Information Sciences 267 (2014) 119–139

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A multi-objective optimization approach for the integration and test order problem

Wesley Klewerton Guez Assunção [a,b], Thelma Elita Colanzi [a,c], Silvia Regina Vergilio [a,*], Aurora Pozo [a]

[a] Computer Science Department, Federal University of Paraná (UFPR), Polytechnic Centre, CP 19.081, CEP: 81531-980, Curitiba, Brazil
[b] Campus of Toledo, Technological Federal University of Paraná (UTFPR), Street Cristo Rei 19, CEP: 85902-490, Toledo, Brazil
[c] Computer Science Department (DIN), State University of Maringá (UEM), Av. Colombo 5790, Bloco C-56, CEP: 87020-900, Maringá, Brazil

A R T I C L E   I N F O

Article history:
Received 3 June 2012
Received in revised form 16 October 2013
Accepted 29 December 2013
Available online 9 January 2014

Keywords:
Search-based algorithm
Integration testing
Multi-objective optimization

A B S T R A C T

A common problem found during the integration testing is to determine an order to integrate and test the units. Important factors related to stubbing costs and constraints regarding to the software development context must be considered. To solve this problem, the most promising results were obtained with multi-objective algorithms, however few algorithms and contexts have been addressed by existing works. Considering such fact, this paper aims at introducing a generic approach based on multi-objective optimization to be applied in different development contexts and with distinct multi-objective algorithms. The approach is instantiated in the object and aspect-oriented contexts, and evaluated with real systems and three algorithms: NSGA-II, SPEA2 and PAES. The algorithms are compared by using different number of objectives and four quality indicators. Results point out that the characteristics of the systems, the instantiation context and the number of objectives influence on the behavior of the algorithms. Although for more complex systems, PAES reaches better results, NSGA-II is more suitable to solve the referred problem in general cases, considering all systems and indicators.

1. Introduction

A testing strategy usually includes a set of phases with distinct objectives. First of all, the unit testing focuses each unit, which is considered the smallest part of an application to be tested. After this, the integration testing phase should occur, to find problems related to the interactions among units. In this phase, the units are combined and tested in aggregates according to the test plans.

A common problem found during integration testing is to determine an order (or sequence) to integrate and test the units. Such sequence can affect the order in which units are developed; the design and execution of test cases; the order in which integration faults are revealed; and the number of required stubs for the units that possibly are not available, but from which the unit being tested depends on.

Most times the stubbing construction is expensive and error-prone. Hence, approaches for determining the best orders and solving the integration and test order problem with minimal efforts and costs are fundamental. This can be corroborated observing the great number of related works found in the literature that address several kinds of applications in different

* Corresponding author. Tel.: +55 41 33613681; fax: +55 41 33613205.
E-mail addresses: wesleyk@inf.ufpr.br (W.K.G. Assunção), thelmae@inf.ufpr.br (T.E. Colanzi), silvia@inf.ufpr.br (S.R. Vergilio), aurora@inf.ufpr.br (A. Pozo).

0020-0255/$ - see front matter © 2014 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.12.040

software development contexts: component-based [21]; object-oriented (OO) [3,40]; aspect-oriented (AO) [14,33]; and product line oriented [20].

Those works use different kind of models to represent the units to be integrated and their dependency relationships. The stubbing cost can be measured by the number of required stubs and other measures generally based on coupling and cohesion that quantify the cost to construct the necessary stubs. Most works are based on graphs, and implement graph-based algorithms [1,3,6,22,26,28,36,37,41]. Other ones [2,5,13,14,39] are search-based. These last ones used search-based algorithms and modeled the integration and test order problem as an optimization problem with many related factors, characteristics associated to the software development and the stubbing process.

In experiments [5,14,39], search-based algorithms presented better performance than traditional ones, aiming at avoiding local optima. The most promising results were obtained with multi-objective algorithms [2,13,14,39]. Multi-objective optimization algorithms are being widely applied in several areas, such as Search-Based Software Engineering (SBSE) [12,19], to solve problems with many interdependent interests (objectives). For the integration and testing order problem, several algorithms have been explored and with different objectives. The results reported in [39] show that the evolutionary algorithms present the best performance in the context of OO software. In [2,13], we also used evolutionary algorithms with four coupling measures, achieving satisfactory solutions. Our previous work [13] introduced a multi-objective and evolutionary approach for the AO software context. The approach was evaluated with four real systems and with the algorithms NSGA-II and SPEA2.

Following up on our previous works, in this paper we describe the approach introduced in [13] making it more general and applicable to any type of unit to be integrated. The steps of the approach are illustrated and evaluated in two usage contexts for OO and AO software, with a not yet explored MOEA, the PAES [25] algorithm. In the evaluation we investigate some points related to the use of the approach in practice such as: the instantiation context, selection of appropriated algorithms and objectives. In this sense the main research questions addressed in this paper are:

RQ1: Does PAES achieve better solutions than NSGA-II and SPEA2 to solve the integration and test order problem? The approach can be applied with any algorithm, but it is necessary to choose one. The performance of a multi-objective algorithm to solve a specific problem is, in general, only known experimentally. NSGA-II and SPEA2 achieved good results in other experiments [2,13,14] to solve the referred problem, but it is possible that other algorithms achieve even better results. PAES was chosen in our study because: (i) it presents a different evolution strategy; and ii) in some very known SBSE problems, such as scheduling, it achieved better results than NSGA-II and SPEA2 [9]. Moreover, these three algorithms belong to the current state-of-the-art of multi-objective evolutionary algorithms which use dominance relation in the selection operator. These algorithms do not need objective normalization or advanced decomposition methods like MOEA/D for instance.

RQ2: Does the number of objectives influence on the results achieved by the algorithms? The approach allows the use of several objectives, which ideally better models the problem. However, there are some evidences that the algorithms performance decreases in the presence of more than three objectives [23,30]. Despite of this, in our previous work [2,13] we obtained good solutions with four objectives, but maybe the use of two objectives is a better choice. Hence this question aims at investigating possible differences in the performance of each algorithm working with two and four objectives.

RQ3: How do the algorithms perform in each instantiation context of the approach? This question helps in the selection of an algorithm according to the instantiation context of the approach, investigating that whether the usage context influences on the behavior of each algorithm.

So, in order to answer those questions, we performed an empirical study whose focus is to evaluate the performance of different multi-objective algorithms, considering OO and AO usage contexts and two set of objectives, one composed by two measures and other composed by four measures. In addition to this, the evaluation includes another algorithm: PAES, not previously used.

This paper is organized as follows. Section 2 describes the algorithms used in our work. Section 3 overviews works that deal with the integration and test order problem. In Section 4 the approach, called MOCAITO, is introduced. Sections 4.1, 4.2, 4.3, 4.4 illustrate each step of MOCAITO in OO and AO contexts. Section 5 describes how the empirical evaluation was conducted. The results are evaluated in Section 6 according to our research questions. Finally, Section 7 contains the conclusions and future works.

2. Multi-objective evolutionary algorithms

A multi-objective problem depends on diverse factors (objectives) that are in conflict and, in general, does not have a single solution. Hence, diverse good solutions exist and these solutions are named non-dominated and form the Pareto front [29]. In most applications, the search for the Pareto optimal is NP-hard [24], then the optimization problem focuses on finding an approximation set (PFApprox), as close as possible to the Pareto front.

Variants of GA adapted to multi-objective problems were proposed. A GA is a heuristic inspired by the theory of natural selection and genetic evolution [18]. From an initial population (candidate solutions), basic operators are applied to evolve the population, generation by generation. Through the selection operator more copies of those individuals with the best objective function values 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 offspring population created from the selection, crossover and mutation replaces the parent population.

Next we describe the algorithms used in our work. They were chosen because they are well known MOEAs [11] and adopt different evolution and diversification strategies.

2.1. Non-dominated Sorting Genetic Algorithm (NSGA-II)

NSGA-II [15] is an algorithm based on GA with a strong elitism strategy. Its pseudocode is presented in Fig. 1. For each generation, NSGA-II sorts the individuals from parent and offspring populations and creates several fronts, considering the non-dominance among individuals (lines 10 and 11). The first front is composed by all non-dominated solutions. After removal of solutions belonging to the first front, the second front is composed with the solutions which become non-dominated. In the same way, the third front is formed by the solutions that become non-dominated after the removal of the solutions belonging to the first and the second fronts, and so on until all solutions are classified.

Another operator named crowding distance is applied in order to maintain the diversity of the solutions of each front (line 12). 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.

The selection operator (line 17) uses both sorting procedures: front and crowding distance. 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).

2.2. Strength Pareto Evolutionary Algorithm (SPEA2)

SPEA2 [43] 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); 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).

```
Procedure NSGA-II
   Input: $N', g, f_k(X) \triangleright N'$ members evolved $g$ generations to solve $f_k(X)$
 1 Initialize Population $\mathbb{P}'$;
 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 [11]).

```
Procedure SPEA2
  Input: $N', g, f_k(X) \triangleright N'$ members evolved $g$ generations to solve $f_k(X)$
  Initialize Population $\mathbb{P}'$;
  Create empty external set $\mathbb{E}'$;
  for $i = 1$ to $g$ do
      Compute fitness of each individual in $\mathbb{P}'$ and $\mathbb{E}'$;
      Copy all nondominated individual from $\mathbb{P}'$ and $\mathbb{E}'$ to $\mathbb{E}'$;
      Use the truncation operator to remove elements from $\mathbb{E}'$ when the
      capacity of the file has been exceeded;
      If the capacity of $\mathbb{E}'$ has not been exceeded then use dominated
      individuals in $\mathbb{P}'$ to fill $\mathbb{E}'$;
      Perform binary tournament selection;
      Apply crossover and mutation;
  end
```

Fig. 2. Pseudocode of SPEA2 (adapted from [11]).

2.3. Pareto Archived Evolution Strategy (PAES)

The multi-objective algorithm PAES (*Pareto Archived Evolution Strategy* [25]) works with a population concept that is different from other evolutionary algorithms strategies, since only one solution is maintained in each generation. The strategy to generate new individuals is to use only the mutation operator, as observed in line 3 of pseudocode presented in Fig. 3, which is an algorithm for a minimization problem. Since the algorithm works with only one solution by generation there is no possibility to use the crossover operator. Similarly to SPEA2, PAES works with an external archive, populated with the non-dominated solutions found in the evolutionary process.

In each generation, PAES creates new children solutions that are compared with the parent solution. In the comparison, if the child solution is dominated by the parent solution, the child is discarded (lines 4 and 5). If the child solution dominates the parent solution, the child takes the place of the parent, to be used in the next generations, and child is added to the external archive (lines 6 and 7). If the child solution is dominated by any solution of the archive, the child is discarded (lines 8 and 9). In the case of none of the solutions (parent, child and archive) is dominant, the solution that maintains the diversity among the solutions is chosen to remain in the evolutionary process (lines 10 and 11).

If the external archive size is exceeded, a diversity strategy is applied on the set of solutions in order to remove the similar solutions and to maintain wide the exploitation of a search space.

3. The integration and test order problem

As mentioned before we can find in the literature many works that address the integration and test order problem in different application contexts: OO and AO software; component-based; and product line oriented. In this section an overview of such works is presented. We find in the literature a lot of works addressing this problem in the OO context, which has recently been subject of a survey [40]. Generally the strategies are based on two types of dependencies graphs: TDG (Test Dependency Graphs [37]) and ORD (Object Relation Diagrams [27]). When there are no cycles in the dependencies graphs, a solution that does not require stubs is found by a simple inverse topological sort of the graph, but most programs have dependence cycles. To break cycles two kind of algorithms are generally used: graph based algorithms, called here, traditional algorithms [1,3,6,22,26,28,36,37] and search-based algorithms (metaheuristics) that search in a huge space an optimization problem solution [2,4,5,13,14,39].

These algorithms mentioned above can be used in the component integration testing context. The term component usually refers to a software unit in any grain size. Hence, a class can be considered a component. However, when the unit is a

