Multiobjective Evolutionary Algorithms NSGA-II and NSGA-III for Software Product Lines Testing Optimization

Muhammad Abid Jamil
College of Computer and Information Science
Umm Al Qura university
Makkah, Saudi Arabia
majamil@uqu.edu.sa

Ahmad Alhindi
College of Computer and Information Science
Umm Al Qura university
Makkah, Saudi Arabia
ahhindi@uqu.edu.sa

Muhammad Arif
College of Computer and Information Science
Umm Al Qura university
Makkah, Saudi Arabia
mahamid@uqu.edu.sa

Mohamed K Nour
College of Computer and Information Science
Umm Al Qura university
Makkah, Saudi Arabia
mknour@uqu.edu.sa

Normi Sham Awang Abubakar
KICT , International Islamic university, Kuala lumpur, Malaysia
nsham@iium.edu.my

Tareq Fahad Aljabri
College of Computer and Information Science
Umm Al Qura university
Makkah, Saudi Arabia
tareq@uqu.edu.sa

***Abstract*—Software Product line (SPL) engineering methodology utilizes reusable components to generate a new system for a specific domain. In fact, the product line establishes requirements, reusable components, architecture, and shared products to develop new products' functionalities. In order to maintain high quality, there is a need for a thorough testing process. Each product in SPL having a different number of features need to be tested. Hence, the testing process of SPL can utilize a multi-objective optimization algorithm to optimize the testing process. This research, reports on the performance of a multi-objective Evolutionary Algorithms Non-Dominated Sorting Genetic Algorithm II (NSGA-II) and NSGA-III on Feature Models (FMs) to optimize SPL testing.**

Keywords—Search-Based Software Engineering, Software Testing, Software Product Line, Feature Models, Multi-objective Evolutionary Algorithms, Non-Dominated Sorting Genetic Algorithm II and III (NSGA-II and NSGA-III).

I. INTRODUCTION

The product lines engineering relies on components reusability. The process consists of two stages, a) Domain Engineering, b) Application Engineering. The domain engineering phase does analysis and design of domain for developing core assets. The final products in application engineering are being generated using core assets with needs if mentioned by the customer. This stage follows three methodologies like; application requirements, application design and application coding [1].

The basic principle to underpinned the Software Product Line (SPL) engineering is the generation of main assets carried out in the domain engineering phase and these assets configuration is done by the application engineering phase. There exist variation points in core assets that help in the configuration process. Even various requirements for different final software products are fulfilled by configuring these variation points. This configuration process disseminates in application engineering [2].

Figure 1 shows the process of Software Product Line Engineering (SPLE) where during SPL development there is the selection of different products and after configuration of products, we have final products with bounded or valid features.

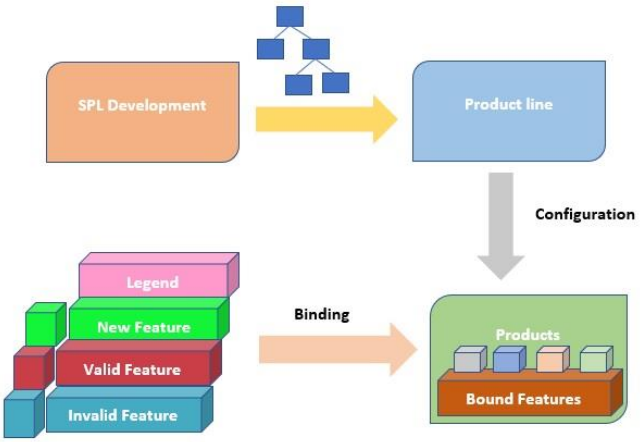

Fig 1: Software Product Line Engineering

In a high-cost software development environment, efficient testing strategies take to play an important role. Hence, the share of testing costs increases in an organization using software product lines (SPL) as the development costs for each product decreases. The SPL is the derivation of a variety of products from a huge product platform, therefore testing of SPL is to be considered as a complex and expensive activity. For the complexity of SPL testing, there is also needed to determine this aspect in regard to what needs to test in the platform as well as in separate products [4].

Therefore, a large number of test cases in the software product line process becomes a challenge. This is a reality that product line testing becomes difficult when there is the growth of variants in the product. Hence, system test increases exponentially when the number of developed products also increases. Hence, it makes the product line testing impracticable [3].

