ICIT 2015 The 7th International Conference on Information Technology
doi:10.15849/icit.2015.0114 © ICIT 2015 (http://icit.zuj.edu.jo/ICIT15)

Smart OptiSelect Preference Based Innovative Framework for User-in-the-Loop Feature Selection in Software Product Lines

Ahmed Eid El Yamany
College of Computing and Information Technology
Arab Academy for Science, Technology, and Maritime Transport
Cairo, Egypt
Ahmedeid100@gmail.com

Mohamed Shaheen Elgamel
College of Computing and Information Technology
Arab Academy for Science, Technology, and Maritime Transport
Alexandria, Egypt
cshaheen@hotmail.com

***Abstract*—Smart OptiSelect is a multi-objective evolutionary optimization and a machine learning based framework for software product lines feature selection. It serves in the direction of filling the gap between software product lines search based feature selection optimization and real life utilization by stakeholders. OptiSelect enables system analysts and project managers to select best features to implement to meet their dynamic and always changing objectives by offering plenty of multi-objective optimized solutions that complies with these objectives. Smart OptiSelect created the availability for providing various versions of result sets based on user experience in a more comprehensive working flow. Smart OptiSelect is enabled to interactively figure out user's preferences and help to reach more convenient solutions that should best draw out the user's desires and express his organization goals.**

Keywords— User-in-the-loop (UIL); Software Product Lines; Feature Models; Optimal Feature Selection; Multi-objective Optimization; Search-Based Software Engineering; Machine Learning; Pareto Front; Non-Dominant Solutions

I. INTRODUCTION

Smart OptiSelect is a continuous result of research experiments that investigated the best ways to empower the user in the process of feature model configuration. Two targets are achieved through this version: 1) Narrowing the gap between product lines search based optimization and real life cases to provide real utilizations to software stakeholders. 2) Provide a preference based framework which can understand the user's needs and provide effective suggestions based on them.

Smart OptiSelect is an interactive framework. Users are enabled to dynamically load feature models, apply adjustments to feature attributes, set objectives and desirable thresholds, and interact by selecting preferred solution among optimization cycles.

Smart OptiSelect is a continuing effort of the previously proposed Opti-Select [1] through enhancing the workflow using machine-learning techniques to intelligently extend preferences, hybrid multi-objective optimization, and adding new features as setting user's objective thresholds.

The optimization process takes place in an incremental form. After each round of optimization, the user is provided with a concise presentation of the multiple solutions thus make up the Pareto Front, allowing the user to mark their preferred ones to focus on producing related solutions in the following iterations.

This work discusses the features and the workflow steps of Smart OptiSelect. An overview of the used algorithms and techniques and how they work together to achieve the user's goals is provided. The rest of the paper is organized as follows; Section II illustrates Smart OptiSelect workflow steps, points of interactions with the user, and processing stages. Section III describes the algorithms and methodologies, why they are selected, and how they orchestrated to work within Smart OptiSelect. Section IV displays a survey comparing users' satisfaction with the results of different techniques. Section V summarizes the proposed framework's contributions to achieve a preference based User-in-the-Loop solutions for search based product lines features optimization. It also covers an overview of some future directions and plans.

II. SMART OPTISELECT WORKFLOW

Smart OptiSelect point of strength lays in the ability to bring together most empowered multiobjective optimization algorithms proven to produce best search based product lines features optimization results [2]. This is done side by side with machine learning techniques in one single interface frame work giving the user the widest capability to be a part of the optimization process itself as shown in Fig. 1. This framework takes place through a tuned process to fit users' interactions.