```
Procedure PAES
  Input: $f_k(X)$
1 repeat
2     Initialize Single Population parent, $C$, and add to archive, $\mathbb{E}'$;
3     Mutate $C$ to produce child $C'$ and evaluate fitness;
4     if $C \prec C'$ then
5         Discard $C'$;
6     else if $C \succ C'$ then
7         Replace $C$ with $C'$, and add $C'$ to $\mathbb{E}'$;
8     else if $\exists_{c'' \in \mathbb{E}'}(C'' \prec C')$ then
9         Discard $C'$;
10    else
11        Apply test $(C, C', \mathbb{E}')$ to determine which becomes the new current solution
          and whether to add $C'$ to $\mathbb{E}'$;
12    end
13 until termination criteria is met ;
```

Fig. 3. Pseudocode of PAES (adapted from [11]).

software subsystem the approaches are required to deal with cost functions that can take into account metrics relevant at different levels of architectural abstraction. The work of Hewett and Kijsanayothin [21] introduces an approach to integrate components. The goal is to minimize the number of stubs. An approach that allows the use of coupling measures across boundaries of large-scale components is described in [20]. The approach was evaluated for two Java product lines.

In the AO context, the units to be integrated are either classes or aspects. Other dependency relations were introduced, as well as integration strategies. The work of Ceccato et al. [8] uses a strategy in which the classes are first tested without aspects. After this, the aspects are integrated and tested with the classes. Solutions based on graphs have been also investigated. Ré et al. [33,34] propose an extended ORD to consider dependency relations between classes and aspects, and different graph-based strategies to perform the integration and test of AO software. They are: (i) combined: aspects and classes are integrated and tested together; (ii) incremental+: first only classes are integrated and tested, and after only aspects; (iii) reverse: applies the reverse combined order; and (iv) random: applies a random selected order. As a result of the study, the combined and incremental + strategies performed better than the other ones, producing a lower number of stubs.

A limitation observed in traditional algorithms is that most of them recursively identify strongly connected components in the graph (SCCs) and in each SCC remove one dependency that maximizes the number of broken cycles [5]. They optimize the decision without determining the consequences on the ultimate results. In many cases the obtained solutions are sub-optimal. Other limitation is that traditional algorithms are very difficult to be adapted to consider different factors that may influence the stubbing costs. To allow the use of different kind of constraints and coupling measures, search-based approaches have presented the most promising results. Firstly, Genetic Algorithms (GAs) were used [5] with fitness functions based on coupling measures given by the number of attributes and methods to be emulated in the stub. A fitness function based on aggregation of objectives was used to deal with the problem that is in fact multi-objective, and to determine the best weights for the objectives is not an easy task, mainly for complex cases.

To overcome these limitations, solutions based on multi-objective algorithms were proposed. Works on this subject are the most related to ours and are summarized in Table 1. Cabral et al. [7] investigated a solution based on the multi-objective Ant Colony Optimization (PACO). The multi-objective approach presents better solutions than the aggregation functions of Briand et al. Furthermore, this approach does not need weights adjustments and generates a set of good solutions to be used by the tester. In [39] the authors extended previous work and compared three different multi-objective algorithms: NSGA-II, MTabu (Multi-objective Tabu Search) and PACO, considering number of attributes and methods. NSGA-II, the evolutionary one, obtained the best results. The NSGA-II and SPEA2 algorithms were also used with four coupling measures by Assunção et al. [2] and obtained satisfactory solutions.

Motivated by these preliminary results we explored in a previous work the use of the MOEAs NSGA-II and SPEA2 to establish integration test orders for AO programs. We have investigated the performance of these algorithms with two measures [14]. In [14] we compared the MOEAs with the approaches based on a simple GA and a traditional algorithm (Tarjan's algorithm). The results obtained by the MOEAs were also the best ones. In [13] we introduce a multi-objective and coupling-based approach, and investigated the use of such approach in the AO context with four measures. We did not observe any decrease in the performance of the algorithms.

By analyzing the works from Table 1 the following points are observed.

- Solutions based on search based algorithms present better results than traditional ones [7,39,14]. For this reason MOCAITO suggests the use of multi-objective algorithms.
- MOEAs present better results than other algorithms [39]. Due to this our evaluation uses evolutionary algorithms.
- The most used algorithms are the MOEAs NSGA-II and SPEA2. NSGA-II is one of the most known and used MOEA. NSGA-II and SPEA2 are considered classical methods to be used in multi-objective optimization [11]. However, other evolutionary algorithms could be evaluated. Considering this, we now evaluate a not previously used algorithm, PAES. PAES has a different evolution strategy and in a recent study obtained better results than NSGA-II and SPEA2 for a known Software Engineering scheduling problem [9].
- NSGA-II and SPEA2 present good results working with 4 objectives [2,13] but these results were not compared with the results obtained with 2 objectives. Due to this, the evaluation herein described, we used two and four objectives.
- We observe that two development contexts were addressed, OO and AO systems, however the performance of the algorithms in such contexts were not compared. In addition to this, other development contexts should be addressed, such as development oriented to components and software product lines. Hence, a generic approach for different contexts is required.

Table 1
Works with multi-objective approach.

REF	Context	Algorithms	Objectives	Systems
[7]	OO	GA, PACO	Attributes, methods	5 (Small systems)
[39]	OO	GA, NSGA-II, MTABU, PACO	Attributes, methods	5 (Small systems)
[2]	OO	NSGA-II, SPEA2	Attributes, methods, return types, parameters types	4 (Large systems)
[14]	OA	Tarjan, GA, NSGA-II, SPEA2	Attributes, operations	4 (Large systems)
[13]	OA	NSGA-II, SPEA2	Attributes, operations, return types, parameters types	4 (Large systems)

These points are motivations to our approach and research questions of our evaluation, subjects addressed in next sections.

4. The MOCAITO approach

This section introduces an approach for solving the integration and test order problem considering different kind of units to be integrated and that can be used in different contexts. The approach is based on multi-objective optimization and it is called MOCAITO (*Multi-objective Optimization and Coupling-based Approach for the Integration and Test Order problem*). MOCAITO includes a set of steps, presented in Fig. 4, which produce artifacts and allow the use of different coupling measures (objectives).

First of all, a dependency model, which consists on the input to MOCAITO, is constructed to represent the dependency relations to be considered. An example of dependency model to be used is the ORD. Other required step is the definition of a cost model. The cost model is generally based on software measures whose values are used by the fitness functions of the algorithms, and need to be collected since they are in fact the objectives to be minimized. After this, multi-objective optimization algorithms are applied. They can work with constraints given by the tester, which can make an order invalid. The algorithms generate as output a set of good solutions for the problem (orders). The orders can be ranked according to some priorities of the tester and/or constraints associated to the software development.

MOCAITO is a generic approach and can be instantiated for a specific usage context. In this paper, we present examples and results of such instantiation in the OO and AO contexts. The next sections illustrate each step of the approach in both contexts.

4.1. Construction of the dependency model (Input)

The input to MOCAITO approach is information about dependencies between modules. So, the first step of the approach produces a representation for the dependency relations to be considered during the multi-objective optimization. The model is constructed according to the context, and kind of unit to be tested. Different constraints to some kind of dependency can also be represented. The dependencies can be between classes, between methods, between methods and classes, between aspects and between classes and aspects, between components, services and so on.

