Empir Software Eng (2016) 21:2546–2600
DOI 10.1007/s10664-015-9399-z

A comparative study of many-objective evolutionary algorithms for the discovery of software architectures

Aurora Ramírez[1] · José Raúl Romero[1] · Sebastián Ventura[1]

Published online: 28 September 2015

Abstract During the design of complex systems, software architects have to deal with a tangle of abstract artefacts, measures and ideas to discover the most fitting underlying architecture. A common way to structure such complex systems is in terms of their interacting software components, whose composition and connections need to be properly adjusted. Along with the expected functionality, non-functional requirements are key at this stage to guide the many design alternatives to be evaluated by software architects. The appearance of Search Based Software Engineering (SBSE) brings an approach that supports the software engineer along the design process. Evolutionary algorithms can be applied to deal with the abstract and highly combinatorial optimisation problem of architecture discovery from a multiple objective perspective. The definition and resolution of many-objective optimisation problems is currently becoming an emerging challenge in SBSE, where the application of sophisticated techniques within the evolutionary computation field needs to be considered. In this paper, diverse non-functional requirements are selected to guide the evolutionary search, leading to the definition of several optimisation problems with up to 9 metrics concerning the architectural maintainability. An empirical study of the behaviour of 8 multi- and many-objective evolutionary algorithms is presented, where the quality and type of the returned solutions are analysed and discussed from the perspective of both the evolutionary

Communicated by: Marouane Kessentini and Guenther Ruhe

✉ José Raúl Romero
jrromero@uco.es

Aurora Ramírez
aramirez@uco.es

Sebastián Ventura
sventura@uco.es

[1] Department of Computer Science and Numerical Analysis, University of Córdoba, Córdoba, 14071, Spain

performance and those aspects of interest to the expert. Results show how some many-objective evolutionary algorithms provide useful mechanisms to effectively explore design alternatives on highly dimensional objective spaces.

Keywords Software architecture discovery · Search based software engineering · Many-objective evolutionary algorithms · Multi-objective evolutionary algorithms

1 Introduction

Software architects face some of the most difficult decisions in the earliest stages of project execution, since they must guarantee that their designs satisfy multiple quality criteria. At this point, only a few concrete items of information are known about the final system, even though its structure already has to be determined (Bosch and Molin 1999). In this scenario, the discovery of software architectures constitutes an abstract and complex task, where software engineers have to identify high-level architectural artefacts, like components and interfaces, from a prior analysis model. As a result, an architectural specification is obtained comprising the underlying system structure, which will serve throughout the software development to add later functionality, to get suitable designs or to ensure proper maintenance (Ducasse and Pollet 2009).

Along the design phase, software architects have to deal with multiple design alternatives that need to be factually evaluated in order to find those solutions that best meet the system requirements. However, characteristics like modularity and reusability clearly hamper the decision making process because of the lack of quantifiable measures that really would represent the quality of the resulting software designs. Therefore, efforts have been directed towards the definition of specific metrics at the architectural level (Narasimhan and Hendradjaya 2007; Sant'Anna et al. 2007) like those already existing and commonly accepted for more concrete artefacts (Bansiya and Davis 2002). Nevertheless, selecting the most suitable subset of metrics is a complex task usually accomplished by experts only guided by their own experience. Since many different design criteria have to be simultaneously balanced, they may affect the manner in which architectural constraints are considered to meet the software architects expectations.

The automatic evaluation of design alternatives constitutes an interesting application field for Evolutionary Computation (EC) according to the rationale used by Search Based Software Engineering (SBSE) (Harman et al. 2012), where traditional Software Engineering (SE) activities, usually performed by humans, are formulated as search problems to be solved using metaheuristics. In the last years, EC has been successfully applied to the resolution of combinatorial problems in Software Engineering like requirements selection (Durillo et al. 2011) or resource allocation in project scheduling (Luna et al. 2014). In this sense, other SE tasks have demanded the attention of more sophisticated EC approaches in order to provide a higher performance in terms of the quality of the solutions and the satisfaction of the engineer's expectations when more than one or two objectives should be considered (Yao 2013).

In this context, a trade-off between different objectives is required, resulting in a set of equivalent solutions among which the expert will decide. Well-known multi-objective evolutionary algorithms (MOEAs) (Zhou et al. 2011), like SPEA2 (Strength Pareto Evolutionary Algorithm 2) and NSGA-II (Nondominated Sorting Genetic Algorithm II), usually arise as the first option when SE practitioners want to adopt a multi-objective approach

(Sayyad and Ammar 2013). Less often, comparative studies of different MOEAs are presented with the aim of providing a further analysis of the abilities of each proposal. These studies are really useful to gain empirical evidence of the adequacy of a specific algorithm to a given problem domain (Luna et al. 2014; Zhang et al. 2013).

These studies have traditionally considered optimisation problems with 2 or 3 predefined objectives, though there exist many SE tasks whose nature clearly demands the definition of a large number of objectives. Therefore, many-objective evolutionary algorithms appear to be an interesting approach for dealing with this kind of situation (Purshouse and Fleming 2007; von Lücken et al. 2014). The current interest within the EC community in tackling highly dimensional objective spaces creates an opportunity to benefit from novel and powerful algorithms to support software design. Traditionally, many-objective approaches have been tested on benchmarks considering a continuous search space (Bader and Zitzler 2011; Yang et al. 2013; Zhang and Li 2007). For this specific kind of problems, many-objective evolutionary algorithms have shown to clearly outperform MOEAs in terms of the standard performance metrics defined within the field of multi-objective optimisation (Coello Coello et al. 2007; Zitzler et al. 2004). Only a few authors have more deeply studied the performance of many-objective approaches in real-world environments like those appearing in SE and, consequently, SBSE practitioners seem to be less familiar with this kind of algorithms. In this sense, the works presented in Kalboussi et al. (2013), Mkaouer et al. (2014), and Sayyad et al. (2013) are the first proposals that are actually focused on the application of a specific many-objective evolutionary algorithm in testing, refactoring and software product lines design activities, respectively.

The greater the number of factors required to reach architectural decisions is, the more dependent the performance, in terms of quality and feasibility, of the obtained architectural solution becomes on the selection of the most appropriate optimisation approach. Therefore, even though many-objective approaches have proven capable of providing improvements in other related fields of application, the intrinsic characteristics of each algorithm make it necessary to thoroughly study them with the aim of identifying those aspects having a greater influence on the satisfactory resolution of a given design problem. In this context, an empirical analysis allows offering a more in-depth comprehensive overview of the behaviour of different types of algorithms when dealing with a different number and sort of objectives.

In Ramírez et al. (2014), we presented an initial discussion on the performance of 5 multi- and many-objective evolutionary algorithms. This research, performed on a set of 6 different design metrics using 6 representative case studies, highlighted the suitability of this new branch of algorithms in terms of their evolutionary performance and ability to support the decision making process as the number of objectives increased. Nevertheless, we consider it necessary to broaden and deepen in different ways this previous analysis in order to get more valuable findings on the performance of different families of many-objectives algorithms applied to the automatic discovery of software architectures. Firstly, the number of problem instances has been increased to 10 real-world system specifications. Secondly, the study focuses on the maintainability of software systems, considering up to 9 metrics to represent the optimisation objectives. These design metrics quantify diverse related non-functional properties of an architectural specification. The experimental framework, which includes a detailed discussion on the existing dependencies between objectives, provides a complete study of the most suitable characteristics of the different families of algorithms to deal with the evolutionary discovery of software architectures. Amongst these families, 8 different algorithms have been chosen to carry out this empirical research. Due to the very wide range of algorithms within the multi-objective optimisation field, such an analysis has served to articulate the distinctive features of the families of algorithms that best fit into the problem domain under study.

The rest of the paper is structured as follows. Section 2 presents some background, focused on multi- and many-objective optimisation and SBSE. Section 3 introduces the conceptual framework related to software architecture design and the search problem being addressed. The evolutionary model is explained in Section 4, including the definition of the evaluation objectives. The experimental framework is described in Section 5. Section 6 presents the obtained outcomes, whereas a discussion from the perspective of the decision making process is given in Section 7. The threats to the validity are explained in Section 8. Finally, some concluding remarks are pointed out in Section 9.

2 Background

The key concepts of multi- and many-objective optimisation are introduced in this section, as well as the evolutionary algorithms used for this comparative study. Additionally, the prior application of these multiple objective approaches in the SBSE field is reviewed.

2.1 Multi- and Many-objective Evolutionary Optimisation

Solving a multi-objective optimisation problem (MOP) consists in looking for solutions characterised by a set of decision variables that can find the most optimal trade-off among a set of objective functions, which are usually in conflict with each other (Coello Coello et al. 2007). One initial approximation to address a MOP is to turn it into a single-objective problem by formulating a new objective function that calculates the weighted sum of all the objectives. Similarly, other approaches have proposed its resolution by keeping only one objective and considering the rest of them as constraints. Even though they are just simple proposals, they have handicaps like the weight selection or requiring the reformulation of the problem (Deb 2001).

Therefore, any search algorithm specifically devoted to address MOPs will deal with objectives independently, so it can return a set of optimal solutions. Each solution is determined by a different trade-off between all the objectives, in which the improvement achieved by one could imply the loss of others. This idea is formally stated by the Pareto dominance definition (Coello Coello et al. 2007) as follows: a solution a is said to dominate another solution b if a has equal or better objective values for all the objectives and a better value for at least one objective than b. If that condition cannot be satisfied by either a or b, then both solutions are referred as non-dominated, meaning that they are incomparable or equivalent. The set of non-dominated solutions constitutes the Pareto set (PS), whereas the mapping of these solutions to the objective space is called Pareto front (PF). Finding a fitting approximation to the PF is a twofold task (Deb 2001). On the one hand, a multi-objective approach is requested to find non-dominated solutions that are close to the front, which is known as the convergence to the true PF. On the other hand, the resulting PF should be properly distributed within the objective space, making a good spread of non-dominant solutions important. Having both ideas in mind, the search algorithm should be able to provide an interesting set of high-quality solutions for the decision maker to choose among.

