EasyChair Preprint

№ 5974

Experimenting with Evolutionary Algorithms to Reduce Feature Model Configuration Steps

Dalia Owdeh and Abdel Salam Sayad

July 1, 2021

Experimenting with Evolutionary Algorithms to Reduce Feature Model Configuration Steps

Dalia Owdeh
Master Program in Software Engineering
Birzeit University
Ramallah, Palestine
dowdeh2@gmail.com

Abdel Salam Sayad
Master Program in Software Engineering
Birzeit University
Ramallah, Palestine
abed.sayyad@gmail.com

***Abstract*— In the software engineering world, software product lines constitute an approach to building reliable software systems. These use feature models to capture, develop, and document shared software for a base system. One of the main challenges when using feature models to derive new products configuration is a way of selecting a configuration that takes under consideration the minimum number of steps and minimum decision-making cost, taking into account resource constraints. To satisfy the challenges of optimizing the configuration selection technique, in this paper, we present an assessment approach that makes use of genetic algorithms to generate the best product configurations from feature models. Our empirical outcomes reveal the effectiveness of the proposed approach in obtaining product configurations that meet the feature model constraints with minimum steps and decision cost, consequently, assist customers in selecting the product configuration that fits their requirements.**

Keywords—Software engineering, software product line, feature model, product configuration, genetic algorithm.

I. INTRODUCTION

Designers define product families using feature models (FMs) within the so-called software product lines (SPLs) where the FM displays the product line features together according to certain values and constraints [1]. The importance of SPLs is coming from the need to reuse software systems in order to create new versions of products, so the products can be produced more easily and at a lower cost [2]. Depending on the SPLs the product is defined by identifying specific features from feature model through a process called feature configuration process [3], where this done by selecting and deselecting features from that model taking into account FM constraints govern.

A SPL product, defined by a FM with a feature set, is equivalent to a full product configuration where only the selected features are defined and the implicitly deleted features are deleted. This approach can be adopted in solving the problem of reducing the configuration steps of the multi-step configuration, such as producing a series of intermediate configurations in which the configuration path goes from specific feature to another. The mentioned configuration process provides an ordered list of configuration steps represent by the configuration path which identified the possible steps, so through it, we can go from the initial feature to the desired final feature without violating the FM constraints.

Software engineers face challenges of generating product configurations by using conventional mathematical configuration method (especially when configuring big size FM) , consequently, they face impossibility of evaluating the resulted huge number of the potential configurations due to the un-reasonability of the time taken to evaluate each of them to decide which is the best (according to number of steps and number of decision made) to be selected. Accordingly, we motivated to propose a technique to generate good quality solutions of configurations without generating all the potential configurations, therefore, we skip the problem of evaluating all possible configurations. To achieve that, we proposed using research-based software engineering (SBSE) techniques, specifically we used evolutionary genetic algorithms (EGA) to obtain a best set of configurations that fit our multi-objective optimization problem. The basic of evaluation process depended on trading off the two mentioned desirable incompatible objectives; reducing the number of configuration steps and reduce a decision cost taking into account the constraints involved in the FM.

Since there are no prior ways to solve such a problem, therefore, we have contributed to find a way to reduce configuration steps in exchange for a lower cost of decision-making, accordingly we made it easy for software engineers to generate a good quality configurations at a reasonable time regardless of the FM size. Therefore assisting the customers to select a configuration that fit their expectations, requirements, and preferences.

The rest of the paper is organized as follows: section 2 highlights the knowledge background is needed to accomplish our work; section 3 summarizes the related works for our research problem; section 4 describes experimental setup and procedures; section 5 demonstrates results; section 6 analyses the empirical results found; and section 6 presents concluding remarks.

II. BACKGROUND

A. Feature models (FMs)

Fig. 1 shows a FM called Banana model inspired by Sefika Efeoglu. A feature model is a tree of features, every node in the tree represents a feature and has one parent except the root feature that does not has a parent (e.g., 'BananaOs'). A terminal feature (e.g., 'BananaFS) is a leaf and a non-terminal feature (e.g., 'Browser) is an interior node of a feature graph [4], [5]. A nonterminal feature acts as synthesis of features that are its descendants.