In our example, as well as in previous works [2,39], we use the ORD. In this graph the vertexes correspond to the classes and the edges to the relations. We use the same dependency relations proposed in [4]: As (association), Ag (aggregation), Cp (composition), Us (use) and I (inheritance). Other examples of models to be used in the OO context are: TDG [37] and ETDG [41]. Other dependencies relations can be also represented, such as polymorphism, and so on.

In the ORD extension [33] the vertexes can also represent aspects, and other kind of dependency relations that consider specific characteristics of the AO programming are possible. They are: C (crosscutting association): represents the association generated by a pointcut with a class method or other advice; As (association): occurs between objects involved in pointcuts; U (use): generated by a relation between advices and pointcuts, and between pointcuts; It (inter-type declaration): occurs when there are inter-type relationships between aspects and the base class; I (inheritance): represents inheritance relationships among aspects or among classes and aspects.

Fig. 5 contains an example of extended ORD. In the left side of the figure are the vertexes representing only class and their relations, the extended part is on the right representing the aspects and the new dependency relations introduced by Ré et al. [34].

Graphs representing higher-level dependencies in the software architecture can also be used. In the product line context variabilities mechanisms and variant points and other coupling dependencies should be considered.

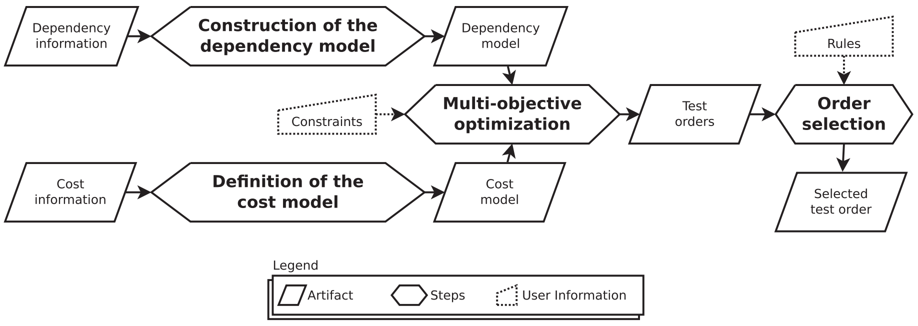


Fig. 4. MOCAITO steps.

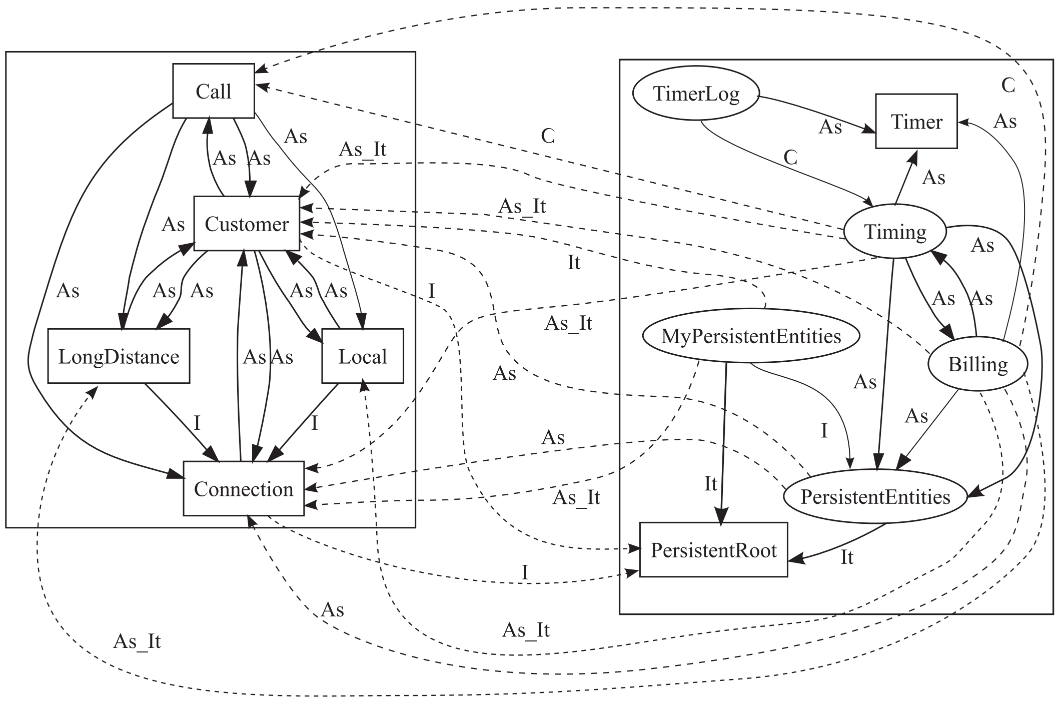


Fig. 5. Example of extended ORD (extracted from [34]).

4.2. Definition of the cost model (Objectives)

This step defines a cost model based on different coupling measures that qualify a dependency and contribute to increase stubbing costs. The measures are in fact the objectives of the fitness functions.

In this paper, we evaluate the approach in the OO and AO contexts using coupling measures. Considering this, we use lower abstraction level metrics, defined to measure the dependencies between server and client modules: number of attributes, number of operations, number of distinct return types and number of distinct parameter types. The two first measures are traditionally used in related works [7,14,34,39]. The two last ones were used by [1,2,13].

Hence, considering that: (i) a unit is a module to be tested that can be either classes or aspects; (ii) m_i and m_j are two coupled modules; and (iii) the operation term represents class methods, aspect methods and/or aspect advices, the used coupling measures are defined as follows:

- Number of attributes (A) = The number of attributes locally declared in m_j when references or pointers to instances of m_j appear in the argument list of some operations in m_i, as the type of their return value, in the list of attributes (data members) of m_i, or as local parameters of operations of m_i (adapted from [5,39]). This complexity measure counts the (maximum) number of attributes that would have to be handled in the stub if the dependency were broken.
- Number of operations (O) = The number of operations (including constructors) locally declared in m_j that are invoked by operations of m_i (adapted from [5]). This complexity measure counts the number of operations that would have to be emulated if the dependency were broken.
- Number of distinct return types (R) = Number of distinct return types of the operations locally declared in m_j that are called by operations of m_i. Void is not counted as return type.
- Number of distinct parameter types (P) = Number of distinct parameters of the operations locally declared in m_j, called by operations of m_i. When there is overloading operations, the number of parameters is the sum of all distinct parameter types among all implementations of each overloaded operation. So, it is considered the worst case, represented by situations in which the coupling consists of calls to all implementation of a given operation.

4.3. Multi-objective optimization (constraints and optimization)

This step is the application of a multi-objective optimization algorithm. In the literature different kind of algorithms are available and can solve efficiently the problem. However, based on the works reported in the literature (Section 3) we suggest the use of evolutionary algorithms based on Pareto concept of dominance [29]. According to this, we use in our work three GA based algorithms: NSGA-II, SPEA2 and PAES.

In this step some constraints that should be implemented by the algorithms are provided by the tester. These constraints are related to restrictions imposed by the software development context and environment, or other ones to ease the test activity. For example, there can be groups of units that should be developed and tested together. The work of Briand et al. [5] and our previous work [2,14,13] adopt that inheritance and inter-types declarations dependencies should not be broken. Other examples of constraints to be considered are the strategies to integrate aspects and classes proposed by Ré et al. [34]. For instance, in the incremental + strategy aspects and classes are integrated in a separate way. Details about the implementation of the algorithms and constraints used in our evaluation are in Section 5.2.

4.4. Order selection (output)

The output of the multi-objective optimization is a set of solutions representing test orders that have the best trade-off between the objectives. In this step, the tester selects an order from the set of non-dominated solutions produced automatically by the algorithms. This selection should be based on constraints and priorities related to the software development, such as test goals, available resources, and contractual restrictions.

To illustrate this step, consider the set of solutions obtained by a MOEA presented in Table 2. This table presents the cost of the solutions according Section 4.2 and were obtained in the experiments described in the next section (program JHotDraw). If Solution 1 were chosen, it would be necessary to create stubs for emulating dependencies of 27 attributes, 19 operations with a total of 3 types of returns and 21 different types of parameters. On the other hand, if Solution 2 of the same algorithm were chosen, the corresponding values would be 30, 10, 1 and 16. Remember that those measures were defined in a generic way to be used in the OO and AO contexts. Thus, the operation term is related to classes methods, aspects methods or aspects advices. As we have mentioned before, the obtained solutions are non-dominated. All of them are good solutions and the tester needs to select an order according to his (or her) preferences.

The selection of a solution should be based on the measure that the tester needs to prioritize, the characteristics of the system, or other factors associated to the development process. He (or she) can conduct the integration test by prioritizing either attributes, operations, returns or parameters complexities. If the program under test has complex attributes to be emulated, Solution 1 from Table 2 should be selected since it has the smallest number of attributes. If the emulation of operations or types of returns is complicated, Solution 2 should be selected. Likewise Solution 3 prioritizes P. If the tester does not want to prioritize a specific measure, the best option is to use Solution 4 that represents the best compromise among the four

Table 2
Example of obtained solutions.

Solution	(A, O, R, P)	Order
1	(27,19,3,21)	**87 24 96** 84 114 25 56 99 47 121 173 77 115 102 101 49 180 0 105 17 26 116 191 186 9 62 131 79 86 118 48 187 157 188 104 190 138 12 58 15 19 6 103 184 112 127 53 36 63 168 10 93 81 23 7 111 147 133 143 136 20 161 119 128 149 76 85 142 154 108 100 46 91 196 89 50 88 97 98 42 41 183 82 160 164 51 141 13 61 163 106 57 185 189 155 123 11 22 34 37 68 3 152 195 28 33 170 78 158 90 14 148 32 92 16 75 30 117 69 66 165 67 166 18 73 59 29 139 172 39 27 5 193 140 31 21 159 74 178 64 181 124 **182** 80 70 146 55 125 135 134 120 194 153 107 144 71 45 2 35 162 175 4 1 54 43 150 8 113 40 167 72 110 192 83 52 156 38 **151 65** 122 179 129 132 130 176 94 60 171 137 109 44 **169** 177 145 95 126 174
2	(30,10,1,16)	**87 96 24** 186 84 48 47 102 133 17 131 9 77 25 103 23 116 99 101 56 114 184 164 100 138 180 149 188 191 190 163 117 147 85 142 76 15 115 123 108 12 16 34 196 187 121 0 22 79 143 6 127 165 112 168 14 93 160 105 18 64 53 128 49 82 13 36 10 50 161 141 106 154 157 104 92 118 140 86 37 62 51 119 26 66 58 155 146 57 **182 69 68** 89 181 21 136 162 80 195 148 67 42 109 11 45 98 156 189 124 61 3 28 170 19 97 153 40 158 73 152 33 185 125 159 46 30 29 27 75 31 120 63 175 173 74 32 20 144 194 71 59 43 192 81 107 5 52 83 78 54 183 7 166 111 135 139 145 172 39 41 4 35 55 132 95 134 130 60 91 178 1 70 167 94 90 150 110 113 2 177 122 174 44 88 179 171 126 193 72 38 176 129 137 8 **65 151 169**