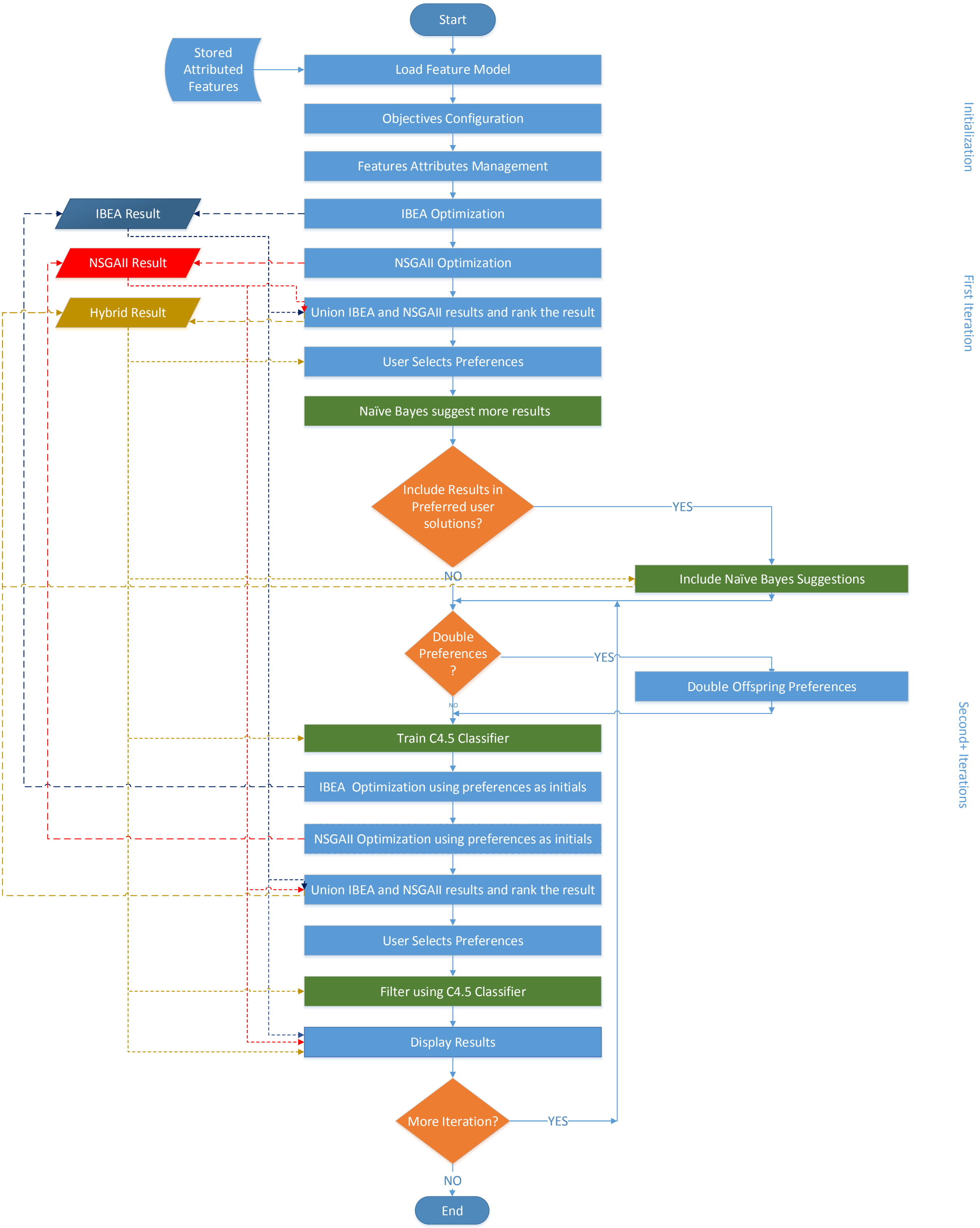


Fig. 1. Smart OptiSelect workflow diagram

doi:10.15849/icit.2015.0114 © ICIT 2015 (http://icit.zuj.edu.jo/ICIT15)

A. Loading and Saving Attributed Features.

The Simple XML Feature Model (SXFM) format was defined by the SPLOT website [3]. Smart OptiSelect implemented a module for dynamically reading and saving feature models in SXFM formats to decrease the time of changing the test model though configuration file or through hard coded instructions.

In Order to provide the user a capability for managing and saving changes over features' attributes. The proposed framework introduced an attributed feature model file format as shown in Fig. 2. It can attach a dynamic series of attributes to each feature in the model.

B. Objective Configuration

Smart OptiSelect has a predefined set of quality attributes for enabling the user to dynamically set optimization objectives and targets. Objectives targets are enabled through setting threshold for each objective as shown in Fig. 3.

The user is allowed to specify objectives being optimized prior to any optimization runs or between runs. This gives the user the power to use a desired solution set resulting from some objectives optimization at specific time as an offspring for a specific objective optimization.

```
ID,Desired,Excluded,UsedBefore,Cost,Defffects,Usability
web_portal,true,false,false,10.0,10,10
basic,true,false,true,16.0,3,50
html,true,false,false,20.0,3,0
```

Fig. 2. Saved faeature attributes format sample

During the optimization process, each solutions is dynamically evaluated based on the current objectives' settings by calculating their related attributes values.

C. Feature Attributes Management

Based on the selected objectives, the users are allowed to edit the corresponding attributes for each feature and define if a certain feature is forced to appear in all solutions or even to be excluded from all solutions as shown in Fig. 4.

Feature attributes management window is designed to be smart enough to help the user manage consequences of forcing existence and discarding existence of features by generating and applying corrective actions based on the behaviors of the user. It checks for user's opinion if more than one corrective option is available as shown in Fig. 5.

D. Multiobjective Optimization

Based on previous researches [4], IBEA [5] has been proven to perform better than the rest of the multiobjective algorithms in optimizing multiobjective problems related to product lines models and feature selection optimization as it pays most attention to user indicators without violating domain constraints. NSGA-II [6] came next in overall result quality.

Smart OptiSelect made advantage of both algorithms and provided innovative hybrid technique based on running both IBEA and NSGA-II separately within limited time. Then the results of both algorithms are merged employing Pareto front ranking [7].