A FM is systematic hierarchically and is graphically drawn as an AND-OR feature graph .there are a constraints called cross-tree constraints (CTC) which used to act non-hierarchical synthesis rules including mutual exclusion (excludes) , mutual dependency (requires) and mutual relationships [6] . There are six CTC in the Fig. 1.

Finding a configuration of terminal features help in derivation products from a FM [5], thus feature selection represents a specific product satisfying the customer's expectation and requirements. Additionally, we can calculate the actual resources consumption and product benefits from a set of terminal features [7].

A feature configuration is valid if a feature selection is valid, in other words, if the feature selection allowed within the FM constraints. The constraints of the FM defined by the relationships and connections between the features and its group of children.

The rules of feature selection may summarize as follows: if a feature selected, then its parent should be also selected. So, if a feature selected then all of its mandatory children participating in an And-group should be selected. Also, if a feature has an Or-group selected then at least one child should be selected for it. Likewise, if the feature selected in an Alternative-group then exactly one child should be selected. For example 'SupportModel' requires the selection of either 'Remote' or 'Other' Childs but not the both.

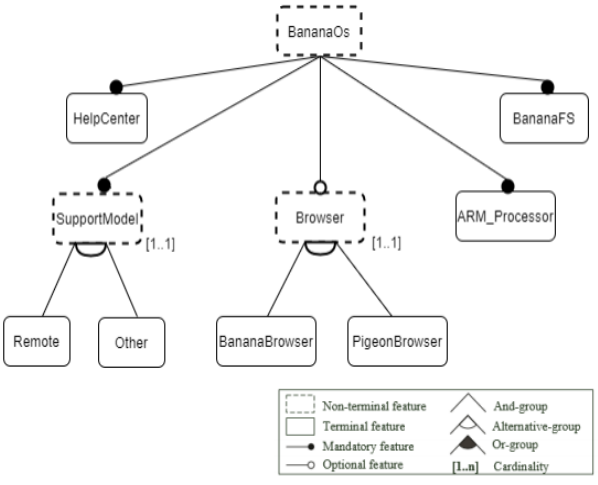

Fig. 1: Example of feature model for the BananaOS SPL

B. Multi-objective optimization problem (MOOP)

A solution to a multi-objective problem is the non-dominated Pareto front. In this research the problem has two-objective, and the solutions are represented by a curve described by an infinite number of points [8]. We have been tracking the Pareto using the direct way, which is by tracking points on the peaks, taking into account Keeping the minimization of both objectives as well as possible, within the framework of the trade-offs between the two objectives.

C. Evolutionary multi-objective optimization algorithms

Evolutionary algorithms (EAs) are areas of decision-making with multiple objectives, where the best decisions must be made with trade-off between different objectives. This algorithm is very attractive in analyzing multi-objective problems using classic methods. The algorithms start their work with a set of randomly selected solutions called initial population. Next generations begin to arise by Offspring process through some operators such as mutation, crossover and selection [9].

1) Genetic algorithms (GAs)

Researchers proposed the concept of heuristics to override unreasonability problem results. Using algorithms based on heuristics does not mean obtaining an exact optimal solution, but it will provide a set of close solutions within an acceptable and reasonable time. To reach high-quality solutions in multi-objective problems, GAs adopt the principle of executing operations several times and do not stop until reach the constraints completion, including the predetermined number of iterations. The principle of working with multi-objective algorithms is to evaluate each objective in order to compare the resulted solutions and determine which are best. The reason for comparing two objectives is to determine whether one of them dominates the other, to ensure that, all the objective's values must be the best possible (according to maximizing or minimizing the objectives) or for a specific solution to be equal to another solution, to confirm that it dominates. Thus the algorithm compares all the solutions and finally gives a list of the solutions that are not dominated [10]. GAs belong to a large category of EAs, it generate a solutions for multi-objective optimization problems through its natural evolution by applying some operations, Fig. 2. In this work, we used Non-dominated sorting GA II to optimize the research problem.