3	(40,15,6,12)	**87 96 24** 186 84 133 48 47 102 17 131 9 77 25 103 23 116 99 101 56 100 114 180 184 149 164 138 188 163 190 191 117 147 85 142 76 15 115 123 108 12 6 16 34 127 196 165 187 93 160 121 0 112 22 128 79 143 82 105 49 18 53 36 13 10 50 161 141 106 66 154 64 62 155 146 26 140 37 157 104 86 58 **182 69** 57 181 92 170 89 71 80 195 162 42 19 63 11 51 40 20 125 67 118 144 175 194 43 45 156 189 124 61 3 173 14 74 32 21 119 168 59 28 98 158 73 153 136 68 97 152 33 185 109 148 159 46 120 29 192 81 30 27 75 31 107 5 52 83 78 54 183 7 166 111 135 139 145 172 39 41 4 35 55 134 70 132 130 60 91 178 167 94 90 150 110 113 2 177 122 174 44 193 95 88 179 171 176 129 72 38 137 1 8 **65 151 169** 126
4	(29,11,3,16)	**87 96 24** 186 84 48 133 47 102 17 131 9 77 25 103 23 116 99 101 56 100 114 180 184 149 164 138 188 190 191 112 163 168 117 147 85 142 76 15 115 123 108 12 6 16 34 127 196 165 14 187 93 160 0 105 79 18 64 53 121 128 143 49 82 22 13 36 10 50 161 141 106 154 157 104 92 118 175 140 86 37 62 51 119 26 66 58 155 146 57 **182 69 68** 173 74 32 89 181 21 20 136 162 80 195 148 67 42 109 63 11 45 98 156 189 124 61 3 28 158 73 152 33 185 125 159 46 30 170 19 29 27 75 31 120 194 59 144 139 145 97 153 81 78 183 107 5 52 192 83 54 40 43 7 166 111 135 132 172 39 41 71 95 4 35 55 134 130 60 91 178 1 70 167 94 90 150 110 113 177 2 174 44 88 179 171 126 193 72 38 176 129 137 8 **65 151 169** 122

Table 3
Used systems.

System	Language	Units		Dependencies	LOC
		Classes	Aspects		
BCEL[a]	Java	45	–	289	2999
JBoss[b]	Java	150	–	367	8434
JHotDraw[c]	Java	197	–	809	20273
MyBatis[d]	Java	331	–	1271	23535
AJHotDraw[e]	AspectJ	290	31	1592	18586
AJHSQLDB[f]	AspectJ	276	25	1338	68550
HealthWatcher[g]	AspectJ	95	22	399	5479
Toll System[h]	AspectJ	53	24	188	2496

[a] http://archive.apache.org/dist/jakarta/bcel/old/v5.0/: library that enables its users to analyze, create, and manipulate (binary) Java class files.
[b] http://www.jboss.org/jbossas/downloads.html – subsystem used: management of JBoss (version 6.0.0M5): is a Java application server
[c] http://sourceforge.net/projects/jhotdraw/ – package used: org.jhotdraw.draw of JHotDraw (version 7.5.1): framework for the creation of drawing editors
[d] http://code.google.com/p/mybatis/downloads/list: data mapper framework that makes easier the use a relational database with OO application
[e] http://sourceforge.net/projects/ajhotdraw/ (version 0.4): is an AO refactoring of the JHotDraw
[f] http://sourceforge.net/projects/ajhsqldb/files/ (version 18): is an AO refactoring of HSQLDB, which is a database manager developed in Java
[g] http://www.aosd-europe.net/ (version 9): collects and manages public health related to complaints and notifications
[h] http://www.comp.lancs.ac.uk/ greenwop/tao (version 9): is a concept proof for automatic charging of toll on roads and streets

objectives, since it does not have any objective with extreme values. It is important to note that due to the interdependence among the measures, to decrease the cost related to an objective may involve to increase another one.

Other factors can influence the tester's decision. For example, suppose that a team are working in units 182, 69 and 68 that need to be tested in sequence. Solutions 2 and 4 satisfy this restriction.

The tester can use the solutions in another way to get useful information about the program under test. We can observe that all orders suggest units 87, 96 and 24 should be first tested. This fact points out that these units should be first available. In another hand, in all orders units 65, 151 and 169 are not in the beginning. Moreover, three orders suggest those units should be the last ones.

5. Empirical evaluation description

This section describes how the empirical evaluation was conducted: used systems, implementation of the algorithms, and quality indicators considered to compare the algorithms. The methodology employed is similar to the adopted in our previous work [2,13,14], however, all the algorithms were configured and re-executed, hence, the results herein reported are not the same, since the algorithms are not deterministic.

5.1. Used systems

The study used eight real systems. Table 3 presents some information about these systems, such as number of modules (classes for Java programs; classes and aspects for AspectJ), dependencies and LOC.

To collect the measures, a parser based on AJATO[1] was developed to identify the existing dependencies between modules from programs code. The parser receives the Java/AspectJ code as entry and returns the syntactic tree code. From this tree, the associations, uses, inheritances, advices, pointcuts and inter-type declarations dependencies are identified. At the end, the parser generates as output matrices that are used as input to the MOEAs, described next.

5.2. Implementation of the algorithms

Similarly to the previous works [2,13,39], the chosen representation to the problem (chromosome) is a vector of integers, where each vector position corresponds to a unit. The size of the chromosome is equal to the number of units of each system. Thus, being each unit represented by a number, an example of a valid solution for a problem with five units is $\{2,4,3,1,5\}$. In this example, the first unit to be tested and integrated would be the unit represented by number 2.

The input data of the algorithm consist of matrices related to the cost model. They are matrices associated to: dependencies between units and each metric of the cost model. Based on the dependency matrices, precedence constraints are defined. In this work, such constraints are do not break inheritance and inter-types declarations dependencies. This means that base units must precede child units in any test order. The matrices are used to calculate the fitness of each solution, where the

[1] http://www.teccomm.les.inf.puc-rio.br/emagno/ajato/.

sum of dependencies between the units for each measure corresponds to an objective, so the goal is to minimize all objectives.

The algorithms were implemented from the version available at JMetal 3.0 [16]. The crossover operator is the Two Points Crossover. In this technique, two points are selected randomly, and the genes inside them are swapped in the children. The remaining genes are used to complete the solution, from left to right. The mutation operator adopted uses the Swap Mutation technique where two genes are randomly selected and are swapped in the child. The use of crossover and mutation operators can generate test orders that break the precedence constraints between the modules (dependencies of inheritance, aggregation and inter-type declaration). The strategy adopted to deal with these constraints consists to check the test order, and if an invalid solution were generated, the module in question is placed at the end of the test order. In our AO context implementation, aspects and classes are integrated together. This corresponds to the strategy combined of Ré et al. [34].

The methodology adopted to configure the parameters was extracted from our previous works [2,13,14]. Hence, some parameters have similar values, except for PAES, which had not been applied before. The parameters used are presented in Table 4.

In order to answer our research questions, each evolutionary algorithm was executed 30 times for each system, first considering only two measures: A and O (named here **Experiment 2M**); and after considering four measures: A, O, R, and P (named **Experiment 4M**). The parameter number of fitness evaluations was used as stopping criterion. All the algorithms executed the same number of fitness evaluations in order to analyze whether they can produce similar solutions when they are restricted to the same resources (number of fitness evaluations).

In Experiment 2M there were three matrices as input that were read from text files: (i) dependencies between modules; (ii) measure A; and (iii) measure O. In Experiment 4M there were more two matrices for measures R and P. As mentioned before, each measure corresponds to an objective and the goal is to minimize all objectives.

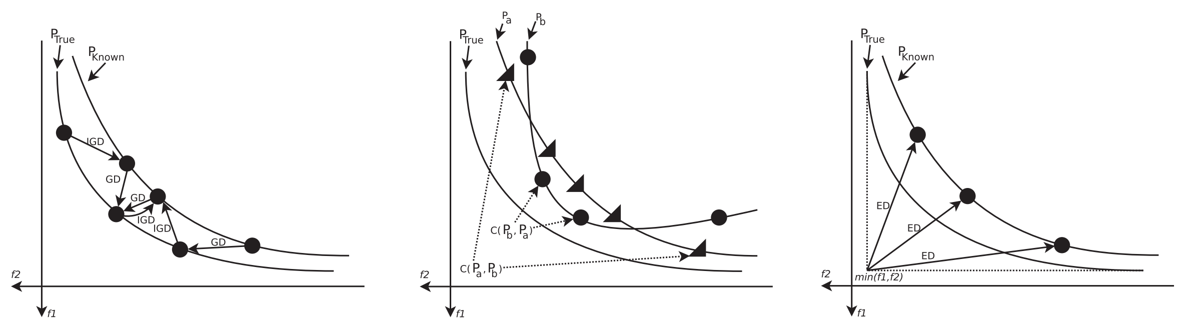
5.3. Quality indicators

The results of the MOEAs are evaluated using quality multi-objective indicators and appropriate statistical analysis as suggested by [44,17]. Four quality indicators were chosen: Coverage (C) [25], Generational Distance (GD) [38], Inverse Generational Distance (IGD) [32] and Euclidean Distance from an ideal solution (ED) [10]. The C and ED indicators were used in [14]. Considering that the MOEAs deteriorate their performance with more than two objectives, we decided to add two other quality indicators: GD and IGD, to analyze the convergence and diversity of the algorithms regarding to the Pareto front. Some of these quality indicators need the Pareto Front True (PF_{true}), however, in real problems PF_{true} is not known. In these cases, it is common to use the non-dominated solutions found by all MOEAs in all runs [42,44].