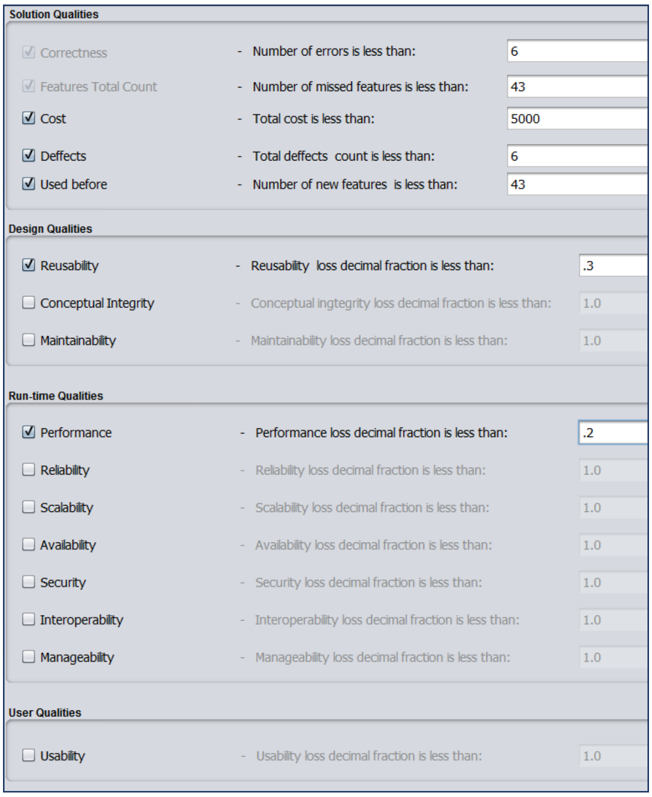
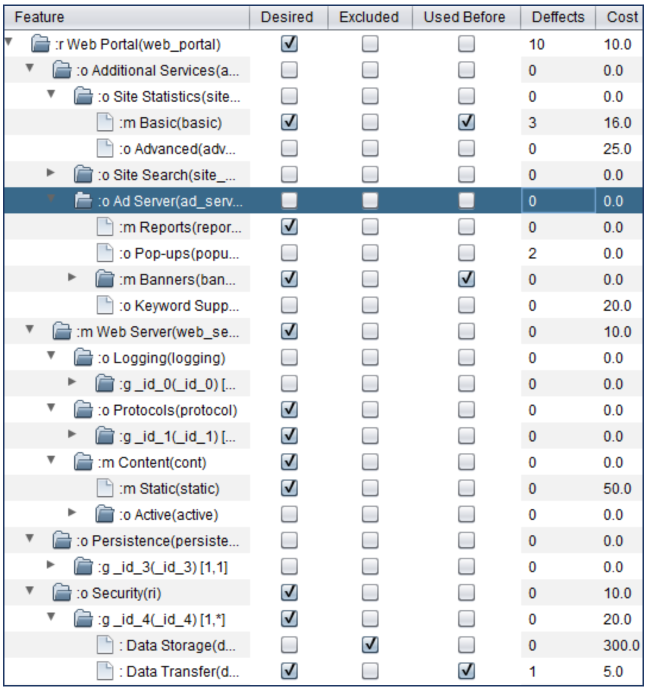


Fig. 3. Configuration sample of the objectives being optimized



Feature	Desired	Excluded	Used Before	Deffects	Cost
:r Web Portal(web_portal)	✓			10	10.0
:o Additional Services(a...				0	0.0
:o Site Statistics(site...				0	0.0
:m Basic(basic)	✓		✓	3	16.0
:o Advanced(adv...				0	25.0
:o Site Search(site_...				0	0.0
:o Ad Server(ad_serv...				0	0.0
:m Reports(repor...	✓			0	0.0
:o Pop-ups(popu...				2	0.0
:m Banners(ban...	✓		✓	0	0.0
:o Keyword Supp...				0	20.0
:m Web Server(web_se...	✓			0	10.0
:o Logging(logging)				0	0.0
:g _id_0(_id_0) [...				0	0.0
:o Protocols(protocol)	✓			0	0.0
:g _id_1(_id_1) [...	✓			0	0.0
:m Content(cont)	✓			0	0.0
:m Static(static)	✓			0	50.0
:o Active(active)				0	0.0
:o Persistence(persiste...				0	0.0
:g _id_3(_id_3) [1,1]				0	0.0
:o Security(ri)	✓			0	10.0
:g _id_4(_id_4) [1,*]	✓			0	20.0
: Data Storage(d...		✓		0	300.0
: Data Transfer(d...	✓		✓	1	5.0

Fig. 4. Feature attributes management window

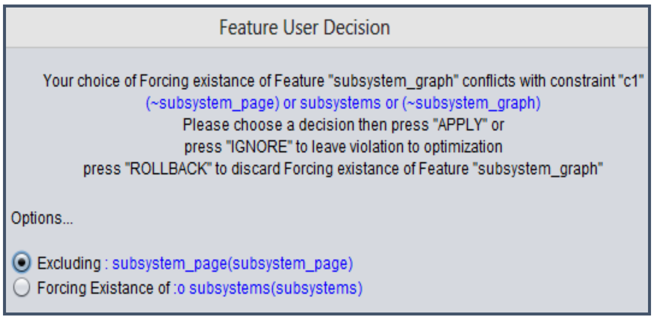


Fig. 5. Attributes management corrective actions list sample

E. User Preference Selection

Smart OptiSelect users are enabled to select a subset of the solutions from the total result set as preferred solutions as shown in Fig. 6. Selected preference are used as an initial offspring population for the next optimization cycles to force the optimization cycle to focus around the selected solutions along with the repeating cycles.

