Multi-Objective Test Case Selection: Local Search Approaches for the NSGA-II algorithm

Luciano Soares de Souza[1]

[1]Instituto Federal do Norte de Minas Gerais (IFNMG) - *Câmpus* Pirapora
R. Dr. Humberto Malard, 1355, 39270-000, Pirapora - MG - Brasil

luciano.souza@infnmg.edu.br

Abstract. *The software testing process can be very expensive and it is important to find ways in order to reduce its costs. Test case selection techniques can be used in order to reduce the amount of tests to execute and this way reducing the costs. Search algorithms are very promising approach to deal with the test case selection problem. This work proposes new hybrid algorithms for multi-objective test case selection by adding local search mechanisms into the NSGA-II algorithm. The results showed that some of the mechanisms were capable of improve the NSGA-II algorithm.*

Resumo. *O processo de testes de software pode ser bastante caro, portanto é importante se encontrar formas de reduzir os custos desse processo. Técnicas de seleção de casos de teste podem ser usadas de forma a reduzir a quantidade de testes a executar e dessa forma reduzir os custos. A utilização de algoritmos de busca são uma maneira promissora de resolver o problema de seleção de casos de teste. Esse trabalho propõe novos algoritmos híbridos de busca para seleção multiobjetivo de casos de teste através da inserção de mecanismos de busca local no algoritmo NSGA-II. Os resultados mostraram que alguns dos mecanismos foram capazes the introduzir melhoras no algoritmo NSGA-II.*

1. Introduction

Software testing is an expensive and time consuming process, which may reach about 40% of total costs involved in software development [Ramler and Wolfmaier 2006]. As such, automation emerges as the key solution for improving the efficiency and effectiveness of the testing process, as well as to reduce its costs.

Fortunately, it is possible to identify, in test suites, redundant test cases concerning a piece of code. Thus, we can envision ways to reduce the suites in order to fit the available resources without seriously compromising the coverage of the adequacy criterion, and thus the quality of the testing process.

The task of reducing a test suite based on a *selection criterion* is known as *Test Case selection*. Given an input test suite, test case selection aims to find a relevant subset regarding the adopted test adequacy criterion, such that the test cases that do not improve the reduced suite coverage can be eliminated. Clearly, the selection criterion relies upon the coverage of the adopted adequacy criterion.

Test case selection is not easy or trivial since there may be a large number of combinations to consider when searching for an adequate subset. A very promising approach to deal with this problem relies upon the use of search optimization techniques

(see [Harman 2011]), which is the focus of this research. Here, the aim is to search for a subset of test cases which optimizes a given objective function (i.e., the given selection criterion).

Within this work we addressed the test case selection problem using two selection criteria: (1) branch coverage, and (2) execution cost (time). In this light, the TC selection problem here was treated as a multi-objective optimization problem.

The focus of this work was to investigate whether local search mechanisms were capable of improve the results of the Non-Dominated Sorting Genetic Algorithm (NSGA-II) [Deb et al. 2000]. For that, we combined three local search strategies with the NSGA-II algorithm in order to create three new hybrid algorithms: (1) NSGA-II-FSBE, (2) NSGA-II-1opt and (3) NSGA-II-AG.

2. Problem Formulation

In this work, the solution is defined as a binary vector representing a candidate subset of TCs to be applied in the software testing process. Let $T = \{T_1, \ldots, T_n\}$ be a test suite with n test cases. A solution is defined as $\mathbf{t} = (t_1, \ldots, t_n)$, in which $t_j \in \{0, 1\}$ indicates the presence (1) or absence (0) of the test case T_j within the subset of selected TCs.

As said, two objective functions were adopted: branch coverage and execution cost. The branch coverage (function to be maximized) consists of the ratio (in percentage) between the amount of code branches covered by a solution $\mathbf{t}$ and the amount of branches covered by T. In turn, the execution cost (function to be minimized) represents the amount of time required to execute the selected suite. Finally, the proposed algorithms are used to deliver a good Pareto frontier regarding the objective functions.

