Binary Multi Objective PSO and GA for Adding New Features into an Existing Product Line

Abdullah Al Mamun, Faisal Al-Naseef

Collage of Computer Science & Engineering
King Fahd University of Petroleum and Minerals
Eastern Province , Dhahran 31261, Saudi Arabia

***Abstract*—Software product line evolution is a decision-making problem where it is determined that which features are the best candidates for the different product. In this paper, we propose for finding qualified features for next release. In this case, the qualification of features is determined by the product value, product integrity, and AND dependencies between features. This study may help a software manager in feature selection. For this purpose, first, we encoded the feature model into a binary string. Then we able to find possible alternatives by using binary multi-objective genetic search algorithm NSGA-II and particle swarm optimization PSO. From these options, the product line manager can quickly select the best-fitted solutions according to the requirements. In this implementation, a feature model ONLINE ELECTRONIC SHOPPING from Software Product Line Online Tool SPLOT is used where the total number of features is 290. Assume that 136 features are already implemented in the existing software. Finally, 30 features among 154 are selected in five product variants successfully. The empirical result shows that GA performs better than PSO.**

***Keywords*—SPL, NRP, PSO, new features.**

I. Introduction

The companies involved in software development usually deal with many constraints while they want to maximize profits and satisfy customers as many as possible. In order to achieve this goal, software development companies need to create software product lines and seek to identify the most important features that must or should be added for existing systems in the next release. This is known as the next release problem: what features in their customer′s systems would be developed first in the next release, given some constraints coming from available resources and customers. Accordingly, they always need to compromise between the cost it causes and the achievement of what the customers require. The useful approach for addressing various customer′s demands of customized software systems is to use SPL. Software Product Line is the art of the reuse in which a considerable number of software tools, libraries, programming codes and designs can be reused. The aim of SPL is to construct a group of products that they most probably may have common features as well as diverse features in which those products are to be different from each other. In addition, the need for providing various products is to satisfy segments of customers and in order to reduce the cost with overcoming the challenge of speeding up the products that are about to be released in the market as possible. Deploying products is not a real problem as such. However, making decisions about what new features should or must be added to existing products for being developed is the most crucial matter. Hence, the question is how to select which products to be best candidates for achieving new features.There should be a necessary way in which the Next Release problem in SPL can be highly reduced as well as its results can be systematically produced in such a way reliable results can be achieved. The objective is to reach to a result that is sub optimal since our solution depends on meta-heuristics optimization techniques. Hence, what will be using for solving such a problem is a decision support software tool that, we think, may help to obtain promising results for NRP in SPL.

In this work, we are going one-step further from the work that was published in [1] by using a different function model as well as considering a required dependency. Moreover, the PSO algorithm, which stands for Particle Swarm Optimization, is used to utilize this technique in order to produce valuable outcomes to solve a research statement problem, which is "how new feature requests should be integrated into an existing SPL".

A. Next Release Problem in Software Requirement Engineering

Software requirement engineering is a large field. Next release problem is an open question and closely related to this area. Software requirement engineering discusses the various method which is used to collect customer requirements and manage them. Also, they discuss that how can maximize the client satisfaction. Similarly, next release problem is a solution which tells you that how and which requirements are going to release in next deadline, which is playing an important role in satisfying customer.

II. Related Works

A. Next Release Problem

The next release problem is firstly proposed by Bagnall et al [2]. It is the problem in which software companies are involved in improving customers software systems according to their demands by selecting the most important features they need among many others to be added in the next release of their new systems. That is, the goal is to find the ideal set of requirements that compromise between the cost it causes, resource constraints, requirement inter-dependencies, and the achievement of what the customers requires. To address this problem, they formalize the problem into a mathematical model, and try to get the best solution for that problem. When building a model, various considerations can be taken into account. In their work, prerequisites of features, features nature with respect to time needed and required effort, and customer′s value to the company are considered with a high-level intention. They also found one major challenge when addressing this problem is that the problem is proven to be NP-hard problem, which means it would take a very long time to come up with an exact solution of a given mathematical model. Therefore, meta-heuristic methods such as genetic algorithm are needed and more popular. Zhang et al [3] generalized the original NRP problem, which is a single-objective optimization into a multi-objective NRP problem, defined mathematically in [4]. It is argued that the multi-objective formulation is more realistic, since requirements engineering is characterized by the presence of complicated or contradictory demands from the customers.