Evolutionary algorithms (EAs) are a metaheuristic technique inspired on the principles of natural evolution that deals with a set of solutions to solve optimisation problems (Boussaïd et al. 2013). Each solution is characterised by its genotype, i.e. the structure used to encode the decision variables that need to be managed during the problem resolution.

In contrast, the phenotype symbolises the real-world representation of its genotype. The algorithm begins with the random creation of a set of candidate solutions, known as population. Each solution, also called individual, is then evaluated considering how well it can solve the optimisation problem according to its fitness function. The rest of the evolution is performed by an iterative process in such a way that individuals in the population are modified in each iteration, also called generation, using genetic operators until the stopping condition, e.g. a maximum number of generations, is reached. With this purpose, some individuals are selected to act as parents, from which new solutions are generated according to the specific EC paradigm. The most commonly used genetic operators are crossover and mutation. The former is applied to create offspring tending to be similar to their parents, whereas the latter is performed to introduce diversity among offspring. New solutions have to be evaluated and compared against one another in order to decide whether they should be part of the next population of individuals. Finally, the best individual found in terms of its fitness function is returned. Notice that several paradigms have emerged from the general concept of Evolutionary Computation, such as genetic algorithms (GAs), evolution strategy (ES), evolutionary programming (EP) and genetic programming (GP) (Boussaïd et al. 2013). On the one hand, GA and GP apply both crossover and mutation. In the latter, individuals represent executable programs encoded using a tree structure instead of a fixed-length genotype as proposed by GAs. On the other hand, ES and EP do not usually implement any crossover operator, EP being characterised by the use of domain-specific encodings.

Algorithm 1 Pseudocode of a multi-objective evolutionary algorithm

```
Require: $maxGenerations$, $populationSize$
Ensure: $paretoSet$
 1: $population \leftarrow$ initialisePopulation($populationSize$)
 2: evaluate($population$)
 3: $archive \leftarrow$ initialiseArchive($population$)
 4: $generation \leftarrow 0$
 5: while $generation <= maxGenerations$ do
 6:     $parents \leftarrow$ matingSelection($population \cup archive$)
 7:     $offspring \leftarrow$ variation($parents$)
 8:     evaluate($offspring$)
 9:     $population \leftarrow$ environmentalSelection($population \cup offspring \cup archive$)
10:     $archive \leftarrow$ updateArchive($offspring \cup archive$)
11:     $generation++$
12: end while
13: $paretoSet \leftarrow$ extractNonDominatedSolutions($population$, $archive$)
14: return $paretoSet$
```

Evolutionary computation has traditionally been a way to efficiently solve complex search problems having a unique fitness function. Nevertheless, they can be also adapted to cover a multidimensional objective space. In fact, the application of EAs for the resolution of multi-objective problems, i.e. multi-objective evolutionary algorithms (Coello Coello et al. 2007), has been widely studied for the resolution of real-world applications. They work well in objective spaces with 2 or 3 objectives, in both combinatorial and continuous optimisation problems. MOEAs perform in general the same iterative process than EAs. However, a specific terminology is often used Zitzler et al. (2004), and they require some additional steps to be executed, as shown in Algorithm 1. Observe that MOEAs provide a mating selection method to pick individuals to act as parents. After the generation of offspring through a domain-specific variation mechanism, i.e. genetic operators, environmental selection is performed to select the individuals that will survive in the next generation. The former can determine an overall fitness value, which is independent from the objective values and represents a quality criterion to classify individuals, usually promoting the selection of non-dominated solutions. The latter applies some diversity preservation techniques in order to choose between equivalent solutions. In addition, an external population, called archive, can be maintained throughout the evolution containing the best found solutions. Finally, the Pareto set is returned by extracting non-dominated solutionsl from either the population or the archive. Two well-known examples of MOEAs adopting this latter structure are SPEA2 (Zitzler et al. 2001) and NSGA-II (Deb et al. 2002).

In the last decades, important efforts have been devoted to deal with the increasing complexity of MOPs, especially with those having a large number of objectives, known as many-objective optimisation problems. Although the actual difference between multi- and many-objective problems has not been clearly stated in the literature (Purshouse and Fleming 2007), some works initially considered that having more than 2 objectives can be considered as a many-objective problem (Adra and Fleming 2011). Nevertheless, most authors agree today with the idea that many-objective problems require the presence of at least 4 objectives (Deb and Jain 2014; von Lücken et al. 2014; Zhou et al. 2011). In this paper, this latter criterion is followed in order to distinguish among optimisation problems. Even so, notice that algorithms originally conceived as multi-objective have served to address many-objective problems, whereas other authors use the term multi-objective to refer to the algorithm, and the term many-objective to the optimisation problem. Here, algorithms will be categorised as prescribed by their respective authors. There are studies which conclude that proposals like SPEA2 and NSGA-II tend to decrease their performance as the number of objectives increases, at least when solving continuous optimisation benchmarks (Khare et al. 2003; Praditwong and Yao 2007). In highly dimensional objective spaces, the Pareto dominance loses the efficiency required to guide the search, since a great number of population members become non-dominated solutions. For this reason, many-objective optimisation problems require new techniques to deal with such complexity (Schutze et al. 2011; von Lücken et al. 2014).

Recently, different strategies have been applied in order to improve the performance of EAs for solving many-objective problems, including the adaptation of some already existing algorithms (Adra and Fleming 2011; Wang et al. 2014) or the design of new frameworks (Hadka and Reed 2013). Some of the proposed features are related to the modification of the dominance principle (He et al. 2014) or the inclusion of specific diversity preservation techniques (Wang et al. 2014), among others. Thus, they are usually classified in families in accordance with the selected technique (Wagner et al. 2007).

2.2 Selected Algorithms

Eight evolutionary algorithms have been selected for this comparative study. Two multi-objective evolutionary algorithms, SPEA2 and NSGA-II, are included as baseline algorithms, their original implementation being maintained. Additionally, other representative proposals, especially well-suited to deal with many-objective problems, were chosen from the different families of algorithms in accordance to the literature (von Lücken et al. 2014; Wagner et al. 2007). Notice that for this selection, algorithms could not consider any previous assumption about domain-specific elements like the encoding or the genetic operators, since both characteristics need to be configured according to the problem domain under study.

SPEA2 (Zitzler et al. 2001) assigns a fitness value based on a strength rate and a density estimator strategy. The former counts the number of dominated and non-dominated solutions considering each individual, whereas the latter uses a clustering approach to select more diverse individuals. Both values are combined and used in the mating selection. Non-dominated solutions are stored in a fixed-size archive using a truncation method if it is required, while the new population will be set using all the offspring.

NSGA-II (Deb et al. 2002) involves a sorting method in order to rank the solutions in fronts considering the dominance between them. Both this ranking and a crowding distance are used in the mating selection. During the environmental selection, individuals are progressively kept by fronts, the crowding distance being used to decide which individuals are selected when a front cannot be stored entirely. This algorithm does not define an external archive, so the non-dominated solutions are extracted from the final population.

MOEA/D (Multiobjective Evolutionary Algorithm based on Decomposition) (Zhang and Li 2007) proposes a decomposition approach where the multi-objective problem is divided into several scalar problems to be simultaneously optimised, looking for a better diversity among solutions. More specifically, MOEA/D associates each subproblem to one individual in the population by means of a weighted vector. Individuals are randomly chosen in the mating selection, whereas the neighbourhood information is determined on the weighted vectors during the environmental selection in order to match each offspring to the subproblem that it can best solve. The Tchebycheff approach is applied to evaluate individuals over their specific problem, computing its distance to a reference point. This algorithm uses an archive to keep all the non-dominated solutions found throughout the evolution process. Originally proposed to solve MOPs, MOEA/D is frequently used in many-objective problems, as well as to try out new proposals (Deb and Jain 2014; von Lücken et al. 2014; Yang et al. 2013).

A common technique to avoid the poor selection pressure of the Pareto dominance criterion is to divide the objective space into hypercubes or grids. Then, a relaxed dominance relation, usually referred as the ϵ-dominance, is defined over the resulting landscape. Thus, solution a ϵ-dominates solution b if a belongs to better or equal hypercubes for all the objectives and to a better hypercube for at least one objective than b. This idea is explored in ϵ-MOEA (Deb et al. 2003), a steady-state algorithm which establishes a fixed length of the hypercubes for every objective, named ϵ_i. In each iteration, one parent from the current population and another one from an archive of solutions are selected to generate one or more offspring. These new solutions will survive depending on the hypercubes to which they belong and those already filled with the solutions previously saved in the external population. Similarly to MOEA/D, ϵ-MOEA was originally conceived to deal with 2 or 3 objectives, although it has turned into a reference algorithm due to its notable performance to address many-objective optimisation problems (Wagner et al. 2007; Yang et al. 2013).

GrEA (Grid-based Evolutionary Algorithm) (Yang et al. 2013) is a many-objective evolutionary algorithm based on the notion of landscape partition, though grids are here dynamically created considering the objective values in the current population. Initially, GrEA calculates some properties of the individuals based on the grids. Such information is used by the mating selection in order to obtain the most promising reproducible solutions. Inspired by NSGA-II, GrEA also ranks the population by fronts during the environmental selection, but it uses its own specific distance and diversity metrics to discard equivalent solutions. GrEA does not define an external archive, the returned solutions being extracted from the final population.