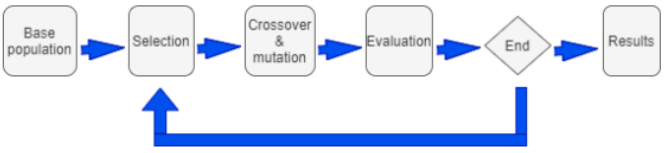

Fig. 2: Main steps of GAs

a) Non-dominated sorting Genetic algorithm II (NSGA-II)

Debt and et.al [11] developed NSGA-II in 2002 to optimize the multi-objective problem as they combined the genetic algorithm with a non-dominant sorting

approach, to take advantage of this sorting's role in multi-objective optimization. This algorithm contains three parts for selecting a new generation of individuals: non-dominant sorting, density estimation, and crowded comparison [12] .

D. *Problem definition*

To obtain configurations that contain the minimum number of steps within an acceptable time, we must order a features in a certain way, therefore the concept of permutation is the primary solution key in evaluating configurations by achieving the best percentage of trade-off between the objectives, which represent the inputs of the problem:

- Configuration step: it is a set of features that have relationships with each other, if these features are arranged consecutively within the configuration, then they follow the same step, so, feature permutation affects the number of steps and the number of features within each step, where whenever the number of features increases within one step, whenever the number of the configuration steps decreases.
- Configuration decision: it is a decision of selecting or deselecting features according to user preferences, such decisions must respect the constraints imposed by the FM [6]. In this work, we pay more attention to the totality of decision evaluations through a series of steps that make up the configuration.

We have adopted a specific approach in counting the configuration steps and evaluating the cost of decision making within the configuration. Our problem adaption represented via two main processes, thus they are:

- Step counting: This process explains how to count steps within a configuration. There are constraints governing the step selection , in addition to the order of the feature in that configuration which governs the length of the step.
- Decision evaluation: It depends on two metrics; the length of the step (the number of features in the step),and the number of decisions within a single step. Decision evaluation calculated by : *Step length * Number of decisions within a step*.

According to the above evaluation, whenever the step length increase, whenever the decision evaluation increase also, which what we call the decision cost for that configuration.

In order to deal with our research problem, we used the official website of the SPL, where we downloaded the FMs that we wanted to use as a data set represents the problem, then we identified the problem through the following steps:

- We converted the FM from an XML file to a feature model tree, thus the data formed became readable.
- We excluded basic features which found in each configuration and kept them in a different file (called basic skeleton) because they are not considered as a relevant part of the evaluation.
- We took the features that were left and kept them in a list.
- We counted the remaining features where their number represents the length of the configuration path.
- We identified fitness functions that perform an evaluation for each objective as follows:

  Objective O1: Minimize number of configuration steps:

  $$Minimize\ |\{f1,f2,\ldots,fa\}1\ ,\ \{fa+1,fa+2,\ldots\}2,\ \ldots\ldots\ ,\{\ldots\ldots\}S| = S$$

  Where S=Number of steps and {...} 1 ∩ {… }2….∩ { ..}s = Φ

  And {… }1 ∪ {…} 2 ….∪ {…} s = {all features} And 1 <= S <=n where n number of features

  Objective O2: Minimize number of decision taking. It rely upon the decision in the group that contains a greater number of features is more than the decision in the smaller number of feature group:

  $$Minimize\ \sum_{i=1}^{n}(d_i l_i)$$

  Where n= number of the groups

  And d= number of decision in the group

  And l= length of the group And 1 <= l <=n

III. RELATED WORKS

Many literary focused on using EAs to solve problems related to SPL, as a number of researchers applied GAs to improve feature selection in the SPL. Kumari et al. [13] Presented feasible and efficient GA for automated product configuration based on the most reusable feature in order to derive an optimal product configuration. They found that GA can produce effective, efficient and fast results rregardless of the increase in the size of the FM. Also, Guo et al. [14] proposed using GA for features selection in order to derive automated product derivation in SPL, where their derived algorithm gave 45-99% a better solution and reduced 45-97% time consumption than exact and heuristic algorithm. The same challenge was tackled using different technique, as White et al. [15] provided a polynomial time approximation algorithm for selecting a highly optimal set of features that adheres to a set of resource constraints, their approach required significant computing time for large-scale problems, In addition for a higher resource tightness required , therefore, their approach has not proven as an effective approach to optimize the problem compared to EAs usage.