The GD indicator [38] is used to calculate the distance from an approximation of the Pareto Front found by the algorithm (PF_{known}) to PF_{true}. So, GD is an error measure used to examine the convergence of an algorithm to PF_{true}. IGD [32] is an indicator based on GD, but with the goal of evaluating the distance from PF_{true} to PF_{known}, i.e., the inverse of GD. Fig. 6(a) presents

Table 4
Algorithms parameters.

Parameter	NSGA-II	SPEA2	PAES
Population size	300	300	300
Mutation rate	0.02	0.02	1
Crossover rate	0.95	0.95	–
Archive size	–	250	250
Number of fitness evaluations	60,000	60,000	60,000



(a) GD and IGD (b) Coverage (c) ED

Fig. 6. Quality indicators.

an example of GD and IGD indicators. For these indicators, values closer to 0 are desired, since 0 indicates that all points of PF_{known} are points on PF_{true} for GD or PF_{known} found all the points on PF_{true} for the IGD.

The C indicator [25] measures the dominance between two sets of solutions. C (PF_a, PF_b) represents a value between 0 e 1 according to how much the set PF_b is dominated by set PF_a. Similarly, we compare C (PF_b, PF_a) to obtain how much PF_a is dominated by PF_b. Fig. 6(b) presents an example of C indicator for a minimization problem with two objectives. For instance, C (P_a, P_b) corresponds to 0.5 because the P_b set has two of its four elements dominated by P_a set. Value 0 for C indicates that the solutions of the former set do not dominate any element of the latter set; on the other hand, value 1 indicates that all elements of the latter set are dominated by elements of the former set.

The ED indicator has the goal of finding the closest solutions to the best objectives, i.e., the ideal solution. An ideal solution has the minimum value of each objective, considering a minimization problem [10]. These minimum values are obtained from all non-dominated solutions. Fig. 6(c) depicts an example of ED for a minimization problem with two objectives.

5.4. Statistical analysis

In the validation of the proposed approach, every algorithm runs several times and the results of the quality indicators GD and IGD are compared using the Friedman test [17] from the R tool [31]. The Friedman test is a non-parametric statistical test used to compare multiple data sets. In this test, the observations are ranked. For example, each execution of each algorithm generates a block of data for each indicator that is independent and it can be ranked among the other indicator values. The test does not use the data values, but the rankings obtained for each sample in the data set. The null hypothesis is that there is no difference between the algorithms. Thus, if the null hypothesis is rejected, the test shows that there is a difference between the sets analyzed. In our analysis 5% significance level is used ($P - value <= 0.05$). After, the post-hoc test [35] from R tool is applied, using the package pgirmess. The post-hoc test indicates if there is a statistical difference between the different blocks of the data sets, to identify which blocks obtained the best values, the averages of the indicators are used.

6. Analysis

In this section the results are presented and analyzed in order to answer our research questions.

Table 5 presents the main results obtained by each algorithm and system in Experiments 2M and 4M. Column 2 presents the cardinality of PF_{true}, formed by the non-dominated solutions obtained considering all algorithms executions. The average number of solutions found by each algorithm per run is also presented, as well as, in parentheses, the number of solutions of the set PF_{known}. In addition, the average of runtime, in seconds, and the standard deviation (in parentheses) are presented for all systems.

A preliminary analysis of this table shows that JBoss, Health Watcher and Toll System have only one solution in PF_{true}, independently of the used objectives. Moreover, all the algorithms find this solution in almost all runs. For these three systems, the average number of solutions is greater than 1 (e.g. 1.07) because in some runs the algorithms found more than one solution. They are examples of systems in which the objectives are not in conflict. This does not happen with the other systems with greater number of dependencies and LOC. For them PF_{true} contains a greater number of solutions in both experiments. It is also noticeable that the greater the number of objectives, the greater this cardinality. The main difference with

Table 5
Number of solutions and runtime.

System	PF_{true}	NSGA-II		SPEA2		PAES	
		# Solutions	Runtime	# Solutions	Runtime	# Solutions	Runtime
Experiment 2M							
BCEL	29	28.73 (29)	5.37 (0.04)	28.93 (29)	184.51 (21.88)	25.70 (29)	**1.88** (0.07)
JBoss	1	1.00 (1)	19.25 (0.18)	1.00 (1)	2666.66 (585.32)	1.00 (1)	**10.46** (0.08)
JHotDraw	3	1.40 (3)	31.29 (0.18)	1.23 (2)	3213.17 (677.40)	1.83 (2)	**19.06** (0.13)
MyBatis	63	60.60 (63)	79.18 (0.33)	58.60 (57)	132.41 (15.55)	43.00 (54)	**52.30** (0.20)
AJHotDraw	7	4.57 (6)	81.44 (0.41)	4.63 (6)	1375.29 (418.06)	5.87 (7)	**53.53** (0.31)
AJHSQLDB	40	31.53 (35)	67.13 (0.21)	26.40 (36)	101.65 (3.86)	26.23 (40)	**44.57** (0.23)
Health Watcher	1	1.00 (1)	13.00 (0.17)	1.00 (1)	2897.21 (744.19)	1.07 (1)	**6.72** (0.07)
Toll System	1	1.00 (1)	7.10 (0.08)	1.00 (1)	3541.92 (804.71)	1.00 (1)	**2.72** (0.01)
Experiment 4M							
BCEL	37	37.43 (37)	**5.91** (0.05)	36.70 (37)	123.07 (18.84)	39.30 (37)	6.58 (1.25)
JBoss	1	1.00 (1)	18.73 (0.20)	1.00 (1)	2455.35 (612.18)	1.13 (1)	**10.69** (0.62)
JHotDraw	11	8.40 (10)	29.85 (0.34)	9.63 (9)	922.99 (373.98)	10.47 (19)	**24.29** (1.50)
MyBatis	789	276.37 (941)	**74.03** (0.87)	248.77 (690)	128.88 (2.65)	243.60 (679)	104.30 (7.91)
AJHotDraw	94	70.03 (79)	75.05 (0.57)	68.87 (78)	195.56 (28.22)	40.73 (84)	**62.07** (2.16)
AJHSQLDB	266	156.63 (360)	**62.34** (0.53)	119.10 (52)	104.29 (0.68)	145.97 (266)	75.62 (5.27)
Health Watcher	1	1.00 (1)	12.72 (0.15)	1.00 (1)	2580.39 (596.29)	1.07 (1)	**8.27** (0.58)
Toll System	1	1.00 (1)	7.33 (0.09)	1.00 (1)	3516.71 (570.76)	1.07 (1)	**4.10** (0.75)

respect to the PF_{true} cardinality occurs for MyBatis. Maybe in this system the methods have more parameters and return values to be emulated.

We can also observe that SPEA2 requires a higher runtime with greater standard deviation than NSGA-II and PAES and that there is no increase in the execution time for all algorithms in Experiment 4M. A simple analysis shows that all the algorithms can be used to efficiently solve the problem, however the quality indicators were applied and used in next section to better compare the algorithms.

The results of C indicator are presented in Table 6. The solutions of the MOEA that appears in a row have the value of domination on the solutions of the MOEA that appears in the column. Values greater than 0.5 are significant and indicate more than 50% of dominance. The significant values are highlighted in bold.

Regarding to Experiment 2M, we observe significant difference in three systems: MyBatis, AJHotDraw and AJHSQLDB. For system MyBatis, the solutions achieved by NSGA-II dominate almost 67% of SPEA2 solutions and 72% of PAES solutions; the SPEA2 solutions dominate almost 63% of PAES solutions; and the PAES solutions do not dominate significantly the solutions of any other MOEA. For system AJHotDraw, the NSGA-II solutions dominate all SPEA2 solutions; PAES solutions dominate 83% of NSGA-II solutions and all SPEA2 solutions (100%); the SPEA2 solutions do not dominate any solutions of any other MOEA. The results for the system AJHSQLDB are similar to the AJHotDraw results, despite of the SPEA2 solutions also dominate all NSGA-II solutions.

Table 7 presents the results for GD and IGD indicators. These results are the average and the standard deviation of GD and IGD of the thirty PF_{approx} sets achieved by each MOEA. To verify the MOEAs that present significant difference, the Friedman test [17] was used, with confidence level of 95%.

Table 8 contains the results of the Friedman statistical test. *P*-values that attest significant difference among the MOEAs were highlighted in bold. The systems JBoss, Health Watcher and Toll Systems were omitted, since for them the MOEAs have achieved only one solution in PF_{true}.

For Experiment 2M, according to the statistical test, there is significant difference in GD and IGD for four systems: BCEL, JHotDraw, AJHotDraw and AJHSQLDB. Considering both GD and IGD, NSGA-II and SPEA2 are equivalent and overcame PAES for systems BCEL and JHotDraw the MOEAs, whereas PAES is the best for systems AJHotDraw and AJHSQLDB.