For test case design we need some testing optimization strategies and their possible use when the test cases are

978-1-7281-4082-7 /19/$31.00 ©2019 IEEE

increasing in number. These strategies should be useful in the testing process to achieve the quality level of the product line organization [4].

For reusable components, a high level of quality is needed but this is not clear how much testing of reusable components can aid to minimize testing liabilities for every product. We must consider these issues at this level about the testing of generic components, the tests order and testing of different possible variants. The organization of testing activities can be complex due to the fact that the software process in SPL is divided into two phases and thus testing may be divided into different parts in the SPL testing process [17].

This research investigates the use of Multiobjective Evolutionary Algorithms (MOEA) in SPL testing. This paper explores how MOEA algorithms can obtain the optimal number of test cases that provide high coverage with less testing efforts. The paper compares the results using two well-known algorithms Non-Dominated Sorting Genetic Algorithm II and III (NSGA-II and NSGA-III). The paper shows the results that using this approach improves the SPL testing process significantly by reducing the number of test cases and maximizing the coverage.

The solution optimality in Multi-Objective (MO) optimization is formed with the idea of Pareto dominance [15]. Zitzler et al. [11] demonstrated that the performance metrics could determine an acceptable implication will be superior as an alternate in place the Pareto dominance case. Therefore, in space of other metrics, the Pareto dominance indicator with n-ary indicator is recommended in such cases.

The remaining part of this paper has been organized in the following way. Section 2 describes NSGA-II and NSGA-III algorithms with their utilization in search-based software engineering. Section 3 defines how to formulate our problem. Section 4 mentions about the different terminologies used. Section 5 discusses the results and in sections 6 and 7, limitations and conclusions about the research have been described.

II. Background and Related Work

In this section, first, a brief introduction of the NSGA-II and NSGA-III algorithm has been presented.

Deb et al. [18] suggested non-dominated sorting genetic algorithm –II (NSGA-II) having three basic properties, 1) it exploits an elitism law, 2) It focuses non-dominated solutions, 3) It utilizes the fully developed quality of being developed preserving mechanism. This algorithm permits that how the combination of the primary population (Pt) and secondary population (Qt) are sorted to fulfill the domination rule, while F1 comprises of Non-dominated solutions.

Deb et al. [6] recommended the many-objective algorithm known as NSGA-III. The fundamental idea is the same as the NSGA-II with the purpose of the selection technique [7]. Figure 2 explains the pseudo-code of NSGA-III working with a specific generation t. First, for a particular domain, there is random initialization of parent population P_t of size N. After that offspring population Q_t has been generated with binary tournament selection, crossover and mutation operators. Then both populations have been joined in order to make the structure of the parent population and then arrange or sort with respect to their domination level. After this, there is sampling, or selection of best N members chosen from joined population to make the parent population for the next generation

As compare to NSGA-II, the NSGA-III works using the technique of a set of reference points Z^r. For the selection of a well-distributed set of points, the NSGA-II utilizes the crowding distance approach or measure. While there is a provision of reference point Z^r in the NSGA-III so that the remaining member's selection can be possible as shown in Figure 2.

For the achievement of members selection, there is a need for normalization of objective values and reference points in order to attain the identical range. Then subsequently there is computation of the orthogonal distance between a member in S_t and every reference line. It means that there is a combination of the ideal point and a reference point.

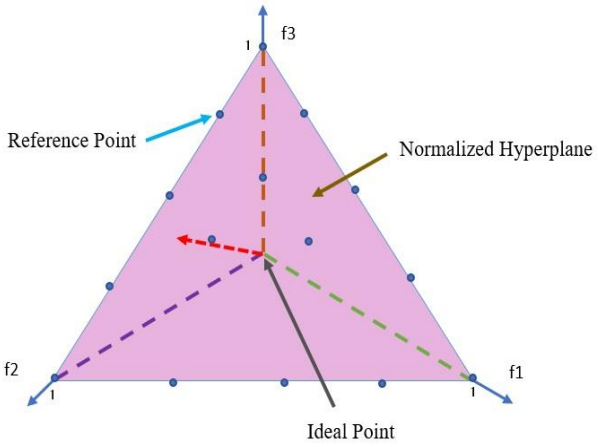

Fig 2: NSGA-III Normalized Reference Plane [13]

Mkaouer, Mohamed Wiem, et al. [9] discussed the issues of search-based software engineering using the NSGA-III optimization approach that handles 15 different objectives required to be optimized. In their research work, they determined the effective solutions utilizing 15 quality metrics. They applied their technique on different seven large open-source systems and inferred correct results up to more than 92% for code smells.