Indicator-based methods propose the use of indicators to guide the evolution process. The selected indicator can represent the preferences of the decision maker, so the evolution is heavily oriented towards a region of interest. Some popular indicator of the quality of a Pareto set approximation, e.g. the hypervolume, can be considered. This kind of algorithms has become an interesting alternative to address many-objective optimisation, since they transform the original problem into a task consisting in optimising the quality indicator (von Lücken et al. 2014). One representative approach here is IBEA (Indicator-based Evolutionary Algorithm) (Zitzler and Künzli 2004), which defines a general evolutionary algorithm where any binary indicator could be used as a fitness function. The selected measure is a key factor, since the fitness function is heavily involved in both the mating and the environmental selection procedures. For instance, the authors proposed the application of the ϵ-indicator, representing the minimum distance by which a set of solutions should be translated in each dimension such that another set of solutions is weakly-dominated. It requires a less complex operation than hypervolume, which could be prohibitive in terms of computational effort when a large number of objectives is set. During the mating selection, binary tournaments are performed to generate the set of parents. For each tournament, two randomly selected individuals are compared with each other according to their fitness values, so the best individual is returned as parent. The worst individuals are discarded during the environmental selection.

Another relevant proposal is HypE (Hypervolume Estimation Algorithm) (Bader and Zitzler 2011), which allows the optimisation of many-objective problems replacing the exact value of the hypervolume with an estimated value based on Monte Carlo simulations. This value represents the portion of the hypervolume to be actually attributed to each solution and used to determine the winner of each binary tournament in the mating selection process. Environmental selection is based on the minimum loss of hypervolume, discarding those solutions that contribute the least to the overall hypervolume as a way to promote convergence to the most promising solutions in terms of this indicator. Neither IBEA nor HyPE makes use of an archive of non-dominated solutions.

Finally, NSGA-III (Deb and Jain 2014) is a many-objective evolutionary algorithm which looks for the improvement of the performance of its predecessor, NSGA-II, when dealing with highly dimensional objective spaces. It is classified as a reference-point-based method, being characterised by the promotion of those individuals that are close to a set of reference points supplied by the user or uniformly generated by the algorithm. In NSGA-III, the crowding distance of NSGA-II is replaced by a new method, where each individual is associated to the closest reference point. In this way, diversity is preserved attending to the survival of, at least, one representative solution for each reference point. Apart from belonging to a different family of many-objective approaches, NSGA-III has shown promising results when coping with complex real-world combinatorial SBSE problems like software refactoring (Mkaouer et al. 2014).

2.3 Multiple Objective Approaches in SBSE

Multi-objective optimisation algorithms have been widely applied in SBSE (Yao 2013), since they have demonstrated their ability to reach a trade-off between objectives, as is often the case in Software Engineering. Focusing on search-based software design (Räihä 2010), i.e. the field within SBSE that proposes the application of metaheuristics to the design and improvement of different software artefacts, some classical MOEAs have already been used. Object-oriented analysis using SPEA2 (Bowman et al. 2010), software module clustering executing the Two-Archive algorithm (Praditwong et al. 2011), or the application of NSGA-II to code refactoring (Ouni et al. 2013) represent some examples of software design activities that have been solved from a multi-objective perspective.

Within the field of search-based software design, software architecture optimisation (Aleti et al. 2013) encompasses those works devoted to automatically specifying, re-designing, discovering or deploying software architectures. Recently, these tasks have also being addressed with multi-objective evolutionary approaches. Grunske (2006) presented a first analysis of the applicability of a multi-objective strategy facing several specific non-functional requirements to the software architecture refactoring problem. In this case, no implementation was proposed.

The generation of a set of alternative architectural specifications by means of a specific multi-objective genetic algorithm is proposed in Räihä et al. (2011). Here, a structural model, specified as a class diagram representing a low level architecture, is derived from its corresponding behavioural specification, modelled in terms of sequence diagrams, which are extracted from the software requirement specification and drawn as use cases. Simultaneously, 4 design metrics related to modifiability and efficiency are optimised in the process. NSGA-II is the algorithm chosen to perform the architectural deployment in Koziolek et al. (2013), looking for a trade-off between availability, performance and cost. Here, the optimisation problem consists of the optimal allocation of components into servers, including the server configuration and component selection.

Although NSGA-II stands out as the preferred algorithm in all the SBSE fields, some comparative studies have recently appeared to empirically decide the most appropriate MOEA for a given design problem. With reference to software architecture deployment, Li et al. (2011) compares SPEA2, NSGA-II and SMS-EMOEA (S Metric Selection Evolutionary Multiobjective Optimisation Algorithm) to obtain the best allocation of software components in terms of CPU utilization, cost and latency. Further, there are some recently proposed studies related to other problem domains. For example, in Zhang et al. (2013) the performance of several variants of NSGA-II were analysed for the priorisation of requirements, considering up to 5 objectives. In Assunção et al. (2014), the authors propose a comparison between SPEA2, NSGA-II and PAES (Pareto Archived Evolution Strategy) to deal with 2 and 4 objectives in the context of software testing. Only del Sagrado et al. (2015), Durillo et al. (2011), and Luna et al. (2014) seem to consider a greater variety of metaheuristics, even though they are actually applied to bi-objective problems, like the selection of requirements or the project scheduling problem.

Classical MOEAs still are the most used algorithms in SBSE for multi-objective problems, though some of these problems really seem to require more than 2 or 3 objectives to be optimised (Sayyad and Ammar 2013). Many-objective approaches have received less attention and their applicability has been only explored in a few proposals, mainly focusing on one specific algorithm for a very certain task. In Sayyad et al. (2013) IBEA was chosen to optimise 5 design characteristics of product lines. Kalboussi et al. (2013) presented a preference-based evolutionary method to deal with 7 objectives of a testing problem, whereas Mkaouer et al. (2014) has recently proposed an interesting solution to the software refactoring, using NSGA-III to simultaneously optimise 15 objectives.

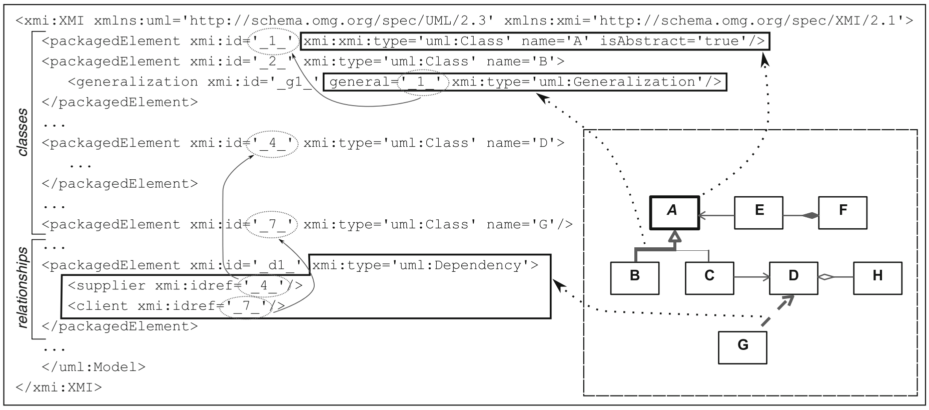
3 Software Architecture Discovery

3.1 Conceptual Framework

The ISO Std. 42010 (ISO 2011b) defines the architecture of a software system as “the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution”. A software architect is in charge of conceiving a system that meets the expected functional requirements, as well as the non-functional requirements like performance or maintainability, at a high level of abstraction. A commonly used approach for developing the software architecture specification is to follow a component-based design, which advocates for the construction of independent, interrelated software artefacts, mostly conceived in favour of reuse. In a component-based architecture, a *component* represents a block of highly cohesive functionality, whose interactions are specified by means of *provided* and *required interfaces* (Szyperski 2002). *Connectors* are links between provided and required interfaces, so they tie a component that supplies some services to another component demanding them.

The tasks related to the architectural design deal with aspects like the understanding, reuse, construction, evolution, analysis and management of the software system (Garlan 2000). From the beginning, an architectural specification is not only required to provide an abstract description of the system, but also to check its consistency and conformance to quality attributes. Nevertheless, the techniques currently used to evaluate software architectures are mostly based on the expert's opinion (Dobrica and Niemela 2002). Actually, the definition of design metrics at the architectural level is still a great challenge, where the ISO Std. 25000 (ISO 2011a) can play an important role as it can serve as a framework to provide a precise guidance on how an architectural model should be assessed.

