Filho and Vergilio *Journal of Software Engineering Research and Development* (2016) 4:4
DOI 10.1186/s40411-016-0030-9

Journal of Software Engineering Research and Development

RESEARCH Open Access

A multi-objective test data generation approach for mutation testing of feature models

Rui A. Matnei Filho* and Silvia R. Vergilio

*Correspondence:
ramfilho@inf.ufpr.br
Federal University of Paraná (UFPR), Computer Science Department, CP:19081, CEP: 81531-970 Curitiba, PR, Brazil

Abstract

Background: Mutation approaches have been recently applied for feature testing of Software Product Lines (SPLs). The idea is to select products, associated to mutation operators that describe possible faults in the Feature Model (FM). In this way, the operators and mutation score can be used to evaluate and generate a test set, that is a set of SPL products to be tested. However, the generation of test sets to kill all the mutants with a reduced, possible minimum, number of products is a complex task.

Methods: To help in this task, in a previous work, we introduced a multi-objective approach that includes a representation to the problem, search operators, and two objectives related to the number of test cases and dead mutants. The proposed approach was implemented and evaluated with three representative multi-objective and evolutionary algorithms: NSGA-II, SPEA2 and IBEA, and obtained promising results. Now in the present paper we extend such an approach to include a third objective: the pairwise coverage. The goal 4 is to reveal other kind of faults not revealed by mutation testing and to improve the efficacy of the generated test sets.

Results: Results of new studies are reported, showing that both criteria can be satisfied with a reduced number of products. The approach produces diverse good solutions and different sets of impacting factors can be considered.

Conclusions: At the end, the tester can either prioritize one objective, by choosing solutions in the extreme points of the fronts or choose solutions with smaller ED values, according to the testing goals and resources.

Keywords: Mutation testing, Multi-objective optimization, Software product line

1 Background

A Software Product Line (SPL) can be defined as a set of products that share common features (Pohl et al. 2005). A feature is related to the software functionalities or system attributes that are visible to the user. Features allow distinguishing products and are important to represent variability. In this sense, different products can be generated by selecting different features. The features are generally expressed in a Feature Model (FM), which allows a hierarchical arrangement of features represented by a tree.

The adoption of the SPL approach in industries is crescent (SEI 2016), due to the associated advantages. With this increasing usage, the demand for specific SPL testing techniques has also been growing (da Mota Silveira Neto et al. 2011). An important activity in this context is the feature testing, which tests if the products derived from the FM

match their requirements. To ensure correctness, ideally, all the products derived from the FM should be tested. However, this is impractical in terms of resources and execution time (Cohen et al. 2006). Then, to select only the most representative products, testing criteria are needed. A criterion provides a way to select and evaluate test data sets that, in the FM context, are sets of products to be tested.

The main criteria used for feature testing of SPL are based on combinatorial testing (Cohen et al. 2006; Henard et al. 2014; Lamancha and Usaola 2010; Perrouin et al. 2010). Such kind of test derives SPL products to test combination of features. For example, the pairwise testing (Cohen et al. 2006) requires that each possible pair of features from the FM is included in at least one product derived for the test.

Recently, fault-based criteria, such as that ones based on mutation testing, have been investigated for variability test using the FM (Ferreira et al. 2013; Henard et al. 2013a). As it happens in the traditional test of programs (Wong et al. 1995), studies show that this kind of criterion is more efficacious, in terms of revealed faults, but also more expensive, in terms of required test cases (Ferreira et al. 2013).

In addition to this, a hard task, associated to the application of a test criterion, is the generation of test data to reach the desired coverage. This task has been successfully solved in the Search Based Software Engineering (SBSE) field (Harman et al. 2012). Search based algorithms, such as the genetic ones, are capable to search, in a huge space, the solution that solves the problem and satisfies some constraints.

Recent surveys (Harman et al. 2014; Matnei Filho and Vergilio 2014) show that the use of search-based algorithms for SPL engineering has raised interest in the last two years, mainly for configuration of products, evolution and adaptation of the FM, and also selection of products for testing. Works on this last subject address minimization (Wang et al. 2013), prioritization (Wang et al. 2014) and generation (Ensan et al. 2012; Henard et al. 2013b) of products (test cases), taking into account different factors: number of test cases, pairwise coverage, number of revealed faults, and other ones related to costs. The mutation score is used only in the work of Henard et al. (2014), which implements the (1+1) Evolutionary Algorithm (EA) algorithm, and considers the operators defined in (Henard et al. 2013a).

Most existing works have some limitations. The main one is that they deal with the problem by using an aggregation function and a single-objective algorithm, generally evolutionary one. However the problem is in fact multi-objective, impacted by many factors. Due to this, multi-objective algorithms are more suitable. Such algorithms are based on the Pareto dominance concept (Pareto 1927) and produce a set of good solutions, that represent the best trade-offs between the objectives.

We can find in the literature works that propose multi-objective approaches (Harman et al. 2014). Among them we can mention the work of Lopez-Herrejon et al. (2013) that uses a multi-objective algorithm and two objectives: number of products and pairwise-coverage but, in general, multi-objective approaches do not address mutation testing.

To overcome such limitation, in a previous work (Matnei Filho and Vergilio 2015), we introduced a multi-objective and mutation based approach to generate sets of products for the variability test of FMs. The approach reached good results considering two objectives: the size and mutation score of the derived sets. However, there are other factors that impact on the testing cost and efficacy, and need to be considered. For instance, the works found in the literature does not generate test sets to satisfy both mutation and pairwise

testing. Such goal is very important because studies reported by Ferreira et al. (2013) show that both criteria should be used in a complementary way, since they can reveal different kind of faults.

Motivated by this fact and to improve the efficacy of the generated test sets, this paper now extends the previous work by instantiating and evaluating the approach with a new objective: the pairwise coverage. The idea is to obtain a set of products to the feature testing with a minimal number of test cases, factor related to the cost, and high mutation score and pairwise coverage, factors related to the quality and efficacious to reveal faults.

The work uses the mutation operators and tool introduced in (Ferreira et al. 2013). The approach encompasses two main characteristics: i) introduces a new representation for the population, where an individual (solution) is a set of products, differently of most existing works, where an individual represents a product; and ii) allows the use of different objectives. The approach is implemented with three different algorithms, traditionally used by SBSE works: NSGA-II (Deb et al. 2002), SPEA2 (Zitzler et al. 2001), and IBEA (Zitzler and Künzli 2004). The performance of such algorithms are compared, according to quality indicators of the optimization field. The obtained solutions are also evaluated with respect to the three objectives. In all cases, solutions that kill all the non-equivalent mutants are obtained. We observe a reduced number of products required to mutation testing, mainly for FMs that derive the greatest number of products.

This work is organized as follows. First we present concepts from the multi-objective optimization field and we review mutation testing of SPLs, the adopted mutation approach and tool. After this, we introduce the test data generation approach: population, search operators, fitness, and implementation aspects, and we also present a use example of the introduced approach. Following, we describe how the evaluation was conducted, including the research questions, FMs used, algorithms configuration and, in the sequence, we also present and analyze the results. Finally at the end, we present related work and conclusions of the work, showing future research directions.

1.1 Multi-objective optimization

Optimization problems that are impacted by many factors are called muti-objective. For them, it is not always possible to find only a solution that optimizes all objectives simultaneously. This is because the objective functions, associated to diverse metrics, are usually in conflict, thus, a set of good solutions is generated, usually following Pareto dominance concepts (Pareto 1927). This set forms the approximation to the Pareto Front (PF_{approx}), which is composed by different non-dominated solutions. Given a set of possible solutions, the solution A dominates B if, the value of at least one objective in A is better than the corresponding objective value in B, and the values of the remaining objectives in A are at least equal to the corresponding values in B.

We observe that the generation of test data sets is a multi-objective problem, impacted by many factors. For example, in our case, to reach a high score Sc implies in a higher cost, in terms of number of test cases t. A solution A with ($t = 15, Sc = 38$ %) dominates the solution B with ($t = 16, Sc = 38$ %), since the same score was reached with a lower number t. However, A does not dominate C with ($t = 18, Sc = 56$ %), since C is better considering the score. A solution is said non-dominated if it is not dominated by any other solution.

1.1.1 Multi-objective algorithms

In order to solve a multi-objective problem, multi-objective algorithms have been successfully applied. Variants of Genetic Algorithms (GAs) are widely used in SBSE (Harman et al. 2012). A GA is a heuristic inspired by the theory of natural selection and genetic evolution. The search is started with an initial population composed by some solutions of the search space. From this population, search operators are applied consisting of selection, crossover and mutation. Such operators are specific for the representation adopted for the problem. They iteratively generate new solutions from existing ones, until some stopping condition is reached. Through the selection operator, copies of those individuals with the best values of the objective function are selected to be parent. So, the best individuals (candidate solutions) will survive in the next population. The crossover operator combines parts of two parent solutions to create a new one. The mutation operator randomly modifies a solution. The descendant population, created from the selection, crossover and mutation, replaces the parent population. At the end, the best solution found is returned.

In our work we use three most representative algorithms (Coello et al. 2006): NSGA-II (Non-dominated Sorting Genetic Algorithm) (Deb et al. 2002), SPEA2 (Strength Pareto Evolutionary Algorithm) (Zitzler et al. 2001), and IBEA (Indicator-Based Evolutionary Algorithm) (Zitzler and Künzli 2004). Each algorithm adopts different evolution and diversification strategies and were chosen in our study because they are well known Multiobjective Evolutionary Algorithms (MOEAs) (Coello et al. 2006) and largely used in the SBSE field (Harman et al. 2012). Next, they are described briefly.

Non-dominated sorting genetic algorithm (NSGA-II) The algorithm NSGA-II (Deb et al. 2002) (see Fig. 1) is a MOEA based in GA with a strong elitism strategy. For each generation, NSGA-II sorts the individuals from parent and offspring populations, considering the non-dominance, creating several fronts (Lines 10 and 11 of Fig. 1). The first front is composed by all non-dominated solutions. The second one has the solutions dominated

```
Procedure NSGA-II
  Input: N', g, f_k(X) ▷ N' members evolved g generations to solve f_k(X)
1 Initialize Population ℙ';
2 Generate random population - size N';
3 Evaluate Objectives Values;
4 Assign Rank (level) based on Pareto - sort;
5 Generate Child Population;
6   Binary Tournament Selection;
7   Recombination and Mutation;
8 for i = 1 to g do
9     for each Parent and Child in Population do
10        Assign Rank (level) based on Pareto - sort;
11        Generate sets of nondominated solutions;
12        Determine Crowding distance;
13        Loop (inside) by adding solutions to next generation starting from
          the first front until N' individuals;
14    end
15    Select points on the lower front with high crowding distance;
16    Create next generation;
17      Binary Tournament Selection;
18      Recombination and Mutation;
19 end
```

Fig. 1 Pseudocode of NSGA-II (adapted from (Coello et al. 2006))

by only one solution. The third front has solutions dominated by two other solutions, and the fronts are created until all solutions are classified.