The proposed framework tries to enrich the population for the next iteration based on user selections at the current iteration. It uses any of the machine learning techniques to classify the rest of the non-selected and undisplayed solutions and see if they match the user current selections. Naïve Bayes [8] has been employed as one of the classification techniques. The user is then asked if he wants to add the suggested solutions to be considered in next iterations as shown in Fig. 7.

F. Iterating and Machine Learning

Smart OptiSelect uses final user preferred decisions selected from total result set to build and train a c4.5 classifier that aims to figure out user's preferences [9] to be used to filter result sets through next iterations.

Application repeats optimization cycle and apply user thresholds preferences filters and display different results to user to indicate if there are similar solutions to selected ones should be also selected by user.

G. Displaying Result

Smart OptiSelect provides four types of results to be displayed to the user after each iteration for comparative purposes: IBEA Result – NSGAII Result – Hybrid Result – C4.5 Filtered results.

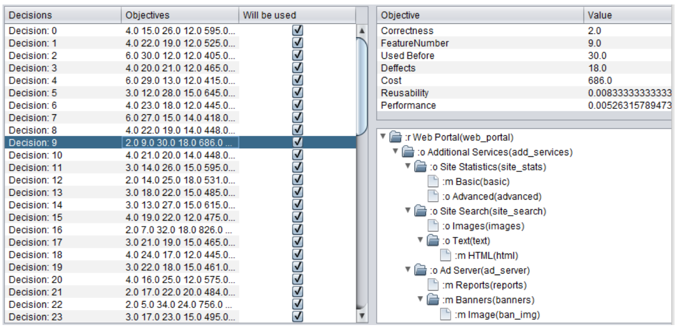


Fig. 6. Solution resultset sample enables the user to select preferred solutions

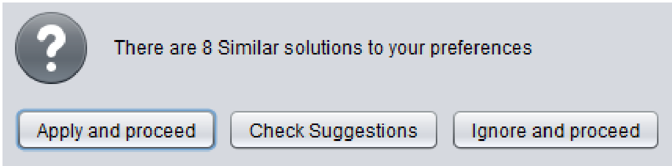


Fig. 7. A sample of Naïve Bayes suggestions to the user

Each result is displayed in a window detailing solution's objective values and solution features details.

III. Algorithms Used, How and Why?

Smart OptiSelect uses hybrid of Multiobjective optimization and machine learning algorithms to achieve effective User-In-the-Loop preference based framework.

A. IBEA

Indicator-based evolutionary algorithm (IBEA) is a multi-objective evolutionary algorithm that can be combined with arbitrary indicators. In contrast to existing algorithms, IBEA can be adapted to the preferences of the user and, moreover does not require any additional diversity preservation mechanism; such as fitness sharing to be used. IBEA calculates domination value (i.e. amount of dominance) based on indicator (e.g. hypervolume). It favors objectives, i.e. user preferences.

A comparison among various multi-objective search-based software engineering methods was performed by A. Sayyad et al. [10]. It has shown that IBEA performs much better in product line feature optimization than methods in widespread use especially with increased number of optimization objectives. IBEA works best since it makes most use of user preference knowledge. It also generates far more products without violations of domain constraints.

To adopt IBEA, jMetal [11] IBEA library was used by Smart OptiSelect through formatting feature model trees attributes into indexed formats ready for the evaluation process. Then, it passes problem to IBEA in a binary-encoded-problem format. IBEA generates selected/non-selected features list for each decision based on the optimization of features selection using hyper volume indicator.

doi:10.15849/icit.2015.0114 © ICIT 2015 (http://icit.zuj.edu.jo/ICIT15)

B. NSGAII

NSGA-II [12] is a multi-objective evolutionary algorithm which uses a non-dominated sorting for optimizing multi-objective problems. It is able to find high spread solutions in all problems. It pays special attention towards creating a diverse Pareto-optimal front within low computational requirements, elitist approach, and parameter-less sharing approach.

NSGA-II Calculates distance to the closest point for each objective. The fitness is the product of these distances. It favors higher fitness, i.e. more isolated points. It favors absolute domination and more spread out solutions.

NSGA-II came second after IBEA in optimizing product lines feature models [10] achieving better spread and hyper volume rather than rest of multi-objective evolutionary algorithms.