Ideally, the architecture specification of the system should be maintained throughout the entire development process, evolving as the software does. Frequently, uncontrolled changes in the requirements or the inclusion of a new functionality are directly reflected in the most concrete artefacts, like source code, leading to out-of-date, incoherent, and useless software documentation. As a result, understanding the original architecture becomes a complicated task (Ducasse and Pollet 2009). Although diverse methods have been proposed to recover the architectural specification, most of them start with the source code, meaning that the recovery process can only be conducted once the system implementation is completed. If code is not available, the discovery of the original architecture could only be done manually in accordance to the analysis models. As this process strongly relies on the expertise of software engineers, providing semi-automatic approaches could serve to assist them, especially in exploring different design alternatives in terms of the expected quality attributes.



```
<xmi:XMI xmlns:uml='http://schema.omg.org/spec/UML/2.3' xmlns:xmi='http://schema.omg.org/spec/XMI/2.1'>
   <packagedElement xmi:id='_1_' xmi:xmi:type='uml:Class' name='A' isAbstract='true'/>
   <packagedElement xmi:id='_2_' xmi:type='uml:Class' name='B'>
      <generalization xmi:id='_g1_' general='_1_' xmi:type='uml:Generalization'/>
   </packagedElement>
   ...
   <packagedElement xmi:id='_4_' xmi:type='uml:Class' name='D'>
      ...
   </packagedElement>
   ...
   <packagedElement xmi:id='_7_' xmi:type='uml:Class' name='G'/>
   ...
   <packagedElement xmi:id='_d1_' xmi:type='uml:Dependency'>
      <supplier xmi:idref='_4_'/>
      <client xmi:idref='_7_'/>
   </packagedElement>
   ...
   </uml:Model>
</xmi:XMI>
```

Fig. 1 General outline of the XMI file

3.2 The Search Problem

In Ramírez et al. (2015), the discovery of software architectures is presented as a search problem, where an analysis model represented in form of a class diagram (OMG 2010) saved in XMI format provides the input information in terms of classes and their relationships (see Fig. 1). In this case, the search problem is solved by a single-objective evolutionary algorithm, which is able to generate one single candidate solution representing a high-level architectural specification in terms of components, interfaces and connectors, and depicted by a component diagram. The discovery process is guided by the following rules:

– A component is represented by a cohesive group of classes. The search algorithm should be able to discover the groups of classes that best integrate the different functionalities of the interacting system. Relationships between components are mapped into interacting interfaces. Two constraints should be satisfied in this regard: *any class should belong to only one component*, and *the resulting architectural specification should not contain any empty element*.
– Once the classes are divided up among components, directed relationships between pairs of classes, each belonging to a different component, would represent a candidate interface. The types of the relationships are those defined by UML 2: associations, dependencies, aggregations, compositions and generalisations. Here, two exceptional cases could cause the interface not to be created: either the navigability of the relationship is unspecified, or the relationship represents a data abstraction modelled by a generalisation. Two additional constraints should be also considered: *any component should define at least one interface*, and *mutually dependent components, i.e. a pair of components each one providing services to the other, should not be created.*
– A connector is a link between a pair of required-provided interfaces belonging to different components.

4 Optimisation Approach for the Discovery Process

According to Aleti et al. (2013), the proposed approach can be classified as an architecture optimisation method, where a search-based algorithm is applied on a number of architectural artefacts in order to get the best trade-off between certain quality attributes. More specifically, our evolutionary model falls into the category of multi-objective approaches (*dimensionality* category), making use of a high-level technique, i.e. metaheuristics, to find approximate solutions (*optimisation strategy*). Similarly to other methods, constraint handling techniques are applied, and UML 2 is used as a notation for the architectural specification. What distinguishes our approach from other existing architecture optimisation methods are those aspects related to the problem domain, i.e. quality attributes and decision variables (*degrees of freedom*). The discovery of software architectures was recently proposed as a search problem in Ramírez et al. (2015), being addressed from a single-objective perspective.

The architectural specification is not constructed from scratch during the search-based discovery process. Instead of being derived from requirements, the identification of the system functional blocks and their abstract specification are carried out from an earlier analysis model. A similar approach is the decomposition process proposed by Lutz (2001), where an evolutionary algorithm is used to find the best distribution of classes into modules. This proposal is founded on clustering, so the system structure is viewed as a graph and, besides, valuable analysis information, such as the presence of abstract artefacts or the existence of different types of relationships among the classes, is omitted during the search. Nonetheless, the proposed procedure is not intended to precisely simulate what the engineer would do to build the architecture, e.g. defining interacting interfaces. Similarly, software modularisation (Praditwong et al. 2011) deals with the organisation of code into packages and modules, working at a lower level of abstraction. Although the three optimisation approaches share the idea of searching the optimal organisation of some kind of software artefacts in other units of construction, each problem defines its own objectives and constraints, leading to several differences among them regarding their respective fitness landscapes.

The elements that constitute the evolutionary algorithm addressing the discovery process are introduced next. Firstly, the encoding and the initialisation are explained. Secondly, the objective functions and the genetic operator, composed by five different mutation procedures, are presented. Crossover is not applied because the evolutionary model presented here adopts the evolutionary programming paradigm.

4.1 Encoding and Initialisation

The individuals of the population represent design solutions in the form of a component-based software architecture. A simple genotype/phenotype mapping constitutes an essential element in favour of interpretability. Additionally, since no prior assumption of the architectural structure to be returned is made, a flexible encoding is also required in order to be able to manage architectural solutions having different number of components. There-

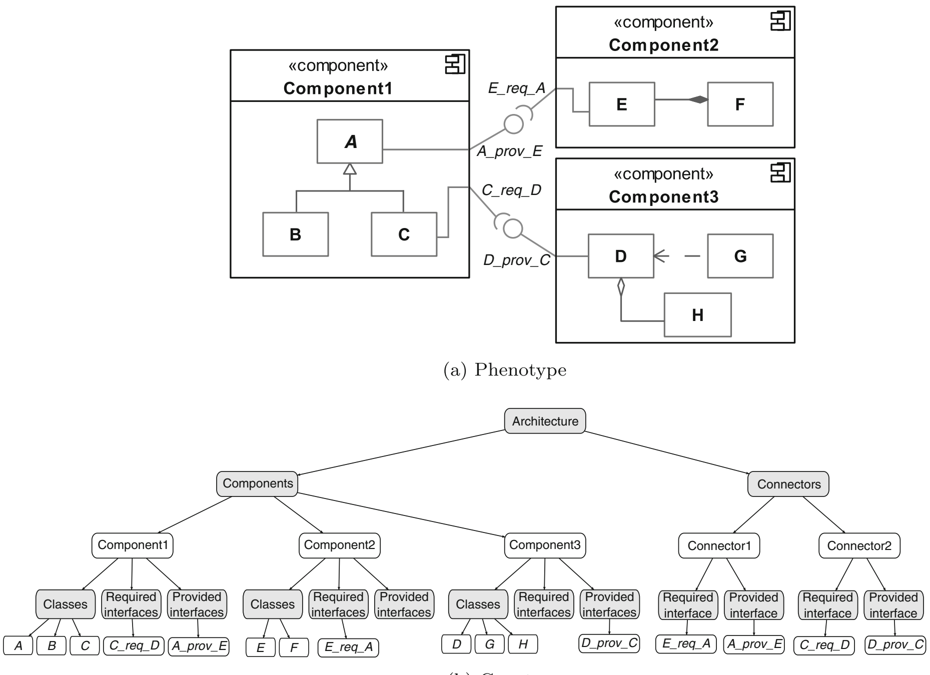


Fig. 2 The component diagram in (**a**) is mapped into the tree structure depicted in (**b**)

fore, a tree structure is used to encode each architectural model, allowing a comprehensible decomposition of the involved software artefacts as established in the problem statement. As depicted in Fig. 2, shaded nodes stand for mandatory elements, whereas the rest of nodes vary from one individual to another, i.e. components and connectors and their internal composition, vary from one individual to another. Additional data about the input class diagram, such as the original relationships between classes and its abstractness are kept by the algorithm. Since this information does not vary from one individual to another, it is not explicitly represented in the genotype.

The initialisation process randomly generates component-based architectures for a given input analysis model. For each solution, a random number of components is chosen between a minimum and a maximum value that is configurable by the software architect. All the classes within the input class diagram are arbitrarily distributed among components. As a constraint none of these components can be empty. Interfaces and connectors are also set considering the existing relationships between classes belonging to different components. The obtained individuals do not need to strictly satisfy all the constraints related to the interaction between components, since they are not checked at this point in order to promote a quicker and more flexible initialisation step. Unfeasible individuals should be turned into feasible ones or be discarded as the evolution elapses, right after the application of the mutation operator and the selection mechanisms.

4.2 Evaluation Objectives

Nine software quality metrics are mapped into their respective objective functions, which serve to guide the evolutionary process. These metrics quantify diverse non-functional requirements related to the maintainability of the software system, which stands out as one of most important quality criteria for component-based software architectures. The software product quality model specified by the ISO Std. 25000 (ISO 2011a) defines *maintainability* as “the degree to which the software product can be modified”, where modifications “may include corrections, improvements or adaptation of the software to changes in the environment, and in requirements and functional specifications”. This standard document defines the sub-characteristics into which the maintainability can be decomposed to effectively measure a number of related design aspects. The specific maintainability terms applicable to the nature of the addressed problem are the following:

- *Modularity*, which is defined as “the degree to which a system is composed of discrete components such that a change to one component has minimal impact on other components”.
- *Reusability*, which establishes “the degree to which an asset can be used in more than one software system or in building other assets”.
- *Analysability*, which determines “the degree to which the parts of the software to be modified can be identified”.

All these terms need to be translated into quantifiable metrics that will then be selectively combined to be simultaneously optimised by the evolutionary algorithm. On the basis of the ISO Std. 25000, a review of the literature was conducted to select those quality metrics that best suited an architectural specification and could be directly computed over the architectural solutions. These metrics are explained next.