B. Release Problem in Software Product Lines

Nowadays, Software Product Lines are getting popularity because of the benefits that it provides and many papers have been published regarding its own problems. Software Product Line (SPL) development is a new approach to software engineering that aims at the development of a whole range of products. This approach has been shown to deliver benefits in terms of effort reduction, quality improvement, and time-to-market reduction [5]. People are trying to address problems appearing from SPL, including release problem. Since SPL considers nearly as a new area in Software Engineering, most problems that were solved particularly for single separated products need to be also resolved in the new area. NRP model for SPL are generally more complex than the original release problem model. In addition to cost and feature values, several considerations might be taken. For example, Karimpour et al [1] propose “product integrity” as one aspect that needs to be maximized in SPL. In their work, integrity is defined as the degree to which the features are perceived as cohesive, and measured by nearest common predecessor (NCP) between two concrete features. Thum et al[6] considered specialization, refactoring, and generalization relationships between features. Ullah et al[7] proposed that “structural impact” should be minimized during updating a current system. A lot of works related to NRP in SPL use the genetic algorithm and its variations when solving their models. Guo et al[8] mentioned GA is rather generic and may not be completely suitable for SPL feature selection problem, so they proposed a modified GA method called GAFES (GA-based AI Approach to Optimized Feature Selection in SPLs). Many other works create multi-objective optimization models for SPL and use the NSGA-II algorithm [9] to solve them [10], [5], [7], and [3]. Based on the work by Karimpour et al[1], we propose a model that considers AND feature dependencies. In their work, those dependencies are ignored. Some feature configurations coming from their method may violate dependencies between features. By considering the AND feature dependencies, our model would suggest more correct feature configurations. Seeing some alternatives other than NSGA-II, we noticed that several other search-based methods are still not well explored in software engineering subject. We found that PSO algorithm is more efficient than GA in many works in petroleum engineering, mechanical engineering, and so on [9], [10], [11], and [12]. Up to our knowledge, no one has done PSO multi-objective algorithm to solve release problem, so our contribution will be in an application of multi-objective PSO algorithm [13] for SPL release problem.

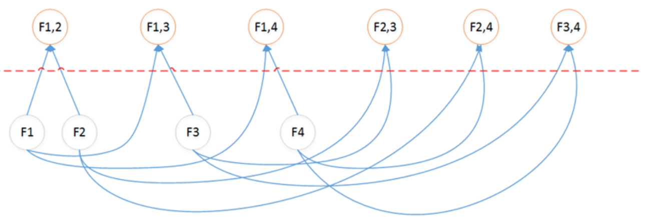


Fig. 1. Presentation of features dependency.

III. Problem Formulation

Our goal is to find the best feature configuration of product variants given a feature model. Building a model from [1], we will measure how good a feature configuration is based on its product value and integrity. Assume the set of all product variants is called P where $p_i \epsilon P. P$ is the number of variants. F is the set of all features and $f_i \epsilon F. w_i$ is the weight of product i. $V(p_i)$ and $D(p_i)$ return the value and integrity of product p_i. The problem is to maximize the following objectives:

$$r_{value}(P) = \frac{1}{\sum_{j=1}^{p} w_j} \sum_{i=1}^{p} w_i V(p_i) \tag{1}$$

$$r_{intg}(P) = \frac{1}{\sum_{j=1}^{p} w_j} \sum_{i=1}^{p} w_i D(p_i) \tag{2}$$

r_{value} is the weighted average value of the all variants in the product line. In real world product lines, different variants may have different importance. This is specified using w_i which is the weight of product p_i