In the same context of using GA, Alidra et.al [16] highlighted on reuse in software systems, they developed their solution at design time, as they tackled the reusable components by focusing on the concepts of them. They assigned the software

product family of GA and derived GA line from it. On the other way, Afzal et.al [17] used algorithms by defining a real-world feature model and optimized different sizes of small and large FMs, as they used GAs to reduce inconsistencies in limitations between features.

Some researchers tackled the product configuration issues far from EAs, where White et al.[21] tackled the challenge of creating SPL configuration that meets arbitrary software requirements, they propose to do contributions on multi-step configuration study for SPL, the first one by create a formal model and map it with constraints satisfaction problem (CSP), the second one is show how the solution of CSP problem can be derived by constraints solver, and the third contribution is to do empirical results showing that their CSP-based technique can solve multi-step configuration problems involving hundreds of features in seconds. Authors could prove scalability for their model by prove the ability to scale to hundreds of features and multiple steps.

Bagheri et al. [22] Searched about product line configuration into one concrete product based on the stakeholders requirements. They proposed building theoretical and technological tools to help the stakeholders understanding the interactions of a product line features, also, gathering the usage of each feature for the stakeholders then help them making decisions in order to dynamically building a decision model. Their approach were be able to facilitate the application engineers' task by helping them understand the utility of the features available in the product line and identifying their preferences.

Tan et.al [23] tacked the issue of the relationships among configurable features that need to be considered to select the desired product features. They proposed to reduce configuration features by selecting a small set of main features, thus configuring a new product should depend on this set which is based on features dependencies, therefore, the decisions made for that set mean decisions for the rest of product line features. Authors approach led to improve product line configuration, in addition to that, using their approach is less likely to make configuration mistakes because of depending on relationships feature dependencies.

Chohan et al. [18] addressed two challenges in developing software engineering, namely, the software 'lack of optimization and re-use features, so, they proposed a tool for improving the SPL by applying multi-objective optimization techniques. Sharma et al. [19] proposed using SPL business process modelling notation. They introduced different dimension of BPMN and extended dimension to configurable BPMN (C-BPMN) used to model many variations of a project system. They have developed framework and utilized it to realize SPLs using BPMN. Also, Sharma et al. [20] continued their previous work and proposed a new version of BPMN, called C-BPMN. Companies can specialize configuration process model as they need, also, they have done a comparison between configurable bbusiness pprocess mmodeling nnotation and configurable Event Process Chain (C-EPC).

IV. EXPERIMENTATION

A. Setup

The objective of this step is to implement the problem of reducing FM configuration steps and its resolution with genetic algorithms using the JMetal library version 5.8. An important major step in preparing for the experiment, is to define the research questions that the experimental study aims to investigate, as follows:

RQ1: how to get the best configuration in SPL?

Using GAs, we can get a high-quality solution, and in some cases, it may be the best solution. We experimented with JCS FM, which represents a small size FM. It produced 96 valid configurations, which are difficult to evaluate to determine the optimal solution one, otherwise, experimenting by using GAs, will produce the best set of solutions as may produce the optimal solution also.

RQ2: can we generate the optimal solution?

The optimal solution can only be calculated with trivial FM in which finding optimal solutions does not affect the development of a software or system, therefore we have tended to work in another way, which is GAs

RQ3: what is the best approach to reach the best solution? Why?

GAs are the best way may get to reach the best solutions that are close to the optimal solution. If we assume that evaluating one configuration in GAs consumes approximately the time that it consumed to evaluate another configuration using traditional computational methods, the process of evaluating all the configurations occupied in this model with the two methods is basically incomparable due to the unequal time spent to complete the assessments as a whole, in addition, the size of The feature model does not affect the evaluation execution time and consumes only seconds.

RQ4: Why we used NSGA-II?

NSGA-II algorithm is good solution for multi-objectives problems, as it is available, works very quickly , and depends on elitism in selecting and multiplying individuals, so it was more effective and safest for us to resort to this algorithm to optimize the problem of reducing configuration steps.

RQ5: How to evaluate the configuration in SPL?

The evaluation of configurations in terms of the number of steps and the number of decisions depends on the permutation between features and their order within the configuration, where we devised an approach to count the steps within the configuration while evaluating the cost of decisions- making at the same time.