For the solutions of the same front, another sort is performed using the crowding distance to maintain the diversity of solutions (Line 12 of Fig. 1). The crowding distance calculates how far away the neighbors of a given solution are and, after calculation, the solutions are decreasingly sorted. The solutions in the boundary of the search space are benefited with high values of crowding distance, since the solutions are more diversified but with fewer neighbors.

Both sorting procedures, front and crowding distance, are used by the selection operator (Line 17 of Fig. 1). The binary tournament selects individuals of lower front; in case of same fronts, the solution with greater crowding distance is chosen. New populations are generated with recombination and mutation (Line 18 of Fig. 1). The computational complexity of NSGA-II is $O(MN'^2)$, where M is the number of objectives and N' the population size (Deb et al. 2002).

Strength Pareto evolutionary algorithm (SPEA2) SPEA2 (Zitzler et al. 2001) is also a multi-objective algorithm based on GA (Fig. 2). In addition to its regular population, SPEA2 uses an external archive that stores non-dominated solutions found at each generation.

In each generation a strength value for each solution is calculated and used by the selection operator. The strength value of a solution *i* corresponds to the number of *j* individuals, belonging to the archive and the population, dominated by *i*. The fitness of a solution is the sum of the strength values of all its dominators, from archive and population (Line 4 of Fig. 1); 0 indicates a non-dominated individual, whereas a high value points out that the individual is dominated by many other ones. After the selection of individuals (Line 8), new populations are generated by recombination and mutation (Line 9).

During the evolutionary process, the external archive, which is used in the next generation, is filled with non-dominated solutions of the current archive and population (Line 5). When the non-dominated front does not correspond exactly to the size of the archive, two cases are possible: a too large new archive or a too small one. In the former case, a truncation procedure is performed (Line 6); first the distances from the solutions to their neighbors are calculated, then, the nearest neighbors are removed. In the second case, the dominated individuals from the current archive and population are copied to the new archive (Line 7). The worst run-time of the truncation procedure is $O(M^3)$ (Zitzler et al. 2001), where M is given by the sum of the population and external archive sizes, but SPEA

Procedure SPEA2

Input: $N', g, f_k(X) \triangleright N'$ members evolved g generations to solve $f_k(X)$

```
1  Initialize Population P';
2  Create empty external set E';
3  for i = 1 to g do
4      Compute fitness of each individual in P' and E';
5      Copy all nondominated individual from P' and E' to E';
6      Use the truncation operator to remove elements from E' when the
       capacity of the file has been exceeded;
7      If the capacity of E' has not been exceeded then use dominated
       individuals in P' to fill E';
8      Perform binary tournament selection;
9      Apply crossover and mutation;
10 end
```

Fig. 2 Pseudocode of SPEA2 (adapted from (Coello et al. 2006))

can have different implementations and, in most cases, behavior similar to NSGA-II (Deb et al. 2002).

Indicator-based evolutionary algorithm (IBEA) IBEA (Zitzler and Künzli 2004) is a multi-objective algorithm based on indicators (Fig. 3). Basically, a weight is assigned to each solution found, according to the quality indicators, favoring the user optimization objectives. IBEA performs binary tournaments for mating selection and implements environmental selection by iteratively removing the worst individual from the population and updating the fitness values of the remaining individuals.

The algorithm has as input values α that represents the population size, N maximum number of generations and k fitness scaling factor. To each population individual, a fitness value is calculated based on the k factor (Line 5 of Fig. 3). That individuals with worst values of fitness are removed from population (Lines 7 and 8). The other individual fitness are updated. If the generation counter is bigger than the maximum number of generations the decision vector is returned with the non-dominated individuals of population P, and the execution is finished (Line 11). Else, a binary tournament selection is performed with replacement on P, in order to fill the temporary mating pool P' (Line 14). The individuals P' suffer mutations and crossover, and resulting offspring is added to population P (Line 15). After that, the generation counter is incremented and the algorithm returns to Line 2. The complexity of the algorithm is $O(\alpha^2)$, with regard to the populatio size α (Zitzler and Künzli 2004).

1.1.2 Quality indicators

Quality indicators are used in the optimization field to evaluate the obtained solutions and to compare the algorithms performance. To calculate these indicators, generally three sets of solutions are used: 1) PF_{approx}: formed by non-dominated solutions returned by one execution of the algorithm; 2) PF_{known}: combination of all PF_{approx} obtained through different executions of an algorithm, removing dominated and repeated solutions; 3)PF_{true}: represents the Optimal Pareto Front to the problem. In our case this set is unknown.

Procedure IBEA

Input: α, N, k

1 Generate an initial population P of size α
2 Set the generation counter m to 0
3 **while** *population size P does not exceed α* **do**
4 **foreach** *Population Individual* **do**
5 Calculate *fitness* values of individual
6 **end foreach**
7 Choose an individual $x \in P$ with the smallest *fitness* value
8 Remove x from the population P
9 Update the *fitness* values of the remaining individuals
10 **if** $m \geqslant N$ *or another stopping criterion is satisfied* **then**
11 Consider individuals not dominated in P as the decision vector
12 **else**
13 Perform binary tournament selection with replacement on P in order to fill the temporary mating pool P'
14 Apply recombination and mutation operators to the mating pool P' and add the resulting offspring to P
15 Increment the generation countr m and go to line 5
16 **end if**
17 **end while**

Fig. 3 Pseudocode of IBEA (adapted from (Zitzler and Künzli 2004))

Due to this, and following the literature (Zitzler et al. 2003), this set was formed by all sets PF_{known} obtained with different algorithms by removing dominated and repeated solutions. The set PF_{true} is in fact an approximation to the real front.

In our work, we assessed the number of solutions and performed an analysis of the Pareto fronts, in order to evaluate the capability of the algorithms in finding a great number and diversified solutions. In this sense, we calculated the Error Ratio (ER) (Van Veldhuizen 1999) quality indicator. It corresponds to the ratio between the PF_{known} elements that are present and those that are not present in PF_{true}.

We also used the hypervolume (Zitzler et al. 2003) as the main quality indicator and the Kruskal Wallis (significance level of 95 %) (Derrac et al. 2011) as non-parametric statistic test. Hypervolume measures the coverage area of a known Pareto front (PF_{known}) on the objective space regarding a nadir (reference) point. The higher the hypervolume, the better the Pareto front. Hypervolume was used because it evaluates a set of solutions generated by multi-objective algorithms regarding convergence and diversity, besides being one of the most used indicators in the literature (Bringmann et al. 2014). In addition, hypervolume is $\triangleright$-completeness compliant (Zitzler et al. 2003), which means that, hypervolume can measure if a Pareto front is better than another in terms of dominance relation (Zitzler et al. 2003). It is a useful indicator for what we are trying to assess, since we want to find which resulting Pareto Front is the best one by comparing fronts generated by different algorithms. Furthermore, for calculating the hypervolume, only the objective values of the solutions and the nadir point is needed as entry.

Taking into account that the tester will choose only one solution from the set of non-dominated ones, we analyzed the solutions with the lowest Euclidean Distance (ED) from the ideal solution. Equation 1 shows how ED is calculated. ED represents the distance between two points $P = (p_1, p_2, \ldots, p_n)$ and $Q = (q_1, q_2, \ldots, q_n)$ in an Euclidean Space n-dimensional.

$$ED = \sum_{i=1}^{n} (p_i - q_i)^2 \tag{1}$$

ED is used here as preference criterion to help the tester in the selection of a solution. The lower the ED value, the better the trade-off between the objectives.

1.2 Feature testing of SPL

As mentioned before the demand for specific testing techniques and tools for SPL is crescent. In the feature testing, the goal is to derive the most representative set of products from the FM. In this section we describe two criteria that can be used for this task.

1.2.1 Pairwise testing

We can find many works in the literature addressing combinatorial testing in the SPL context (Cohen et al. 2006; 2008; Lamancha and Usaola 2010; McGregor 2001; Oster et al. 2011; Perrouin et al. 2010; Uzuncaova et al. 2010). The pairwise testing is a well known and used kind of combinatorial test, and due to this was adopted in our work. To derive the pairs, we use the the Combinatorial tool[1] that implements, among other algorithms, the AETG algorithm, introduced by Cohen et al. (1997). The size of an AETG test set grows logarithmically in the number of test parameters. To illustrate how the pairs are derived, consider the FM of Fig. 4, for the SPL CAS (Car Audio System (Weißleder

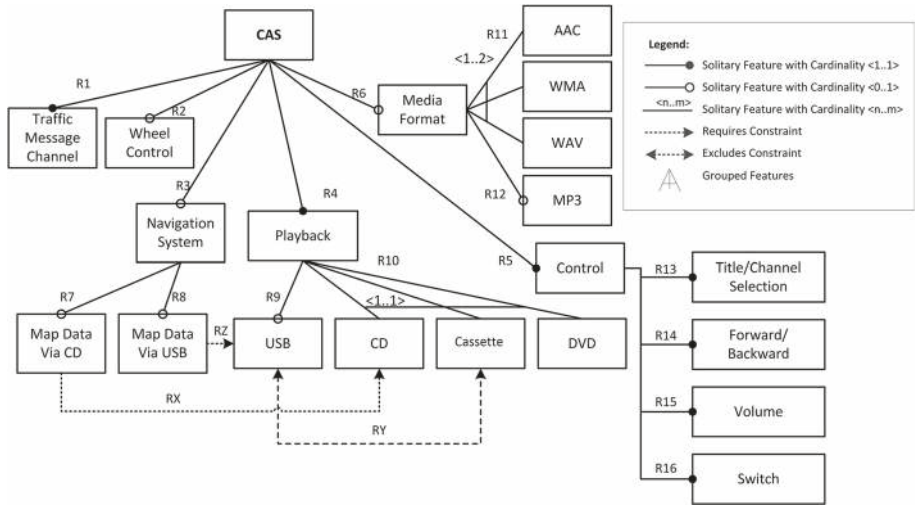

Fig. 4 Feature diagram of the Car Audio System (adapted from (Weißleder et al. 2008))

et al. 2008)), and Table 1, which presents the set of products generated by pairwise testing using such tool. In such table the valid products are represented only in terms of their variabilities. In this sense, Traffic Message Channel that is a obligatory characteristic does not appear, as well as Control. Product 1 includes the 14 pairs (Wheel Control,Map Data via CD), (Wheel Control,CD), (Wheel Control,AAC), (Wheel Control,USB), (Wheel Control,MP3), (Map Data via CD,CD), (Map Data via CD,AAC) and so on.

1.2.2 Mutation testing in the SPL context

The mutation testing has been recently explored in the SPL context. Mutation operators are used to describe faults that can be present mainly in the FM (Ferreira et al. 2013; Henard et al. 2013a). Henard et al. (2013a) introduced two mutation operators. They are defined to generate dissimilar test cases and reveal faults in the FM. A limitation of this work is that the operators are not oriented to common faults that can be present in the FM. Other one is that the work does not explore the use of such operators for generation of a test set. They are only used to assessment of test sets.