JMetal [11] NSGA-II library was used by Smart OptiSelect as following:

- Feature model attributes tree is reformatted into an indexed array to speed up evaluation processes.
- Problem is passed to NSGA-II as a binary-encoded problem using selected/non-selected features for each decision.
- NSGA-II generates optimized solution set based on maximizing the spread of features attributes.

C. Hybrid Optimization

Smart OptiSelect runs both optimization algorithms independently for a fixed amount of time rather than fixed amount of evaluations to control the performance and to guarantee each of optimization algorithms is not waiting for other. Then both algorithms solutions are merged, ranked and filtered.

For achieving this merging process, employing Pareto front ranking [13] gave a way to extract non-dominated solutions with highest ranks from multiobjective optimization hybrid solutions.

After each phase of the optimization process, solutions generated by both algorithms are plotted on the fitness space as shown in Fig. 8. J Metal Library [14] is used to sort, filter and extract first rank of Pareto front optimum solutions.

D. Naïve Bayes

Naïve Bayes [15] classifier is selected for providing suggestions to the user based on his preferred solutions selected from totals solutions result set.

The Naive Bayes algorithm is a simple probabilistic classifier that calculates a set of probabilities by counting the frequency and combinations of values in a given data set.

The probability of a specific feature in the data appears as a member in the set of probabilities derived by calculating the frequency of each feature value within a class of a training data set. The training dataset used to train a classifier algorithm by using known values to predict future, unknown values.

Although Naïve Bayes performed consistently worse than C4.5 [16], it remained true to its reputation and sufficient enough for being used for providing suggestions to the user for following reasons:

- Its probabilistic nature depending on counting frequency and combination given in training set suited well the problem in hand as training dataset is the same of test dataset.
- It can build models from extremely small feature sets [17].
- Its simplicity and fairly competitive performance make it the best alternative.

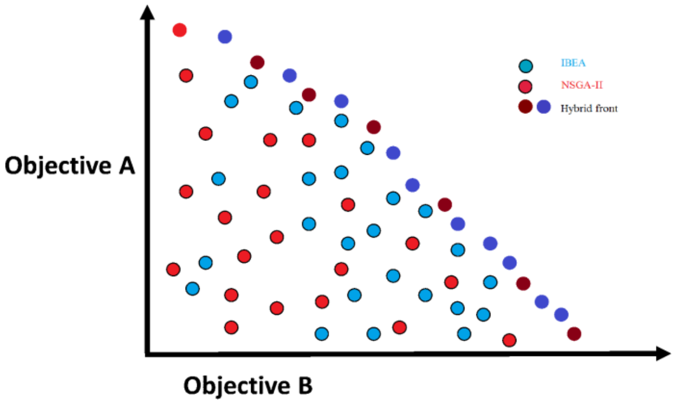


Fig. 8. Hybrid ranked non-dominant optimization solutions.

Smart OptiSelect used Naïve Bayes through following implementation: Given a set of r decision vectors D = {d_1, …, d_r}, classified along a two C classes, C={c_1,c_2} for representing Selected/Non-Selected classes, Bayesian classifiers estimate the probabilities of each class ck given a decision dj as:

$$P(c_k|d_j) = (P(c_k)P(\vec{d}_j \mid ck)) \,/\, P(\vec{d}_j) \qquad (1)$$

In eq. 1, $P(\vec{d}_j)$ is the probability that a randomly picked decision has vector d_j as its representation, and $P(c_k)$ the probability that a randomly picked decision belongs to ck.

$P(\vec{d}_j \mid ck)$ is the product of the probabilities of each feature that appears in the decision. So, $P(\vec{d}_j \mid ck)$ may be estimated as:

$$P(\vec{d}_j \mid ck) = \prod_{i=1}^{|T|} P(F_{ij} \mid c_k) \qquad (2)$$

Where, $\vec{d}_j = (f_{1j}, \ldots, f_{|T|j})$.

For classifying datasets, Weka library implementation was adopted by OptiSelect. Weka is a data mining library contains many machine leaning algorithms [18].

Smart OptiSelect uses Weka Naïve Bayes library through the following steps:

doi:10.15849/icit.2015.0114 © ICIT 2015 (http://icit.zuj.edu.jo/ICIT15)

1) For each decision, the application checks each feature in it and format it into binary map represents presence and absence of that feature.
2) Training Naïve Bayes using every decision and its corresponding category (Selected/Non-Selected).
3) While testing a decision, algorithm calculates the probability of each feature of the test decision.
4) The test decision is classified into Selected/Non-Selected categories on the basis of probability.

E. C4.5

C4.5 [19] is adopted by Smart OptiSelect to build user preferences decision tree based on user's preferred solutions. This decision tree evolves along optimization increments and is used to determine the user preferences. During each framework cycle, the results from the optimization process are filtered using the C4.5 built preference decision tree during previous cycles.

C4.5 may perform slightly worse than Support Vector Machine and Random Forest algorithms in terms of output quality, yet it is the most convenient to be used by Smart OptiSelect for its superiority in building models from extremely small feature sets [17].