A. Product Value

One of the primary targets of system analyst is the achieving maximum product value. A priority of the feature or vote by the stockholder can be the trivial approach to assign product value. This value may depend on selecting of other feature. For example, the value of one feature can increase or decrease concerning the value of other feature. So, if it is relative, the value can be dogged by choosing of other features and if not then we use the popular stockholder vote by default. However, instead of just summing up of feature values that are implemented to evaluate a product value, we also consider AND dependencies between features. We assume there will be AND feature dependencies between two components at most, which means no more than two features have a required dependency for each required dependency. To evaluate a product value, the total number of independent features after performing required dependencies at most need to be counted first. To illustrate, the features will be divided into two names, fundamental features (up) as individual features and dependent features (down) as combined features. The total numberof at most dependent features can be defined by the following formula:

$$C_(a, b) = \frac{a!}{b!(a! - b!)} \tag{3}$$

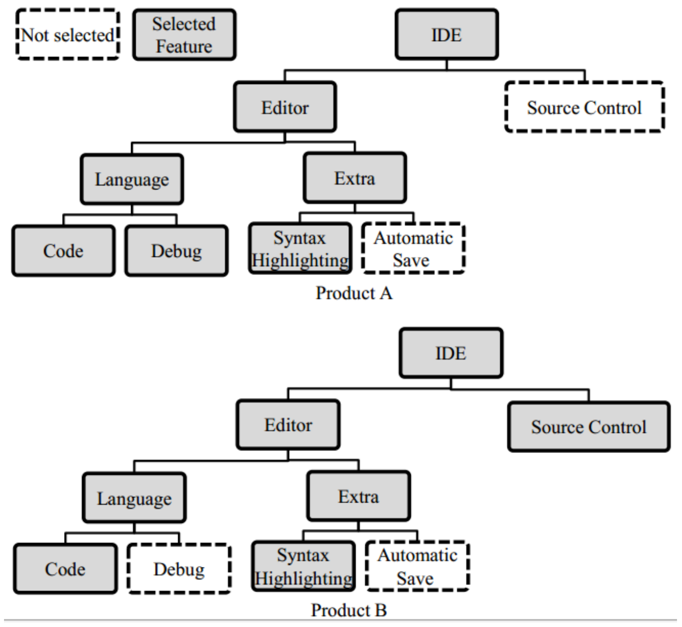


Fig. 2. Two products of a SPL. Product A has higher integrity compared to product B because of the distribution of the concrete features in sub trees

Where a, b represents the total number of fundamental features and the number of dependency between features respectively. As illustrated in Fig. 1, there are four fundamental features, which are F1, F2, F3, and F4. When combining each two features with one required dependency, there will be six combinations as a result, which are $F_{1,2}, F_{1,2}, F_{1,3}, F_{1,4}, F_{2,3}, F_{2,4}$ and $F_{3,4}$ at most. Therefore, for 5 fundamental features the maximum total dependent features would be 10, 15 for 6 and 21 for 7 as more examples. Hence, only dependent features can be used for calculating the objective functions of the product value.

B. Product Integrity

According to the definition in [**?**], the product integrity is a degree to which the feature of a product are seemed as interconnected. It is observed that the products with higher integrity have features with harmony among them rather than loosely coupled. A standard feature model is used to measure product integrity. For example, a sample tiny feature model is given in Fig. 2 that represents a fantasy IDE with two product variants A and B. All features are elective with no cardinality or cross-tree constraints for the simplicity. Due to the hierarchical characteristics of the feature model, the product configurations having congestion on sub-trees is more semantically related. The features in product A are more intimately correlated to one another compared to the features in product B. Hence, the product has higher integrity.