The work of Ferreira et al. (2013) introduced a set of operators that describe different faults associated to the FM and related to the feature management. This set is defined based on classes of typical faults that can be present in the FM. They are related to

Table 1 Products required by pairwise testing to the FM of Fig. 4

bf Id	Characteristics
1	Wheel Control,Map Data via CD,CD,AAC,USB,MP3
2	Wheel Control,Map Data via USB,Cassette,WMA
3	Wheel Control,Map Data via USB,DVD,WAV,USB,MP3
4	Navigation System,Map Data via CD,CD,WMA,USB,MP3
5	Navigation System,Map Data via CD,Cassette,WAV,USB
6	Navigation System,Map Data via CD,DVD,AAC
7	Navigation System,Map Data via USB,CD,AAC,MP3
8	Media Format,Map Data via CD,Cassette,AAC,USB,MP3
9	Media Format,Map Data via CD,DVD,WMA,USB,MP3
10	Media Format,Map Data via USB,CD,WAV

incorrect cardinality of solitary features and set relations, incorrect definition of grouped relations, incorrect definition of constraints, such as depends and excludes relation. The work also introduced a mutation process to use the operators as a test criterion, for evaluation and selection of test cases. This process includes steps which are similar to the steps followed in the traditional mutation testing of programs. First of all, mutants for FM are generated by applying the mutation operators and introducing only a simple modification each time. It is important to note that the FM mutants are valid diagrams.

Consider again the FM of Fig. 4. Figure 5 contains an example of mutant generated by the operator AFS (Add Feature to a Set relation (solitary feature to grouped)). In the mutant, the feature USB was added to the set relation, previously composed only by features CD, Cassette and DVD. The operator and corresponding mutant describe a possible fault in the FM.

A test set T is given by a set of products to be used in the test. If T is available, it can be evaluated by the produced score, by executing the mutants. If not, the mutants can be used to construct T. A test data (product) is "executed" with an FM analyzer. A mutant is considered dead if the validation of a product by using the mutant produces a different result from the validation of the same product against the original FM.

To illustrate the mutation process, consider again Fig. 5 and the test case of Fig. 6. The product is valid for the original FM, however it is not valid for the mutant. In this case, this product (test data) kills the mutant. A product that is invalid according to the original FM and valid according to the mutant is also capable to kill the mutant. At the end, the mutation score is calculated, given by the number of generated and dead mutants. If both models, original and mutant ones, validate the same set of products, they are considered equivalent. Equivalent mutants are not counted to calculate the score.

A tool, named FMTS (Feature Model Testing System) was implemented to support the proposed process. It works with the framework Feature Model Analyzer (FaMa) (FaMa 2014), which is responsible to validate the models and execute the test data. A valid FM satisfies some properties like satisfability, and does not have inconsistencies such as dead features. The tool supports the FODA notation (Kang et al. 1990), extended and cardinality-based FMs (Czarnecki et al. 2005). The input is an XML file that represents the FM. The tester can provide a percentage for each operator to be used in the mutant generation. If a percentage of 0.5 is provided to operator AFS, only 50 % of the possible mutants

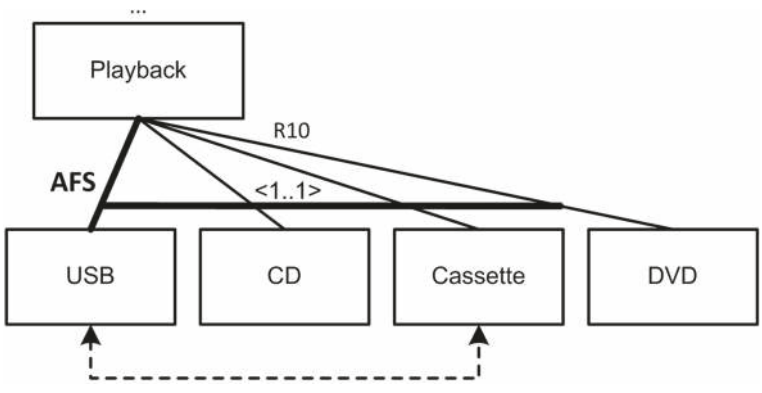

Fig. 5 An example of mutant (extracted from (Ferreira et al. 2013))

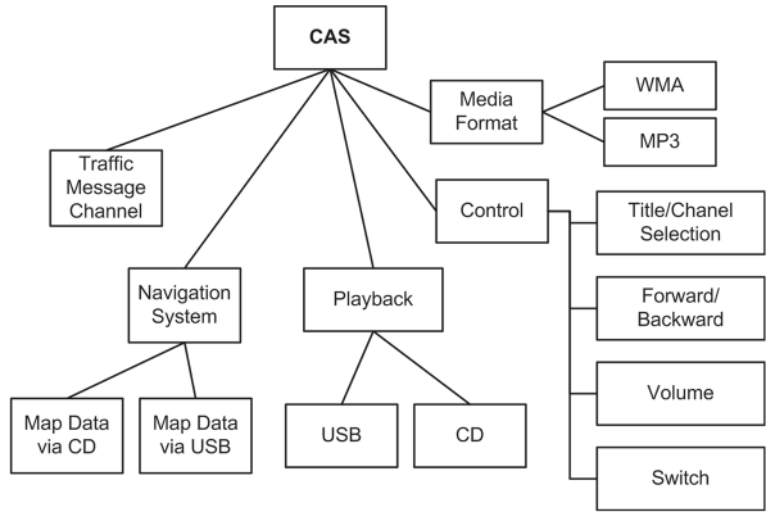

Fig. 6 Example of product (test data) to the FM of Fig. 4

to this operator are generated. The tester can also provide a test data (or a test data set) in XML format to be executed by the tool, to check the score, as well as, to set a mutant as equivalent. By using FMTS, Ferreira et al. (2013); Ferreira (2013) conducted an experiment to evaluate their operators considering factors such as efficacy, strength and cost. In such works we can find the definition of the operators, what operators are more expensive, what generate more equivalent mutants and so. In addition to this, a comparison with pairwise testing was performed showing that both criteria can reveal different kind of faults and considered incomparable in terms of inclusion relation. Mutation testing is, in general more expensive and difficult to satisfy.

In our work, we used the set of operators proposed by Ferreira et al. (2013) and the tool FMTS. This set is more complete and describes more faults. It can be applied as a testing criterion in a testing process that is automated. Our approach is introduced in the next section.

2 Methods

2.1 The search based approach

This section describes our search-based approach. The main goal is to generate sets of products, which are sets of test data to satisfy the mutation testing of FMs and also to consider different factors that can impact the test data generation. This is a complex problem that can be efficiently solved by multi-objective optimization algorithms.

According to Harman et al. (2012), to implement a search-based solution to a problem, we need the following main ingredients: i) an adequate representation to the solution, that needs to be represented in a way that can be manipulated by the algorithm; ii) search-based operators to improve the solutions and explore the search space; and iii) an adequate way to evaluate the quality of the solution, that is, the fitness function (objective function). Such ingredients of our approach are described next.

2.1.1 Population representation

The individual (solution) in the population represents a set of n products, as illustrated in Fig. 7. In such figure, the individual X contains three products (test data), identified by

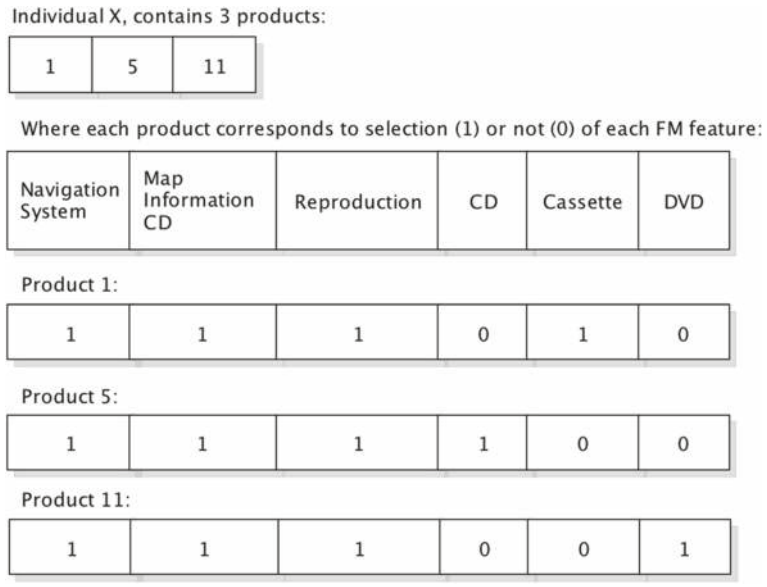

Fig. 7 Individual representation

integer numbers: (1, 5, 11). Each product is represented by a binary vector, where each element of this vector corresponds to a feature of the FM. The value (1) represents that the corresponding feature is selected for the product, and (0) represents that the feature is not selected. In the example the product 1 does not contain the features CD and DVD.

2.1.2 Search operators

The crossover operator proposed is called ProductCrossover. It works recombining two individuals X and Y, called parents, randomly selected from the population. Being n_X and n_Y the size of, respectively, individuals X and Y, if $n_X > 1$ and $n_Y > 1$ the individuals are divided into two parts. The size of the first part is equal to $n \; div \; 2$ and the second one is equal to $n - (n \; div \; 2)$. If $n \; mod \; 2 == 0$, both parts will have the same size. For example, if $n_X = 8$ (X contains 8 products) the first part after division contains the first four products and the second part contains the last four. If $n_X = 5$, the first part contains the first two products and the second one the last three. After division the parts of the parents are combined to form the offspring, as illustrated in Fig. 8. If $n_X == 1$, the division results in two equal parts, containing the same information and used in the combination, as shown in Fig. 9.

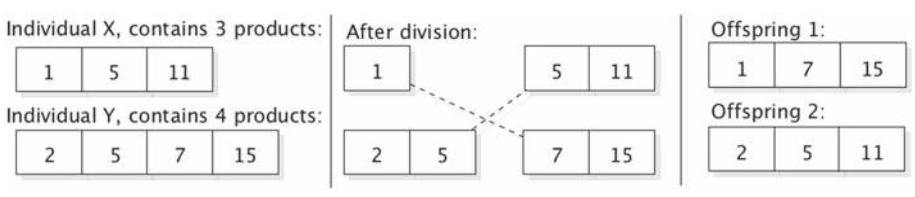

Fig. 8 Crossover operator example

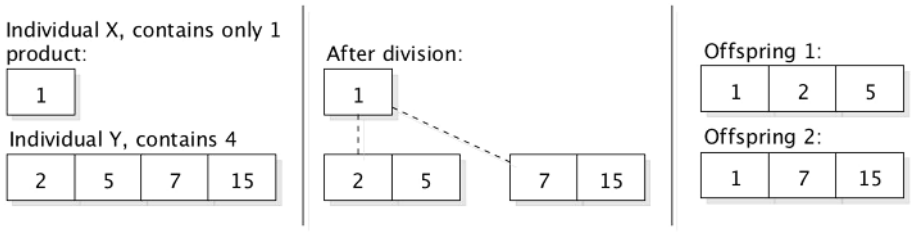

Fig. 9 Crossover operator with $n_X = 1$

The mutation operator proposed is called ProductMutation and works with three different types of mutation: addition, removal and swap. In the addition, a product randomly selected is inserted in the individual. In the removal a product of the individual is randomly selected and removed. In the swap, a product is randomly removed from the individual, and another one, also randomly selected, which does not belong to the individual is added. Figure 10 shows application examples of this operator.

