Computers, Materials & Continua
DOI: 10.32604/cmc.2022.029096
Article

Tech Science Press

Maintain Optimal Configurations for Large Configurable Systems Using Multi-Objective Optimization

Muhammad Abid Jamil[1,*], Deafallah Alsadie[1], Mohamed K. Nour[1] and Normi Sham Awang Abu Bakar[2]

[1]Department of Computer Science, Umm Al-Qura University, Makkah, Saudi Arabia
[2]Kulliyyah of Information and Communication Technology, International Islamic University, Malaysia
*Corresponding Author: Muhammad Abid Jamil. Email: majamil@uqu.edu.sa
Received: 25 February 2022; Accepted: 06 May 2022

Abstract: To improve the maintenance and quality of software product lines, efficient configurations techniques have been proposed. Nevertheless, due to the complexity of derived and configured products in a product line, the configuration process of the software product line (SPL) becomes time-consuming and costly. Each product line consists of a various number of feature models that need to be tested. The different approaches have been presented by Search-based software engineering (SBSE) to resolve the software engineering issues into computational solutions using some metaheuristic approach. Hence, multiobjective evolutionary algorithms help to optimize the configuration process of SPL. In this paper, different multi-objective Evolutionary Algorithms like Non-Dominated Sorting Genetic algorithms II (NSGA-II) and NSGA-III and Indicator based Evolutionary Algorithm (IBEA) are applied to different feature models to generate optimal results for large configurable. The proposed approach is also used to generate the optimized test suites with the help of different multi-objective Evolutionary Algorithms (MOEAs).

Keywords: Software product line; search-based software engineering; metaheuristic; multiobjective evolutionary algorithms; feature model

1 Introduction

The work described in this paper is also mentioned in Software Product Lines testing optimization using the Multiobjective Evolutionary Algorithms (MOEAs) like Non-Dominated Sorting Genetic Algorithm II (NSGA-II) and NSGA-III [1]. Mostly, software engineers are developing such software families comprised of identical systems with many variations. Software Product Line Engineering (SPLE) methodology generates a diversity of quality software products in less time frame and cost [2,3]. In other words, the SPL methodology can also be described as a set of different systems sharing and managing features. Moreover, those are adapted according to specific market requirements [4]. SPLE process combines reusable components in order to produce a newer product for a particular

domain [5]. This approach identifies the requirements, architecture, and different reusable components, and all these artifacts help to develop the specific components of the products' functionalities [6,7].

The goal of the SPL methodology is to increase the product quality while reducing the time and cost of production by reusing already tested core assets. To achieve the objectives in this process, artifacts like requirements specification documents, analysis diagrams, architectures, reusable components, tests, strategies, and maintenance methods are put together [7,8]. Software Product Line Engineering (SPLE) has been introduced for feature analysis [9,10]. The feature models help to deal with variability that can arise in SPLE [11,12].

Fig. 1 shows the methodology of the SPLE process, the reusability of components is considered as the main approach in SPLE. This methodology has been divided into two phases, (1) Domain Engineering where analysis, design, and implementation have been done to develop the core assets of the specific domain, (2) Application Engineering phase build the final products as per requirements of the customer [1,7,13].

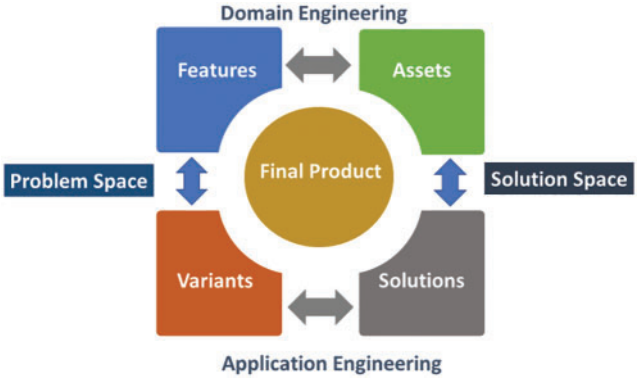


Figure 1: Software product line engineering process

The large number of variations points in SPL makes its development process challenging because product line development differs from single product development. Hence, the variability is a natural aspect of the modeling process [14]. The software industry uses Software product lines (SPL) methodology to make the software development process more effective because of faster production in less time. But testing of SPL needs more research due to the infeasibility of testing of individuals' systems components. The organization of software product line variation points originates the process of test case generation [15–17]. Therefore, variability is considered as the core point in product line development. But major challenges regarding SPL testing is to handle the situation when required as test cases become large in numbers due to product variants [16]. During the test case design, there requires strategies for testing optimization and the utilization of these strategies would be useful when test cases become large and also helps to maintain the quality of the products [15]. The optimization techniques would also be helpful to optimize the bigger size of the regression test suits [18].

However, the feature models are used to represent the software product line and Mendonc et al. [19] discussed that Satisfiability (SAT) solvers originate clear solutions for SPL feature models' reasoning. The SAT solver produces a valid number of configurations for a feature model. The technique is adopted by the SAT solver to translate a feature model into the Boolean formula to generate the configurations [20]. There has been needed to adopt efficient testing techniques in high-cost and sensitive software development environments. In SPL, a different number of products can

be derived from a single product line. Hence, the testing process of SPL can be expensive and time-consuming. The complex SPL testing process needs to figure out which product features should be tested [21].