IV. Methodology

A. The Genetic Search Algorithm

Assigning features has been a challenging problem. The reason is that when the number of features is even small for instance, combinations of various features grow in an exponential manner. The problem would be more complex once the quality and feature model constraints are allowed to be taken into consideration. Applying these constraints in SPL makes it harder. Hence, the problem is NP-hard for obtaining optimal feature selection as in [2]. In this work we applied NSGA-II algorithm for finding the best feature configuration given the feature model. For the implementation, we use jMetal library, which is a Java library containing a sort of meta-heuristic algorithms.

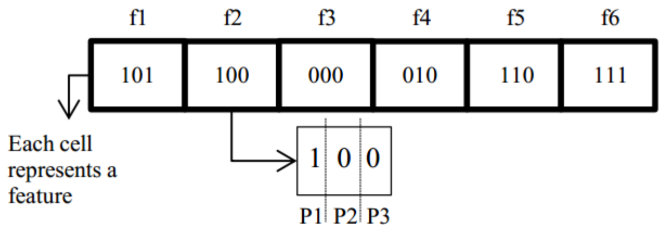


Fig. 3. Representing multiple products using a single chromosome.

B. The Particle Swarm Optimization (PSO) Algorithm

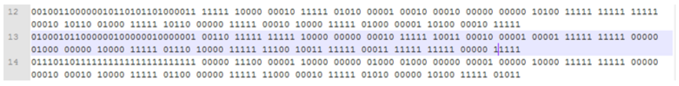
In addition to NSGA-II, we tried to explore the effectiveness of using PSO in solving our release problem model. Original PSO only works with continuous data space and doesnt work when the input are binary numbers. However, there are some works that modify PSO so that it can deal with binary data. We tried to follow an binary PSO implementation defined by [13]. The implementation was written in Java, using some components in jMetal library.

C. In feasible Chromosomes

All chromosomes, created in the evolution progression are not valid due to some constraints in model, even with optimality. However, NSGA-II allows constraint in the Pareto front calculation. Hence, then infeasible products are discarded in the evaluation process. Worth mentioning that this filtering process is very complicated. Thus, a simple limitation violation filtering means with some degree of capabilities is implemented in this research. For instance, few obligatory constraints and cross-tree constraints are no longer supported. Although a complicated constraints checker implementation is out of scope.

D. Encoding

To form a chromosome as a presence of product feature, a single data structure is used for whole product line. It is a long binary string in which each bit represents the inclusion or exclusion of that feature of a product variant. A simple encoding is illustrated in the Fig 3 where the product line consists of six features and three product variants as P1, P2, and P3. Each binary bit tells you about selection. For instance, feature f2 is not selected for product P2 and P3 whereas it is selected only for product P1. Only new features are changed their position while encoding in the evaluation process, since legacy feature are suppose to unchanged.



```
001001100000010110101101000011 11111 10000 00010 11111 01010 00001 00010 00010 00000 00000 10100 11111 11111 11111
00010 10110 01000 11111 10110 00000 11111 00010 10000 11111 01000 00001 10100 00010 11111
010001011000000100000010000001 00110 11111 11111 10000 00000 00010 11111 10011 00010 00001 00001 11111 11111 00000
01000 00000 10000 11111 01110 10000 11111 11100 10011 11111 00011 11111 11111 00000 11111
011101101111111111111111111111 00000 11100 00001 10000 00000 01000 01000 00000 00001 00000 10000 11111 11111 00000
00010 00010 10000 11111 01100 00000 11111 11000 00010 11111 01010 00000 10100 11111 01011
```

Fig. 4. The output solutions of NSGA-II after maximizing two objective functions.