Intra-modular Coupling Density (icd) This metric was proposed by Gupta et al. (2012) and establishes a trade-off between cohesion and coupling, as shown in (1). icd is defined as the ratio between internal and external relations in the components. ci_i^{in} represents the number of interactions between classes inside the component i, whereas ci_i^{out} is the number of external relations, i.e. the number of candidate interfaces. Here, the ratio between ci_i^{in} and ci_i^{out} is weighted by the fraction of classes taking part in these relationships. Finally, icd is calculated for the overall architecture as the average of all the icd_i values, all these terms varying in [0,1].

$$icd_i = \frac{\#classes_{total} - \#classes_i}{\#classes_{total}} \cdot \frac{ci_i^{in}}{ci_i^{in} + ci_i^{out}} \quad icd = \frac{1}{n} \cdot \sum_{i=1}^{n} icd_i \tag{1}$$

External Relations Penalty (*erp*) This metric is inspired by the *Interface Violations* metric (Krogmann 2010). erp computes the number of existing relationships between classes belonging to each possible pair of different components, i and j, that could not be defined as candidates interfaces (see (2)). More precisely, these relationships are those associations (as), aggregations (ag) and compositions (co) that do not explicitly specify their navigability, as well as generalisations (ge). A weighted sum (w_x) is calculated to emphasise those relations that strongly harm the overall quality of the architectural solution at the software engineer's discretion.

$$erp = \sum_{i=1}^{n} \sum_{j=i+1}^{n} (w_{as} \cdot n_{as_{ij}} + w_{ag} \cdot n_{ag_{ij}} + w_{co} \cdot n_{co_{ij}} + w_{ge} \cdot n_{ge_{ij}}) \tag{2}$$

Instability (*ins*) The instability metric is intended to create highly independent components, leading to more separated functionality blocks. Although the concept of instability was originally defined by Martin (1994) to evaluate the stability of object-oriented designs, it can be also considered at the architectural level. In this way, the overall instability of the architecture is obtained as the average value of the instability of each single component (ins_i) (see (3)). In order to calculate this metric, the terms ac_i and ec_i need to be first computed. ac_i represents the afferent coupling of component i, i.e. the number of components that require its services, whereas ec_i stands for the efferent coupling, i.e. the number of external components that provide their services to component i (Sant'Anna et al. 2007).

$$ins_i = \frac{ec_i}{ec_i + ac_i} \quad ins = \frac{1}{n} \cdot \sum_{i=1}^{n} ins_i \tag{3}$$

Encapsulation (*enc*) This metric is inspired by the *Data Access Metric* proposed by Bansiya and Davis (2002), which serves to evaluate the encapsulation of one class within an object-oriented design. With the aim of adapting this concept to the architectural level, the encapsulation of each component i, enc_i, is computed as the ratio of its hidden classes, i.e. those that do not participate in any interaction with classes belonging to other components in the architecture, and the total number of contained classes. The overall encapsulation is calculated as the mean of the enc_i values (see (4)).

$$enc_i = \frac{\#inner_{classes}}{\#total_{classes}} \quad enc = \frac{1}{n} \cdot \sum_{i=1}^{n} enc_i \tag{4}$$

Critical Size (*cs*) This metric, presented in Narasimhan and Hendradjaya (2007), intends to minimise the number of large components. To this end, cs counts the number of critical components with respect to their size (cc_{size}), this value being returned by the $size()$ function and calculated as the percentage of classes comprised by the component in relation to the entire model. A threshold value is required to determine if the corresponding component would be critical or not (see (5)).

$$cc^{i}_{size} = \begin{cases} 1 & \text{if size}(i) > \text{ threshold} \\ 0 & \text{otherwise} \end{cases} \qquad cs = \sum_{i=1}^{n} cc^{i}_{size} \tag{5}$$

Critical Link (*cl*) Similarly to cs, cl considers the number of critical components, cc_{link}, with respect to the number of their interactions within the architecture (Narasimhan and Hendradjaya 2007). As can be seen in (6), the links refer to the number of provided interfaces and a threshold should be established.

$$cc^{i}_{link} = \begin{cases} 1 & \text{if } \#provided\ interfaces_i > \text{ threshold} \\ 0 & \text{otherwise} \end{cases} \qquad cl = \sum_{i=1}^{n} cc^{i}_{link} \tag{6}$$

Groups/Components Ratio (*gcr*) The gcr metric, presented in (7), has been adapted from the *Component Packing Density* (cpd) metric, defined by Narasimhan and Hendradjaya (2007). cpd determines the ratio between the number of constituents, i.e. those elements that compose the components, and the total number of components in the architecture. Here, the constituents are the groups of connected classes ($\#cgroups$) inside each component.

$$gcr = \frac{\#cgroups}{\#components} \tag{7}$$

Abstractness (*abs*) This measure (Krogmann 2010) is defined as the ratio of abstract classes inside a component, abs_i (see (8)). A global value of the abstractness distribution for a given architectural solution, abs, can be obtained by computing the mean value of abstractness for each component.

$$abs_i = \frac{\#abstract\_classes_i}{\#classes_i} \qquad abs = \frac{1}{n} \cdot \sum_{i=1}^{n} abs_i \tag{8}$$

Component Balance (*cb*) Reducing the presence of critical components is an important factor in software design, though it could lead to an excessive number of tiny components hampering the comprehension of the resulting system structure. cb is proposed in Bouwers et al. (2011) to quantify the balance among components, taking into consideration their respective sizes. It is based on two terms, the system breakdown (sb) and the component size uniformity (csu), as shown in (9), where csu is defined as the Gini Coefficient, i.e. a statistical measure of dispersion. The $volume(c)$ function counts the number of classes per component c, C being the set of components in the architecture. csu varies in the range [0,1], the unity being its optimum value, meaning that all the components have the same size. sb represents a linear function in the range [0,1], which benefits the proximity to a number of components, μ, and penalises those architectures having just one or too many components (ω). Since the problem definition determines the minimum and maximum number of components to be comprised by the candidate architectures, ω should be set to this maximum value, and the theoretical minimum threshold, 1, should be replaced

Table 1 Main characteristics of objective functions

Metric	Min/Max	Quality attribute	Range of values	Design goals
icd	max	modularity	$[0, 1]$	Small components with high cohesion
erp	min	modularity	$[0, *]$	Large components with low coupling
ins	min	modularity	$[0, 1]$	Components with few interactions
enc	max	modularity	$[0, 1]$	Components with hidden classes
cs	min	modularity	$[0, n]$	Small or medium-sized components
cl	min	modularity	$[0, n]$	Components with few provided interfaces
gcr	min	reusability	$[1, *]$	Connected classes within each component
abs	max	reusability	$[0, 1]$	Components with abstract classes
cb	max	analisability	$[0, 1]$	Equal-sized components

by the minimum number of components. Similarly, the μ value will be equal to the average value between these maximum and minimum values.

$$\begin{aligned} sb(n) &= \begin{cases} \frac{n-1}{\mu-1} & \text{if } n < \mu \\ 1 - \frac{n-\mu}{\omega-\mu} & \text{if } \mu < n < \omega \\ 0 & \text{if } n \geq \omega \end{cases} \\ csu(C) &= 1 - Gini(\{volume(c) : c \in C\}) \\ cb(S) &= sb(|C|) \cdot csu(C) \end{aligned} \tag{9}$$

Table 1 summarises the main characteristics of the 9 objective functions, including whether they are to be minimised (min) or maximised (max), and a brief description of the design goals being promoted. Each metric is related to maintainability characteristics defined above, and the range of possible values is also provided. The symbol $*$ is used to represent that the upper bound of erp and gcr is dependent on the features of the problem instance, whilst n indicates the number of components within the architecture.

Table 2 shows the list of design goals being promoted ($\uparrow$) or demoted ($\downarrow$) by each metric. The symbol $-$ represents that the corresponding metric has no influence on that goal. As can be seen, most of the trade-offs between metrics are related to the size of components and the way in which components would preferably interact with each other. For example, those metrics that look for decreasing the number of interactions between components, in terms of both interfaces and external relationships, i.e. coupling, tend to demote the internal component cohesion due to an excessive encapsulation of functionalities.

4.3 Mutation Operator

The execution of a crossover operator could produce a significant amount of unfeasible solutions, requiring the implementation of repair methods to satisfy those constraints that had been already checked during the initialisation process. It would be a time-consuming procedure, without which unfeasible individuals could be generated even late in the search, making the convergence to the feasible region of the search space considerably more difficult. Only a domain-specific mutation operator serves to explore the different architectural solutions throughout the evolutionary search (Ramírez et al. 2015). A weighted roulette is used to select one of the five proposed mutation procedures. More specifically, these procedures simulate those architectural transformations that could be applied to a given individual, i.e. the parent, to generate an offspring.

- *Add a component*. A new component is added to the architecture. Creating such a new component requires extracting some classes from other components. More specifically, if a component would contain several unconnected groups of related classes, some of them could be randomly selected to create the new component. Hence, the ultimate aim of this procedure is to improve the distribution of functionalities among components.
- *Remove a component*. This procedure implies that one component in the current architecture will be discarded. In this case, its inner artefacts are arbitrarily distributed among the remaining components. A component having a great number of dependencies with others is a preferable candidate to be removed.
- *Merge two components*. This procedure involves the construction of a new component by taking two existing components from the current architecture. Since the aim of this procedure is to attempt to concentrate functionalities, the first selected component would be one with a large number of external dependencies. Then, a second component is selected at random.
- *Split a component*. This procedure is designed with the aim of avoiding excessively large components. One component is randomly selected and split into two components, forcing an interaction between both of them to remain in the form of a new interface. More precisely, an internal relationship is chosen at random to act as the candidate interface. Next, each inner class is assigned to one component or another depending on how it relates to the interface ending, i.e. classes providing and requiring the service are separated.
- *Move a class*. This procedure executes the simple movement of one class from one component to another. Notice that this procedure does not apply any change in the structure of the solution, so the transformation is done completely at random.