After the application of a search operator, repeated products are removed from the generated individuals.

2.1.3 Fitness

The goal is to obtain high mutation scores but also to consider other factors (objectives). In this section we describe the objectives used in this work, they are related to the cost, given by the number of test cases and quality given by the pairwise coverage.

The problem is transformed in a minimization problem and we use three functions that produce values between 0 and 1, associated to: size of the individual, number of corresponding alive mutants, number of uncovered pairs. They are calculated as described next.

The score is calculated as for the mutation testing of programs, and corresponds to the relation between the number of dead mutants and the total number of mutants, given in Eq. 2.

$$A_X = 1 - \frac{DM_X}{AM} \tag{2}$$

where A_X is the fitness value A of the individual X; DM_X is the number of dead mutants, killed by executing the set X; AM is the number of active mutants being considered, that is, this set can be obtained from the set of generated mutants by discarding those mutants that are not valid (anomalous) or that generate equivalent sets of products, when compared with the original FM.

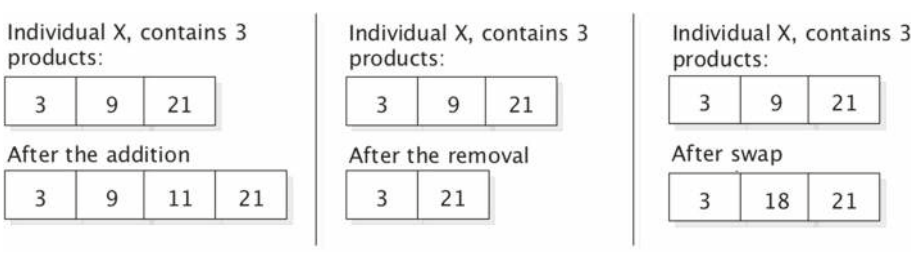

Fig. 10 Mutation operator examples

The fitness function associated to the pairwise coverage is given by Eq. 3.

$$P_X = 1 - \frac{PC_X}{P} \tag{3}$$

where P_X is the fitness value P of the individual X; PC_X is the number of covered pairs, by executing the set X; and P is the number of valid pairs generated. The pairs need to satisfy the restrictions present in the FM being tested.

The number of test cases is calculated according to the size of the individual, which represents the set of products.

$$S_X = \frac{n_X}{n} \tag{4}$$

where S_X is the fitness value for the individual (set) X; n_X is the number of products in X; and n is the number of products derived from the original FM.

2.1.4 Implementation aspects

To implement and evaluate our approach, we used three multi-objective algorithms that performed well in the SPL context: NSGA-II (Deb et al. 2002), SPEA2 (Zitzler et al. 2001) and IBEA (Zitzler and Künzli 2004). They were implemented by configuring jMetal (Durillo and Nebro 2011). jMetal is a Java based framework that implements, among others, the three mentioned algorithms. It allows an easy integration with other tools. It includes default operators and solutions for known optimization problems, and can be instanced for new ones. In addition to this, it implements some quality indicators, such as hypervolume.

As mentioned before, in this work to calculate the fitness of the solutions, we used the tool FMTS (Ferreira et al. 2013) and its operators, and the AETG algorithm implemented in the Combinatorial tool. FMTS works with FaMa analyzer. However, its is important to highlight that the approach is independent of the set of mutation operators and testing tool. To do this, it is only necessary to implement other fitness procedures.

The diagram under test is provided, following FaMa, in XML format. An array of features present in the diagram is generated and used by the AETG algorithm to generate the pairs. After this, a procedure is executed to discard invalid pairs, considering the restrictions of the FM. A module to check if a product covers a given pair was also implemented, to evaluate the fitness of an individual.

By using FMTS a set of mutants is generated. FMTS only generated valid mutants according to FaMa. From this set, and by using FaMa, equivalent mutants can be discarded. That is, they are discarded if the set of products derived by the mutant and the set derived by the original FM are the same. The mutants that can not be killed by valid products according to FM can also be discarded to ensure that the final set is composed by only valid products. The set AM is composed by the remaining set of mutants.

To reduce execution time during the evolution process and to avoid many fitness calculations, the implementation uses auxiliary matrices, MA and MP that maintain, respectively, the mutants killed and the covered pairs by a product. These matrices are generated in the initial phase and have n entries, where n can be either the number of products generated by FaMa for the FM, or a smaller number provided by the tester. This mechanism allows scalability, avoiding the manipulation of a huge number of products.

The initial population is generated by randomly selecting products in the matrices, but other methods to obtain such population can be evaluated in future works. The size of

each individual is also randomly set. In the evolution process the search operators (mutation and crossover) are applied according to the rates provided by the tester. The method used to select the individuals is binary tournament.

In the case of n smaller than the total number of products, new products can be generated in the evolution process, which are not in the matrices MA and MP. If such products are not valid according to FaMa they are discarded, otherwise, their fitness are calculated and such information is added to the matrices.

2.2 Using the approach

This section presents examples using the test data generation approach, considering the FM of the SPL CAS (Fig. 4) and the NSGA-II algorithm.

First of all, the FM is provided in the XML format. The user can do this informing the file location through the "load" command of FMTS. After that, the user configures the "problem", command with a set of parameters. The definition of the problem corresponds to the definition of the algorithms parameters: population size, maximum number of evaluations, size of the external file, crossover and mutation rates. To adjust such parameters and to ease the use of the approach in practice, we recommend the use of default parameters found in the literature. This recommendation is supported by studies in the software testing domain, reported in (Arcuri and Fraser 2011), which analyze the effects of a tuning and shows that the use of default parameters is reasonable and a justified choice. The user also needs to chose the objectives to be used in the evolution process. In the next subsections we present examples that consider two and three objectives.

2.2.1 Two-objective use

To illustrate the use with two objectives, suppose that the user wants to satisfy the mutation testing with a reduced number of test cases. Hence, he/she selected the corresponding fitness functions, the algorithm NSGA-II, and provided the following set of parameters values: 50 to population size, 10,000 to maximum number of evaluations, 0 to extension file size (NSGA-II does not use an external file), 0.1 to crossover rate (corresponding to 10 %) and 0.5 to mutation rate (corresponding to 50 %).

In this case, no value for maximum number of products is provided. This means that all the products generated by FaMa were used to compose matrix MA. To generate the mutants a percentage of 100 % was also used in FMTS for all operators. This means that all the possible mutants were generated. The number of active mutants AM to be killed by the test sets is 227, discarding the equivalent mutants.

After the execution of NSGA-II algorithm with the above mentioned parameters, a set of 13 non-dominated solutions was generated. These solutions form the approximation to the Pareto Front, the PF_{known} set. The frontier is depicted in Fig. 11. Axis X corresponds to the objective $S(x)$ and axis Y corresponds to $A(x)$.

Table 2 describes in detail each non-dominated solution found. The second column shows the fitness values reached for the solution, considering the normalized values, produced by the functions presented in Section 2.1.3. The third column shows the fitness values transformed to better visualization, corresponding to the pair: number of products and mutation score, presented in a coverage percentage of dead mutants.

Through the table and figure, it is possible to see that the proposed approach generates a set of optimal solutions. If the user wants the smallest product set, the first solution is

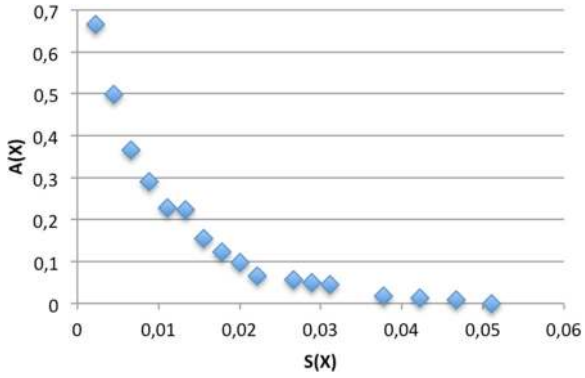

Fig. 11 PF_{known} obtained for SPL CAS with NSGA-II algorithm and two objectives

the best option, however it has a low mutation score. If the user wants the highest score, the last solution is the best choice, however it includes 44 products. All the solutions are good considering a point of view. The user needs to select the solution that better covers his/her requirements and needs.

Solutions with the best trade-off between the objectives are that ones located in the knee regions of the Pareto front (Solutions 6–10). These solutions have the best values of ED (Euclidean Distance), considering a minimization problem and the ideal solution as the point (0,0). This indicator can be used as a good way to select a solution.

We can observe that solutions with the best ED values are not in the extreme points of the Pareto front and are not associated to a mutation score of 100 %. Maybe the tester wants to prioritize this objective and choose solutions with score greater than 98 %. In this, case Solution 11 could be chosen. It has 19 products and 98.678 % of coverage. Some products of this solution are in Table 3. The ids of the such products are {15, 18, 34, 87, 107, 113, 134, 143, 154, 255, 257, 264, 279, 287, 312, 364, 388, 414, 416}.

2.2.2 Three-objective use

If the user additionally wants a high pairwise coverage he needs to select the three fitness functions, besides all the other parameters. Suppose that he/she provided the following

Table 2 NSGA-II 2-Objective Solutions for CAS

Id	Fitness values	
	(*S*, *A*)	(#products; score(%))
1	(0.002; 0.665)	(1; 33.48)
2	(0.004; 0.529)	(2; 47.137)
3	(0.007; 0.366)	(3; 63.436)
4	(0.009; 0.348)	(4; 65.198)
5	(0.011; 0.229)	(5; 77.093)
6	(0.016, 0.154)	(7;84.581)
7	(0.018, 0.141)	(8; 85.903)
8	(0.024, 0.084)	(11; 91.630)
9	(0.026, 0.079)	(12; 92.070)
10	(0.029, 0.048)	(13: 95.154)
11	(0.042, 0.013)	(19;98.678)
12	(0.053, 0.004)	(24; 99.559)
13	(0.098, 0.0)	(44;100.0)

Table 3 Products included in the 2-Objective Solution Number 11

ID	Included characteristics
15	TRAFFIC-MESSAGE-CHANEL, WHEEL-CONTROL, NAVIGATION-SYSTEM, MAP-DATA-VIA-CD, MAP-DATA-VIA-USB, PLAYBACK, USB, CD, CONTROL, TITLE-CHANEL-SELECTION, FORWARD-BACKWARD, VOLUME, SWITCH
18	TRAFFIC-MESSAGE-CHANEL, NAVIGATION-SYSTEM, PLAYBACK, CASSETTE, CONTROL, TITLE-CHANEL-SELECTION, FORWARD-BACKWARD, VOLUME, SWITCH
34	TRAFFIC-MESSAGE-CHANEL, NAVIGATION-SYSTEM, PLAYBACK, CD, CONTROL, TITLE-CHANEL-SELECTION, FORWARD-BACKWARD, VOLUME, SWITCH, MEDIA-FORMAT, AAC
87	TRAFFIC-MESSAGE-CHANEL, WHEEL-CONTROL, NAVIGATION-SYSTEM, PLAYBACK, USB, DVD, CONTROL, TITLE-CHANEL-SELECTION, FORWARD-BACKWARD, VOLUME, SWITCH, MEDIA-FORMAT, MP3, AAC
107	TRAFFIC-MESSAGE-CHANEL, WHEEL-CONTROL, PLAYBACK, CASSETTE, CONTROL, TITLE-CHANEL-SELECTION, FORWARD-BACKWARD, VOLUME, SWITCH, MEDIA-FORMAT, WMA

parameters to NSGA-II: 50 to population size, 10,000 to maximum number of evaluations, 0 to extension file size, 0.1 to crossover rate and 0.9 to mutation rate. No value for maximum number of products was provided. This means that all the products generated by FaMa were used to compose matrices *MA* and *MP*. To generate the mutants, a percentage of 100 % was also used in FMTS for all operators. The equivalent mutants and invalid pairs were discarded, totaling a number of 227 active mutants and 420 pairs.