B. Implementation

we expanded the infrastructure of the library to comply with the problem of reducing the configuration steps in the FMs, accordingly, we established the following structures; Fig. 4:

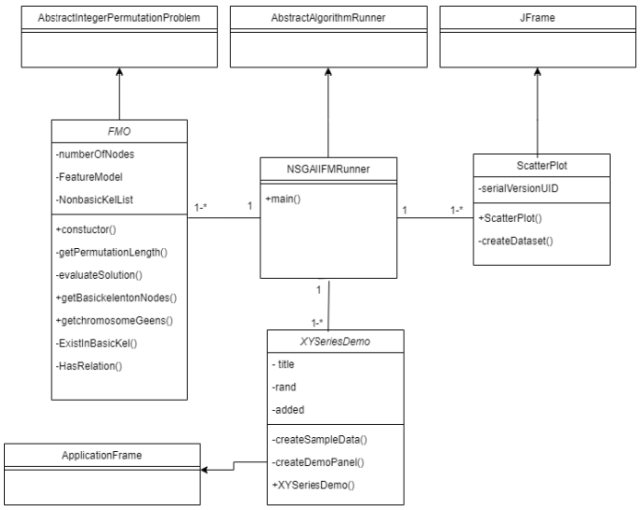

Fig. 4: class diagram of the problem domain

To evaluate the two competitive objectives of the research, we developed a technique for counting the steps and evaluating the cost of decisions making. Basically, we need to go over permutation concept, thus it consider as a key for our approach. Previously, in the problem definition section, we explained the evaluation approach , here we implement this, see figure 6.

Simplified Evaluation Algorithm

Data Solution: The solution to evaluate,FeatureModel

Result: Fitness values for objectives

```
Initialize Variables
Create StepGroup contains first gen
For each gen in Solution
        If gen has relation with StepsGroup
            Add Gen to StepGroup
            Calculate Number of Decisions in current group
        Else
                Increase Numberofsteps by 1
                Calculate TotalGroupDecision
                Increase SolutionDecisionRank by TotalGroupDecision
                Clear StepGrou
         End If
End For
Return Numberofsteps, SolutionDecisionRank
```

Fig. 5: implementation for the objective evaluation idea

C. Input files

In order to facilitate the experiments, we have implemented the functionality in charge of reading the FM, as it converts it from an XML file to another readable file. While the problem definition is formulated therefore the input data of chromosomes will be ready for GA use. The empirical study consists of three realistic FMs differs in feature size and constraints: (JCS; 12 features, 25% ECR), (MLX; 37 features, 40% ECR) and (Windows8o; 451 features, 30% ECR).

D. Output

In order to achieve the experiment objectives, the output of the experiment will be an array of two elements, each element represents an objective, the first represents the number of steps while the second represents the cost of decisions, these results act as fitness values and are store in a file called (FUN.tsv), the number of arrays in the file represents the population size that was predefining in the experiment. Therefore, the algorithm will keep the best solutions corresponding to the mentioned fitness values and store them in a file called (VAR.tsv) . The output of the experiment as a whole will be stored in a file called (allexpVal.tsv) , where the number of resulting arrays represents the number of experiment repeated time. In addition, we defined a method to decode the configuration as we use the indexes of them in the GA, this method used a file called (GeenArrayForDecode.tsv) , to return the feature names.

E. Experiment protocol

As an attempt to solve the research problem which the researchers have not yet dealt with, nor by any method, we devised a specific approach that takes into account the FM constraints to count steps and evaluate decisions, also, to prove the effectiveness of the approach, we defined the problem and linked it with runner algorithm then conduct the experiment three times, each with different FM. Each FM were executed by the same algorithm; NSGA-II. the experiments were executed 10000 evaluations of 100 population size, thus we were could catch the best 100 individuals for every FM, which means the best 100 configurations.