The NSGA-III approach produced remarkable results with 31 runs of statistical analysis of experiments when compared with other many-objective techniques (NSGA-II, IBEA and MOEA/D). Hence, finally concluded that NSGA-III outperforms for automated refactoring problem. From a large space of a set of candidates, there is a selection of best-refactoring solution for refactoring problem optimization [9]. Typically, in SBSE techniques there are mostly two or three metrics that are considered as objective functions in order to discover code smells and fix them [9]. But in reality, there exist many kinds of code smell and to figure out or detect each smell there is a need for a specific set of metrics.

The researchers et al. [9] proposed to utilize the high number of metrics (i.e. 15 metrics) and for every single metric, there is a representation of every objective function. Therefore, in the case of refactoring problem with 15-

objectives was a challenge to solve it with common or standard multi-objective approaches. For this, they devised to use the vector-oriented representation [9].

But here in our approach, we work on testing a feature model (Counter-Strike) of products selected from the SPL repository [10] mentioned later in the paper.

III. Adapting NSGA-II and NSGA-III for Software Product Line Testing Problem

This research work focuses to adapt the NSGA-II and NSGA-III algorithms for the feature testing of the selected feature model. Our approach will help to reduce the testing efforts by configuring the products. The following objectives will be achieved for our proposed work.

A. Objective One

To maximize the pairwise coverage.

O1(x) = p_coverage(x),

The above function calculates the coverage of number of features pairs.

B. Objective Two

To minimize the number of products.

O2(x) = p_min(x),

This function finds number of products m.

C. Objective Three

To reduce the testing cost.

O3(x) = t_cost(x),

This function computes the cost in term of testing.

IV. Method Description

Feature Model Testing

For the analysis and representation, the feature modeling approach has been utilized. But in the case of SPL testing, there is a need for two important aspects regarding the feature models, a) a well-prescribed criterion should be defined for automatic extraction of test configurations, b) there should be proper or specific relation between assets of SPL. A feature needs to contain its concrete information with the purpose to combine it with other features to make an absolute product formation. For this purpose, an SAT Solver is used to generate a valid number of products [14].

Terminologies Descriptions

The terminologies used in our work have been described in the following Figure 3.

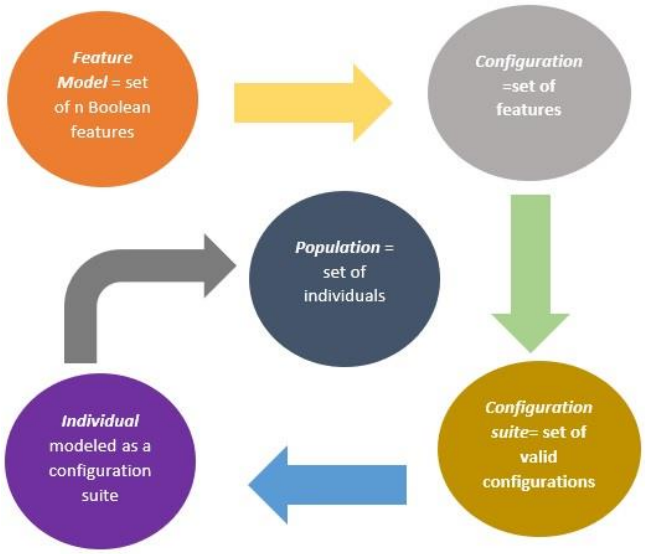

Fig 3: Terminologies Definitions used in Research work

- *Feature Model* is a set of features with specific constraints to be evaluated true.
- A *Configuration* is set that shows the presence of a feature in FM for a product in SPL.
- A *Configuration Suite* is set of valid configurations.
- An *Individual* is modeled as a configuration suite.
- A *Population* is a set of individuals.

TABLE I. Characteristics of the Feature Model Used

Feature Model	Features	Configurations	Number of pairs
CounterStrike	24	18176	833

Table 1 represents the feature model Counter-Strike [12] selected in our work and its main characteristics like the number of pairs and configurations have been shown.

V. Experiment And Results

The following Table II shows the parameter settings for the algorithms used in our research work. The population for both MOEA algorithms has been selected 200 with other parameter values mentioned in Table II.