After the execution of NSGA-II algorithm with the above mentioned parameters, a set of 23 non-dominated solutions was generated. The fitness of these solutions are presented in Table 4. The fitness values are normalized and also in terms of percentage.

We can observe that the smallest set contains only 1 product and the largest 45. This last one is associated to a coverage of 100 % for both criteria, but if desired, the user can select the option with best ED value, in this case, Solution 17, with 21 products, score of 98.678 % and a pairwise coverage of 98.907 %. Some products included in this solution are in Table 5. The ids of the such products are {12, 16, 29, 61, 101, 134, 149, 180, 201, 204, 252, 262, 268, 277, 278, 298, 329, 361, 368, 420, 445}.

It is interesting to observe, by checking the ids of the products that compose the 2-objective and 3-objective solutions with best ED, that the sets are disjunct. This shows the difficult to solve the problem, since many solutions are possible. The approach offers the best ones considering the desired objectives.

2.3 Evaluation description

This section describes how our approach was evaluated: research questions, target FMs, how the experiment was organized, and parameters used.[2]

2.3.1 Research questions

The evaluation was guided by the following two main research questions.

- **RQ1:** How are the solutions produced by the approach with respect to the objectives? To evaluate this question the solutions produced by each algorithm are considered and evaluated according to the corresponding fitness values. Since the goal is to satisfy the mutation testing criterion, solutions associated to the best scores are analyzed.
- **RQ2:** Which algorithm is the best to solve the problem? To evaluate this question the algorithm that produced the best solutions for each FM is identified. To do this, we

Table 4 NSGA-II 3-objective solutions for CAS

Id	Fitness values	
	(*S*, *A*, *P*)	(#prod.; score(%); pairwise cover)
1	(0.002; 0.687; 0.426)	(1; 31.278; 57.377)
2	(0.004; 0.533; 0.279)	(2; 46.696; 72.131)
3	(0.007; 0.396; 0.246)	(3; 60.352; 75.410)
4	(0.007; 0.427; 0.142)	(3; 57.269; 85.792)
5	(0.009; 0.317; 0,191)	(4; 68.282; 80.874)
6	(0.011; 0.189; 0.027)	(5; 81.057; 97.268)
7	(0.018; 0.141; 0.082)	(8; 85.903; 91.803)
8	(0.018; 0.145; 0.071)	(8; 85.463; 92.896)
9	(0.02; 0.110; 0.049)	(9; 88.987; 95.082)
10	(0.02; 0.145; 0.038)	(9; 85.463; 96.721)
11	(0.024; 0.079; 0.005)	(11; 92.070; 99.454)
12	(0.029; 0.062; 0.027)	(13; 93.833; 97.268)
13	(0.031; 0.044; 0.005)	(14; 95.595; 99.454)
14	(0.036; 0.062; 0)	(16; 93.833; 100)
15	(0.038; 0.040; 0.011)	(17; 96.035; 98.907)
16	(0.04; 0.035; 0.011)	(18; 96.476; 98.907)
17	(0.047; 0.013; 0.011)	
18	(0.049; 0.031; 0)	(22; 96.916; 100)
19	(0.049; 0.026; 0.005)	(22; 97.357; 99.454)
20	(0.058; 0.013; 0)	(26; 98.678; 100)
21	(0.073; 0.009; 0)	(33; 99.119; 100)
22	(0.084; 0.004; 0)	(38; 99.560; 100)
23	(0.1; 0; 0)	(45; 100; 100)

used hypervolume and error ratio (ER), quality indicators from the optimization field described in Section 1.1.3.

2.3.2 Feature models used

We used four FMs extracted from the SPLOT repository (Mendonça et al. 2009). Such FMs were used in related work (Ferreira et al. 2013) because they contain different kind of

Table 5 Products included in the 3-objective solution - number 17

ID	Included characteristics
12	TRAFFIC-MESSAGE-CHANEL, NAVIGATION-SYSTEM, MAP-DATA-VIA-CD, PLAYBACK, USB, CD, CONTROL, TITLE-CHANEL-SELECTION, FORWARD-BACKWARD, VOLUME, SWITCH
101	TRAFFIC-MESSAGE-CHANEL, WHEEL-CONTROL, NAVIGATION-SYSTEM, MAP-DATA-VIA-USB, PLAYBACK, USB, CD, CONTROL, TITLE-CHANEL-SELECTION, FORWARD-BACKWARD, VOLUME, SWITCH, MEDIA-FORMAT, WMA
204	TRAFFIC-MESSAGE-CHANEL, PLAYBACK, USB, DVD, CONTROL, TITLE-CHANEL-SELECTION, FORWARD-BACKWARD, VOLUME, SWITCH, MEDIA-FORMAT, MP3, AAC, WMA,
329	TRAFFIC-MESSAGE-CHANEL, WHEEL-CONTROL, NAVIGATION-SYSTEM, MAP-DATA-VIA-USB, PLAYBACK, USB, DVD, CONTROL, TITLE-CHANEL-SELECTION, FORWARD-BACKWARD, VOLUME, SWITCH, MEDIA-FORMAT, MP3, AAC, WAV
445	TRAFFIC-MESSAGE-CHANEL, WHEEL-CONTROL, PLAYBACK, USB, DVD, CONTROL, TITLE-CHANEL-SELECTION, FORWARD-BACKWARD, VOLUME, SWITCH, MEDIA-FORMAT, MP3, AAC, WMA, WAV

constructs that can be present in the FMs to allow the evaluation of the proposed mutation operators. In addition to this, such repository and FMs were used in different works from literature. The first FM is associated to the SPL CAS (Car Audio System) (Weißleder et al. 2008) to manage automotive sound systems. JAMES (Benavides et al. 2005): SPL for collaborative web systems; Weather Station (WS) (Beuche and Dalgarno 2012): SPL for weather systems; and E-Shop: an E-commerce SPL (Segura et al. 2010). Some information about the FMs of these SPLs are presented in Table 6: number of features, number of binary and grouped relations, number of includes and excludes constraints, and number of products. E-Shop has the greatest number of products. Such number is impacted by the number of optional and grouped features. On the other hand, includes and excludes constraints have a negative impact.

The matrices *MA* and *MP* were initially generated containing all the valid products for each FM, not being necessary to set the number maximum of products *n*. The last columns of Table 6 present the number of active mutants *AM* and the number of valid pairs *P*. The sets *AM*s differ from the sets used in (Ferreira et al. 2013). This is because, the set does not include equivalent mutants and other ones that are killed by invalid products. This ensures generation of a set with only valid products according to the FM being tested.

2.3.3 Experiment organization

To answer the research questions and evaluate the use of the approach with different sets of objectives, we created two experiments. The first experiment, named here E2O (Experiment with 2 Objectives), used two objectives related to mutation score and number of test cases, given respectively by Eqs. 2 and 4. The goal is to obtain high scores with reduced cost.

The second experiment, named here E3O (Experiment with 3 Objectives), was conducted to evaluate the performance by using more than two objectives. The goal is to obtain high scores, with reduced cost, but also covering all pairs of features. The fitness values are given by Eqs. 2, 3, and 4. In this case the complexity of the search is greater, as well, as the difficult to obtain the solutions. Hence, the goal is to analyze if the approach is capable to reach solutions with score and pairwise coverage of 100 %.

2.3.4 Parameters setting

Before the experiment execution, a parameter tuning was performed. We used the tuning instead of fixed parameters because, in this way, it is possible to compare the best performance of the algorithms.

The following parameters were adjusted: population size; maximum number of evaluations; external file size; crossover and mutation rates. Based on the literature (Arcuri and

Table 6 Feature models properties

FM	#Feat.	#Binary relations	#Grouped relations	#Includes	#Excludes	#Prod.	#*AM*	#*P*
CAS	21	12	2	1	1	450	227	420
JAMES	14	5	3	1	1	68	106	182
WS	22	5	6	0	0	504	357	462
EShop	22	8	6	1	1	1152	394	462

Briand 2011), the values variation were defined as in Table 7. The stop criterion adopted was the number of evaluations.

These parameters were adjusted for each algorithm and each FM. In this stage, 81 different settings to the NSGAII algorithm were created and 162 settings for the algorithms SPEA2 and IBEA (both use the external file). For each algorithm 10 executions were made. Therefore, 29,160 executions were performed.

After tuning, the best settings were selected based on the best mean values of hypervolume (Zitzler et al. 2003). To those averages that did not reach statistic significance through the Kruskal-Wallis test (Kruskal and Wallis 1952), with significance level of 95 %, the fastest setting was selected. Table 8 shows the best settings founded to each FM/algorithm. Then, those parameters were used in the experiments, that executed each algorithm 30 times, since they are non-deterministic.

2.3.5 Threats to validity

We consider the SPLs size as the main threat of our study. Nevertheless, the used SPLs indicate that the approach can produce good solutions to satisfy the mutation testing of FMs.

We have not evaluated the mechanism to allow scalability, that is, to provide a limited number n of products to be manipulated by FaMa and optimization algorithm. This should be better investigated in future experiments with greater FMs. In our experiment the FMS are small and all the equivalent mutants could be identified comparing the set of products derived by FaMa. But a limitation that we can expect in the future experiments is the difficulty (or even impossibility) of automatically determining the equivalent mutants. In this way, it will be not possible to reach a 100 % score. A mechanism implemented by FMTS to allow the user to set equivalent mutants can help in this task.

The execution time of the algorithms is mainly impacted by the fitness evaluation procedures, the use of matrix M can help to reduce this time and to scale our implementation to greater FMs.

The choice of parameters for the algorithms is always a hard task. To tunning the algorithms and reduce the associated threats, we followed recommendations found in (Arcuri and Briand 2011). Since search algorithms include random variations, we repeated the experiments 30 times, to reduce the possibility the results were obtained by chance.

As mentioned before the approach was instantiated with the mutant operators generated by FMTS and pairwise implemented in the Combinatorial tool. Due to this the implementation works with FMs valid according to FaMa. We think this is not a problem because such framework considers most FM constructs, and includes cardinality-based FMs. Further investigations can use other testing tools.