C4.5 is based on inductive logic programming methods, constructing a decision tree based on a training set of data and using an entropy measure to determine which features of the training cases are important to populate the leaves of the tree.

The algorithm first identifies the dominant attribute of the training set and sets it as the root of the tree. Second, it creates a leaf for each of the possible values the root can take. Then, for each of the leaves it repeats the process using the training set data classified by this leaf. The core function of the algorithm is determining the most appropriate attribute to best partition the data into various classes.

Smart OptiSelect uses C4.5 through the following steps:

1) After each iteration, C4.5 is trained to build decision tree using user selected preferred decisions as a training set using two classes (Selected/Non-Selected).
2) After finishing each next optimization cycle, each decision is tested using the C4.5 built decision tree to calculate decisions belonging to the user's preferences class, resulting in a filtered solution result set.

F. Mechanism Design Methodologies

Feature management conflict control: During the phase of feature attributes' management, the user is allowed to configure forcing and excluding specific features. This type of management may violate feature model mandatory constraint or cross tree constraints.

The pseudo code shown in Fig. 9 illustrates how the application deals with such probable conflicts.

Pre-optimization indexing: Performance and memory management are essential especially when searching large feature model trees attached with dynamic multi-objective attributes. A sorted index array is introduced to hold references for tree features nodes as shown in Fig. 10.

```
IF control is exclusion/forcing THEN
    Get all successors/parents affected nods
    FOR each _node in affected nods
        If _ node exclusion/forcing causes conflict
        THEN
            Get all corrective alternatives
            If corrective alternative count > 1 THEN
                Notify the user
            ELSE
                Perform corrective action
            END IF
        END IF
    END FOR
```

Fig. 9. Feature management consecuences control pseudo code

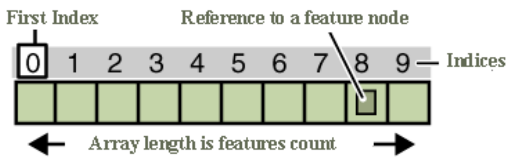


Fig. 10. Features Indexed List

Features tree is traversed using depth first algorithm once prior to optimization iteration to generate a sorted index array. This Index provides O (1) direct access to features properties and attributes. Hence, evaluation processes and search performance are optimized by avoiding tree repetitive search and tree diving recursion overhead which costs O (N). Thus, attributes are demoralized to a binary array.

Tree mutation probability: Based on the knowledge of SPLOT feature model tree structure, Tree mutation using special tree mutation probability parameter is used [20]. It aims to prevent mutations which violates feature model constraints and performs mutations with paying respect to feature model tree structure and constraints as shown in Fig. 11.

Usually, in the experiments, we set tree mutation probability to 0 to prevent tree structure and constraints violation while mutation. We also experimented raising the tree mutation probability parameter to %20 which resulted in more diversity in results but less correct solutions due to correctness thresholds.

ICIT 2015 The 7th International Conference on Information Technology
doi:10.15849/icit.2015.0114 © ICIT 2015 (http://icit.zuj.edu.jo/ICIT15)

```
FOR each bit in the decision string
    IF rand (0, 1) < mutation_probability THEN
        IF Deselecting root feature OR Deselecting a mandatory child feature whose parent is selected, or Group cardinality is violated AND rand (0,1) < tree_mutation_probibility
        THEN
                Do not mutate
        ELSE
                Flip this bit
                IF selecting (turning on) a feature THEN
                        Turn on children (a minimum skeleton)
                Else IF deselecting (turning off) a feature THEN
                        Turn off all children
                END IF
        END IF
    END IF
END FOR
```

Fig. 11. Tree mutation procedure pseudo code

IV. USERS SATISFACTION RESULTS

Smart OptiSelect can display four result sets formats of solutions:

- IBEA optimization Result
- NSGAII optimization Result
- Hybrid Pareto front optimization result
- C4.5 preferences filtered result

A survey has been created among 20 specialized software project managers, software architects and system analysts. Each of them has run through the framework for five iterations then was asked to express his satisfaction with different versions of output. User satisfaction is expressed in terms of solutions richness and its relevance to the scope. Each user was only allowed to select one result set as the best result set based on his satisfaction for each iteration round. We calculated average satisfaction for each number of iterations round. Sometimes, one result version achieved much higher satisfaction than others. Other times, more than one result were nearly equaled as shown in Table 1.

TABLE I. SYSTEM OUTPUTS USERS SATISFACTION

	0 Iteration	1 Iteration	2 Iterations	3+ Iterations
IBEA Result	☑	☑	☐	☐
NSGAII Result	☐	☐	☐	☐
Hybrid Result	☑	☑	☑	☐
C4.5 Result	☐	☐	☑	☑

The results have shown that:

IBEA results we generally more satisfying than NSGAII results because they made more attentions to users' objectives.

Hybrid results attracted attention as it displayed interesting decision solutions added from NSGA-II.