For Experiment 4M, the statistical test denotes significant difference for the results of the indicator GD for four systems: BCEL, MyBatis, AJHotDraw and AJHSQLDB. For the indicator IGD there is significant difference in five systems: the same four systems with difference in indicator GD (BCEL, MyBatis, AJHotDraw, AJHSQLDB) and JHotDraw. Regarding to indicator GD, NSGA-II and SPEA2 are equivalent and better than PAES for systems BCEL and MyBatis, NSGA-II is the best for AJHotDraw, and PAES is the best for AJHSQLDB. Regarding to indicator IGD, for the systems BCEL, JHotDraw, MyBatis and AJHotDraw, the MOEAs NSGA-II and SPEA2 are better, and for the system AJHSQLDB, PAES is the best.

Considering the indicator ED, Figs. 7 and 8 present graphs showing the number of solutions for ED. In these pictures, it is possible to verify which MOEA has the greatest concentration of closest solutions to the ideal solution.

Table 6
C indicator for PF_{know} sets.

System	MOEA	NSGA-II	SPEA2	PAES	System	NSGA-II	SPEA2	PAES
Experiment 2M								
BCEL	**NSGA-II**	–	0.0344828	0.482759	AJHotDraw	–	**1**	0.142857
	SPEA2	0	–	0.448276		0	–	0
	PAES	0	0	–		**0.833333**	**1**	–
JBoss	**NSGA-II**	–	0	0	AJHSQLDB	–	**1**	0
	SPEA2	0	–	0		0	–	0
	PAES	0	0	–		**1**	**1**	–
JHotDraw	**NSGA-II**	–	0	0	Health Watcher	–	0	0
	SPEA2	0	–	0		0	–	0
	PAES	0	0	–		0	0	–
MyBatis	**NSGA-II**	–	**0.666667**	**0.722222**	Toll System	–	0	0
	SPEA2	0.206349	–	**0.62963**		0	–	0
	PAES	0.285714	0.403509	–		0	0	–
Experiment 4M								
BCEL	**NSGA-II**	–	0	0.189189	AJHotDraw	–	**0.807692**	**0.952381**
	SPEA2	0.027027	–	0.216216		0.0506329	–	**0.916667**
	PAES	0	0	–		0	0.0128205	–
JBoss	**NSGA-II**	–	0	0	AJHSQLDB	–	0.307692	0
	SPEA2	0	–	0		**0.602778**	–	0
	PAES	0	0	–		**1**	**1**	–
JHotDraw	**NSGA-II**	–	0	**0.947368**	Health Watcher	–	0	0
	SPEA2	0	–	**0.947368**		0	–	0
	PAES	0	0	–		0	0	–
MyBatis	**NSGA-II**	–	0.0231884	**0.976436**	Toll System	–	0	0
	SPEA2	**0.894793**	–	**0.963181**		0	–	0
	PAES	0	0	–		0	0	–

Table 7
GD and IGD indicators.

Indicator	System	NSGA-II		SPEA2		PAES	
		Average	Standard deviation	Average	Standard deviation	Average	Standard deviation
Experiment 2M							
GD	BCEL	**0.002893**	0.002286	**0.003143**	0.001946	0.009942	0.004169
	JBoss	0	0	0	0	0	0
	JHotDraw	**1.133704**	2.599180	**1.206938**	2.793913	4.568271	4.152712
	MyBatis	0.013541	0.006589	0.016307	0.009722	0.014445	0.006950
	AJHotDraw	0.802435	0.667415	0.758403	0.619125	**0.454011**	0.286958
	AJHSQLDB	0.298636	0.099864	0.403558	0.120089	**0.045367**	0.021198
	Health Watcher	0	0	0	0	0.080800	0.312494
	Toll System	0	0	0	0	0	0
IGD	BCEL	**0.002638**	0.002015	**0.003039**	0.002078	0.011752	0.004477
	JBoss	0	0	0	0	0	0
	JHotDraw	**1.213778**	2.976485	**1.322521**	3.141205	5.069102	4.745460
	MyBatis	0.013799	0.006501	0.015677	0.008448	0.017287	0.009075
	AJHotDraw	0.854634	0.966271	0.771509	0.485032	**0.392923**	0.294614
	AJHSQLDB	0.309241	0.189022	0.422809	0.178879	**0.038639**	0.021898
	Health Watcher	0	0	0	0	0.058486	0.227020
	Toll System	0	0	0	0	0	0
Experiment 4M							
GD	BCEL	**0.001984**	0.001211	**0.002085**	0.001144	0.011376	0.003297
	JBoss	0	0	0	0	0.069611	0.381275
	JHotDraw	0.303920	0.190225	0.408509	0.304008	0.487535	0.281376
	MyBatis	**0.006973**	0.001603	**0.007766**	0.001727	0.016785	0.002996
	AJHotDraw	**0.034252**	0.013653	0.044081	0.014161	0.052848	0.020588
	AJHSQLDB	0.058134	0.022112	0.086932	0.034211	**0.013284**	0.004251
	Health Watcher	0	0	0	0	0	0
	Toll System	0	0	0	0	0.081650	0.447214
IGD	BCEL	**0.001919**	0.001320	**0.001681**	0.001114	0.018785	0.005098
	JBoss	0	0	0	0	0.063221	0.346276
	JHotDraw	**0.249168**	0.258185	0.380334	0.456610	0.493869	0.330265
	MyBatis	**0.010104**	0.003140	**0.008608**	0.003607	0.024483	0.002652
	AJHotDraw	**0.036723**	0.016616	**0.046578**	0.018384	0.070992	0.026762
	AJHSQLDB	0.054444	0.020275	0.084633	0.035610	**0.016285**	0.004057
	Health Watcher	0	0	0	0	0	0
	Toll System	0	0	0	0	0.074536	0.408248

In Experiment 2M, PAES achieves more solutions closest to the ideal solution for MyBatis (Fig. 7(c)), AJHotDraw (Fig. 7(d)) and AJHSQLDB (Fig. 7(e)). For JHotDraw (Fig. 7(c)) NSGA-II finds more closest solutions. For BCEL NSGA-II and SPEA2 find more closest solutions. For Experiment 4M NSGA-II and SPEA2 achieve more closest solutions to the ideal solution to BCEL (Fig. 8(a)) and JHotDraw (Fig. 8(b)). For MyBatis (Fig. 8(c)) SPEA2 finds more closest solutions and NSGA-II for AJHotDraw (Fig. 8(d)).

6.1. RQ1: solutions found by PAES versus other solutions

The first question is concerned about the solutions found by PAES in relation to the solutions found by NSGA-II and SPEA2. Table 9 supports this analysis by presenting the MOEAs with the best results for each indicator in each system. The cells with value "All" indicate that the three MOEAs have the same behavior. We observe that this happens for the less complex systems JBoss, HealthWatcher and Toll System, independently of objectives used. Considering other systems and the indicators, we conducted separate analysis as follows.

- Considering the indicator C (Table 6) and systems where there is difference, we can see in Figs. 9 and 10 the solutions on the search space respectively for Experiments 2M and 4M. In the case of Experiment 2M and MyBatis (Fig. 9(a)) all solutions are in the same area, but NSGA-II solutions have better values. For AJHSQLDB (Fig. 9(b)) it is clear that PAES achieves the best solutions.
  In Fig. 10 (Experiment 4M) the objectives are represented in two pictures. For MyBatis (Fig. 10(a) and (b)) the SPEA2 and NSGA-II solutions are similar, although SPEA2 solutions are the best. PAES solutions are more spread on the search space. In Fig. 10(c) and (d) we can observe that again the best solutions were achieved by PAES for AJHSQLDB. We can summarize the results related to the indicator C considering the highest coverage achieved by each MOEA for each system. In the case of Experiment 2M, PAES obtained the best results for the systems with significant difference among the MOEAs. Regarding to Experiment 4M, NSGA-II and SPEA2 had similar results.

Table 8
Friedman test results for indicators GD and IGD.

Indicator	System	Compared MOEAs	Experiment 2M				Experiment 4M			
			p-value	Observed differ.	Critical differ.	Difference	*p*-value	Observed differ.	Critical differ.	Difference
GD	BCEL	NSGA-II × SPEA2	**6.0e−10**	5	18.54	False	**1.1e−10**	5	18.54	False
		NSGA-II × PAES		46	18.54	True		47.5	18.54	True
		SPEA2 × PAES		41	18.54	True		42.5	18.54	True
	JHotDraw	NSGA-II × SPEA2	**4.7e−05**	2	18.54	False	0.05799	12	18.54	False
		NSGA-II × PAES		26	18.54	True		51	18.54	True
		SPEA2 × PAES		28	18.54	True		39	18.54	True
	MyBatis	NSGA-II × SPEA2	0.29131	1	18.54	False	**5.0e−11**	12	18.54	False
		NSGA-II × PAES		11	18.54	False		51	18.54	True
		SPEA2 × PAES		10	18.54	False		39	18.54	True
	AJHotDraw	NSGA-II × SPEA2	**0.00357**	1	18.54	False	**0.00357**	22	18.54	True
		NSGA-II × PAES		23	18.54	True		23	18.54	True
		SPEA2 × PAES		22	18.54	True		1	18.54	False
	AJHSQLDB	NSGA-II × SPEA2	**2.0e−11**	16	18.54	False	**1.1e−11**	18	18.54	False
		NSGA-II × PAES		37	18.54	True		36	18.54	True
		SPEA2 × PAES		53	18.54	True		54	18.54	True
IGD	BCEL	NSGA-II × SPEA2	**6.8e−10**	3	18.54	False	**0.01364**	2	18.54	False
		NSGA-II × PAES		45	18.54	True		20.5	18.54	True
		SPEA2 × PAES		42	18.54	True		18.5	18.54	False
	JHotDraw	NSGA-II × SPEA2	**2.8e−05**	1	18.54	False	**0.01364**	2	18.54	False
		NSGA-II × PAES		26.5	18.54	True		20.5	18.54	True
		SPEA2 × PAES		27.5	18.54	True		18.5	18.54	False
	MyBatis	NSGA-II × SPEA2	0.13089	1	18.54	False	**1.2e−10**	6	18.54	False
		NSGA-II × PAES		14	18.54	False		42	18.54	True
		SPEA2 × PAES		13	18.54	False		48	18.54	True
	AJHotDraw	NSGA-II × SPEA2	**0.00022**	6	18.54	False	**8.8e−06**	14	18.54	False
		NSGA-II × PAES		24	18.54	True		37	18.54	True
		SPEA2 × PAES		30	18.54	True		23	18.54	True
	AJHSQLDB	NSGA-II × SPEA2	**5.0e−11**	12	18.54	False	**6.6e−11**	17	18.54	False
		NSGA-II × PAES		39	18.54	True		35	18.54	True
		SPEA2 × PAES		51	18.54	True		52	18.54	True

- Considering the indicators GD and IGD (Table 7) in general, NSGA-II and SPEA2 have the same performance and they are better than PAES for almost all systems. The single case where PAES is the best for these two indicators is for AJHSQLDB.
- Regarding the indicator ED, all MOEAs achieve solutions with lower ED for some system.

For Experiment 2M, considering all systems, in addition to achieve the closest solution for four systems, we can state that PAES can achieve more solutions with lower ED than the other two MOEAs. On the other hand, in Experiment 4M PAES achieves the best results for ED only for AJHSQLDB (Fig. 8(e)). From these results we can observe that, in the presence of many objectives (4), NSGA-II and SPEA2 can find a greater number of closest solutions to the ideal solution.