Table 7 Parameters variation

Parameter	Values
Population size	50; 100; 200
Max number of evaluations	10,000; 30,000; 50,000
External file size	50; 100; 200
Crossover rate	10 %; 50 %; 90 %
Mutation rate	10 %; 50 %; 90 %

Table 8 Best configuration settings founded

Feature models	Algorithm	Pop. Size	Max. Eval	Ext. File.	% Crossover	% Mutation
			Experiment 2O			
CAS	NSGAII	50	10,000	-	10 %	50 %
	SPEA2	50	10,000	50	90 %	10 %
	IBEA	50	10,000	50	10 %	50 %
JAMES	NSGAII	100	10,000	-	50 %	10 %
	SPEA2	50	10,000	50	50 %	50 %
	IBEA	50	10,000	50	10 %	50 %
WS	NSGAII	50	10,000	-	10 %	10 %
	SPEA2	50	10,000	50	10 %	90 %
	IBEA	100	10,000	100	10 %	50 %
EShop	NSGAII	200	10,000	-	50 %	50 %
	SPEA2	100	10,000	50	90 %	50 %
	IBEA	100	10,000	50	10 %	10 %
			Experiment 3O			
CAS	NSGAII	50	10,000	-	10 %	90 %
	SPEA2	200	10,000	50	90 %	50 %
	IBEA	50	10,000	50	10 %	50 %
JAMES	NSGAII	50	10,000	-	90 %	90 %
	SPEA2	50	10,000	50	90 %	90 %
	IBEA	50	10,000	50	50 %	50 %
WS	NSGAII	100	10,000	-	50 %	10 %
	SPEA2	200	10,000	50	10 %	10 %
	IBEA	100	10,000	50	10 %	90 %
EShop	NSGAII	200	10,000	-	10 %	50 %
	SPEA2	100	10,000	50	90 %	90 %
	IBEA	50	10,000	50	90 %	50 %

3 Results and discussion

In this section the experimental results are presented and analyzed. Tables 9 and 10 show the number of solutions obtained, respectively, through experiments E2O ad E3O. The first column of each table displays the used FM, while the second one displays the amount of solutions in the real front (PF_{true}). The other columns display the amount of solutions in the PF_{known} obtained for each algorithm. The amount of the PF_{known} solutions that are present in the front PF_{true} is also presented separated by "/", ER values are displayed in parenthesis. The bold results are the best ones.

Through Table 9 we can observe that for E2O, NSGAII obtained in three (out four) FMs the greatest number of solutions in the PF_{know} followed by SPEA2. However in all those

Table 9 Pareto fronts - experiment E2O

Feature models	PFtrue	PFknown		
		NSGAII	SPEA2	IBEA
CAS	17	13/6 (0.54)	**16/9 (0.44)**	15/2 (0.87)
JAMES	8	**10/4** (0.60)	10/3 (0.70)	8/4 **(0.50)**
WS	18	**20/8** (0.60)	17/6 (0.65)	19/8 **(0.58)**
EShop	21	**20/7** (0.65)	18/9 **(0.50)**	15/6 (0.60)

Table 10 Pareto fronts - experiment E3O

FM	PFtrue	PFknown		
		NSGAII	SPEA2	IBEA
CAS	20	23/3 (0.87)	**21/13 (0.38)**	17/4 (0.76)
JAMES	13	12/2 (0.83)	**13/7 (0.42)**	13/5 (0.62)
WS	28	23/5 (0.78)	**24/18 (0.25)**	26/6 (0.77)
EShop	29	37/15 (0.59)	**20/10 (0.50)**	18/4 (0.78)

cases NSGA-II obtained lower ER values. This implies that despite the great number of solutions found by NSGA-II, few of them belong to the approximation of the real front.

Both of SPEA2 and IBEA algorithms obtained the best ER values. For CAS, SPEA2 obtained more diversity of non-dominated solutions resulting 16 solutions. From these solutions, 9 of them are present in the PF_{true}, which implies in a 0.44 ER value. This is the best ER value, which represents that most of the found SPEA2 solutions belong to PF_{true}. For EShop, the algorithm did not find the greatest number of solutions, 18 against 20 obtained by NSGAII, however, again it obtained the best ER value. For JAMES and WS, even with low diversity of found solutions, IBEA algorithm showed the best ER values.

When we analyze E3O (Table 10), NSGA-II obtained the greatest number of solutions in the PF_{know} for two systems CAS and EShop, and IBEA for the other two. Similarly to what happened for E2O, this does not imply in higher ER values. SPEA2 presented the highest ER values for all systems. This means that SPEA2 generated the greatest number of solutions in PF_{know} that dominate the other solutions.

To provide another quantitative analysis of the experimental results, Tables 11 and 12 show the hypervolume mean obtained, respectively, in both experiments. In those tables, the "=" sign represents hypervolume means that do not show statistic significance according to Kruskal-Wallis test, with 95 % of significance. The statistical test was performed with all thirty execution means. Bold values correspond to the best ones and in parenthesis the standard deviation is shown.

We can observe in Table 11 (E2O) that just for JAMES the best hypervolume mean obtains a statistical significance. For the other FMs, there is no statistical difference between the algorithms. However, this does not happen in E3O (Table 12). In the presence of three objectives there is difference between the algorithms for all almost cases, except between NSGA-II and SPEA for JAMES, where both algorithms are equivalent. For the other systems, SPEA2 is the best for CAS and WS, and NSGA-II is the best for EShop. IBEA is not the best for any system.

Table 13 shows the runtimes, in seconds, to each algorithm in each FM. Bold values are the best ones. About the runtime, IBEA seems to be the best option in both experiments. In E2O, IBEA obtained the best values in three FMs. Just for JAMES other algorithm, NSGAII, obtained the best value. In E3O, IBEA runtime is the best for CAS, JAMES and

Table 11 Hypervolume means - experiment E2O

FM	*Hypervolume*		
	NSGAII	SPEA2	IBEA
CAS	= 0.9661 (0.0176)	**= 0.9736 (0.0129)**	0.9641 (0.0178)
JAMES	**0.9414 (0.0061)**	0.9334 (0.0071)	0.9322 (0.0059)
WS	= 0.9753 (0.0102)	= 0.9718 (0.0135)	**= 0.9799 (0.0045)**
EShop	**= 0.9888 (0.0049)**	= 0.9843 (0.0098)	0.9821 (0.0083)

Table 12 Hypervolume means - experiment E3O

FM	*Hypervolume*		
	NSGAII	SPEA2	IBEA
CAS	0.9672 (0.0125)	**0.9819 (0.0022)**	0.965 (0.013)
JAMES	**= 0.9213 (0.0093)**	= 0.9211 (0.0139)	0.9166 (0.0122)
WS	0.977 (0,0059)	**0.9819 (0.003)**	0.9768 (0.0063)
EShop	**0.9885 (0.0025)**	0.9842 (0.0078)	0.976 (0.0096)

WS, and NSGA-II is the best for E-Shop. We can notice that SPEA2 runtimes were the worst ones for all systems in both experiments.

As mentioned before the ED indicator can be used by the tester to choose the solutions with best trade-off among the objectives. The tester can also use a solution with the smallest number of products, sacrificing the number of dead mutants. It is possible to choose other solutions according to the tester's preferences. For example, solutions with a 100 % coverage, in cases where the required reliability is very high, or solutions that satisfy other constraints related to organizational or contractual restrictions. Such preferences can be incorporated in procedures that allow automatic selection after the evolution process or during the optimization. This subject has been investigated in the area of preference-based algorithms (Bechikh et al. 2015) and should be explored in future works.

For our analysis, we consider that it is more interesting to take some solutions with a mutation coverage greater than a value ϵ, such as $\epsilon = 94$ %, and compare them to solutions with a 100 % coverage. The fitness values of the chosen solutions are presented respectively, for E2O and E3O, in Tables 14 and 15. For all FMs, all the algorithms found solutions with 100 % of coverage in both experiments. The number of products in those solutions are in Table 16. Bold values are the best ones. In the case of E2O, the best values for the number of products were obtained by NSGA-II for three of the FMs. IBEA was the best for CAS. In a general case, IBEA seems to be the best option. In E3O, the best values were obtained for SPEA2 for all systems, but the other algorithms also found good values.

Comparing both experiments, we can notice that the addition of a new objective did not impact in the number of required products.

However, if we observe Table 14 and analyze solutions with score greater than 98 %, we can see the impact on the cost to increase the score in 2 %. For example, if we take SPL CAS and NSGA-II, we can observe that to this increase, it is necessary 25 additional products (44-19). This impact can be observed for all FMs and algorithms. If the reliability required is high, maybe this additional cost is justified.

We can observe in Table 15 that the same happens in E3O. If we take the algorithm IBEA and CAS, we can see that we needed 30 additional products to increase the score in

Table 13 Average Runtimes

FM	Execution Time					
	E2O			E3O		
	NSGAII	SPEA2	IBEA	NSGAII	SPEA2	IBEA
CAS	73	105	**60**	135	219	**102**
JAMES	**5**	39	12	58	54	**17**
WS	98	139	**85**	429	246	**122**
EShop	245	273	**189**	**325**	348	399

Table 14 Fitness values of solutions with score greater than 94 % - E2O

FM	Algorithms		
	NSGAII	SPEA2	IBEA
CAS	(19; 98.678)	(17; 98.238)	(17; 96.916)
JAMES	(7; 94.340)	(8; 97.170)	(5; 94.340)
WS	(18; 98.880)	(17; 98.039)	(17; 98.039)
EShop	(32; 99.239)	(29; 98.985)	(24; 98.477)

3.084 % and to increase the pairwise coverage only in 0.544 %. Hence, using solutions of Table 14 and 15 seems to be a good choice for many cases.

3.1 Discussion

This section discusses the main findings of our evaluation, comparing experiments E2O and E3O and providing answers to our research questions.

During the evaluation we can observe the main advantages of our approach, compared with a single-objective one. A multi-objective algorithm produces diverse good solutions and different sets of impacting factors can be considered. At the end, the tester has a set of possible solutions and can either prioritize one objective, by choosing solutions in the extreme points of the fronts or choose solutions with smaller ED values, according to the testing goals and resources.

Answering RQ1, we can observe that independently of the experiment, it is possible to kill more than 98 % of the mutants with a reduced number of products. This guideline can be used by the tester to select the solutions and is a good choice, since the goal is to satisfy mutation test. However as showed in last section, if the reliability required to the application is high, an extra effort in terms of test cases can be justified. In the worst case, SPL CAS and algorithm SPEA2, we observed that to increase 2 % in the score it was necessary to increase the test set in 50 %. But in other cases, such as for SPL WS and IBEA, it was necessary an increase in the test set of 20 %. This is an advantage of the approach, which is capable to generate different solutions with different trade-offs including solutions with 100 % of coverage for both criteria. The tester can choose according her needs. A simple-objective algorithm generates only a solution and this selection is not possible.

If the tester wants to prioritize the score, we observe that in both experiments, for all SPLs, all the algorithms found solutions with a score of 100 %. A reduced number of products is required. In E2O, if we would take the solutions generated by IBEA, 23 products were required for CAS, 11 for JAMES, 34 for WS and 44 for EShop. These numbers represent, respectively, 5, 16, 6, and 3 % of the total number of products derived from the FMs being tested. These numbers are lower than the ones reported in the approaches from the literature (Ferreira et al. 2013).