During the first iterations, Users were more satisfied with IBEA and hybrid results as they have more decisions displayed than filtered result sets by C4.5.

Starting from second iteration, most of users - who paid an interest in certain solutions' features - found that the C4.5 results were more convenient to their needs.

V. RELATED WORK DISCUSSIONS AND COMPARISON

Botterweck G. [21] feature configuration tool S2T2 Configurator integrates a visual interactive representation of the feature model and a formal reasoning engine that calculates consequences of the user's actions and provides formal explanations. Still it didn't provide a multi-objective support nor incremental configuration.

FAMA [22] is a framework for the automated analysis of feature models integrating some of the most commonly used logic representations and solvers proposed for automated analyses of feature models.

The Feature Model Plugin (FMP) [23] is implemented as an Eclipse plug–in. It supports configuration based on feature diagrams. But it does not have the analysis of FMs among its main goals. It does not support attributed feature models.

CaptainFeature is a feature modelling tool using the FODA notation to render and configure feature diagrams. It does not support the automated analysis of FMs.

\ [24] is a lightweight yet expressive language for structural modeling: feature modeling and configuration, class and object modeling.

TABLE II. SUMMARY OF FEATURE CONFIGURATION PROPOSAL

	Feature Model Representation	Iterative Configuration	Multi-Objective Optimization	Hybrid Optimization	Features Automated Analysis	Attributed Feature Model	Machine Learning Based Preferences
S2T2	☑	☐	☐	☐	☐	☐	☐
FAMA	☑	☐	☐	☐	☑	☐	☐
FMP	☑	☐	☐	☐	☐	☐	☐
CaptainFeature	☑	☐	☐	☐	☐	☑	☐
Clafer	☑	☐	☑	☐	☑	☑	☐

Opti-Select	☑	☑	☑	☐	☑	☑	☐
Smart OptiSelect	☑	☑	☑	☑	☑	☑	☑

VI. CONCLUSION AND FUTUTRE WORK

Smart OptiSelect pays more attention to user preferences by recoding his selections and training the framework incrementally to narrow the results around selected decisions and solutions.

Smart OptiSelect is considered an innovative framework as it is the first in the field of product-lines-search-based-optimization to adopt and purpose the following techniques and algorithms, as well as merging their outputs together consistently in one frame work application:

- Incremental optimization: The user can run feature-selection optimization process in increments allowing the user to adjust both the objectives and attributes in the middle of the optimization process, and to set preferred solutions.

- Hybrid Optimization: The Innovative technique utilizing the superiority of IBEA and NSGA-II [25] [26] in the field of search-based-product-line-optimization, as well as merging and filtering their results using Pareto front ranking.

- Utilization of machine learning techniques such as Naïve Bayes and C4.5 for their capability to build classifiers and decision trees to produce preference-based-solutions inspired by the user's selections among optimization increments.

Through our continuous research and development, our future steps will be:

- Using machine learning techniques to train classifiers to learn the user's objectives classification and categorization. This may vary as a simple objective or a certain relation between some features rather than his preferred features.

- Utilization of newly proposed 10-WS-C4.5-TDM-NB-TDMR [27] for user's preferences classification problem.

- Examining scalability of the results obtained with larger feature models, such as the Linux kernel feature model (part of LVAT repository [28]) composed of 6888 features.

ACKNOWLEDGMENT

Our thanks to Dr. Abdel Salam Sayyad, Dr. Tim Menzis and to Dr. Hany Ammar from West Virginia University for their valuable advices and providing access to benchmarks.

REFERENCES

[1] Yamany, El, Ahmed Eid, Mohamed Shaheen, and Abdel Salam Sayyad. "OPTI-SELECT: an interactive tool for user-in-the-loop feature selection in software product lines." In Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools-Volume 2, pp. 126-129. ACM, 2014.

[2] Sayyad, Abdel Salam, Tim Menzies, and Hany Ammar. "On the value of user preferences in search-based software engineering: A case study in software product lines." In Software Engineering (ICSE), 2013 35th International Conference on, pp. 492-501. IEEE, 2013.

[3] Mendonca, Marcilio, Moises Branco, and Donald Cowan. "SPLOT: software product lines online tools." In Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications, pp. 761-762. ACM, 2009.

[4] Sayyad, Abdel Salam, Joseph Ingram, Tim Menzies, and Hany Ammar. "Optimum feature selection in software product lines: Let your model and values guide your search." In Combining Modelling and Search-Based Software Engineering (CMSBSE), 2013 1st International Workshop on, pp. 22-27. IEEE, 2013.

[5] Zitzler, Eckart, and Simon Künzli. "Indicator-based selection in multiobjective search." In Parallel Problem Solving from Nature-PPSN VIII, pp. 832-842. Springer Berlin Heidelberg, 2004.

[6] Sadeghi, Javad, Saeid Sadeghi, and Seyed Taghi Akhavan Niaki. "A hybrid vendor managed inventory and redundancy allocation optimization problem in supply chain management: An NSGA-II with tuned parameters." Computers & Operations Research 41 (2014): 53-64.