For software product line testing, the big number of variants results in a large number of test cases which is a challenge for the testing procedure. During product development, the number of test cases increases exponentially which makes the testing process infeasible [1]. This study explores how Multiobjective Evolutionary Algorithms (MOEA) can help to optimize the product line configurations. The research also inquiries to generate the optimal number of test cases with the aim of maximum coverage and to minimize the testing exertions. The three renounced Multiobjective Evolutionary Algorithms (MOEA), Non-Dominated Sorting Genetic Algorithm II, Non-Dominated Sorting Genetic Algorithm III, and an Indicator-Based Evolutionary Algorithm (IBEA) have been utilized to achieve the testing objectives. These three evolutionary algorithms apply to three different product line feature models to generate optimal results. The Pareto dominance technique has been adopted for solution optimality [22]. For the accuracy of results, performance metrics and quality indicators have been utilized [23].

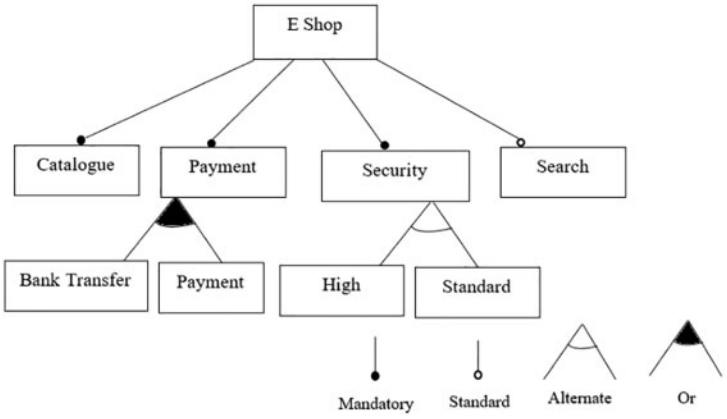


Figure 2: Feature model for E-shop product line [24]

The article has been organized as follows. Section 2 discusses the Feature Model and Multiobjective Evolutionary Algorithms NSGA-II, NSGA-III, and IBEA. The research problem and different terminologies mention in Section 3. The experiments, results, limitations, and conclusions about the research have been described in Sections 4, 5, and 6 respectively.

2 Background

2.1 Feature Models

Feature Models (FM) are represented in a hierarchical form with associations between different features to follow particular constraints. This research work focuses on model-based configuration of product lines and a feature model that describes a particular product line. According to Iglesias et al. [2] and Pohl et al. [3], a product line is a collection of different products, that can share the same features. Software functionalities or attributes are being represented by a feature. The products are distinguished from each other with the help of features, which allow representing variability. Therefore, it can be concluded that distinctive products might be produced by selecting different features. In an SPL,

features are represented in a hierarchical form like a tree in a single Feature Model (FM) [21]. The following Fig. 2 describes the E-Shop feature model.

2.2 Multiobjective Evolutionary Algorithms (MOEAs)

The requisition of optimization algorithms assists to solve issues concerned with software testing and software engineering, which are known as Search-Based Software Testing (SBST) and Search-Based Software Engineering (SBSE). Many of the problems include concurrent streamlining optimization with some competing goals. Therefore, problems concerning optimization, a set of possible solutions or choices or courses of action need to be explored [25,26]. These solutions are ideal in a large search space when every last bit targets the required results [27,28]. For multi-objective optimization, a significant number of algorithms have been proposed from the last two decades and these algorithms rely on evolutionary algorithms [21]. They were invented with the purpose of single-objective optimization, such as Genetic Algorithms, Evolutionary Strategies, Particle Swarm Optimization, and Differential Evolution. In the proposed research, MOEAs discussed relying on Genetic Algorithms. These MOEAs follow basic features or qualities, like crossover, mutation, selection, and elitism. Following different multi-objective algorithms are considered for optimization problems criteria, which are given as:

2.2.1 NSGA-II Algorithm

Deb recommended non-dominated sorting genetic algorithm–II (NSGA-II) [26]. NSGA-II algorithm has three important properties, (1) an elitism law is exploited by this algorithm, (2) the algorithm relies on non-dominated solutions, and (3) It makes use of a developed mechanism based on preservation. It initializes a population of N size after measuring this rank population. From the ranked population, different criteria are defined to calculate the child population. For N individuals' selection, it creates a combination of parent, rank, and child population.

The pseudocode for the NSGAII algorithm has been presented as follows.

Algorithm NSGA-II Algorithm

Input: N', g, fk (X) $\triangleright N'$ members matured g generations to get solution fk (X)
1 Initialize Population $\mathbb{P}'$;
2 Random population generation–size N';
3 Evaluate Objectives Values:
4 Rank assignment rely on Pareto–sort;
5 Produce Child Population;
6 Apply Binary Tournament Selection;
7 Procedure of crossover and Mutation;
8 for i = 1 to g do
9 for each Parent and Child in Population do
10 Rank assignment rely on Pareto–sort:
11 Produce nondominated solutions sets;
12 Calculate Crowding distance:
13 Inner Loop to add solutions to next-generation begin from the first front until N' individuals;
14 end

(Continued)

Algorithm Continued

```
15          Select points on the lower front with high crowding distance;
16          Produce next generation:
17            Apply Binary Tournament Selection:
18            Procedure of crossover and Mutation;
19 end
```

2.2.2 NSGA-III

Deb proposed the many-objective algorithm known as NSGA-III [29]. The basic working is similar to the NSGA-II using the selection criteria [30]. NSGA-III pseudo-code working has been described with generation t. For a specific domain, the parent population Pt of size N is initialized randomly. After that, with the application of operators like selection, crossover, and mutation operators, offspring population Qt is generated. Then both populations are joined to generate the organization and after that according to the domination level they are arranged and sorted. Then sampling approach is adopted for the best N members selection. The NSGA-III adopts the reference points Zr technique as compared to NSGA-II. The pseudocode for the NSGAIII algorithm has been shown as follows.