The roulette is dynamically built for each parent, only considering those applicable procedures in each particular case. Thus, if the initial individual is a feasible solution, the mutation procedure will always return a solution without duplicate classes or empty components. The minimum and maximum number of components within the architecture is also preserved. For example, the *split a component* operation cannot be executed on an individual representing an architecture that already contains the maximum number of components. The weights of the roulette can be configured by the software architect. Finally, notice that the existing constraints related to interactions between components have to be considered here. For example, if the obtained individual represents an unfeasible solution, the parent is mutated again until a feasible solution is reached or a maximum number of

Table 2 Design goals and trade-offs between metrics

Design goal	*icd*	*erp*	*ins*	*enc*	*cs*	*cl*	*gcr*	*abs*	*cb*
Small components	↑	↓	↑	↓	–	–	↓	–	↓
Large components	↓	↑	↓	↑	↓	–	↑	–	↓
High cohesion between classes	↑	↓	–	↓	–	–	↓	–	–
Low coupling between components	–	↑	–	–	–	–	–	–	–
Few interfaces per component	↓	–	↑	↑	–	↑	–	–	–
High encapsulation	–	–	↑	↑	–	–	↑	–	–
Distribution of abstract classes	–	–	–	–	–	–	–	↑	–

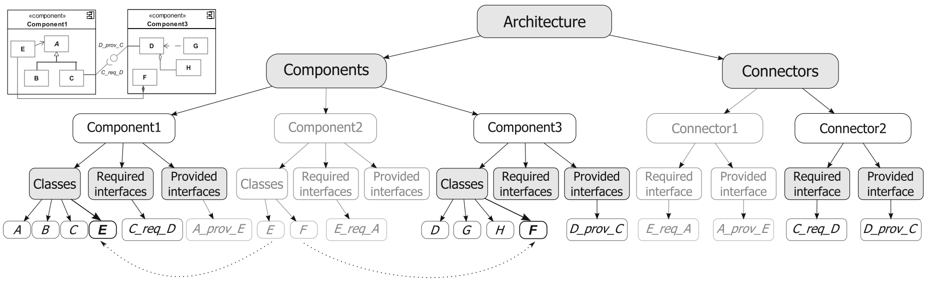


Fig. 3 Example of the mutation operator

attempts is exceeded. In the latter situation, the unfeasible mutant individual could be considered as a valid offspring depending on a probabilistic criterion. In this sense, a dynamic threshold is set at the first generation, and will be decreased along the evolution. If a randomly generated number exceeds the current threshold, the unfeasible individual is added to the offspring pool. On the contrary, the original parent takes his position in the set of descendants.

As a way to illustrate how the mutation operator works, Fig. 3 shows the result of applying the remove component procedure over the solution presented Fig. 2. As can be observed, the second component ($Component2$) within the original individual has been removed, and its classes randomly reallocated. In this example, class E is moved to $Component1$, whilst class F is reallocated to $Component3$. As a result, since classes A and E are now part of the same component, the existing interaction between them becomes internal. No more interfaces are created to link classes E and F, since the navigability of their composition is unspecified.

5 Experimental Framework

All the algorithms and the experimental framework have been coded in Java. Additionally, some public Java libraries have been used to facilitate the implementation. SDMetrics Open Core[1] provides support for parsing XMI files, allowing the extraction of information from the analysis models created with the MagicDraw tool.[2] Preprocessing tasks and internal data structures have been implemented using the Datapro4j library.[3] The evolutionary algorithms have been written using JCLEC (Ventura et al. 2008). Experiments were run on a HPC cluster of 8 compute nodes with Rocks cluster 6.1 x64 Linux distribution, where each node comprises two Intel Xeon E5645 CPUs with 6 cores at 2.4 GHz and 24 GB DDR memory.

In this section, the comparison methodology is presented, including the performance measures used to evaluate the evolutionary algorithms. The parametrisation and set-up, as well as the problem instances used, are also introduced next.

[1] http://www.sdmetrics.com/OpenCore.html

[2] http://www.nomagic.com/

[3] http://www.uco.es/grupos/kdis/datapro4j

5.1 Methodology

To assess the validity of the experiments, every algorithm has been executed 30 times with different random seeds. The 9 proposed design metrics have been combined in groups of 2, 4, 6, 8 and 9 objectives, which make a total of 256 combinations. After executing all these combinations, it is possible to analyse how the selection of a particular subset of metrics, including its cardinality, can affect both the evolutionary performance and the type of obtained architectural solutions.

Regarding the comparison study from an evolutionary perspective, two quality indicators, hypervolume (HV) and spacing (S) have been considered (Coello Coello et al. 2007). HV calculates the hyper-area covered by the Pareto set, whereas S measures the spread of the front. It should be noted that HV requires an objective space in the range [0,1], so the normalisation approach proposed in Luna et al. (2014) is applied after executing the algorithms. More specifically, a reference Pareto front (RPF) is built up considering all the solutions found after executing the 8 algorithms for each problem instance. Then, the objective values are normalised taking as reference the worst value and the best value achieved by any solution within the RPF.

Statistical tests (Arcuri and Briand 2011; Derrac et al. 2011) have served to compare the performance of the algorithms under study. The Friedman test, a non-parametric test that allows comparing multiple algorithms over different problem instances, is executed first. At a specific level of significance, $1-\alpha$, this test establishes the null hypothesis, H_0, denoting that all the algorithms perform equally well. If H_0 is rejected, i.e. significant differences are detected, a post-hoc procedure has to be carried out in order to properly determine those statistical differences between the considered algorithms. Here, the Holm test is performed when H_0 cannot be accepted, a control algorithm being compared to the rest of algorithms. Additionally, the Cliff's Delta test has been executed to assess the magnitude of the improvements using an effect size measurement (Arcuri and Briand 2011). This test performs pairwise comparisons to determine whether such an improvement should be considered negligible, small, medium or large on the basis of specific thresholds (Romano et al. 2006).

Some other measures have been compiled as well in order to precisely state the discussion of results beyond the evolutionary performance. Hence, aspects like the execution time, the size of the Pareto set, or the set of non-dominated solutions are also scrutinised to properly analyse the behaviour of each algorithm concerning the software architect's expectations.

5.2 Parameter Set-up

Before entering into the details of the value assignment to specific parameters of each algorithm, there are some related aspects to be addressed respecting constraint handling and mating selection. On the one hand, notice that constraint handling techniques need to be applied in different ways. A first scenario refers to the way invalid solutions are evaluated. In the case of the algorithms SPEA2, MOEA/D, GrEA, IBEA and HypE, the worst possible fitness value is assigned to every unfeasible individual. A constrained version of NSGA-II was already presented in the original work by Deb et al. (2002), in which feasible solutions were promoted according to a specific comparison method. The same approach has been adopted for ϵ-MOEA, since this algorithm does not declare any fitness function. These two mechanisms guarantee that unfeasible individuals will not prevail over feasible individuals in any selection process.

Table 3 Parameter set-up

Common parameters	
Population size (2, 4, 6, 8, 9 objectives)	100, 120, 126, 330, 495
Max. evaluations (2, 4, 6, 8, 9 objectives)	10,000, 15,000, 20,000, 66,000, 99,000
Min-max. components	2–8
Mutator weights	$w_{add} = 0.2$, $w_{remove} = 0.3$, $w_{merge} = 0.2$, $w_{split} = 0.1$, $w_{move} = 0.2$
erp weights	$w_{as} = 2$, $w_{ag} = 3$, $w_{co} = 3$, $w_{ge} = 5$
cs threshold	0.3
cl threshold	8
SPEA2 parameters	
Parents selector	Binary tournament
External population size	50
k-th neighbour	12
MOEA/D parameters	
Neighbourhood size (τ)	8
Max. replacements (*Nr*)	2
H (2, 4, 6, 8, 9 objectives)	99, 7, 4, 4, 4
ϵ-MOEA parameters	
ϵ values	$\epsilon_{icd} = 0.05$, $\epsilon_{erp} = 5.00$, $\epsilon_{ins} = 0.05$, $\epsilon_{enc} = 0.05$ $\epsilon_{cs} = 1.00$, $\epsilon_{cl} = 1.00$, $\epsilon_{gcr} = 0.10$, $\epsilon_{abs} = 0.05$, $\epsilon_{cb} = 0.05$
GrEA parameters	
Number of divisions (*div*)	12
IBEA parameters	
Scaling factor (*k*)	0.05
HypE parameters	
Number of sampling points (*M*)	10,000

On the other hand, regarding the mating selection phase in the case of MOEA/D, it usually takes two individuals from the neighbourhood of each member of the population to act as parents and generates a unique descendant. However, given that only one mutator operator is executed, just one neighbour is selected to generate the offspring.

The specific configuration parameters of the selected algorithms have been set up according to their respective authors, whilst the domain-specific configuration is identical in all cases. If required, some preliminary experiments have been performed to properly adapt their parameters to the specific needs of the problem domain, e.g. the hypercube lengths in ϵ-MOEA. The final parameter set-up is shown in Table 3. Notice that different values are set for the population size and the maximum number of evaluations to deal with the growing complexity of the optimisation problem when increasing the number of objectives from 2 to 4, 6, 8 and 9. Regarding the population size, it should be noted that MOEA/D requires a uniformly distributed set of weighted vectors, whose size is controlled by the number of objectives and the parameter H. Therefore, the number of weighted vectors obtained for

Table 4 Problem instances and their characteristics

Problem	*#Classes*	*#Relationships*					*#Interfaces*
		As	De	Ag	Co	Ge	
Aqualush	58	69	6	0	0	20	74
Borg	167	44	109	36	38	90	300
Datapro4j	59	3	4	3	2	49	12
ICal4j	190	36	4	3	11	161	70
Java2HTML	53	20	66	15	0	15	170
JSapar	46	7	33	21	9	19	80
JXLS	96	60	10	10	9	45	136
Logisim	253	113	19	46	25	137	248
Marvin	32	5	11	22	5	8	28
NekoHTML	47	6	17	15	18	17	46

each number of combined objectives has determined the population size for the rest of algorithms. For the remaining parameters, they are related to the problem domain, including the minimum and maximum number of components to be considered for the architecture, the parameters required to calculate the objective functions (see Section 4.2) and the weight values related to the mutation procedures (see Section 4.3).

