cove (x) = $1/m \sum_{i=1}^{m} \emptyset_i$

DOI 10.17148/IJARCCE.2016.5453

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

Min cost (X) =$\sum_{i=1}^{n} x_i$. cost (τ_i)

Program	Method	Parato size		Non-dominated solution	
		Mean	St.dev	Mean	st.dev
Calculator	Div-GA	150	2.87	145	2.95
	Add Greedy	49	-	54	-
	VNSGA-II	128	-	118	2.17
Calendar	Div-GA	176	2.99	160	2.99
	Add greedy	132	3.1	112	-
	VNSGA-II	112	-	115	2.10
Clock	Div-GA	198	3.24	175	4.50
	Add Greedy	50	-	45	-
	VNSGA-II	40	-	90	4.24
Flex	Div-GA	306	8.52	303	8.23
	Add Greedy	43	-	70	0
	VNSNA-II	157	70.99	0	-
Grep	Div-GA	162	2.19	140	3.05
	Add Greedy	70	-	9	-
	VNAGA-II	60	-	275	4.79
Gzap	Div-GA	222	3.71	186	13.29
	Add Greedy	59	-	354	-
	VNSGA-II	88	1.36	48	13.48

3-OBJECTIVE

Three-objective test case selections are finding a set of optimal solutions which max previous coverage, in cost, max fault

Max cove (x) = 1/m $\sum_{i=1}^{m} \emptyset_i$

Min cost (X) =$\sum_{i=1}^{n} x_i$. cost (τ_i)

Max fault(x) =1/h $\sum_{i=1}^{h} \varphi_i$

Program	Method	Parato size		Non-dominated solution	
		Mean	St.dev	Mean	st.dev
Calculator	Div-GA	172	3.74	165	3.66
	Add Greedy	52	-	11	-
	VNSGA-II	119	-	0	-
Calendar	Div-GA	185	3.14	144	3.24
	Add greedy	70	48.4	6	48.75
	VNSGA-II	80	4.15	1	4.14
Clock	Div-GA	220	4.08	108	4.74
	Add Greedy	60	-	8	-
	VNSGA-II	70	8.06	0	4.44
Flex	Div-GA	331	-	328	8.04
	Add Greedy	47	44.62	7	-
	VNSNA-II	139	13.06	0	-
Grep	Div-GA	317	-	171	44.21
	Add Greedy	72	46.23	1	-
	VNAGA-II	207	4.97	130	42.73
Gzap	Div-GA	198	-	110	11.30
	Add Greedy	19	-	7	-
	VNSGA-II	195	4.66	0	20.77

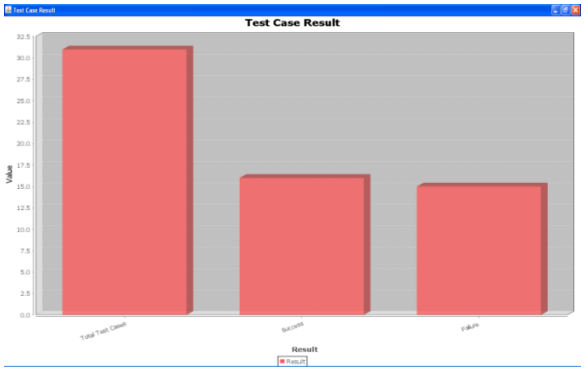
Average Execution Time for Algorithm:

Program	2-Objective		3-Objective	
	Add Greedy	DIV-GA	Add Greedy	DIV-GA
Calculator	1min 2s	2min 10s	1min 10s	2min 10s
Calendar	1min 45s	2min	1min 25s	2min 40s
Clock	1min 20s	2min 30s	1min 40s	2min 45s
Flex	1min 1s	2min 45s	1min 7s	2min 25s

Table: 4.5 Average execution time calculations

S.No	Test case id	Test data	Excepted output	Actual output
1	Tid1	Calculator		
2	Tid2	Calendar		
3	Tid3	Clock		
4	Tid4	Flex		
5	Tid5	Grep		
6	Tid6	Gzip		

Table: 4.6 Result generate



V. CONCLUSION&FUTURE ENHANCEMENT

Proposed DIV-GA (Diversity based genetic algorithm) improves the performance of multi-objective test case selection for solving multi-criteria. The DIV-GA is considered as the best optimizers for multi-objective test case selection problem. . In particular test suite allows not only to generate more optimal trade-offs with respect to the other optimizers when considering two and three test case selection criteria, but its selected sub-test suites turned out to be more cost-effective. The sub-test suites generated are able to reveal more faults at same level of execution cost than the sub- test suites obtained by both the additional greedy algorithm and vNSGA-II. . Using for the DIV-GA to reduce time and cost for the source coding, so be find and reduce minimum space source code to use regression testing for test case selection. This is true in the test case selection problem, but also in other software engineering problems such as test case generation. It may be planned to reduce the cost & time of efficient test case selection in the future work.