TABLE II. Parameters Settings for NSGA-II and NSGA-III

Algorithm	Population	Chromosome	Selection	Crossover Operator	Mutation Operator
NSGA-II	200	Variable-length encoding	Tournament selection (20)	Single point	Random
NSGA-III	200	Variable-length encoding	Tournament selection (20)	Single point	Random

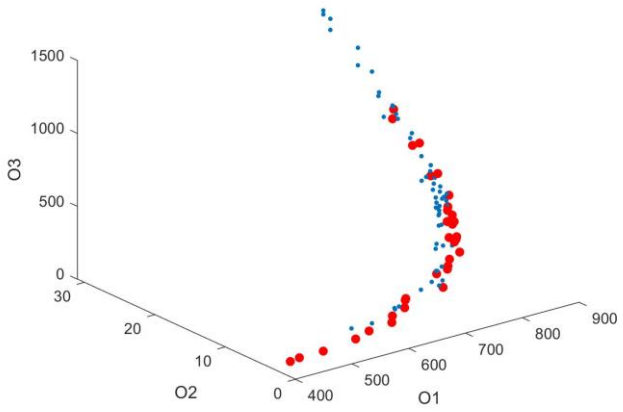

Fig. 4 NSGA-II Pareto Front Counter-Strike FM

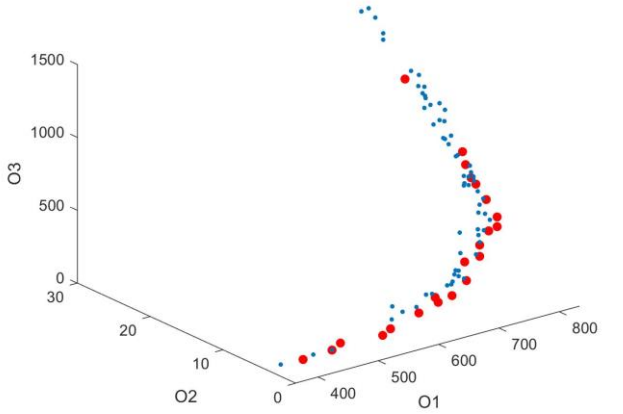

Fig. 5 NSGA-III Pareto Front Counter-Strike FM

TABLE III. COUNTERSTRIKE FEATURE MODEL RESULT COMPARISON FOR PARETO DOMINANCE

Algorithms	Total	Non-Pareto-Dominance	Pareto-Dominance	% Non-Pareto-Dominance	% Pareto-Dominance
NSGA-II	94	21	73	22.3	77.7
NSGA-III	92	13	79	14.1	85.9

The results discussed in Table III, are presented about the Pareto dominance and non-Pareto dominance also discussed by Zitzler et al. [11].

The experiment has been undertaken for one feature models. The size of the population has been set as 200 while generations numbers adjusted as 500. For each algorithm 5 runs were carried for the feature model selected in our case. The results for the Counter-Strike feature model are shown in Fig 4 and Fig 5. While Table III shows the comparing results for NSGA-II and NSGA-III for the Counter-Strike feature model. The table shows that the total number of points for the Pareto front for the last generation for NSGA-II is 94 and 92 for NSGA-III. The non dominated solutions for NSGA-II are 21 which provide 22.3 % compared to 13 solutions for NSGA-III which is about 14%. The results show NSGA-II clearly outperforms NSGA-III in three objectives optimization problems. This is in line with literature which shows NSGA-II performs better in the multi-objective problem where NSGA-III performs better in many-objective problems.

TABLE IV. EVOLUTION OF OBJECTIVES FROM INITIAL GENERATION TO THE FINAL GENERATION

FM	Algorithm	O1: Pairwise Coverage (max.)		O2: No. Configurations (min.)		O3: Cost (min.)	
		Initial	Final	Initial	Final	Initial	Final
Counter-Strike	**NSGA-III**	**809**	**825**	**42**	**32**	**105**	**94**
	NSGA-II	**833**	**849**	**29**	**20**	**83**	**79**

In the above Table IV, we compared the initial and final values determined from experiments for all three objectives and presented our results using the same approach proposed by Henard et al. [12].