5.3 Problem Instances

The experimental study has been carried out over 10 diverse problem instances, each one representing a system design with different complexity in terms of both the number of classes and the interoperability among them. Table 4 provides the complete list of their respective characteristics. The number and types of relationships are enumerated in the column *#Relationships*, where the different types are broken down in turn as prescribed by UML 2: associations (*As*), dependencies (*De*), aggregations (*Ag*), compositions (*Co*) and generalisations (*Ge*). The last column (*#Interfaces*) indicates the number of candidate interfaces, i.e. the number those relationships whose navigability has been explicitly specified.

Most of these instances have been taken from actual real-world software systems working on diverse application domains. Only *Aqualush*[4] is a benchmark used for educational purposes. Some of the software specifications can be accessed from the Java Open Source Code Project[5], whereas *Datapro4j*, *iCal4j*[6] and *Logisim*[7] are available at their respective websites.

[4]http://www.ifi.uzh.ch/rerg/research/aqualush.html

[5]http://www.java-source.net/

[6]http://www.sourceforge.net/projects/ical4j/

[7]http://www.sourceforge.net/projects/logisim/

6 Analysis of Results

In this section, we analyse the results from the perspective of the evolutionary performance. More specifically, we focus on the scalability of the selected algorithms regarding the two quality indicators mentioned in Section 5.1, hypervolume and spacing.

6.1 Analysis of 2- to 4-objective Problems

Table 5 shows the overall ranking obtained by the Friedman test for each selected algorithm in terms of hypervolume. A value represents the arithmetic mean of the ranking positions determined considering all the possible combinations for the given number of objectives. Notice that the lowest values are the best. Their corresponding standard deviation is shown accordingly. Similarly, Table 6 shows the summary of the results for the spacing indicator. Complementary information about the results is reported in the Appendix.

Regarding the HV for 2-objective problems, NSGA-II, ϵ-MOEA and HypE achieve the best ranking positions, obtaining significant differences with the rest of algorithms. On the contrary, SPEA2 and MOEA/D show a better behaviour in terms of the spacing indicator. IBEA and NSGA-III achieve good spacing values only for certain combinations of objectives, which is reflected in a higher standard deviation. Notice that simultaneously having good scores in terms of both hypervolume and spacing becomes a great challenge. Even though a multi-objective evolutionary algorithm tries to find a compromise between the convergence towards a set of non-dominated solutions and the spread of the resulting front, its specific characteristics could encourage the promotion of one criterion at the expense of the other. Specific many-objective approaches do not exhibit any superiority yet, obtaining the best ranking positions for only a few problems.

As for the combinations of metrics, it can be noted that their respective optimisation problems may imply several differences in the performance of the algorithms. Since the software architect could be interested in different aspects of the same quality criteria, e.g. several design metrics are related to modularity, the selection of the specific metrics is also an important decision with respect to the expected architectural solutions. For those problems where erp, gcr, cs or cl are jointly optimised, all the algorithms have achieved similar values in terms of both quality indicators. For example, the selection of erp and cl ends in a tie between SPEA2, NSGA-II, MOEA/D and ϵ-MOEA. Moreover, there are no statistical differences when comparing these algorithms to GrEA and HypE, because all the algorithms obtain either a unique solution or an equivalent Pareto set. In the former case,

Table 5 Mean and standard deviation of Friedman rankings for hypervolume

Algorithm	2 objectives	4 objectives	6 objectives	8 objectives	9 objectives
SPEA2	4.35 ± 0.78	4.66 ± 0.86	5.64 ± 0.29	6.33 ± 0.28	6.70 ± 0.00
NSGA-II	2.02 ± 0.59	1.39 ± 0.40	1.92 ± 0.54	2.66 ± 0.46	2.90 ± 0.00
MOEA/D	3.90 ± 0.81	4.41 ± 0.68	3.75 ± 0.47	3.48 ± 0.43	3.30 ± 0.00
ϵ-MOEA	3.88 ± 1.12	2.78 ± 1.15	1.51 ± 0.55	1.08 ± 0.09	1.00 ± 0.00
GrEA	4.86 ± 0.79	5.30 ± 0.75	5.80 ± 0.41	5.83 ± 0.11	5.50 ± 0.00
IBEA	7.21 ± 0.31	7.64 ± 0.15	7.76 ± 0.06	7.79 ± 0.05	7.90 ± 0.00
HypE	3.04 ± 0.58	3.20 ± 0.75	3.20 ± 0.55	2.82 ± 0.31	2.80 ± 0.00
NSGA-III	6.75 ± 0.49	6.62 ± 0.21	6.49 ± 0.16	6.01 ± 0.29	5.90 ± 0.00

Table 6 Mean and standard deviation of Friedman rankings for spacing

Algorithm	2 objectives	4 objectives	6 objectives	8 objectives	9 objectives
SPEA2	3.65 ± 1.25	2.09 ± 1.08	1.10 ± 0.17	1.37 ± 0.37	1.70 ± 0.00
NSGA-II	4.00 ± 0.82	4.59 ± 0.91	2.77 ± 0.85	1.97 ± 0.34	1.80 ± 0.00
MOEA/D	3.23 ± 1.10	3.16 ± 0.79	5.21 ± 0.50	5.69 ± 0.23	5.30 ± 0.00
ϵ-MOEA	5.58 ± 1.04	3.02 ± 1.49	3.41 ± 0.69	4.06 ± 0.37	4.60 ± 0.00
GrEA	5.37 ± 0.52	6.78 ± 0.45	7.29 ± 0.17	7.28 ± 0.14	6.60 ± 0.00
IBEA	5.73 ± 1.51	7.43 ± 0.42	7.60 ± 0.04	7.63 ± 0.07	7.80 ± 0.00
HypE	4.09 ± 0.72	4.91 ± 0.95	5.11 ± 0.82	4.88 ± 0.57	5.00 ± 0.00
NSGA-III	4.35 ± 1.45	4.02 ± 1.09	3.53 ± 0.51	3.13 ± 0.16	3.20 ± 0.00

every approach finds an optimal solution, since the objectives are not strongly opposed. In the latter case, all the algorithms have some trouble in avoiding local optima. In constrast, the combination of erp and cl metrics with icd, abs, cb or enc implies the formulation of a more complex optimisation problem, where the dominance between solutions is harder to achieve. Therefore, the resulting Pareto set will be comprised of more diverse solutions, where both objectives are satisfied in different ways.

As the number of objectives increases up to four, differences between them become more evident. It is worth noticing that SPEA2 and NSGA-II are still the best algorithms regarding spacing and hypervolume, respectively, though their rankings for the other indicator have increased. As can be seen in Tables 5 and 6, ϵ-MOEA and NSGA-III have improved their overall rankings for both indicators, whereas the rest of algorithms do not experience great changes or their performance even decreases, as in the case of IBEA and GrEA.

MOEA/D and ϵ-MOEA obtain the best ranking for some specific combinations of objectives. For example, MOEA/D has shown good spacing values when cs, cl or ins are considered. Similarly, ϵ-MOEA seems to optimise certain objectives, e.g. icd, ins, abs or cb, more accurately than NSGA-II in terms of HV. These 4 objectives promote the functional distribution among components. On the contrary, if metrics like erp, gcr, enc and cl are selected to define the search problem, the number of components within the architectural solution tends to be minimised as a way to reduce their interactions. In general, ϵ-MOEA provides a better performance than NSGA-II when the optimisation problem is more complex from a design perspective, as the latter tends to promote the discovery of an excessively monolithic architecture.

Table 7 compiles the outcomes obtained by the different algorithms in terms of their performance when a low number of objecives is considered to address the problem under study.

6.2 Analysis of 6- to 9-objective Problems

When algorithms optimising 6 or more objectives are compared, the Friedman test returns significant differences for a large number of combinations, many of them including at least one objective that promotes a design goal that tends to be demoted by the rest of objectives. Some initial tendencies observed for 4-objective optimisation problems are now clearly brought out. To illustrate this point, Fig. 4 depicts the percentage of combinations for which each algorithm reaches the best ranking for hypervolume and spacing, respectively. As can

Table 7 Summary of the evolutionary performance for 2- and 4-objective problems

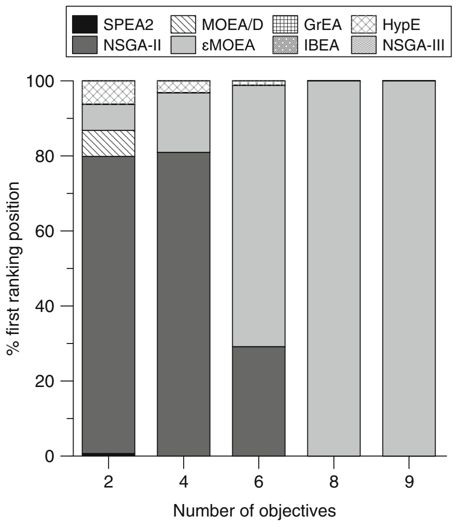
Algorithm	Main findings
SPEA2	• Ability to maintain a wide-spread Pareto front due to the use of density information • Some difficulties to obtain competitive values for the hypervolume
NSGA-II	• Good scalability with respect to hypervolume • Variable behaviour in terms of spacing, especially when increasing to 4 objectives • The nondominated sorting approach has a strong influence on the evolution
MOEA/D	• Spacing values reflect effective exploration of multiple search directions using weight vectors • Low ranking positions for both 2- and 4-objective problems in terms of HV
ϵ-MOEA	• Variable behaviour in terms of HV when optimising combinations of 2 metrics • Better HV than other many-objective approaches like HypE or NSGA-III for 4 objectives • ϵ-dominance contributes to enhance the diversity of the Pareto front
GrEA	• Poor performance regardless of the number and combination of metrics • Similar to NSGA-II regarding S, since the sorting approach strongly promotes convergence
IBEA	• Ability to reach reasonably good spacing values for specific 2-objective combinations • Poor performance when dealing with 4 objectives. • ϵ-indicator is a non-discriminatory criterion to guide the search towards promising solutions
HypE	• Lower HV values than expected regardless of the number of objectives • Good trade-off between HV and S for 2 objectives • Excessive promotion of solutions contributing to HV at certain steps of the evolution • Some regions of interest can remain unexplored by the optimisation process
NSGA-III	• Better performance than NSGA-II only in terms of S due to the use of reference points • Overemphasis on promoting diversity leads to a low performance in HV ranking values

be noted, as the number of objectives increases, ϵ-MOEA emerges as the best algorithm for HV, outperforming NSGA-II, and SPEA2 is the best alternative in terms of spacing, especially when dealing with more than 6 objectives.