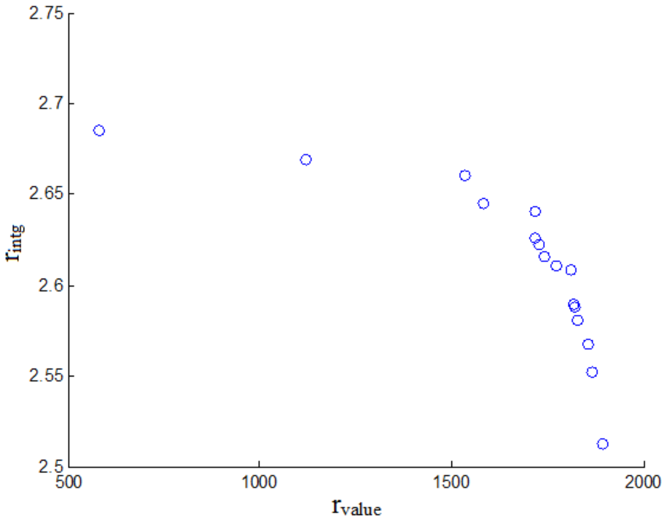


Fig. 5. Pareto front generated by Binary PSO

V. Case Study

A. Dataset

A feature model from SPLOT is used. This feature model consists of 290 features and some dependencies relation between features. We take 136 of the features randomly and assume the 136 features have been implemented in the existing product and the other 154 features are not yet implemented. In this case, our objective is to find feature configurations of implementing 30 features out of 154 features in 5 product variants that maximize the product values and integrity. Therefore, for representing feature configurations as an input for our meta-heuristic algorithm, we created a binary string whose bit represent the feature realization of a feature in a product (so there will be 5 x 30 = 150 bits in a binary string).

B. Results

The set of binary string representing feature configurations of five product variants that maximizing feature values and integrity is shown in the Fig. 4. This binary result sets are generated by the NSGA-II and PSO. The configurations are previously given and the output follow the AND dependencies rule defined in earlier section. If we can see the highlighted string set, first 30 bits are initial chromosomes where 1 represents the availability of a feature and 0 for absence. Next, five bits represent the five variants of each feature. For example, 00110 means first feature f1 absence in product variant P1, P2 and P5 similarly, f1 is presented in variant P3 and P4.

The pareto front generated by NSGA-II and Binary PSO are depicted in Fig. 5 and Fig. 6 respectively where x-axis and y-axis represent the maximum value of product value and product integrity. The graph pareto front generated by

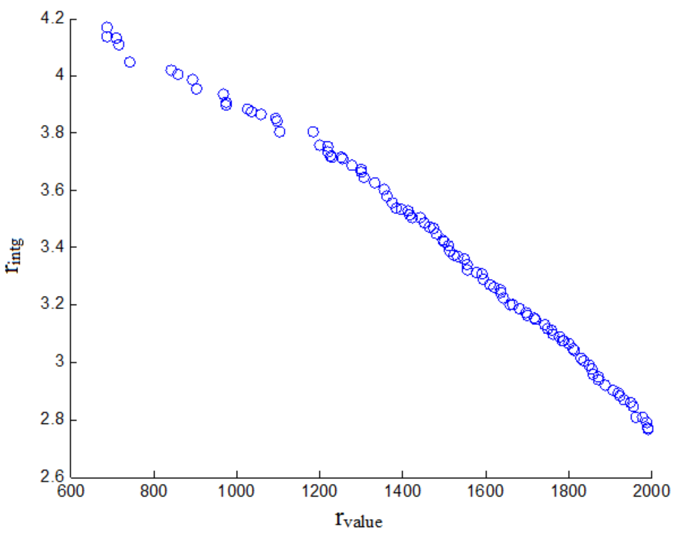


Fig. 6. Pareto front generated by NSGA-II

PSO shows that it generates only sixteen value out of thirty. When $r_{value} = 1050$, PSO can generate only one possible solution which is maximize the r_{intg} value to 2.68 whereas GA generates more than ten solutions and its maximum value is 3.9. The maximum value of r_{intg} is 2.69 in case of PSO where GA generates upto 4.2. The GA generates more possible solution that PSO which turns out that NSGA-II gives more options of good solutions than PSO. Moreover, most solutions generated by NSGA-II aress dominating the solutions from PSO, which means the GA solutions are having much more objective values than solutions given by PSO. By seeing this results, we can see that NSGA-II performs better than PSO when dealing with binary problem, like in our case.