The pseudocode for the NSGAIII algorithm

Algorithm Algorithm Generation t of NSGA-III

```
Input: H structured reference points Zs
Zα represents points, Pt for parent population
Output: Pt + 1
1   St ← Ø, i ←1;
2   Qt ← Crossover + Mutation (Pt)
3   Rt ←Pt ∪ Qt
4   (All fronts F1, F2, . . .)←Non-dominated-sort (Rt)
5   iterate
6      St←St ∪ Fi and i ← i + 1;
7   until |St| ≥ N;
8   Fl ←Fi; /*Last front to be included */
9   if |St | = N then
10          Pt + 1 ←St;
11    else
12          Pt+1 ← U_{j=1}^{l-1} Fj;
      /*Select Points from Fl */
13       K ←N − |Pt + 1|;
       /* Objectives Normalization and generate reference set Zr */
14    Normalize (FM, St, Zr, Zs, Zα)
         /*Relate all member s of St with a reference point*/
         /*π(s): shows closest reference point */
         /* d: distance between s and π(s) */
```

(Continued)

Algorithm Continued

15 [π(s), d(s)] = Associate (St, Zr)
/* Calculate niche count of reference point j $\in$ Zr */
16 $\rho \leftarrow \sum s\epsilon$St / Fl (($\pi$(s) = j) ? 1: 0);
/*Select K members one at a time from Fl to form Pt = 1 */
17 Niching (K, ρj, π(s), d(s), Zr, Fl, Pt + 1);
18 end if
19 end

2.2.3 IBEA Algorithm

The indicator-based evolutionary algorithm (IBEA) procedure works using arbitrary indicators. The IBEA procedure relies on the user's preferences and therefore it does not require the diversity preservation technique. To obtain better results, the IBEA focuses on fitness ranking criteria, therefore it can obtain the best results [31]. The details and working flow of the IBEA algorithm can be explored in [32]. The IBEA procedure has been presented as follows.

Algorithm Procedure for IBEA

Input: α, N, k
1 Define initial population P of size α
2 Generation counter m to 0
3 while population P not greater than α do
4 foreach Population Individual do
5 Measure individual fitness values
6 end foreach
7 Select individual x ϵ P (smallest fitness value)
8 Eliminate x from the P
9 Revise the fitness values for other individuals
10 if alternate stopping criterion satisfied then
11 Selection of non-dominated individuals in P
12 else
13 Fill the mating pool P' while running binary tournament selection with the replacement on P
14 Use crossover and mutation operators on pool P' and sum up the generated offspring to P
15 Addition the generation counter m and start again from line 5
16 end if
17 end while

2.3 Quality Metrics

These metrics guarantee the discovery of quality-oriented solutions. The performance of MOEAs is difficult in terms of assessment and comparison to measure the nature of Pareto fronts.

Hypervolume

The Hype Volume metric is used to present the volume of objective space and this space is dominated by Pareto front A. Here, A is considered as a set of non-dominated solutions and mathematically defined as

$HV(A) = volume(\cup^{|A|}_{i=1} v_i)$ where, v_i is a hypercube defined

between solution $i \in A$ and the reference point R.

A hyper-volume quality indicator is utilized to determine the quality for both the spread and convergence of non-dominates solutions set. However, there is a need for a reference point selection that bounds the dominated region. The higher value of Hypervolume will illustrate the better working of an algorithm. The discussed metric produces a single value that helps to estimate non-dominated solutions. But one unfavorable metric of HV is that it requires a reference point to compare all non-dominated solutions [33].

2.4 SBSE Problems and Selected MOEAs

Search-based software engineering problems have been resolved by different researchers using multiobjective optimization algorithms. For example, Yoo et al. [34] addressed the test case selection problem and explored that Pareto efficient technique can help to resolve or optimize the multiple objectives problem. They devised a two objectives approach that unites the coverage and cost. Meanwhile, for three objectives optimization, they combine the coverage, cost, and fault history [34]. To optimize the two or three objectives for test case selection, three algorithms like single–objective greedy algorithm, the Non-Dominating Sorting Genetic Algorithm (NSGA-II), and an island genetic algorithm variant of NSGA-II, which is called vNSGA-II, have been utilized [34].

Mkaouer recommended and utilized a search-based software engineering technique where he used the NSGA-III algorithm to optimize the 15 distinct objectives. They produced effective results in their proposed work to automated refactoring defining 15 distinguishable metrics. They selected open-source systems available online where they applied their approach and concluded that their technique is capable to generate results of more than 92% for code smells [35]. Sayyad proposed a refinement approach to improve the IBEA's productivity where he introduced the PUSH, seeding, and PULL heuristic methods. The PUSH heuristic forces the evolutionary search identified dependencies utilizing feature models. But, the PULL technique focuses to satisfy constraints, while the seeding technique figures out the correct configurations in the optimization process [36].