[7] Kumar, Rajeev, and Peter Rockett. "Improved sampling of the Pareto-front in multiobjective genetic optimizations by steady-state evolution: a Pareto converging genetic algorithm." Evolutionary computation 10, no. 3 (2002): 283-314.

[8] Ting, S. L., W. H. Ip, and Albert HC Tsang. "Is Naive Bayes a good classifier for document classification?." International Journal of Software Engineering and Its Applications 5, no. 3 (2011): 37.

[9] Quinlan, J. Ross. "Improved use of continuous attributes in C4.5." arXiv preprint cs/9603103 (1996).

[10] Sayyad, Abdel Salam. "Evolutionary Search Techniques with Strong Heuristics for Multi-Objective Feature Selection in Software Product Lines." PhD diss., WEST VIRGINIA UNIVERSITY, 2014.

[11] Nebro, Antonio J., and Juan J. Durillo. "jMetal 4.5 User Manual." (2014).

[12] Deb, Kalyanmoy, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II." Lecture notes in computer science 1917 (2000): 849-858.

[13] Bosman, Peter AN. "On gradients and hybrid evolutionary algorithms for real-valued multiobjective optimization." Evolutionary Computation, IEEE Transactions on 16, no. 1 (2012): 51-69.

[14] Matjelo, Naleli Jubert, Fred Nicolls, and Neil Muller. "Evaluation of Optimal Control-based Deformable Registration Model." In New Trends in Networking, Computing, E-learning, Systems Sciences, and Engineering, pp. 117-124. Springer International Publishing, 2015.

[15] Zhang, Harry. "The optimality of naive Bayes." AA 1, no. 2 (2004): 3.

[16] Dimitoglou, George, James A. Adams, and Carol M. Jim. "Comparison of the C4. 5 and a Naive Bayes Classifier for the Prediction of Lung Cancer Survivability." arXiv preprint arXiv:1206.1121 (2012).

[17] Vatolkin, Igor, Mike Preuß, and Günter Rudolph. "Multi-objective feature selection in music genre and style recognition tasks." In Proceedings of the 13th annual conference on Genetic and evolutionary computation, pp. 411-418. ACM, 2011.

[18] Sharma, Narendra, Aman Bajpai, and Mr Ratnesh Litoriya. "Comparison the various clustering algorithms of weka tools." facilities 4 (2012): 7.

[19] Ruggieri, Salvatore. "Efficient C4.5 [classification algorithm]." Knowledge and Data Engineering, IEEE Transactions on 14, no. 2 (2002): 438-444.

doi:10.15849/icit.2015.0114 © ICIT 2015 (http://icit.zuj.edu.jo/ICIT15)

[20] Linsbauer, Lukas, Roberto Erick Lopez-Herrejon, and Alexander Egyed. "Feature Model Synthesis with Genetic Programming." In Search-Based Software Engineering, pp. 153-167. Springer International Publishing, 2014.

[21] Botterweck, Goetz, Mikolas Janota, and Denny Schneeweiss. "A Design of a Configurable Feature Model Configurator." VaMoS 29 (2009): 165-168.

[22] Benavides, David, Sergio Segura, Pablo Trinidad, and Antonio Ruiz Cortés. "FAMA: Tooling a Framework for the Automated Analysis of Feature Models." VaMoS 2007 (2007): 01.

[23] Czarnecki, Krzysztof, and Chang Hwan Peter Kim. "Cardinality-based feature modeling and constraints: A progress report." In International Workshop on Software Factories, pp. 16-20. 2005.

[24] Antkiewicz, Michał, Kacper Bąk, Alexandr Murashkin, Rafael Olaechea, Jia Hui Jimmy Liang, and Krzysztof Czarnecki. "Clafer tools for product line engineering." In Proceedings of the 17th International Software Product Line Conference co-located workshops, pp. 130-135. ACM, 2013.

[25] Peddabachigari, Sandhya, Ajith Abraham, Crina Grosan, and Johnson Thomas. "Modeling intrusion detection system using hybrid intelligent systems." Journal of network and computer applications 30, no. 1 (2007): 114-132.

[26] Purshouse, Robin C., Kalyanmoy Deb, Maszatul M. Mansor, Sanaz Mostaghim, and Rui Wang. "A review of hybrid evolutionary multiple criteria decision making methods." COIN Report,(2014005), January (2014).

[27] Molano, Viviana, Carlos Cobos, Martha Mendoza, Enrique Herrera-Viedma, and Milos Manic. "Feature Selection Based on Sampling and C4.5 Algorithm to Improve the Quality of Text Classification Using Naïve Bayes." In Human-Inspired Computing and Its Applications, pp. 80-91. Springer International Publishing, 2014.

[28] She, Steven, Rafael Lotufo, Thorsten Berger, Andrzej Wasowski, and Krzysztof Czarnecki. "The Variability Model of The Linux Kernel." VaMoS 10 (2010): 45-51.