C. Limitations

Our approach is to generate automated alternatives solutions from a set of given feature lists depends on product value and integrity. However, this approach has few limitations such as processing time is large though it depends on hardware specification. This method is valid only for a binary multi-objective NP-Hard problem. It will take enormous time to process a massive number of feature sets. To address this limitation cloud hardware can be used.

VI. Conclusion

To rapidly obtain an optimal product configuration, developers need algorithmic techniques to repeatedly produce a suitable feature selection that optimizes preferred product properties. Nevertheless, resulting a feature collection with optimal product ability focus on obligatory constraints is an NP-hard problem. Even if there are many heuristic techniques for many NP-hard problems, they cannot straightforwardly sustain the SPL feature collection optimization with resource constraints as they are not planned to handle various structural and semantic constraints defined in the feature model. This paper presents a modified mathematical model for measuring feature objectives by considering a two-component required dependency as well as using a PSO and GA techniques for achieving sub optimal results. Experiments show that this approach can achieve an average optimality even for large feature models.

References

[1] R. Karimpour and G. Ruhe, "Bi-criteria genetic search for adding new features into an existing product line," in *Combining Modelling and Search-Based Software Engineering (CMSBSE), 2013 1st International Workshop on*. IEEE, 2013, pp. 34–38.

[2] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley, "The next release problem," *Information and software technology*, vol. 43, no. 14, pp. 883–890, 2001.

[3] Y. Zhang, M. Harman, and S. A. Mansouri, "The multi-objective next release problem," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM, 2007, pp. 1129–1137.

[4] X. Cai, O. Wei, and Z. Huang, "Evolutionary approaches for multi-objective next release problem," *Computing and Informatics*, vol. 31, no. 4, pp. 847–875, 2012.

[5] K. Pohl, *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated, 2010.

[6] S. Schulze, T. Thüm, M. Kuhlemann, and G. Saake, "Variant-preserving refactoring in feature-oriented software product lines," in *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*. ACM, 2012, pp. 73–81.

[7] M. I. Ullah, G. Ruhe, and V. Garousi, "Decision support for moving from a single product to a product portfolio in evolving software systems," *Journal of Systems and Software*, vol. 83, no. 12, pp. 2496–2512, 2010.

[8] J. Guo, J. White, G. Wang, J. Li, and Y. Wang, "A genetic algorithm for optimized feature selection with resource constraints in software product lines," *Journal of Systems and Software*, vol. 84, no. 12, pp. 2208–2221, 2011.

[9] R. Rajendra and D. K. Pratihar, "Particle swarm optimization algorithm vs. genetic algorithm to solve multi-objective optimization problem in gait planning of biped robot," in *Proceedings of the International Conference on Information Systems Design and Intelligent Applications 2012 (INDIA 2012) held in Visakhapatnam, India, January 2012*. Springer, 2012, pp. 563–570.

[10] E. Assareh, M. Behrang, M. Assari, and A. Ghanbarzadeh, "Application of pso (particle swarm optimization) and ga (genetic algorithm) techniques on demand estimation of oil in iran," *Energy*, vol. 35, no. 12, pp. 5223–5229, 2010.

[11] A. C. Godinez, L. E. M. Espinosa, and E. M. Montes, "An experimental comparison of multiobjective algorithms: Nsga-ii and omopso," in *Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2010*. IEEE, 2010, pp. 28–33.

[12] L. D. Li, X. Yu, X. Li, and W. Guo, "A modified pso algorithm for constrained multi-objective optimization," in *2009 Third International Conference on Network and System Security*. IEEE, 2009, pp. 462–467.

[13] L. Wang, W. Ye, X. Fu, and M. I. Menhas, "A modified multi-objective binary particle swarm optimization algorithm," in *Advances in Swarm Intelligence*. Springer, 2011, pp. 41–48.