3 Proposed Method

This section describes the research problem concerned with finding optimal configurations for the large software product line. This section provides definitions of terms used in this research work. In particular, this section also defines problem representation (encoding), variation operators (mutation and crossover), objective functions, and constraint handling mechanisms. Before describing the research process for the proposed research, there is a brief discussion about the problem statement i.e., in SPL large composition and configuration of products. For this, the SAT solver technique is applied to three different size feature models using three different MOEAs and a quality metric with specific parameter settings. The terminologies used in the proposed work has been shown in Fig. 3.

3.1 Maintaining Configurations using NSGA-II, NSGA-III, and IBEA

The proposed study utilizes the three MOEAs like NSGA-II, III, and IBEA to obtain optimized results for different product lines. These three algorithms are applied to different sizes of product line feature models. The following objectives will be optimized with the help of these MOEAs which are mentioned in detail in research work [14].

I. Objective One (Maximization of pairwise coverage)

Obj1(x) = coverage(x)

The function determines coverage about the number of feature pairs.

II. Objective Two (Minimization the number of products)

Obj2(x) = minimize(x)

This function calculates the number of products.

III. Objective Three (Minimization of the testing cost)

Obj3(x) = cost(x)

This function determines the testing cost of the product.

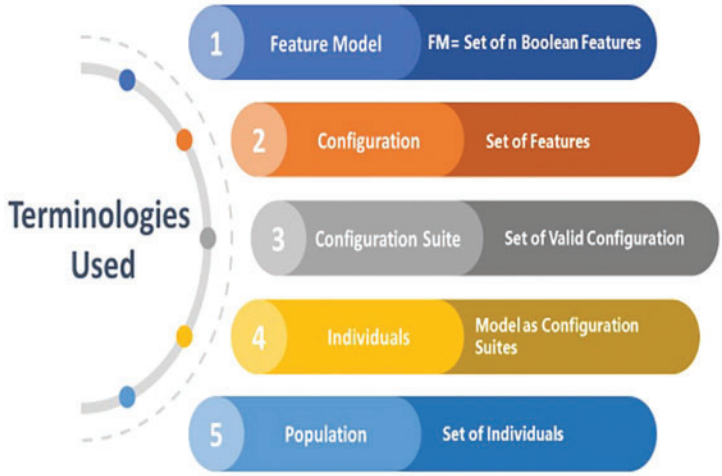


Figure 3: Terminologies used

Fig. 4 shows the workflow of the proposed approach. After the start of the optimization process, a feature model is selected, then SAT solver technique is applied to select the configurations from the feature model. After this a MOEA with parameter values is applied with the defined objective functions to run the experiments. We obtained and organized the results and best optimal values are selected from the whole process.

3.2 Definitions of Terminologies

There have been defined as different terminologies in this section and shown in Fig. 3.

- *The feature Model* consists of different features with some constraints that need to be satisfied.
- *Configuration* is a set of features that represents particular features of a product of product line.
- *Configuration Suite* shows a set of valid configurations.
- An *Individual* is represented as a configuration suite.
- A *Population* is shown as a set of individuals.

3.3 Experiments Organization

This section discusses the experimental setup. There have been selected three different feature models product lines of different sizes. These feature models have been selected from the SPLOT online repository [37]. The characteristics of these feature models have been shown in Tab. 1.

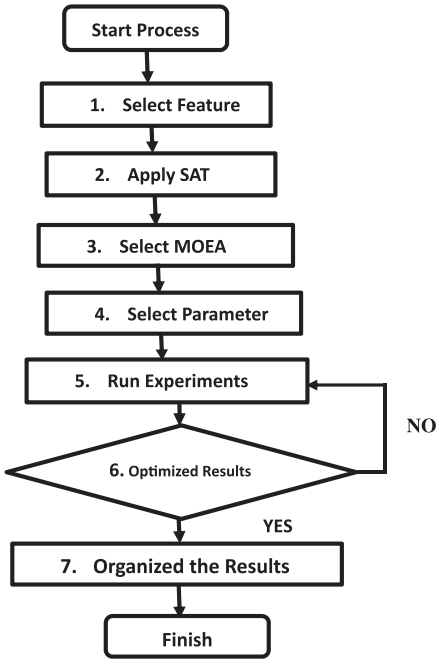


Figure 4: Workflow of the proposed approach

Table 1: Feature models attributes

Feature model	Features	Configurations	Number of pairs
Counter strike	24	18176	208
DS sample	32	73728	1448
Electronic drum	52	331776	2592

Tab. 1 shows the characteristics of feature models selected for experiments. For optimization of the three objectives, the three feature models have been selected named as Counter Strike, DS Sample, and Electronic Drum. Each Feature Model (FM) has a distinct number of characteristics, for example, the total number of features, configurations number, and the number of pairs mentioned in Tab. 1. For the Counter Strike feature model, it has 24 features, 18176 numbers of configurations, and 208 numbers of pairs. Similarly, other feature models' characteristics have been described in Tab. 1. This information about feature models is obtained from the SPLOT repository [37].