Table 15 Fitness values of solutions with coverages greater than 94 % - E3O

FM	Algorithms		
	NSGAII	SPEA2	IBEA
CAS	(21; 98.678; 98.907)	(15; 98.238; 98.907)	(14;96,916;99,453)
JAMES	(7; 94.340; 100)	(7; 97.170; 100)	(8; 99.057; 98.667)
WS	(21; 99.160; 100)	(15; 98.039; 98.462)	(20; 98.599; 98.974)
EShop	(24; 99.492; 99.505)	(31; 100; 100)	(29; 98.477; 99.010)

Table 16 Number of products in the solutions with a 100 % coverage

FM	Number of Products					
	E2O			E3O		
	NSGAII	SPEA2	IBEA	NSGAII	SPEA2	IBEA
CAS	44	47	**23**	45	**41**	44
JAMES	**10**	13	11	13	**11**	13
WS	**31**	35	34	28	**23**	25
EShop	**43**	51	44	40	**31**	45

We observe similar behavior in E3O. Considering SPEA2 solutions, 41 products were required for CAS, 11 for JAMES, 23 for WS and 31 for EShop, representing, respectively, 9, 16, 4, and 2 % of the total number of products derived from the FMs being tested. We observe in both experiments, that the greatest reduction was for EShop, the FM with the greatest number of products, case where the use of our optimization approach seems to be more advantageous. This should be evaluated in future experiments.

Comparing both experiments, we can notice that the use of three objectives does not impact in the cost and in the necessary number of products. In the cases analyzed this number was lower. In other cases (see Table 16) it is either greater or maintain constant, but the differences are not so significant. This fact is maybe due to the nonexistence of a conflict between mutation score and pairwise coverage. In fact, increasing the score results implies increasing pairwise coverage.

The noticeable difference between the experiments is in the number of non-dominated solutions found by the algorithms, given in the set Pf_{true} (see Tables 9 and 10), with an increase of 17, 63, 55 and 38 % for, respectively, CAS, JAMES, WS, and EShop. This shows that the complexity of the problem with three objectives increases, since the search space of solutions is greater, and in such case the use of the approach seems to be indispensable.

With respect to RQ2, the statistical test does not point out difference among the algorithms in the experiment with two objectives, but in the presence of three objectives there is difference between them for almost all cases, SPEA2 presented the best performance in three FMs (out four).

Despite of this, all the algorithms found good solutions and reached 100 % of coverage with reduced number of test cases. Hence, we can conclude that all of them are a good option. However, we can observe some points related to them.

IBEA presented the best runtimes in both experiments, and should be investigated in future works to evaluate scalability.

In E2O, NSGA-II presented greater diversity of solutions, a great number of solutions in PF_{known}, for most cases. NSGA-II also obtained the best values of hypervolume. This means that this algorithm offers to the tester a greater number of solutions with different and extreme values of fitness for the tester.

IBEA and SPEA2 presented greater number of solutions in the PF_{true}, in E2O. In E3O only SPEA2 had this behavior. Then SPEA2 was capable to find the greatest number of non-dominated solutions in both experiments, considering the approximated real fronts.

In short, all the algorithms are capable to find good solutions and can be used in a general case, with many advantages compared to a single-objective algorithm. NSGA-II

should be used in cases where the testing activities have many constraints, such as that ones associated to contractual and development issues. In this way, a great variety of solutions will be available, and the tester can choose the best one according his/her preferences or needs. If a fast execution is required, cases where the FMs has a large number of products, IBEA seems to be the best choice. SPEA2 reached in both experiments the greatest number of non-dominated solutions, this means that it should be used if the tester is only interested in solutions with the best-trade-offs.

4 Conclusions

4.1 Related work

The interest in the application of search-based techniques in the SPL engineering is crescent. A recent survey (Harman et al. 2014) shows an increasing number of papers on this subject in the last three years. Results of other mapping study related to search based selection and configuration of features are presented in (Matnei Filho and Vergilio 2014). They show that this selection considers different objectives. For example, adaptation and evolution, customization according to user preferences, and so on (Henard et al. 2015; Olaechea et al. 2014; Sayyad et al. 2013).

For example, the work of Sanchez et al. (2013) and Karimpour and Ruhe (2013) has as objectives the run time adaptation and evolution of SPLs. The configuration of products is addressed by many works, mainly considering cost, preferences of the user and decision-makers, and violations of the model rules (Cruz et al. 2013; Guo et al. 2011; Pereira et al. 2013; White et al. 2014). The works of Sayyad et al. (2013a,b) evaluate different multi-objective evolutionary algorithms for SPL configuration, considering different factors to select the products: number of violated rules in the FM, cost, number of used features, and number of faults revealed during the testing activity. Multi-objective selection approaches and exact ones are compared in the work of (Olaechea et al. 2014).

The works most related to ours are those ones on search-based testing of FMs. They are described next. The work of Wang et al. (2013) addresses minimization of test cases. They use a GA and an aggregation function of the following factors: number of tests cases, pairwise coverage and capability to reveal faults. In (Wang et al. 2014) the goal is prioritization of test cases. Wang et al. use another aggregation function including cost measures, and compare GA with (1+1) EA and random search. Different GA configurations were evaluated considering different weights in the aggregation function. Other factors, such as execution cost and resources, are also considered in another work by the authors (Wang et al. 2014).

Similarly, Ensan et al. (2012) also use a simple GA with an aggregation function composed by cost and error rate factors. The work of Henard et al. (2013b) also uses a GA with an aggregation function to handle with conflicting objectives in the selection of test products. Cost, pairwise coverage, and number of products are considered.

Testing solutions based on multi-objective algorithms are also found. The work of Lopez-Herrejon et al. (2013) proposes a Pareto solution using pairwise coverage and size of the test suites. However, we can observe that most of them are based on single GAs and aggregation functions. Pairwise coverage, is the only test criterion considered in such functions besides other factors. Mutation testing has been addressed for test data generation only in the work of Henard et al. (2014). In this work the operators proposed in (Henard et al. 2013a) and mentioned in Section 1.2.2 are considered to generate test cases

(products) for testing. The approach implements the (1+1) EA algorithm in conjunction with a constraint solver to check if the products are valid, according to the FM.

The work of Henard et al. (2014) has similar goals to our work since it also considers the mutation score for generating products, but there are some differences: i) our treatment to the problem is multi-objective, since we use multi-objective algorithms that produce a set of good solutions with the best trade-off between score and number of products. The related work uses a single-objective algorithm guided by the score only; ii) we consider a broader set of mutation operators; iii) we work with a population of individuals that are complete sets of products and propose specific operators adequate to this kind of population. In most existing works, the individual used in the population is given by a binary vector, similarly to our representation for product. The related work (Henard et al. 2013a) uses only one individual. In this way, such work produces only a solution. Instead, our approach produces a set of good solutions, which can be used by the tester according to his or her goals.

We can observe that the works do not deal with the mutation-based test data generation as a multi-objective problem. In addition to this, all of them do not address mutation and combinatorial testing at the same time. To satisfy both criteria can improve the quality of the generated test data. Moreover, our approach allows reducing costs.

4.2 Concluding remarks

This paper introduced a multi-objective approach to generate test sets to kill mutants used for the feature testing of SPLs. The approach includes: i) a representation for the individual in the population that allows manipulating a population of test sets; ii) search operators adequate to the introduced representation and to evolutionary algorithms; iii) three fitness functions related to the size of the generated sets, number of dead mutants and number of pairwise coverage. This paper extends our previous work by presenting new experimental results and considering faults that are described by pairwise testing to generate reduced test sets to satisfy mutation testing of FMs. The great advantage of our approach, with respect to works from the literature, is to offer different alternatives for the testing, differently of a single objective approach.

The approach was implemented using three multi-objective algorithms: NSGA-II, SPEA2, and IBEA2 in a framework that works with FMTS and Combinatorial tool. FMTS implements different kinds of mutation operators that describe common FM faults from diverse categories. Combinatorial tool implements the algorithm AETG and pairwise testing. However the approach can be implemented with other evolutionary algorithms and testing tools. In such case other mutation operators could be used, as well as t-wise testing. This should be explored in future works.

We conducted experiments to evaluate the approach with two and three objectives. In both cases, good solutions are produced, representing the best trade-offs between the objectives. The tester can select the solution that better fits his/her needs. It is possible to choose solutions with a score greater than 0.98 and with a reduced number of test cases. In most cases, it was observed that an increase in the score implies also an increase in the pairwise coverage. Other option is to choose solutions associated to the greatest scores. In such cases, we observe in both experiments that the number of test cases of these solutions is not greater than 16 % of the number of products derived from the FM

under testing. The greater the number of products derived from the FM, the greater this reduction is.

In general all the algorithms performed well and obtained similar results. NSGA-II presented more diversity, a greater number of solutions in the PF_{approx}. IBEA and SPEA2 more solutions in the PF_{true}. The results point out that IBEA seems to be a good choice for FMs with a large number of products, since it presented the best runtimes. This should be investigated in future experiments. Such experiments should consider other FMs to investigate the performance of the implementation and the impact of using the matrices associated to the fitness. The results point out that in these cases the approach is more useful. Other objectives, related for example to test cases similarity, can also be investigated.

Endnotes

[1] http://161.67.140.42/CombTestWeb (Cohen et al. 1996).

[2] The artefacts and results obtained with the testing tools used in our evaluation can be found in www.inf.ufpr.br/gres/apoio_en.html.

Abbreviations

AFS, add feature to a set relation; CAS, car automotive system; EA, evolutionary algorithm; ED, euclidean distance; E20, experiment with two objectives; E3O, experiment with three objectives; EShop, e-commerce-Shop; FaMA, feature model analyzer; FM, feature model; FMTS, feature model testing system; GA, genetic algorithm; IBEA, indicator-based evolutionary algorithm; NSGA-II, non-dominated sorting genetic algorithm II; RQ1, research question 1; RQ2, research question 2; SPEA2, strength Pareto evolutionary algorithm 2; SPL, software product line; WS, wheather station; XML, extensible markup language

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

Both authors are equal contributors for the work and for the definition of the approach. RMF is the main responsible for the implementation and experiments execution. SRV helped in the results analysis and writing. All authors read and approved the final manuscript.

Acknowledgements

The authors would like to thank CAPES and CNPq for financial support.