V. Results

The experiment results presented using the JFreeChart library which provides the chart frames, as follows:

Experiment 1: was used JCS FM , the distribution of the best solutions appear throughout the pareto fronts , Fig. (6, 7):

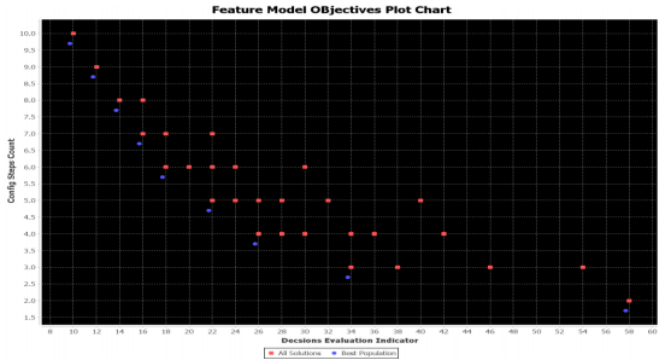

Fig. 6: the solutions distribution and Pareto front when JCS FM were applied.

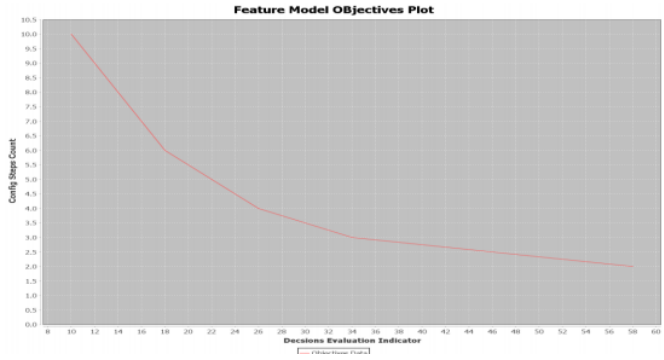

Fig. 7: the Pareto front when JCS FM were applied.

Experiment 2: was used MLX FM, the distribution of the best solutions appear throughout the Pareto fronts , Fig. (8, 9)

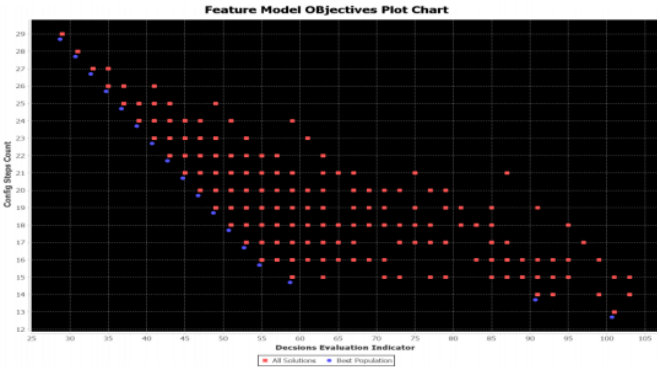

Fig. 8: the solutions distribution and Pareto front when MLX FM were applied.

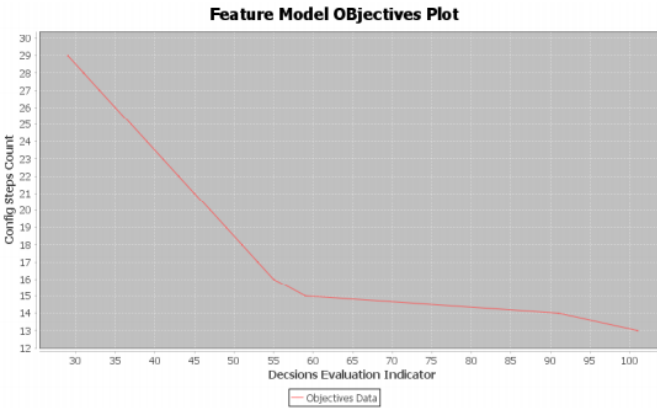

Fig. 9: the Pareto front when MLX FM were applied.

Experiment 3: was used Windows8o FM, the distribution of the best solutions appear throughout the Pareto fronts , Fig. (10, 11):

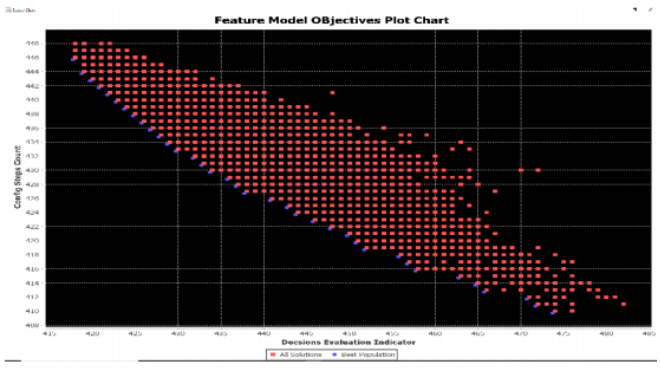

Fig. 10: the solutions distribution and Pareto front when Windows8o FM were applied.