3.4 Experiments Results and Analysis

Tab. 3 shows the elapsed time to produce the solutions that are convergent at a particular generation for each feature model. The table represents the feature models, algorithms used for generation convergence, and elapsed time in milliseconds. From Tab. 2, the same settings as the number of generations, population size have been considered in Tab. 3, while running the experiment.

Fig. 5 shows each feature model convergent generation using different three MOEAs. For example, for the Electric Drum feature model, the solutions generated by the IBEA algorithm convergent at 65 generations. In the same for NSGAII and NSGAIII, the solutions are convergent at generations 74

and 492 generations respectively. It is observed from Fig. 5 that NSGAIII for all the feature models has the capability to convergent solutions at a maximum number of generations.

Table 2: Parameters settings for experiments

Parameters	Values
Size of population	200
Number of generations	500
Crossover rate	60%
Mutation	30%

Table 3: Elapsed time for solutions generation for each feature model

Feature model	Algorithm	Convergent generation	Elapsed time (in milliseconds)
Counter strike	NSGAII	142	57110
	NSGAIII	499	370024
	IBEA	74	97139
DS sample	NSGAII	96	388779
	NSGAIII	498	410025
	IBEA	82	164069
Electronic drum	NSGAII	74	456612
	NSGAIII	492	1691169
	IBEA	65	346673

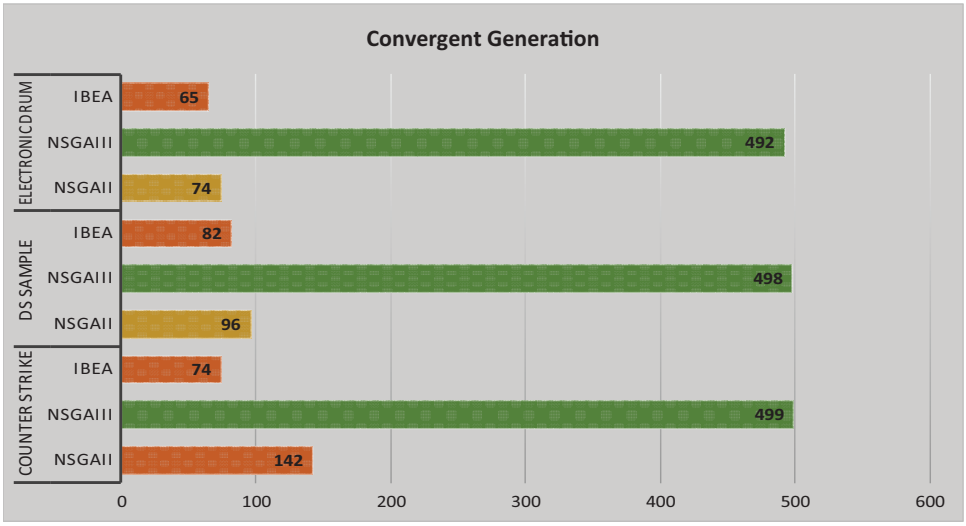


Figure 5: Generation convergent

Tab. 4 results are based on a set of better solutions created, generally following the idea of Pareto dominance [38]. This solution set builds estimation to the Pareto Front, where it also comprises of other

non-dominated solutions. The first column of Tab. 4 shows the feature models, the second column represents the number of solutions in the global Pareto front described as Global PF. The percentage calculation describes the percentage of solutions in the Local PF calculated by each algorithm. The number of Local PF solutions that exist in the front Global PF.

Table 4: Pareto fronts solutions for different feature models

Feature models	Algorithms			
	Global PF	NSGAII	NSGAIII	IBEA
		Local PF		
Counter strike	363	102 (28.1%)	189 (52.1%)	72 (19.8%)
DS sample	566	188 (33.12%)	292 (51.6%)	86 (15.91%)
Electric drum	623	212 (34.02%)	308 (49.4%)	103 (16.5%)

Tab. 5 presents the first 5 and last five fitness values generated by the three different MOEAs, the first five values are presented in italic font while the last five fitness values are shown as underlined. The best fitness values have been bold in this Table. Again, from Tab. 5, it is concluded that better results in terms of maximizing coverage, minimizing products, and minimizing cost are generated by the NSGAII and NSGAIII algorithms.

Table 5: First 5-last 5 (out of 500 runs) fitness values of FMs

No. Of values	Counter strike			DS sample			Electronic drum		
	NSGAII	NSGAIII	IBEA	NSGAII	NSGAIII	IBEA	NSGAII	NSGAIII	IBEA
1	*286.25*	*49.5*	*190.62*	*85.25*	*63.26*	*56.01*	***301.75***	***299.25***	*286.25*
2	*267*	***314.06***	*76.76*	*216.5*	*21.50*	***89.96***	*295*	*282*	*267*
3	*181*	*94.5*	*90.11*	*91.5*	*44.51*	*80.11*	*255.75*	*207.5*	*181*
4	*297.5*	*101.13*	*10.57*	*241.5*	*20.52*	*77.82*	*227.25*	*131*	***297.5***
5	*267*	*168.09*	*66.21*	***300.25***	*57.26*	*69.01*	*241.5*	*271*	*267*
6	258.75	75.22	**297.20**	227.75	286.25	17.87	290.25	287	190
7	269.25	152.13	73.07	170	267	41.94	256.25	189.75	261.75
8	177.5	134.2	26.03	211.75	181	31.58	256.5	286.25	261.5
9	**289.75**	155.15	49.66	172.25	**297.5**	51.14	249.5	267	253.75
10	241.5	144.5	66.21	289.25	267	23.86	261.25	181	280

Tab. 6 shows the hyper-volume indicator with their average values. This metric or indicator generate better quality values [33]. The Hyper-volume metric is used to present the volume of objective space and this space is dominated by non-dominated solutions of Pareto front as discussed in Section 2.3. From the Tab. 7, it is concluded that NSGAIII is the best algorithm to generate better values using the hypervolume indicator.