Received: 28 November 2015 Accepted: 12 June 2016
Published online: 26 July 2016

References

Arcuri A, Briand L (2011) A practical guide for using statistical tests to assess randomized algorithms in software engineering. In: Proceedings of the 33rd International Conference on Software Engineering (ICSE'11). ACM, New York. pp 21–28

Arcuri A, Fraser G (2011) Parameter tuning or default values? an empirical investigation in search-based software engineering. In: Proceedings of the International Symposium on Search Based Software Engineering (SSBSE). Springer, Hungary. pp 594–623

Bechikh S, Kessentini M, Said LB, Ghédira K (2015) Chapter four - preference incorporation in evolutionary multiobjective optimization: A survey of the state-of-the-art. Adv Comput 98:141–207. Elsevier

Benavides D, Trujillo S, Trinidad P (2005) On the modularization of feature models. In: First European Workshop on Model Transformation. www.academia.edu/5750212/On_the_Modularization_of_Feature_Models

Beuche D, Dalgarno M (2012) Software product line engineering with feature models. http://www.pure-systems.com/fileadmin/downloads/pure-variants/tutorials/SPLWithFeatureModelling.pdf. Accessed July 2016

Bringmann K, Friedrich T, Klitzke P (2014) Two-dimensional subset selection for hypervolume and epsilon-indicator. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO. Vancouver, ACM. pp 589–596

Coello CA, Lamont GB, Veldhuizen DAV (2006) Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation). Springer, Secaucus

Cohen DM, Dalal SR, Fredman ML, Patton GC (1997) The AETG system: An approach to testing based on combinatorial design. IEEE Trans Softw Eng 33(7):437–444

Cohen DM, Dalal SR, Parelius J, Patton GC (1996) The combinatorial design approach to automatic test generation. IEEE Softw 13(5):83–88

Cohen MB, Dwyer MB, Shi J (2006) Coverage and adequacy in software product line testing. In: Proceedings of the ISSTA 2006 Workshop on Role of Software Architecture for Testing and Analysis. ROSATEA'06. ACM, New York. pp 53–63

Cohen MB, Dwyer MB, Shi J (2008) Constructing interaction test suites for highly-configurable systems in the presence of constraints: A greedy approach. IEEE Trans Softw Eng 34(5):633–650. doi:10.1109/TSE.2008.50
Cruz J, Neto PS, Britto R, Rabelo R, Ayala W, Soares T, Mota M (2013) Toward a hybrid approach to generate software product line portfolios. In: IEEE Congress on Evolutionary Computation (CEC). IEEE, Cancun, Mexico. pp 2229–2236
Czarnecki K, Helsen S, Eisenecker U (2005) Formalizing cardinality-based feature models and their specialization. Softw Process Improv Pract 10(1):7–29
da Mota Silveira Neto PA, Machado IC, Mcgregor JD, de Almeida ES, de Lemos Meira SR (2011) A systematic mapping study of software product lines testing. Inf Softw Technol 53(5):407–423
Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
Derrac J, Garcìa S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18
Durillo JJ, Nebro AJ (2011) jMetal: A Java framework for multi-objective optimization. Adv Eng Softw 42:760–771
Ensan F, Bagheri E, Gašević D (2012) Evolutionary search-based test generation for software product line feature models. In: Proceedings of the 24th International Conference on Advanced Information Systems Engineering. CAiSE'12. Springer, Berlin. pp 613–628
FaMa FW (2014). http://www.isa.us.es/fama. Accessed July 2016
Ferreira JM (2013) Teste de Linha de Produto de Software Baseado em Mutação do Diagrama de Características. Federal University of Paraná, Curitiba, Paraná, Brazil. in Portuguese
Ferreira JM, Vergilio SR, Quinaia M (2013) A mutation approach to feature testing of software product lines. In: International Conference on Software Engineering and Knowlodge Engineering (SEKE). Knowledge Systems Institute, Boston. pp 231–237
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. J Syst Softw 84(12):2208–2221
Harman M, Jia Y, Krinke J, Langdon WB, Petke J, Zhang Y (2014) Search based software engineering for software product line engineering: A survey and directions for future work. In: Proceedings of the 18th International Software Product Line Conference - Volume 1. SPLC'14. ACM, New York. pp 5–18
Harman M, Mansouri SA, Zhang Y (2012) Search-based software engineering: Trends, techniques and applications. ACM Comput Surv 45(1):1099–1161
Henard C, Papadakis M, Harman M, Traon YL (2015) Combining multi-objective search and constraint solving for configuring large software product lines. In: Proceedings of 37th International Conference on Software Engineering (ICSE 2015). IEEE, Florence. pp 517–528
Henard C, Papadakis M, Perrouin G, Klein J, Heymans P, Traon YL (2014) Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test configurations for software product lines. IEEE Trans Softw Eng 40(7):650–670
Henard C, Papadakis M, Perrouin G, Klein J, Traon YL (2013a) Assessing software product line testing via model-based mutation: An application to similarity testing. In: IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops. IEEE, Luxembourg. pp 188–197
Henard C, Papadakis M, Perrouin G, Klein J, Traon YL (2013b) Multi-objective test generation for software product lines. In: Proceedings of the 17th International Software Product Line Conference. SPLC'13. ACM, New York. pp 62–71
Henard C, Papadakis M, Traon YL (2014) Mutation-based generation of software product line test configurations. In: Le Goues C, Yoo S (eds). Search-Based Software Engineering, Lecture Notes in Computer Science, Vol. 8636. Springer Verlag, Fortaleza. pp 92–106
Kang KC, Cohen SG, Hess JA, Novak WE, Peterson AS (1990) Feature-Oriented Domain Analysis (FODA) Feasibility Study. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania. www.sei.cmu.edu/reports/90tr021.pdf
Karimpour R, Ruhe G (2013) Bi-criteria genetic search for adding new features into an existing product line. In: International Workshop on Combining Modelling and Search-Based Software Engineering (CMSBSE). IEEE, San Francisco. pp 34–38
Kruskal WH, Wallis WA (1952) Use of ranks in one-criterion variance analysis. J Am Stat Assoc 47(260):583–621
Lamancha BP, Usaola MP (2010) Testing product generation in software product lines using pairwise for features coverage. In: Proceedings of the 22Nd IFIP WG 6.1 International Conference on Testing Software and Systems. ICTSS'10. Springer, Berlin. pp 111–125
Lopez-Herrejon RE, Chicano JF, Ferrer J, Egyed A, Alba E (2013) Multi-objective optimal test suite computation for software product line pairwise testing. In: International Conference on Software Maintenance (ICSM). IEEE, Netherlands. pp 404–407
Matnei Filho RA, Vergilio SR (2014) Configuração baseada em busca de linha de produto de software: Resultados de um mapeamento sistemático. In: V Workshop de Engenharia de Software Baseada em Busca (WESB). Congresso Brasileiro de Desenvolvimento de Software (CBSoft), Maceió, AL. In Portuguese
Matnei Filho RA, Vergilio SR (2015) A mutation and multi-objective test data generation approach for feature testing of software product lines. In: Brazilian Symposium on Software Engineering (SBES). Congresso Brasileiro de Desenvolvimento de Software (CBSoft), Belo Horizonte - MG
McGregor JD (2001) Testing a software product line. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania. www.sei.cmu.edu/reports/01tr022.pdf
Mendonça M, Branco M, Cowan D (2009) SPLOT: software product lines online tools. In: 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications. ACM, USA. pp 761–762
Olaechea R, Rayside D, Guo J, Czarnecki K (2014) Comparison of exact and approximate multi-objective optimization for software product lines. In: Proceedings of the 18th International Software Product Line Conference - Volume 1. SPLC'14. ACM, New York. pp 92–101
Oster S, Zink M, Lochau M, Grechanik M (2011) Pairwise feature-interaction testing for SPLs: potentials and limitations. In: 15th International Software Product Line Conference. ACM, Munich. pp 6–168
Pareto V (1927) Manuel D'Economie Politique. Ams Press, Paris

Pereira JA, Figueiredo E, Noronha T (2013) Modelo computacional para apoiar a configuração de produtos em linha de produtos de software. In: V Workshop de Engenharia de Software Baseada em Busca (WESB). Congresso Brasileiro de Desenvolvimento de Software (CBSoft), Brasilia, DF, Brazil. pp 80–89. In Portuguese

Perrouin G, Sen S, Klein J, Baudry B, le Traon Y (2010) Automated and scalable t-wise test case generation strategies for software product lines. In: Software Testing, Verification and Validation (ICST), 2010 Third International Conference On. IEEE, Paris. pp 459–468

Pohl K, Böckle G, van der Linden FJ (2005) Software Product Line Engineering: Foundations, Principles and Techniques. Springer, Secaucus

Sanchez LE, Moisan S, Rigault JP (2013) Metrics on feature models to optimize configuration adaptation at run time. In: International Workshop on Combining Modelling and Search-Based Software Engineering (CMSBSE) 2013 1st. IEEE, San Francisco. pp 39–44

Sayyad AS, Ingram J, Menzies T, Ammar H (2013a) Optimum feature selection in software product lines: Let your model and values guide your search. In: International Workshop on Combining Modelling and Search-Based Software Engineering (CMSBSE). pp 22–27. doi:10.1109/CMSBSE.2013.6604432

Sayyad AS, Ingram J, Menzies T, Ammar H (2013b) Scalable product line configuration: A straw to break the camel's back. In: IEEE/ACM 28th International Conference on Automated Software Engineering (ASE). IEEE, Palo Alto, California. pp 465–474

Sayyad AS, Menzies T, Ammar H (2013) On the value of user preferences in search-based software engineering: A case study in software product lines. In: Proceedings of the 2013 International Conference on Software Engineering. ICSE'13. IEEE, San Francisco. pp 492–501

Segura S, Hierons RM, Benavides D, Ruiz-Cortés A (2010) Automated test data generation on the analyses of feature models: A metamorphic testing approach. In: 3rd International Conference on Software Testing, Verification, and Validation. IEEE, Paris. pp 35–44

SEI (2016) Product line hall of fame. Technical report. http://www.splc.net/fame.html

Uzuncaova E, Khurshid S, Batory DF (2010) Incremental Test Generation for Software Product Lines. IEEE Trans Softw Eng 36(3):309–322

Van Veldhuizen DAV (1999) Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. Air Force Institute of Technology, Wright Patterson AFB, OH, USA

Wang S, Ali S, Gotlieb A (2013) Minimizing test suites in software product lines using weight-based genetic algorithms. In: Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference. GECCO'13. ACM, New York. pp 1493–1500

Wang S, Buchmann D, Ali S, Gotlieb A, Pradhan D, Liaaen M (2014) Multi-objective test prioritization in software product line testing: An industrial case study. In: Proceedings of the 18th International Software Product Line Conference - Volume 1. SPLC'14. ACM, New York. pp 32–41

Weißleder S, Sokenou D, Schlingloff H (2008) Reusing state machines for automatic test generation in product lines. In: Bauer T, Eichler H, Rennoch A (eds). MoTiP '08: Model-Based Testing in Practice. ACM, USA

White J, Galindo JA, Saxena T, Dougherty B, Benavides D, Schmidt DC (2014) Evolving feature model configurations in software product. J Syst Softw 87(0):119–136

Wong WE, Mathur AP, Maldonado JC (1995) Mutation versus all-uses: An empirical evaluation of cost, strength and effectiveness. In: Software Quality and Productivity: Theory, Practice and Training. Chapman & Hall, Ltd., London. pp 258–265

Zitzler E, Künzli S (2004) Indicator-based selection in multiobjective search. In: Parallel Problem Solving from Nature - PPSN VIII. Lecture Notes in Computer Science, Vol. 3242. Springer Verlag. pp 832–842

Zitzler E, Laumanns M, Thiele L (2001) SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems. International Center for Numerical Methods in Engineering, Athens. pp 95–100

Zitzler E, Thiele L, Laumanns M, Fonseca CM, da Fonseca VG (2003) Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans Evol Comput 7(2):117–132