VI. LIMITATIONS OF WORK

This research faced a couple of threats based on the experiments performed. Firstly, about the results generalization that there may be chances for different results for Feature Models discussed here in this paper. The results of two feature models have been presented here to compete with this threat. The implementation can also be a threat to have the concrete or optimized results, hence we make it into different routines. The experimentation performed in iterations also help to generate the results without errors.

VII. CONCLUSION AND FUTURE WORK

The optimization process with the different number of objectives seems to be challenging as when we try to optimize one object may be another not optimize. Here in our research work, we present the real application of NSGA-III and NSGA-II to address the search-based software testing problem. In our approach testing objectives are evaluated with different metrics. Our approach has been evaluated on two different SPL feature models mentioned above selected from the feature model repository [10]. The results generated from our experiments demonstrated that NSGA-III produced better, and scaled results compare to NSGA-II results.

In addition to future work, we plan to adopt NSGA-III with more than three objectives to solve the SPL testing problems and will accomplish our objectives with comparative studies on larger product lines.

Acknowledgment

This work is supported by grant number 17-COM-1-01-0007, Deanship of Scientific Research (DSR) of Umm al Qura University, Kingdom of Saudi Arabia. The authors would like to express their gratitude for the support and generous contribution towards pursuing research in this area.

REFERENCES

[1] Trigaux, Jean-Christophe, and Patrick Heymans. "Software product lines: State of the art." *Product Line ENgineering of food TraceabilitY software, FUNDP-Equipe LIEL* (2003): 9-39.

[2] Cawley, Ciarán, et al. "A 3d visualisation to enhance cognition in software product line engineering." *International Symposium on Visual Computing*. Springer, Berlin, Heidelberg, 2009.

[3] Ibrahim H. Osman. An introduction to metaheuristics. In M. Lawrence and C. Wilsdon, editors, *Operational Research Tutorial Papers*, pages 92–122. Stockton Press, Hampshire, UK, 1995. Publication of the Operational Research Society, Birmingham, UK.

[4] Engström, E., & Runeson, P. (2011). Software product line testing–a systematic mapping study. *Information and Software Technology*, 53(1), 2-13.

[5] Oster, S., Markert, F., & Ritter, P. (2010, September). Automated incremental pairwise testing of software product lines. In *International Conference on Software Product Lines (pp. 196-210)*. Springer Berlin Heidelberg.

[6] Deb, K. and Jain, H. An Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting Approach, Part I: Solving Problems with Box Constraints. In *Proceedings of IEEE Transactions on Evolutionary Computation*. Accepted.

[7] *Deb, Kalyanmoy, et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." IEEE transactions on evolutionary computation 6.2 (2002): 182-197*

[8] Kessentini, M., Kessentini, W., Sahraoui, H., Boukadoum, M., and Ouni, A. 2011. *Design Defects Detection and Correction by Example*. In ICPC'11. 81-90.

[9] Mkaouer, Mohamed Wiem, et al. "High dimensional search-based software engineering: finding tradeoffs among 15 objectives for automating software refactoring using NSGA-III." *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2014.

[10] Mendonca, Marcilio, Moises Branco, and Donald Cowan. "SPLOT: software product lines online tools." *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*. ACM, 2009.

[11] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol*. Comput., vol. 7, no. 2, pp. 117–132, 2003.

[12] Henard, Christopher, et al. "Multi-objective test generation for software product lines." *Proceedings of the 17th International Software Product Line Conference*. ACM, 2013.

[13] Kessentini, M., Kessentini, W., Sahraoui, H., Boukadoum, M., and Ouni, A. 2011. Design Defects Detection and Correction by Example. In ICPC'11. 81-90

[14] D. Le Berre and A. Parrain. The sat4j library, release 2.2, system description. Journal on Satis_ability, Boolean Modeling and Computation(JSAT), 7:59{64,2010.

[15] Goh, Chi-Keong, and Kay Chen Tan. "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization." *IEEE Transactions on Evolutionary Computation* 13.1 (2009): 103-127.

[16] N. Srinivas and K. Deb, "Multiobjective optimization using non-dominated sorting in genetic algorithms," *Evol. Comput*., vol. 2, no. , pp. 221–248, 1994.

[17] VAN DER LINDEN, Frank, and Klaus Pohl. "Software Product Line Engineering: Foundations, Principles, and Techniques." (2005).

[18] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II." *IEEE transactions on evolutionary computation*, *6*(2), 182-197.