Table 6: Indicator hypervolume (HV) average values for three objectives

Indicator	Feature models	Algorithms		
		NSGAII	NSGAIII	IBEA
		Average-values of hypervolume (HV)		
HV	Counter strike	0.7627	**0.8338**	0.2449
	DS sample	0.6365	**0.7530**	0.2315
	Electric drum	0.7885	**0.8749**	0.3008

Table 7: Better MOEA algorithm with respect to indicator (HV)

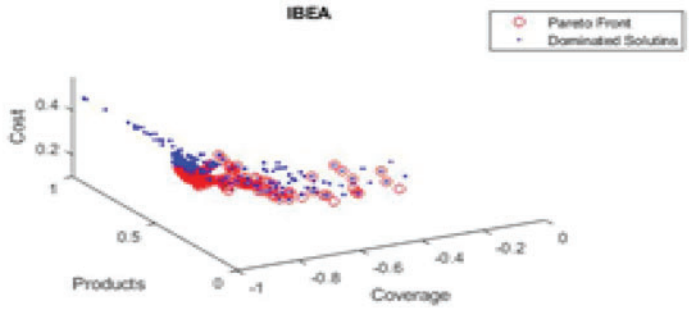
Feature models	Quality indicator Hypervolume (HV)
Counter strike	NSGAII, NSGAIII
DS sample	NSGAII, NSGAIII
Electronic drum	NSGAII, NSGAIII

Tab. 7 assists in analyzing the MOEAs performance in terms of producing the optimal results using a quality indicator (hyper-volume) for each feature model. Tab. 7 inferences are also determined from the results of Tab. 6. For all three feature models, it was discovered that NSGAII and NSGAIII algorithms originate the best values using the hyper-volume indicator. However, considering the Spacing indicator, the NSGAIII is only one algorithm that generates the optimal results for all feature models mentioned in Tab. 7.

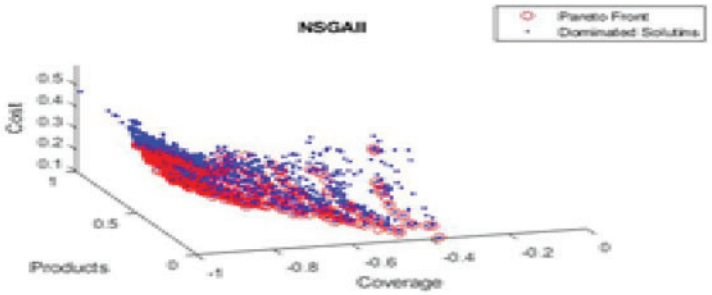
Figs. 6a–6c shows the Pareto Fronts solutions for Counter Strike feature model in the case of three objectives representation i.e., coverage, number of products, and testing cost.

4 Limitations of Work

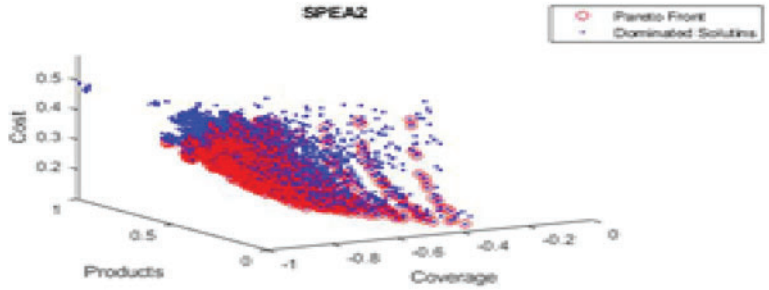
From the conducted experiments it is inferred that there exist potential threats to validity for this work. The generalization of the results presented in this paper can be considered a threat. For different feature models, there is an option to have different results. In the proposed work, three feature models are selected having different sizes to observe the generalization threat and also to ensure the validity of the results on these three selected feature models. Due to implementation, there is a chance to occur other threats, and these errors may influence to generate the optimized results. Hence, the implementation process is divided into subroutines to make sure of fewer errors. The repetition process of experimentation would help to minimize the risk of errors. The size of the product line is considered a risk for the study. However, to satisfy the configuration generation of feature models, it is revealed that good solutions can be generated by the proposed approach. The fitness evaluation procedures are to influence the execution time of the algorithms. It is always challenging to decide on the parameters of the algorithms. The recommendations found in [14] are truly followed to fine-tune the algorithms and mitigate the allied risks. Since fixed random variations are included in the search algorithms, the experiments are repeated 30 times, so that the likelihood results are obtained by chance may be reduced.