REFERENCES

[1]. S. Bates and S. Horwitz, "Incremental program testing using program dependence graphs," in Proceedings of the 20th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, ser. POPL '93. ACM, 1993, pp. 384–396.

[2]. J. Black, E. Melachrinoudis, and D. Kaeli, "Bi-criteria models for all-uses test suite reduction," in Proceedings of the 26th International Conference on Software Engineering, ser. ICSE '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 106–115.

[3]. T. Y. Chen and M. F. Lau, "Dividing strategies for the optimization of a test suite," Inf. Process. Lett., vol. 60, no. 3, pp. 135–141, Nov. 1996.

[4]. A. De Lucia, M. Di Penta, R. Oliveto, and A. Panichella, "On the role of diversity measures for multi-objective test case selection," in Automation of Software Test (AST), 2012 7th International Workshop on, 2012, pp. 145–151.

DOI 10.17148/IJARCCE.2016.5453

[5]. "Estimating the evolution direction of populations to improve genetic algorithms," in Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference. ACM, 2012, pp. 617–624.
[6]. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast elitist multi-objective genetic algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, vol. 6, pp. 182–197, 2000.
[7]. S. Elbaum, A. Malishevsky, and G. Rothermel, "Incorporating varying test costs and fault severities into test case prioritization," in Proceedings of the 23rd International Conference on Software Engineering. Washington, DC, USA: IEEE Computer Society, 2001, pp. 329–338.
[8]. M. J. Harrold, R. Gupta, and M. L. Soffa, "A methodology for controlling the size of a test suite," ACM Transactions Software Engineering and Methodologies, vol. 2, pp. 270–285, 1993.
[9]. D. Jeffrey, "Test suite reduction with selective redundancy," in IEEE International Conference on Software Maintenance (ICSM) 2005. IEEE Computer Society, 2005, pp. 549–558.
[10]. A. G. Malishevsky, J. R. Ruthruff, G. Rothermel, and S. El- baum, "Cost-cognizant test case prioritization," Department of Computer Science and Engineering, Tech. Rep., 2006.
[11]. S. Mcmaster and A. M. Memon, "Call stack coverage for test suite reduction," in IEEE International Conference on Software Maintenance (ICSM) 2005. IEEE Computer Society, 2005, pp. 539–548.
[12]. A. J. Offutt, J. Pan, and J. M. Voas, "Procedures for reducing the size of coverage-based test sets," in In Proc. Twelfth Int'l. Conf. Testing Computer Softw, 1995, pp. 111–123.
[13]. G. Rothermel and M. J. Harrold, "Analyzing regression test selection techniques," IEEE Trans. Software Eng., vol. 22, no. 8, pp. 529–551, Aug. 1996.
[14]. G. Rothermel, M. J. Harrold, J. Ostrin, and C. Hong, "An empirical study of the effects of minimization on the fault detection capabilities of test suites," in Proceedings of the In- ternational Conference on Software Maintenance. IEEE CS Press, 1998, pp. 34–44.
[15]. G. Rothermel, M. J. Harrold, J. von Ronne, and C. Hong, "Empirical studies of test-suite reduction," Journal of Software Testing, Verification, and Reliability, vol. 12, pp. 219–249, 2002.
[16]. G. Strang, Introduction to Linear Algebra, 4th ed. Wellesley-Cambridge Press and SIAM, 2009.
[17]. W. E. Wong, J. R. Horgan, S. London, and A. P. Mathur, "Effect of test set minimization on fault detection effectiveness," in Proceedings of the 17th International Conference on Software Engineering. ACM Press, 1995, pp. 41–50.
[18]. S. Yoo and M. Harman, "Pareto efficient multi-objective test case selection," in Proceedings of the ACM/SIGSOFT Interna- tional Symposium on Software Testing and Analysis. London, UK: ACM Press, 2007, pp. 140–150.
[19]. ——, "Using hybrid algorithm for Pareto efficient multi- objective test suite minimisation," Journal of Systems and Soft- ware, vol. 83, no. 4, pp. 689–701, 2010.
[20]. Q. Zhang and Y.-W.Leung, "An orthogonal genetic algorithm for multimedia multicast routing," Trans. Evol. Comp, vol. 3, no. 1, pp. 53–62, Apr. 1999.
[21]. H. Do, G. Rothermel, and A. Kinneer, "Empirical studies of test case prioritization in a junit testing environment," in 15th International Symposium on Software Reliability Engineering. IEEE Computer Society, 2004, pp. 113–124.
[22]. A. E. Eiben and C. A. Schippers, "On evolutionary exploration and exploitation," Fundam Inform., vol. 35, no. 1-4, pp. 35–50, 1998.
[23]. H. Maaranen, K. Miettinen, and A. Penttinen, "On initial populations of a genetic algorithm for continuous optimization problems," Journal of Global Optimization, vol. 37, no. 3, pp. 405–436, Mar. 2007.
[24]. C. A. CoelloCoello, G. B. Lamont, and D. A. V. Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation). Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

DOI 10.17148/IJARCCE.2016.5453