3. Local Search Approaches

Generally speaking, local search algorithms choose, at each step, the locally best node (which yields the best objective evaluation). The local search algorithms used in this work are the Forward Selection (FS) [Webb 2002], the Backward Elimination (BE) [Webb 2002], the 1-opt [Papadimitriou and Steiglitz 1998] and the Additional Greedy (AG) algorithm [Elbaum et al. 2000].

3.1. NSGA-II-FB

In order to create the NSGA-II-FB algorithm, we introduced the local search procedure as the last step of the main loop as follows:

1. Select 10% of the non-dominated solutions by using a Roulette Wheel;
2. For each selected solution $\mathbf{t}$ randomly select one objective to improve
3. *IF* the selected objective is to be maximized, *THEN* use the FS local search algorithm; *ELSE* (if it is to be minimized) use the BE local search algorithm.

The FS algorithm takes the solution $\mathbf{t}$ and iterates as follows: for each test case t_j not yet present in the current solution $\mathbf{t}$ (i.e., for each $t_j = 0$) a new candidate solution $\mathbf{t}'$ is produced by setting $t_j \leftarrow 1$. For each candidate solution, the previously chose objective function is computed. The candidate solution which yields the highest objective value and is not dominated is then adopted as the new current solution in the search process.

The algorithm stops (1) when all candidate solutions found at an the current iteration are dominated, or (2) when all test cases have been already added to the current solution. Contrarily, the BE algorithm takes the solution **t** and iterates as follows: for each TC present in the current solution (i.e., for each $t_j = 1$) a candidate solution $\mathbf{t}'$ is produced by setting $t_j \leftarrow 0$. The objective function is computed and the candidate solution which yields the lowest objective value and is not dominated is considered as the current solution for the next iteration.

It is important to highlight that all non-dominated solutions found during the local search process are stored within the non dominated solutions of the NSGA-II algorithm.

3.2. NSGA-II-1opt

The NSGA-II-1opt algorithms was created by adding the 1-opt algorithm as local search in the following way:

1. Select 10% of the non-dominated solutions stored in the EA by using a Roulette Wheel;
2. For each solution **t** do:
    (a) For each test case $t_j \in \mathbf{t}$ do:
        i. Flip the bit of t_j (i.e if $t_j = 0$ then $t_j \leftarrow 1$, or the contrary) in order to create the neighborhood solution $\mathbf{t}'$;
        ii. Evaluate $\mathbf{t}'$ according to the objectives and store if it is a non-dominated solution;

3.3. NSGA-II-AG

In order to create the NSGA-II-AG algorithm we adapted the idea of the aforementioned traditional Additional Greedy algorithm in the following way:

1. Select 10% of the non-dominated solutions stored in the EA by using a Roulette Wheel;
2. For each solution **t** do:
    (a) REPEAT
        i. For each test case $t_j \in \mathbf{t}$ where $t_j = 0$ do:
            A. Creates $\mathbf{t}'$ by making $t_j = 1$;
            B. Computes the additional coverage per cost of $\mathbf{t}'$;
        ii. Choose the $\mathbf{t}'$ with the best additional coverage per cost value and update the current solution $\mathbf{t} = \mathbf{t}'$
    (b) UNTIL it is not possible find a $\mathbf{t}'$ that enhances **t**
    (c) Store each non-dominated solution found during the previous steps

4. Experiments and Results

The experiments were performed using 4 programs (*flex*, *grep*, *sed* and *space*) from the SIR (Software-artifact Infrastructure Repository). Each algorithm (NSGA-II, NSGA-II-FB, NSGA-II-1opt and NSGA-II-AG) was executed 100 times with a total of 50000 functions evaluations and they used the parameters' values proposed in [Deb et al. 2000]. Furthermore, three well known multi-objective metrics (Hypervolume - HV, Generational Distance - GD and Inverted Generational Distance - IGD) were used in order to evaluate the algorithms.

Table 1. Mean and standard deviation values for each algorithm and each metric.