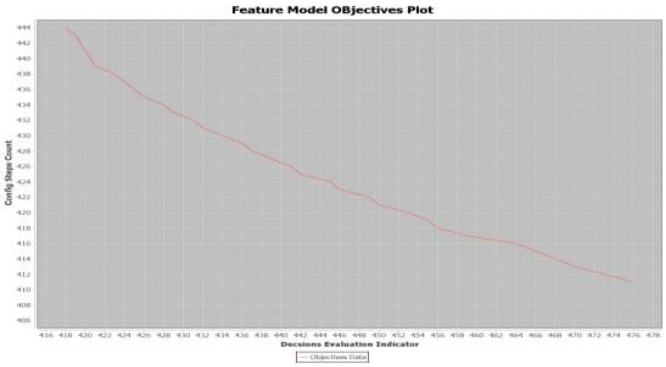

Fig. 11: the Pareto front when Windows8o FM were applied.

VI. Analysis

Through the previously resulted plots, we noticed that the optimal solutions are focused at the bottom of the distribution points, because whenever go towards the Centre from X-axis, the decisions will be less cost, also, whenever go down towards the Centre from Y-axis, the configuration steps will be as less as possible. In other words, if a line is drawn at an angle of 450 from The two-axis then the intersection of the combination of the points with the drawn line from the bottom will represent the optimal solutions, the following figure shows the idea intended, where the optimal solutions were identified at the intersection of the white line with the solutions, Fig. 12.

Figure 31: The idea of identifying the best solution combinations at the intersection of the white line with the solutions

On the other hand, through the previous three experiments, we noticed that the size of the FM almost does not affect the execution time, which leads us to adopt the idea of GAs to solve this problem, On the contrary, if we wanted to extract all the configurations from a non-trivial FM, then this may take an unreasonable time in execution, even if performed using a supercomputer.

Although the comparison is not acceptable here because the sources are mismatches, although, it is good to put table showing the difference between the use of GAs and the conventional mathematical method (which based on the factory method) in finding a set of the optimal solutions to the research problem. The table shows impossibility to generate all the configurations from a non-trivial FM using the conventional method at a reasonable time, consequently, evaluating configuration in this way is illogical and unacceptable. See Table 1 below.

TABLE I. EVALUATING CONFIGURATION USING GA VS MATHEMATICAL METHOD

FM	No. of features	GA	Mathematical method
JCS	12	2.777 s	16.7 month
MLX	37	2.9075 s	81 e^{18} year
Windows80	450	3.718 s	1.7 e^{10000} year

VII. CONCLUSION

After experimenting and analysing the research problem, we concluded the following remarks:

- Getting all constraint-meet configurations to evaluate them according to certain multi objectives can't be solved using the conventional mathematical method for the unreasonable time needed to generate all the configurations where the mentioned technique is more unacceptable with large size FM.
- The GAs is an efficient choice to facilitate selecting a suitable product configuration because it is produce a good quality solutions in very short time.
- The GA may not produce an identified optimal solution but at least produces a set of the best solutions in very short time.
- For each run of GA, the problem may produce a deferent solutions but all the solutions in each execution nearly equal in quality.
- The GA nearly take the same execution time even the FM size changed because the execution time depends on the number of generation required to produce a solution not on the size of the problem.
- The NSGA-II multi-objective algorithm makes an effective trading-off between objectives especially when objectives values are increased or decreased in the opposite way.

ACKNOWLEDGMENT

This research was supervised by a software engineering program at Birzeit university through Prof. DR. Abdel Salam Sayad. The assistance provided by him was greatly appreciated, so I wish to extend my special thanks to him for helping me until finalizing the work.