(a) IBEA generated Pareto Fronts for Three Objectives Optimization



(b) NSGAII generated Pareto Fronts for Three Objectives Optimization



(c) NSGAIII generated Pareto Fronts for Three Objectives Optimization

Figure 6: Number of pareto front solutions for counter strike feature model

5 Conclusion and Future Work

This research work discussed the idea of Pareto efficient multiobjective optimization to discuss the configuration generation problem of large systems. The proposed framework generates better results as compared to state-of-the-art tools [39,40]. The novelty of our proposed approach is to use the framework of three different MOEAs (NSGAII, NSGAIII, and IBEA) in a framework that works with SAT solver to optimize three objectives in the best tradeoff, while other researchers use a genetic algorithm (GA) for different objectives optimization [40–42]. The proposed approach can discover the optimal numbers of solutions in specific time duration (elapsed time in milliseconds as mentioned in Tab. 3). The proposed framework of MOEAs can work on small, medium, and large size SPL feature models. The significant number of non-dominated solutions generated by different MOEAs algorithms after experimentation have been presented in Tab. 4 to validate the novelty of the approach.

The estimation to the Pareto Front, called the Local Pareto front set, is established by these solutions. This is indicative of the fact that the complexity of the problem increases with three objectives because of the large search space, and it is essential to use this approach in such cases. The proposed framework is also capable of generating the optimal fitness values for the solution keeping the normalized values as shown in Tab. 5. These fitness values correspond to the three objectives (1) pairwise coverage (2) the number of products and (3) testing cost. Tab. 5, it is examined that the proposed framework can help to produce an optimal set of solutions with the help of fitness values.

For future work, besides the SAT solver there exists other solvers, evolutionary algorithms, and quality metrics that can be applied on other feature models with different parameter settings as compared to the proposed research to obtain the novel optimized results to enhance this proposed research work.

Acknowledgement: The authors are thankful to Umm Al Qura University, Makkah, Saudi Arabia, for supporting this research work.

Funding Statement: The authors would like to thank the Deanship of Scientific Research at Umm Al-Qura University for supporting this work by Grant Code: (22UQUyouracademicnumberDSRxx).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. A. Jamil, A. Alhindi, M. Arif, M. K. Nour, N. S. A. Abubakar *et al.,* "Multiobjective evolutionary algorithms NSGA-ii and NSGA-iii for software product lines testing optimization," in *6th Int. Conf. on Engineering Technologies and Applied Sciences (ICETAS)*, IEEE, pp. 1–5, 2019.

[2] A. Iglesias, M. Iglesias-Urkia, B. López-Davalillo, S. Charramendieta and A. Urbieta, "TRILATERAL: Software product line based multidomain IoT artifact generation for industrial CPS," in *MODELSWARD*, pp. 62–71, 2019.

[3] K. Pohl, G. Böckle and F. Van Der Linden, *Software product line engineering: Foundations, principles, and techniques*, vol. 1. Springer, 2005.

[4] J. D. McGregor, L. M. Northrop, S. Jarrad and K. Pohl, "Initiating software product lines," *IEEE Software*, vol. 19, no. 6, pp. 24–27, 2002.

[5] J. A. -C. Klünder, P. Hohl, N. Prenner and K. Schneider, "Transformation towards agile software product line engineering in large companies: A literature review," *Journal of Software: Evolution and Process*, vol. 31, no. 5, 2019.

[6] G. Guizzo, T. Elita, C. Silvia and R. Vergilio, "Applying design patterns in the search-based optimization of software product line architectures," *Software & Systems Modeling*, vol. 18, no. 2, pp. 1487–1512, 2019.

[7] J. C. Trigaux and P. Heymans, "Software product lines: State of the art," in *Product Line Engineering of Food Traceability Software*. FUNDP-Equipe LIEL, pp. 9–39, 2003.

[8] J. Debbiche, O. Lignell, J. Krüger and T. Berger, "Migrating Java-based APO-games into a composition-based software product line," in *Proc. of the 23rd Int. Systems and Software Product Line Conf.-Volume A*, pp. 98–102, 2019.

[9] N. Yousaf, M. Akram, A. Bhatti and A. Zaib, "Investigation of tools, techniques and languages for model driven Software Product Lines (SPL):A systematic literature review," *Journal of Software Engineering and Applications*, vol. 12, no. 7, pp. 293–306, 2019.

[10] K. C. Kang, J. Lee and P. Donohoe, "Feature-oriented product line engineering," *IEEE Software*, vol. 19, no. 4, pp. 58–65, 2002.