Aiming at answering RQ1, PAES is the best in all quality indicators for AJHotDraw and AJHSQLDB in Experiment 2M. This algorithm is also the best for AJHSQLDB in Experiment 4M. Those two systems have the greatest numbers of units (classes and aspects) and dependencies. So, PAES has better performance to solve the integration and test order problem for more complex systems. The analysis of the solutions achieved by PAES in the solution space for AJHSQLDB (Figs. 9(b), 10(c) and (d)) corroborates the last statement, since its solutions have the lower values for each objective than solutions found by other MOEAs. Besides, in both experiments, it spent less runtime than the other MOEAS in almost all cases (Table 5).

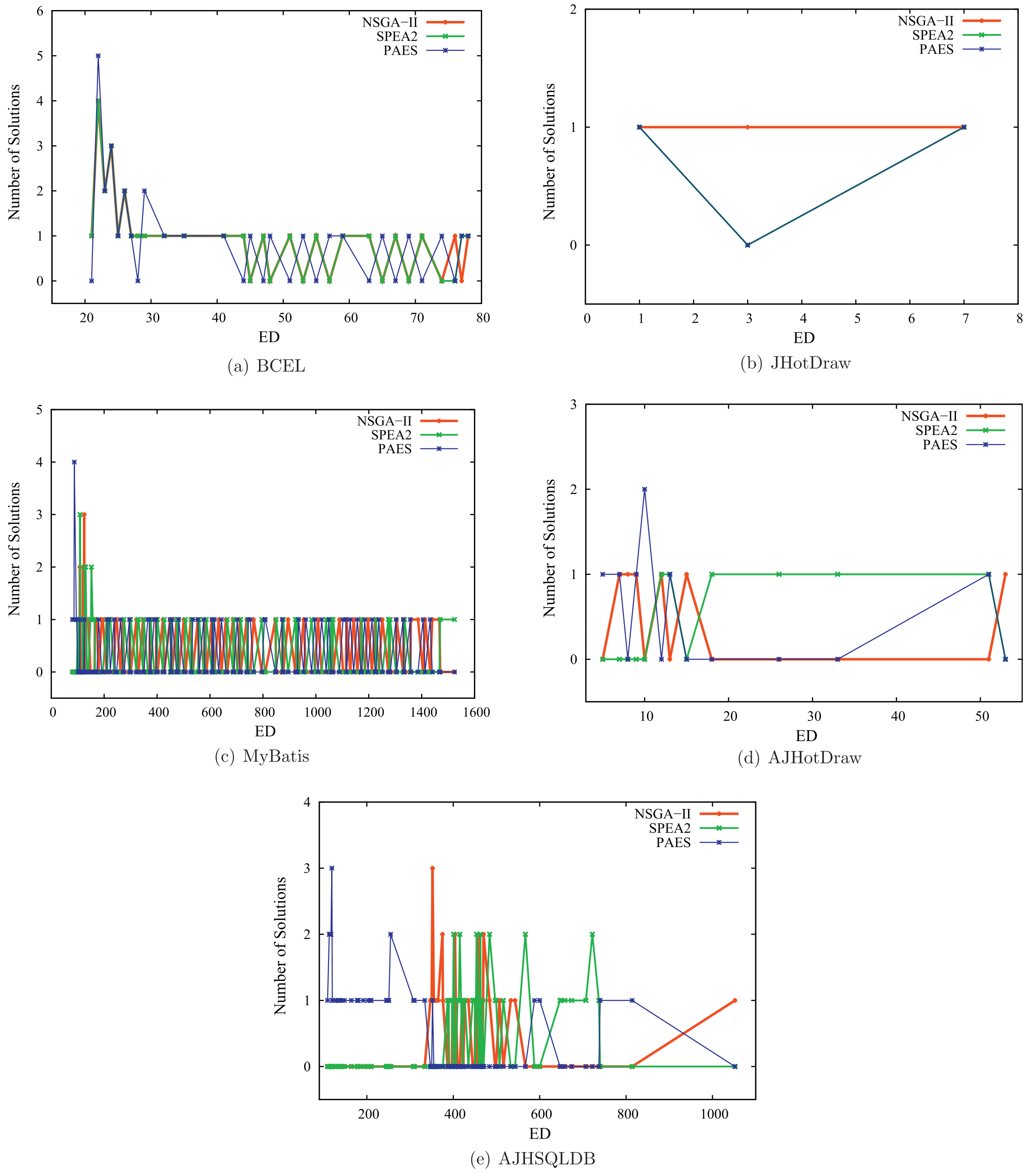
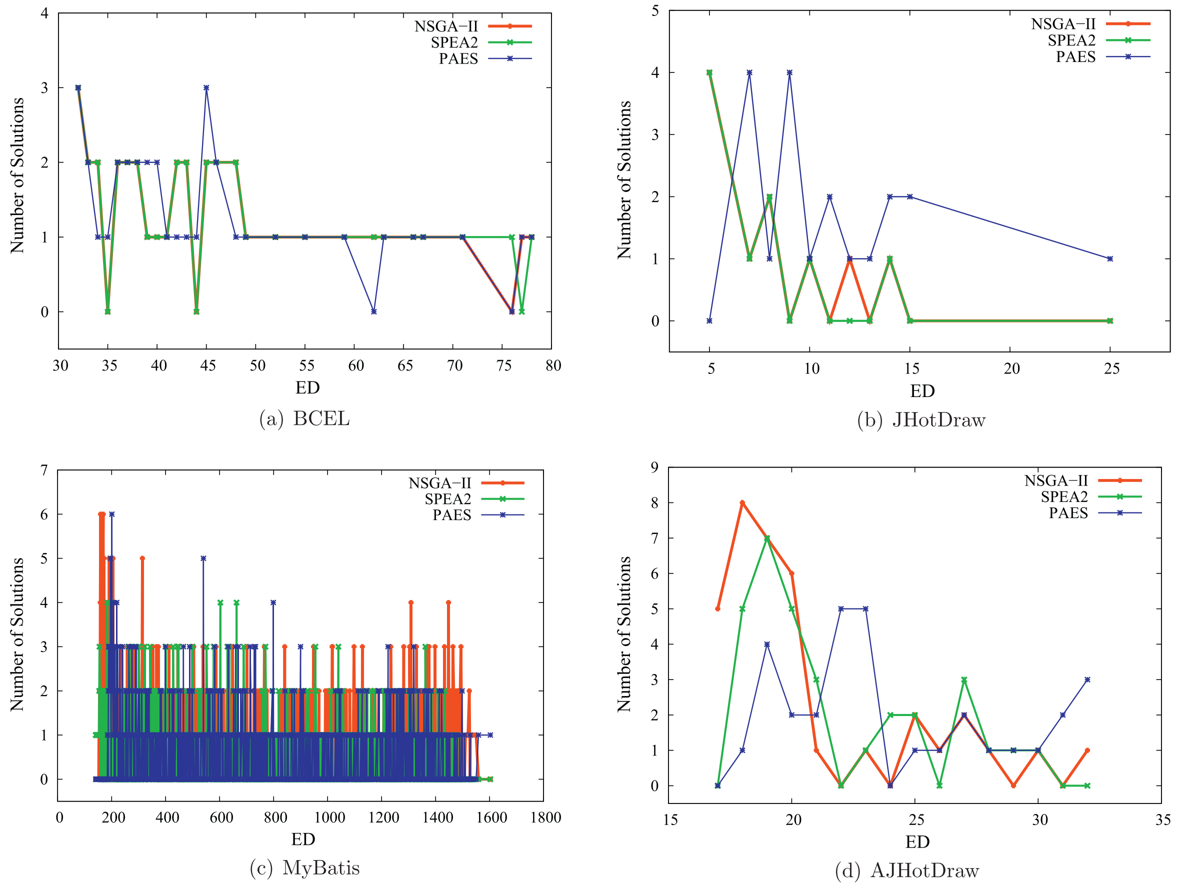


Fig. 7. Number of solutions × indicator ED (Experiment 2M).

6.2. RQ2: influence of the number of objectives

RQ2 asks if the number of objectives influences on the results achieved by the algorithms. Aiming at answering this question, we compared the results of each MOEA in Experiment 4M (four objectives) in relation to its results in Experiment 2M (two objectives) taking into account information from Table 9 and Fig. 11(a) and (b). Fig. 11 presents the graphs that summarize the performance of each MOEA for each quality indicator. In those graphs each bar represents the number of times that a MOEA overcomes other MOEA.



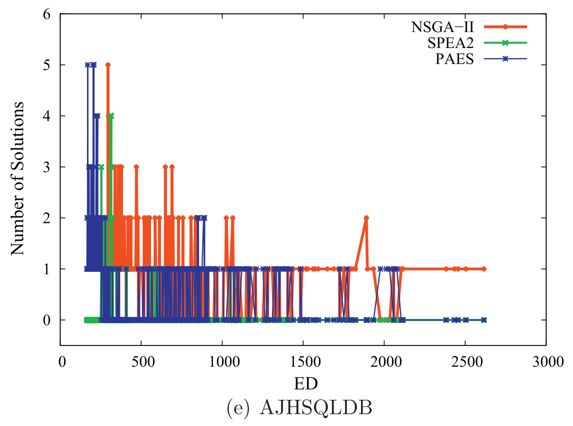


Fig. 8. Number of solutions × indicator ED (Experiment 4M).

It is possible to note that the main difference in performance appears for MyBatis, JHotDraw and AJHotDraw. For the other systems, all MOEAs had the same behavior independently of the number of objectives.

Regarding to AJHotDraw, PAES was the best MOEA in Experiment 2M but it was the worst in Experiment 4M in all quality indicators. NSGA-II had the inverse behavior since it was the worst in Experiment 2M and the best in Experiment 4M. For JHotDraw the three MOEAs have similar behavior despite SPEA2 had a slight better performance with four objectives. Considering MyBatis, SPEA2 also improved its behavior in Experiment 4M, however the other two MOEAs had worse behavior in the same experiment.

So, by analyzing Fig. 11(a) and (b) it is possible to state that the number of objectives influences on the performance of the MOEAs in a certain way, although there is not a default behavior for such influence since the same algorithm sometimes is

Table 9
Better MOEAs by quality indicator.

System	Experiment 2M				Experiment 4M			
	C	GD	IGD	ED	C	GD	IGD	ED
BCEL	All	NSGA-II SPEA2	NSGA-II SPEA2	NSGA-II SPEA2	All	NSGA-II SPEA2	NSGA-II SPEA2	NSGA-II SPEA2
JBoss	All	All	All	All	All	All	All	All
JHotDraw	All	NSGA-II SPEA2	NSGA-II SPEA2	NSGA-II	NSGA-II SPEA2	All	NSGA-II SPEA2	NSGA-II SPEA2
MyBatis	NSGA-II	All	All	PAES	SPEA2	NSGA-II SPEA2	NSGA-II SPEA2	SPEA2
AJHotDraw	PAES	PAES	PAES	PAES	NSGA-II	NSGA-II	NSGA-II SPEA2	NSGA-II
AJHSQLDB	PAES	PAES	PAES	PAES	PAES	PAES	PAES	PAES
Health Watcher	All	All	All	All	All	All	All	All
Toll System	All	All	All	All	All	All	All	All

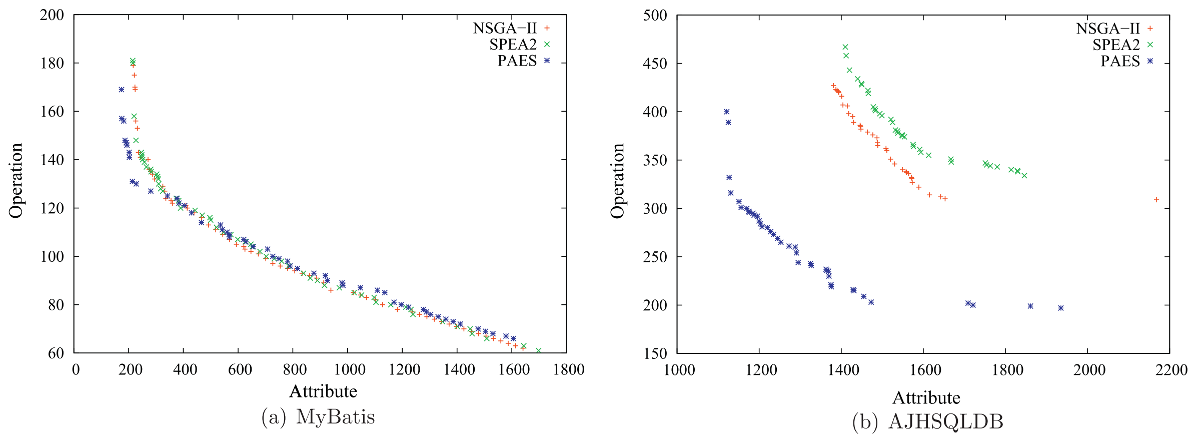


Fig. 9. Search space of Experiment 2M.

better sometimes is worse. Then, it is clear that the characteristics of the system impacts on the performance more than the number of objectives.

6.3. RQ3: influence of the instantiation context on the results

The last question is about how is the algorithms behavior in each instantiation context (OO and AO). We can highlight that:

- PAES achieves the best results only for AO systems (AJHotDraw and AJHSQLDB) in Experiment 2M (Fig. 11(a)).
- For OO systems, NSGA-II and SPEA2 have the same performance with 2 objectives and SPEA2 was the best with 4 objectives (Fig. 11).

So, these two points provide evidences that the instantiation context influences on the performance of each MOEA in addition to the complexity of the system and the number of objectives. For instance, AO context imposes a greater number of dependencies and one more and different kind of constraint, that is the inter-type declarations.

Finally, in the context of our empirical evaluation, PAES fits better to solve the problem for AO systems in the presence of two objectives and with complex systems. From all results of the quality indicators, NSGA-II seems to be more appropriate in general cases because: (i) it has good convergence (GD and IGD indicators), (ii) it finds a set of solutions that cover the solutions found by two other MOEAs (C indicator), (iii) it achieves solutions closer to the ideal solution, and (iv) its good results do not change in the presence of four objectives.

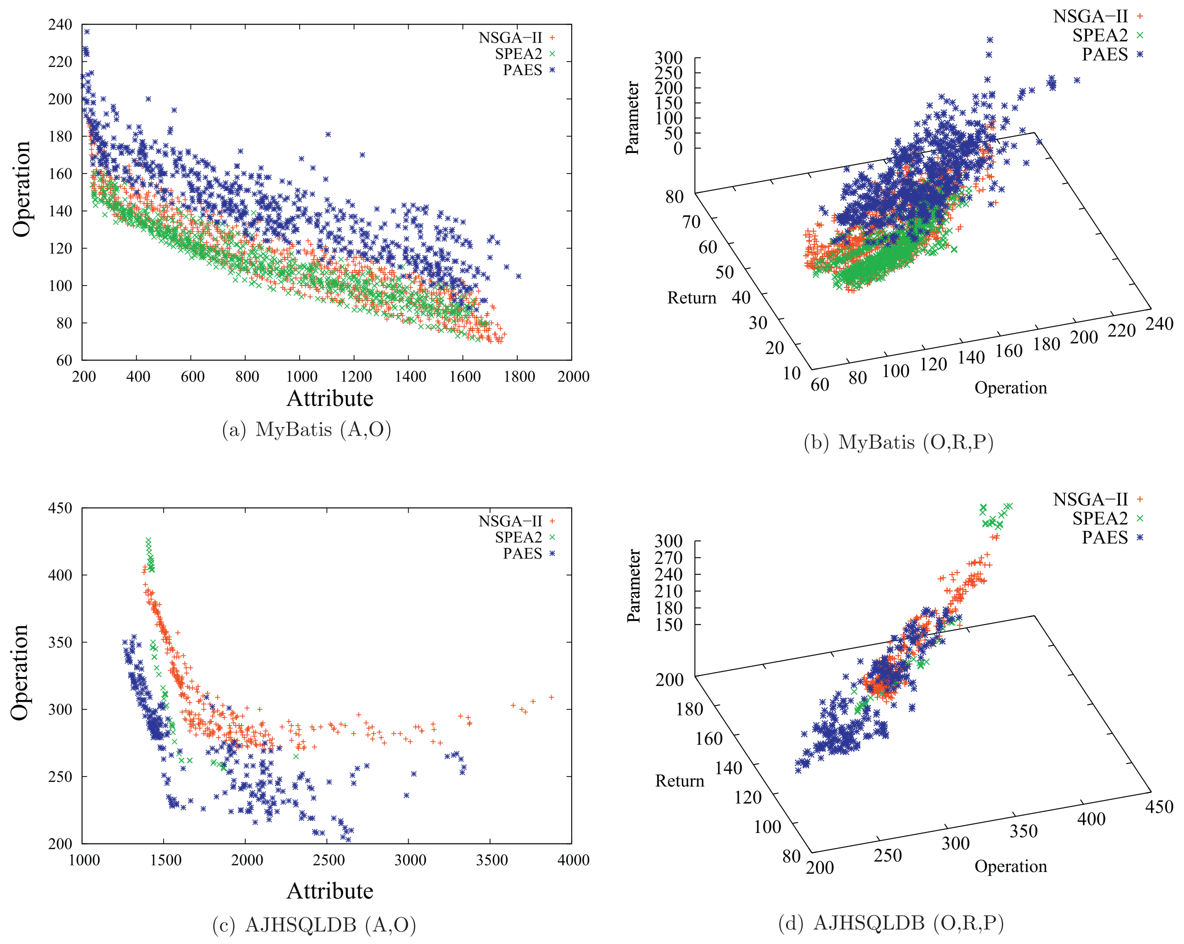


Fig. 10. Search space of Experiment 4M.

6.4. Discussion

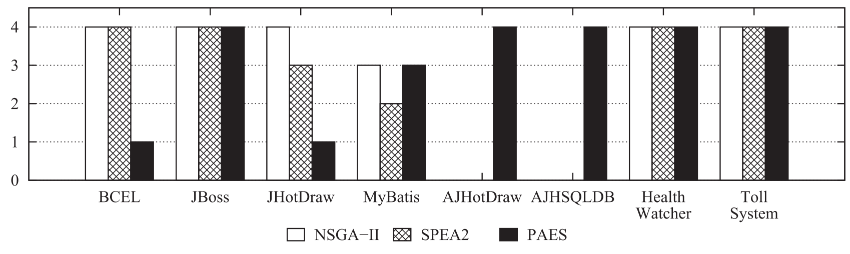
In this section we discuss some points related to the complexity of the systems used in the experiments and the applicability of the proposed approach.

All systems used in the experiment were important to compare the performance of the three MOEAs in the context of the proposed approach. Let's consider a hypothetical situation where one MOEA achieved one solution and the other two MOEAs achieved five or ten solutions for the same simple and small system. In such a situation, it would be observed that these two last MOEAs are not able to satisfactorily solve the problem in this context. In our study, all MOEAs achieved similar and good results even implementing different evolution and diversification strategies.

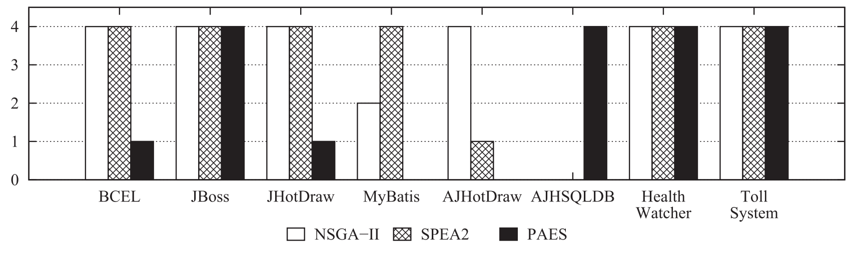
It is difficult to obtain real, industrial systems to perform experiments. In spite of this, obtaining other benchmarks do not guarantee that they would be complex or simple to solve before the approach application, similar what happened in respect to JBoss, Health Watcher and Toll System.

In a practical real application of the approach, the tester works with simple and complex systems. So, a validation study to the approach must involve both kinds of systems. Considering the results obtained in our experiments, it is possible to state that the proposed approach can be applied to either complex or simple systems. In this way, the tester does not need to choose more than one approach for each kind of system or different kinds of instantiation contexts.

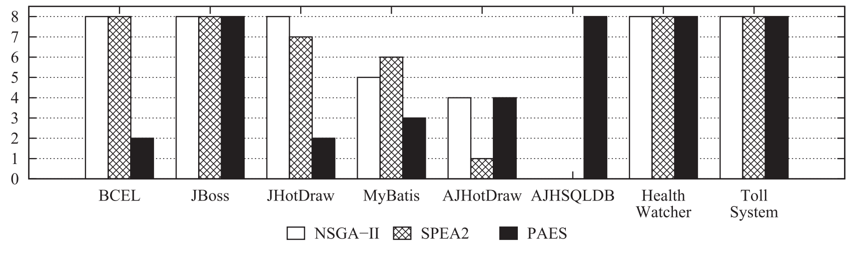
Additionally, even for simple systems, the manual determination of an order to minimize the stub costs implies in a huge effort for the tester. To illustrate this, consider the lowest OA system used in the experiment Toll System, with 53 classes and 24 aspects. For it, there is a number of 1.45E+113 possibilities of different permutations among the modules (classes and aspects) to be analyzed. Since the task of determining a test order is delegated to some MOEA, the tester only needs to concentrate his/her effort on choosing an order achieved by the algorithm.



(a) Experiment 2M



(b) Experiment 4M



(c) Experiments 2M and 4M

Fig. 11. Number of best results $\times$ indicator (C, GD, IGD and ED).

7. Conclusions

This paper presented MOCAITO, an approach for the integration and test order problem in diverse software development contexts, where the units can be components, classes, aspects and so on. Firstly, dependencies among the units need to be represented in a dependency model. In our paper the model used is the ORD, and some most usual dependency relations between classes and between aspects and classes are considered. Relevant information about the order costs is used to construct a cost model. Such model is based on collected measures such as coupling, given by number of attributes, operations, and type and number of parameters. Both models are used as entries to multi-objective optimization algorithms. The algorithms produce a set of non-dominated solutions (good orders) that represent the best trade-off among the objectives. The tester then selects an order according to the testing plans and environment, by using a priority rule.

Results from the empirical evaluation of MOCAITO in OO and AO systems show that the three compared evolutionary algorithms can efficiently solve the problem.

With respect to RQ1, we observe that for simpler systems, there is no difference among the algorithms. However, based on the analysis of the quality indicators GD, IGD, C and ED, NSGA-II seems to be more suitable in most cases considering all systems, with two and four objectives. The algorithm PAES, not explored previously, shows better performance for more complex systems. SPEA2 presents the greatest execution time for all systems and both numbers of objectives.

Regarding RQ2 and RQ3, we observe that the characteristics of the systems, such as, number of dependencies and LOC seem to influence on the number of non-dominated solutions and complexity to solve the problem. The greater the number of objectives the greater this complexity. The development context is related to these characteristics and should be considered.