REFERENCES

[1] Collis, J. and R. Hussey, Business Research: A practical guide for undergraduate and post graduate students. Second edition ed. 2003: Palgrave Macmillan.

[2] Robinson, G., Methods and Techniques in human geography. 1998, Cheschister: Wiley & sons (Robinson, 1998)

[3] B.V, E. (2020). feature-configuration. sciencedirect. Retrieved from https://www.sciencedirect.com/topics/computer-science/feature-configuration

[4] Batory, D. (2005, September). Feature models, grammars, and propositional formulas. In International Conference on Software Product Lines (pp. 7-20). Springer, Berlin, Heidelberg.

[5] Thum, T., Batory, D., & Kastner, C. (2009, May). Reasoning about edits to feature models. In 2009 IEEE 31st International Conference on Software Engineering (pp. 254-264). IEEE.

[6] Feature_model. (2020). Retrieved from wikipedia: https://en.wikipedia.org/wiki/Feature_model.

[7] White, J., Dougherty, B., & Schmidt, D. C. (2009). Selecting highly optimal architectural feature sets with filtered cartesian flattening. Journal of Systems and Software, 82(8), 1268-1284.

[8] Hatzakis, I., & Wallace, D. (2006, July). Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach. In Proceedings of 52 the 8th annual conference on Genetic and evolutionary computation (pp. 1201- 1208).

[9] Farmani, R., Savic, D. A., & Walters, G. A. (2005). Evolutionary multi-objective optimization in water distribution network design. Engineering Optimization, 37(2), 167-183.

[10] Elvassore, V. (2016). Experimenting with generic algorithms to resolve the next release problem (Master's thesis, Universitat Politècnica de Catalunya).

[11] 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.

[12] Tian, Y., Wang, H., Zhang, X., & Jin, Y. (2017). Effectiveness and efficiency of non-dominated sorting for evolutionary multi-and many-objective optimization. Complex & Intelligent Systems, 3(4), 247-263.

[13] Kumari, A. C. (2018). Feature selection optimization in SPL using genetic algorithm. Procedia computer science, 132, 1477-1486.

[14] Guo, J., White, J., Wang, G., Li, J., & Wang, Y. (2011). A genetic algorithm for optimized feature selection with resource constraints in software product lines. Journal of Systems and Software, 84(12), 2208-2221.

[15] White, J., Dougherty, B., & Schmidt, D. C. (2009). Selecting highly optimal architectural feature sets with filtered cartesian flattening. Journal of Systems and Software, 82(8), 1268-1284.

[16] Alidra, A., & Kimour, M. T. (2014). Towards a Software Factory for Genetic Algorithms. International Journal of Computer and Electrical Engineering, 6(1), 44.

[17] Yadav, H., Kumari, A. C., & Chhikara, R. (2020). Feature selection optimisation of software product line using metaheuristic techniques. International Journal of Embedded Systems, 13(1), 50-64.

[18] Chohan, A. Z., Bibi, A., & Motla, Y. H. (2017, December). Optimized software product line architecture and feature modeling in improvement of SPL. In 2017 International Conference on Frontiers of Information Technology (FIT) (pp. 167-172). IEEE.

[19] Sharma, D. K., & Rao, V. (2014, February). Configurable business process modeling notation. In 2014 IEEE International Advance Computing Conference (IACC) (pp. 1424-1429). IEEE.

[20] Sharma, D. K., & Rao, V. (2015, May). Individualization of process model from configurable process model constructed in C-BPMN. In International 53 Conference on Computing, Communication & Automation (pp. 750-754). IEEE.

[21] White, J., Dougherty, B., Schmidt, D. C., & Benavides Cuevas, D. F. (2009). Automated reasoning for multi-step feature model configuration problems. In SPLC 2009: 13th International Software Product Line Conference (2009), p 11-20. ACM.

[22] Bagheri, E., & Ensan, F. (2014). Dynamic decision models for staged software product line configuration. Requirements Engineering, 19(2), 187-212.

[23] Tan, L., Lin, Y., Ye, H., & Zhang, G. (2013, January). Improving product configuration in software product line engineering. In Proceedings of the Thirty-Sixth Australasian Computer Science Conference-Volume 135 (pp. 125- 133).