		HV	GD	IGD			HV	GD	IGD			HV	GD	IGD			HV	GD	IGD
flex	NSGA-II	0.793 (0.012)	**0.018 (0.004)**	0.021 (0.001)	grep	NSGA-II	0.707 (0.013)	**0.025 (0.005)**	0.025 (0.001)	sed	NSGA-II	0.776 (0.021)	0.030 (0.008)	0.019 (0.001)	space	NSGA-II	0.852 (0.017)	0.003 (0.001)	0.016 (0.001)
	NSGA-II-FB	**0.909 (0.064)**	**0.018 (0.011)**	**0.009 (0.006)**		NSGA-II-FB	**0.769 (0.072)**	0.033 (0.013)	**0.018 (0.007)**		NSGA-II-FB	**0.940 (0.085)**	**0.008 (0.009)**	**0.004 (0.006)**		NSGA-II-FB	**0.976 (0.001)**	**0.001 (0.001)**	**0.001 (0.001)**
	NSGA-II-1opt	0.763 (0.011)	0.024 (0.003)	0.022 (0.001)		NSGA-II-1opt	0.672 (0.012)	0.037 (0.005)	0.029 (0.001)		NSGA-II-1opt	0.788 (0.021)	0.027 (0.005)	0.016 (0.001)		NSGA-II-1opt	0.920 (0.009)	0.004 (0.001)	0.006 (0.001)
	NSGA-II-AG	0.697 (0.12)	0.027 (0.009)	0.024 (0.001)		NSGA-II-AG	0.592 (0.011)	0.054 (0.010)	0.038 (0.001)		NSGA-II-AG	0.670 (0.020)	0.069 (0.010)	0.024 (0.001)		NSGA-II-AG	0.746 (0.024)	0.015 (0.004)	0.019 (0.001)

Table 1 shows the mean and standard deviation values for each algorithm and for each metric. The values were compared using the Mann-Whitney U statistical test and the best value for each metric is highlighted.

According to the results from the Table 1, it is possible to see that the NSGA-II-FB outperformed the others in almost all cases. Hence, the FB local search mechanism indeed was able to improve the results of the NSGA-II algorithm in the studied cases. In turn, the 1-opt mechanism was able to improve the results only in some cases. Finally, the AG mechanism was not able to improve the results of the NSGA-II algorithm. In fact, the results of the NSGA-II-AG algorithm were worse than the NSGA-II results.

5. Conclusions

This paper proposes the use of three local search mechanisms in order to create new hybrid algorithms. These algorithms were used in the context of multi-objective test case selection. Experiments, using programs from the SIR repository, were performed in order to evaluate wether the proposed hybrid algorithms were capable of improving the results of the NSGA-II algorithm.

The results showed that the NSGA-II-FB algorithm was able to improve the results of the NSGA-II algorithm in almost all observed cases. Hence it indicates that the FB local search mechanism indeed improved the NSGA-II algorithm. Furthermore, the NSGA-II-FB algorithm was considered the best. Further studies will expand these experiments in order to allow the comparison of the algorithms in more diverse environments.

References

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *Parallel Problem Solving from Nature PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, pages 849–858. Springer Berlin Heidelberg.

Elbaum, S., Malishevsky, A. G., and Rothermel, G. (2000). Prioritizing test cases for regression testing. *SIGSOFT Softw. Eng. Notes*, 25(5):102–112.

Harman, M. (2011). Making the case for morto: Multi objective regression test optimization. In *Fourth International IEEE Conference on Software Testing, Verification and Validation*, pages 111–114. IEEE Computer Society.

Papadimitriou, C. and Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity*. Dover books on mathematics. Dover Publications.

Ramler, R. and Wolfmaier, K. (2006). Economic perspectives in test automation - balancing automated and manual testing with opportunity cost. In *Workshop on Automation of Software Test, ICSE 2006*.

Webb, A. R. (2002). *Statistical Pattern Recognition, 2nd Edition*. John Wiley & Sons.