As future work we intend to perform empirical studies to evaluate the approach in other contexts such as component-based and software product line development. It is also interesting to analyze other multi-objective algorithms with other evolution strategies, such as MOEA/D that decomposes a multi-objective problem into a set of scalar optimization sub-problems. This algorithm performs well when the Pareto front is convex. However, it is not known the kind of Pareto front for each system. The results of the algorithm can provide an insight to answer this question. Other point to be investigated is the addition of other constraints, such as, those ones related to a distributed software development environment. These constraints may reduce the search space and influence on the algorithms performance.

Acknowledgments

We would like to thank Edison Klafke Fillus for his contribution in the experiments. This work was supported by CNPq (Productivity Grant Nos. 303761/2009-1 and 306608/2009-0 and Universal Project Grant Nos. 472510/2010-0 and 481397/2009-4) and by CAPES/Reuni.

References

[1] A. Abdurazik, J. Offutt, Coupling-based class integration and test order, in: International Workshop on Automation of Software Test, May 2006, pp. 50–56.
[2] W.K.G. Assunção, T.E. Colanzi, A. Pozo, S.R. Vergilio, Establishing integration test orders of classes with several coupling measures, in: 13th Genetic and Evolutionary Computation Conference (GECCO'2011), 2011, pp. 1867–1874.
[3] P. Bansal, S. Sabharwal, P. Sidhu, An investigation of strategies for finding test order during integration testing of object oriented applications, in: Proceeding of International Conference on Methods and Models in Computer Science (ICM2CS 2009), 2009, pp. 1 –8.
[4] L.C. Briand, J. Feng, Y. Labiche, Experimenting with genetic algorithms and coupling measures to devise optimal integration test orders, Carleton University, Technical Report SCE-02-03, October 2002.
[5] L.C. Briand, J. Feng, Y. Labiche, Using genetic algorithms and coupling measures to devise optimal integration test orders, in: 14th International Conference on Software Engineering and Knowledge Engineering, July 2002, pp. 43–50.
[6] L.C. Briand, Y. Labiche, Y. Wang, An investigation of graph-based class integration test order strategies, IEEE Trans. Software Eng. 29 (7) (2003) 594–607.
[7] R.V. Cabral, A. Pozo, S.R. Vergilio, A pareto ant colony algorithm applied to the class integration and test order problem, in: 22nd IFIP International Conference on Testing Software and Systems (ICTSS'10), Springer, 2010, pp. 16–29.
[8] M. Ceccato, P. Tonella, F. Ricca, Is AOP code easier or harder to test than OOP code, in: First Workshop on Testing Aspect-Oriented Program (WTAOP), 2005, pp. 16–29.
[9] J. Francisco Chicano, Francisco Luna, Antonio J. Nebro, Enrique Alba, Using multi-objective metaheuristics to solve the software project scheduling problem, in: 13th Genetic and Evolutionary Computation Conference (GECCO'2011), 2011, pp. 1915–1922.
[10] J.L. Cochrane, M. Zeleny, Multiple Criteria Decision Making, University of South Carolina Press, Columbia, 1973.
[11] C.A. Coello, G.B. Lamont, D.A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation), 2006.
[12] T.E. Colanzi, S.R. Vergilio, W.K.G. Assunção, A. Pozo, Search based software engineering: review and analysis of the field in Brazil, Journal of Systems and Software (2012), http://dx.doi.org/10.1016/j.jss.2012.07.041.
[13] T.E. Colanzi, W.K.G. Assunção, S.R. Vergilio, A. Pozo, Tegration test of classes and aspects with a multi-evolutionary and coupling-based approach, in: Third International Symposium on Search Based Software Engineering (SSBSE), 2011, pp. 188–203.
[14] T.E. Colanzi, W.K. Assunção, S.R. Vergilio, A. Pozo, Generating integration test orders for aspect-oriented software with multi-objective algorithms, in: RITA-Revista de Informática Teórica e Aplicada, 2011 (submitted for publication).
[15] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evolut. Comput. 6 (2) (2002) 182–197.
[16] J.J. Durillo, A.J. Nebro, E. Alba, The JMetal framework for multi-objective optimization: design and architecture, in: IEEE Congress on Evolutionary Computation (CEC), Barcelona, Spain, July 2010, pp. 4138–4325.
[17] S. Gárcia, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization, J. Heuristics 15 (6) (2009) 617–644.
[18] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.
[19] M. Harman, S.A. Mansouri, Y. Zhang, Search based software engineering: a comprehensive analysis and review of trends techniques and applications, Technical Report TR-09-03, April 2009.
[20] N.L. Hashim, H.W. Schmidt, S. Ramakrishnan, Test order for class-based integration testing of Java applications, in: Fifth International Conference on Quality Software, September 2005, pp. 11 – 18.
[21] R. Hewett, P. Kijsanayothin, Automated test order generation for software component integration testing, in: 24th IEEE/ACM International Conference on Automated Software Engineering (ASE'09), November 2009, pp. 211 –220.
[22] J. Jaroenpiboonkit, T. Suwannasart, Class ordering tool – a tool for class ordering in integration testing, in: International Conference on Advanced Computer Theory and Engineering, December 2008, pp. 724 – 728.
[23] J. Knowles, D. Corne, Local search, multiobjective optimization and the Pareto archived evolution strategy, in: Third Australia–Japan Joint Workshop on Intelligent and Evolutionary Systems, 1999, pp. 209–216.
[24] J. Knowles, L. Thiele, E. Zitzler, A tutorial on the performance assessment of stochastic multiobjective optimizers, Technical report, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, February 2006, Revised version.
[25] J.D. Knowles, D.W. Corne, Approximating the nondominated front using the Pareto archived evolution strategy, Evolut. Comput. 8 (2000) 149–172.
[26] N.A. Kraft, E.L. Lloyd, B.A. Malloy, P.J. Clarke, The implementation of an extensible system for comparison and visualization of class ordering methodologies, J. Syst. Software 79 (2006) 1092–1109.
[27] D. Kung, J. Gao, P. Hsia, Y. Toyoshima, C. Chen, A test strategy for object-oriented programs, in: 19th Computer Software and Applications Conference, August 1995, pp. 239–244.
[28] D.C. Kung, J. Gao, P. Hsia, J. Lin, Y. Toyoshima, Class firewal, test order and regression testing of object-oriented programs, J. Object-Orient. Program. 8 (2) (1995).
[29] V. Pareto, Manuel D'Economie Politique, Ams Press, Paris, 1927.
[30] R.C. Purshouse, P.J Fleming, Evolutionary multi-objective optimisation: an exploratory analysis, in: Congress on Evolutionary Computation, CEC 2003, 2003.
[31] R Development Core Team, R: a language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria, 2005, ISBN 3-900051-07-0.
[32] I. Radziukyniene, A. Zilinskas, Evolutionary methods for multi-objective portfolio optimization, in: Proceedings of the World Congress on Engineering 2008, vol. II, 2008.
[33] R. Ré, O.A.L. Lemos, P.C. Masiero, Minimizing stub creation during integration test of aspect-oriented programs, in: 3rd Workshop on Testing Aspect-Oriented Programs, 2007, pp. 1–6.
[34] R. Ré, P.C. Masiero, Tegration testing of aspect-oriented programs: a characterization study to evaluate how to minimize the number of stubs, in: Brazilian Symposium on Software Engineering, October 2007, pp. 411–426.
[35] S. Siegel, J. Castellan, Nonparametric Statistics for The Behavioral Sciences, second ed., Mc Graw Hill Int., 1988.

[36] K.-C. Tai, F.J. Daniels, Test order for inter-class integration testing of object-oriented software, in: 21st Computer Software and Applications Conference, 1997, pp. 602–607.
[37] Y.L. Traon, T. Jéron, J.-M. Jézéquel, P. Morel, Efficient object-oriented integration and regression testing, IEEE Trans. Reliab. (2000) 12–25.
[38] David A. van Veldhuizen, Gary B. Lamont, Multiobjective evolutionary algorithm test suites, in: ACM Symposium on Applied Computing, SAC '99, 1999, pp. 351–357.
[39] S.R. Vergilio, A. Pozo, J.C. Árias, R.V. Cabral, T. Nobre, Multi-objective optimization algorithms applied to the class integration and test order problem, Int. J. Software Tools Technol. Trans. (STTT) (2012), http://dx.doi.org/10.1007/s10009-012-0226-1.
[40] Z. Wang, B. Li, L. Wang, Q. Li, A brief survey on automatic integration test order generation, in: Software Engineering and Knowledge Engineering Conference (SEKE), 2011, pp. 254–257.
[41] Z. Wang, B. Li, L. Wang, Q. Li, An effective approach for automatic generation of class integration test order, in: Computer Software and Applications Conference, July 2011, pp. 680 –681.
[42] S. Yoo, M. Harman, Pareto efficient multi-objective test case selection, in: International Symposium on Software testing and analysis, ISSTA'07, 2007, pp. 140–150.
[43] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: improving the strength pareto evolutionary algorithm, Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.
[44] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, V.G. da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Trans. Evolut. Comput. 7 (2003) 117–132.