[11] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak and A. S. Peterson, *Feature-oriented domain analysis (FODA) feasibility study*, Software Engineering Institute, Carnegie Mellon University, 1990.
[12] A. Kicsi, V. Csuvik, L. Vidács, F. Horváth, A. Beszédes *et al.,* "Feature analysis using information retrieval, community detection and structural analysis methods in product line adoption," *Journal of Systems and Software*, vol. 155, no. 4, pp. 70–90, 2019.
[13] A. Vázquez-Ingelmo, F. J. Garcia-Peñalvo and R. Therón, "Taking advantage of the software product line paradigm to generate customized user interfaces for decision-making processes: A case study on university employability," *PeerJ Computer Science*, vol. 5, no. 2, pp. e203, 2019.
[14] M. A. Jamil, "Maintenance of software product line using software testing optimization techniques," Ph.D. dissertation, International Islamic University Malaysia, 2020.
[15] E. Engström and P. Runeson, "Software product line testing: A systematic mapping study," *Information and Software Technology*, vol. 53, no. 1, pp. 2–13, 2011.
[16] R. E. Lopez-Herrejon, J. Ferrer, F. Chicano, A. Egyed and E. Alba, "Computational intelligence and quantitative software engineering," *Computational Intelligence and Quantitative Software Engineering*, vol. 617, pp. 59–87, 2016.
[17] A. Arrieta, J. A. Agirre and G. Sagardui, "Seeding strategies for multi-objective test case selection: An application on simulation-based testing," in *Proc. of the 2020 Genetic and Evolutionary Computation Conf.*, pp. 1222–1231, 2020.
[18] Z. Anwar, H. Afzal, N. Bibi and H. Abbas, "A hybrid-adaptive neuro-fuzzy inference system for multi-objective regression test suites optimization," *Neural Computing and Applications*, vol. 31, no. 11, pp. 7287–7301, 2019.
[19] M. Mendonca, A. Wkasowski and K. Czarnecki, "SAT-based analysis of feature models is easy," in *Proc. of the 13th International Software Product Line Conf.*, pp. 231–240, 2009.
[20] Y. Xiang, X. Yang and Y. Zhou, "Going deeper with optimal software products selection using many-objective optimization and satisfiability solvers," *Empirical Software Engineering*, vol. 25, no. 1, pp. 591–626, 2020.
[21] M. A. Jamil, M. K. Nour, A. Alhindi, N. S. Awang Abhubakar, M. Arif *et al.,* "Towards software product lines optimization using evolutionary algorithms," *Procedia Computer Science*, vol. 163, no. 1, pp. 527–537, 2019.
[22] C. K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," in *Evolutionary Multi-objective Optimization in Uncertain Environments*. Springer, pp. 153–185, 2009.
[23] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.
[24] M. Z. Sahid, A. B. M. Sultan, A. A. A. Ghani and S. Baharom, "Combinatorial interaction testing of software product lines: A mapping study," *Journal of Computer Science*, vol. 12, no. 8, pp. 379–398, 2016.
[25] H. R. Maier, S. Razavi, Z. Kapelan, L. S. Matott, J. Kasprzyk *et al.,* "Introductory overview: Optimization using evolutionary algorithms and other metaheuristics," *Environmental Modelling & Software*, vol. 114, no. 11, pp. 195–213, 2019.
[26] X. R. Zhang, X. Sun, W. Sun, T. Xu and P. P. Wang, "Deformation expression of soft tissue based on BP neural network," *Intelligent Automation & Soft Computing*, vol. 32, no. 2, pp. 1041–1053, 2022.
[27] F. Glover, "Tabu search—Part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.
[28] X. R. Zhang, J. Zhou, W. Sun and S. K. Jha, "A lightweight CNN based on transfer learning for COVID-19 diagnosis," *Computers, Materials & Continua*, vol. 72, no. 1, pp. 1123–1137, 2022.
[29] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2013.
[30] K. Deb, A. Pratap, S. Agarwal and T. A. T. M. Meyarivan, "A fast and elitist multiobjective genetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[31] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms–Part I : A unified Formulation," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 28, no. 1, pp. 26–37, 1998.
[32] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Int. Conf. on Parallel Problem Solving from Nature*, Springer, pp. 832–842, 2004.
[33] E. Zitzler, D. Brockhoff and L. Thiele, "The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration," in *Int. Conf. on Evolutionary Multi-Criterion Optimization*, Springer, pp. 862–876, 2007.
[34] S. Yoo and M. Harman, "Pareto efficient multi-objective test case selection," in *Proc. of the 2007 Int. Symp. on Software Testing and Analysis*, pp. 140–150, 2007.
[35] M. W. Mkaouer, M. Kessentini, S. Bechikh, K. Deb and M. O. Cinnéide, "High dimensional search-based software engineering: Finding tradeoffs among 15 objectives for automating software refactoring using NSGA-III," in *Proc. of the 2014 Annual Conf. on Genetic and Evolutionary Computation*, pp. 1263–1270, 2014.
[36] A. S. Sayyad, T. Menzies and H. Ammar, "On the value of user preferences in search-based software engineering: A case study in software product lines," in *2013 35Th Int. Conf. on Software Engineering (ICSE)*, IEEE, pp. 492–501, 2013.
[37] M. Mendonca, M. Branco and D. Cowan, "SPLOT: Software product lines online tools," in *Proc. of the 24th ACM SIGPLAN Conf. Companion on Object Oriented Programming Systems Languages and Applications*, pp. 761–762, 2009.
[38] V. Pareto and A. S. Schwier, *Manual of political economy*, London: The Macmillan Press, 1927.
[39] A. Arcuri and L. Briand, "Formal analysis of the probability of interaction fault detection using random testing," *IEEE Transactions on Software Engineering*, vol. 38, no. 5, pp. 1088–1099, 2011.
[40] C. Henard, M. Papadakis, G. Perrouin, J. Klein and Y. Le Traon, "Multi-objective test generation for software product lines," in *Proc. of the 17th Int. Software Product Line Conf.*, pp. 62–71, 2013.
[41] R. A. M. Filho and S. R. Vergilio, "A mutation and multi-objective test data generation approach for feature testing of software product lines," in *2015 29th Brazilian Symp. on Software Engineering*, IEEE, pp. 21–30, 2015.
[42] M. A. Jamil and M. K. Nour, "Managing software testing technical debt using evolutionary algorithms," *Computers, Materials & Continua*, 2022.