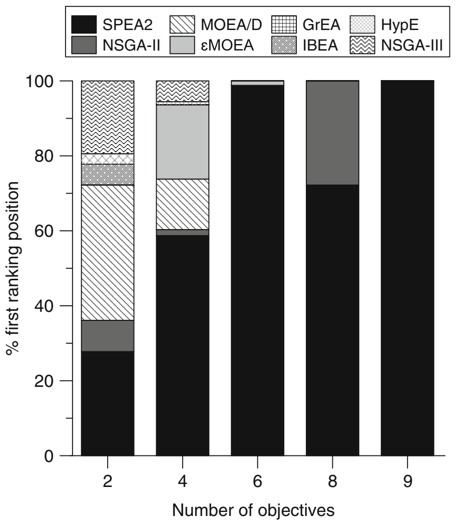
As can be seen from Tables 5 and 6, SPEA2, NSGA-II, GrEA and IBEA report worse HV ranking values than those obtained for 2 and 4 objectives. On the contrary, MOEA/D, ϵ-MOEA, HypE and NSGA-III have experienced an improvement regarding this indicator, though ϵ-MOEA and HypE obtain competitive values when dealing with 8 and 9 objectives.

Additionally, the Cliff's Delta test assesses that the pairwise comparisons between the algorithms are statistically significant in most objective combinations [8]. For the 9-objective problem, a total of 56 test executions per quality indicator were required. In this case, differences in the magnitude of the corresponding indicator have been catalogued as large in 49 and 48 out of the total number of executions made for HV and S, respectively. On the one hand, the effect size obtained when comparing ϵ-MOEA in terms of HV against any other algorithm is always greater than 0.7. On the other hand, regarding SPEA2, the test

[8]Tables 10 (HV) and 11 (S) in Appendix show the results obtained by the Cliffs Delta test for the 9-objective combination. The full results in raw format for all the combinations are available at http://www.uco.es/grupos/kdis/sbse/RRV15



(a) Hypervolume



(b) Spacing

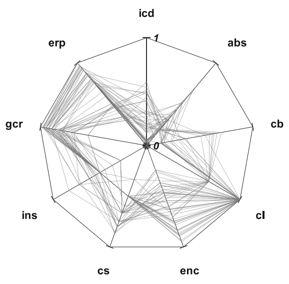
Fig. 4 Percentage of first ranking positions in the Friedman test for hypervolume and spacing

reports an effect size greater than 0.6 for all its comparisons regarding *S*, except when facing NSGA-II.

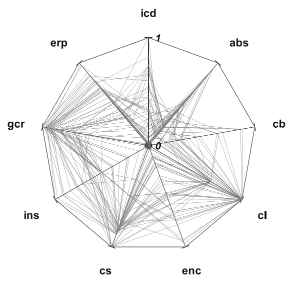
Concerning the performance of all the algorithms with more than 4 objectives, the number of objectives has a deeper impact than the set of selected metrics. As can be observed from the experimental results, the ranking values obtained by each algorithm are very similar regardless of the metric subset being optimised. A progressively decrease of the standard deviation is observed in Tables 4 and 5, whereas the number of different algorithms achieving the top ranking position is lower for 4 or 6 objectives than for 2 (see Fig. 4), even when a larger number of objective combinations was generated. Notice that the specific subset of metrics, as well as the need for a different number of objectives, relies on the software architect's judgement. A detailed analysis on the influence of these aspects is provided in Section 7.2.

Some additional remarks can be pointed out with respect to the joint optimisation of 9 objectives. For instance, Fig. 5 shows an example of the Pareto front obtained by each algorithm for a representative problem instance, *Aqualush*. Notice that the objective space is normalised and all the objectives must be maximised. For the sake of readability, a maximum number of 50 solutions is depicted. The solutions have been randomly selected from the set of all the solutions generated in different executions. Those solutions reaching the global minimum and maximum value of each objective are also included in order to identify the worst and best values obtained.

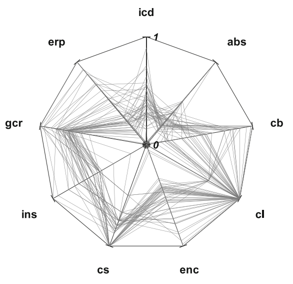
If we look at how lines are distributed, it can be seen that some algorithms tend to focus their search on specific values of particular objectives, while others can reach a wider range of trade-offs among them. More specifically, SPEA2 is unable to reach a proper trade-off among all the metrics, icd, *abs* and *cb* usually being optimised worse than erp, gcr, enc and cl. On the contrary, NSGA-II shows great ability to simultaneously deal with all the objectives. In this case, a wide spectrum of solutions is returned, some of them being are able to reach the best values for several objectives. Regarding MOEA/D, most of the obtained



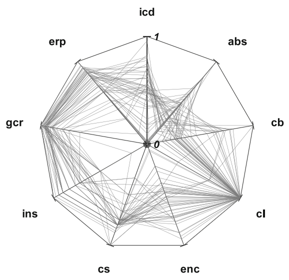
(a) SPEA2



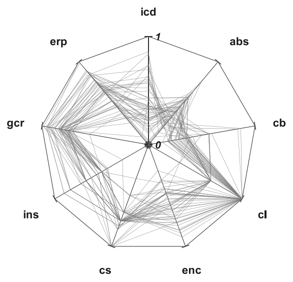
(b) NSGA-II



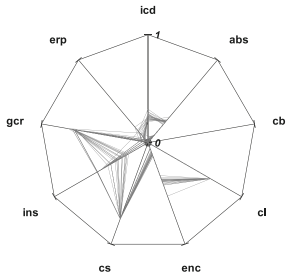
(c) MOEA/D



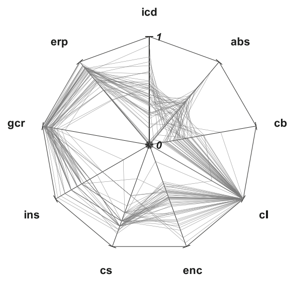
(d) ϵ-MOEA



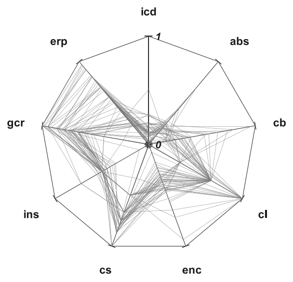
(e) GrEA



(f) IBEA



(g) HypE



(h) NSGA-III

Fig. 5 An example of the Pareto front obtained for the *Aqualush* problem instance

solutions lie on intermediate ranges, gcr, cs and cl being frequently promoted to the detriment of icd and abs. This algorithm is able to obtain great diversity of solutions in terms of erp, enc and cb. Focusing on ϵ-MOEA, it can reach competitive values for all the objectives. Its ability to provide several alternatives for icd, enc and cb is noteworthy, whereas good values are frequently reached for gcr and cl. GrEA and HypE behave similarly. Both algorithms experience difficulties to optimise abs and cb, getting only low or medium values. Besides, some solutions tend to concentrate on certain values of the design metrics, generating very similar solutions. HypE converges disproportionally towards erp, gcr and cl. Solutions returned by IBEA show very poor values for all the objectives. Even cb and cs are usually neutralised in benefit of other metrics, causing the returned solutions to be comprised by components completely unbalanced in size. Finally, NSGA-III has had problems to reach high values of icd and abs, although it is able to provide a wide range of values for erp, gcr, cs and enc.

Table 8 Summary of the evolutionary performance for 6-, 8- and 9-objective problems

Algorithm	Main findings
SPEA2	• Valuable diversity preservation when a larger number of objectives is optimised • Unable to converge to the PF as well as other approaches can
NSGA-II	• Behaviour changes when dealing with 8 and 9 objectives • Overtaken by ϵ-MOEA in terms of HV, but shows an improvement in S • Crowding distance may have a deeper impact on individual survival than the sorting method
MOEA/D	• Exploration of an excessive number of directions without favouring any of them • Not all the subproblems are useful to solve the global problem in a discontinuous space
ϵ-MOEA	• Best algorithm with respect to HV, but the observed S variability still remains • The many-objective approach with the best trade-off between both quality indicators
GrEA	• Low ranking values for both quality indicators • No evidence of improvement as the number of objectives increases
IBEA	• Low ranking values for both quality indicators, like GrEA • Outperformed by all other algorithms
HypE	• Intermediate ranking values for both quality indicators • Equivalent to the corresponding control algorithms for specific combinations
NSGA-III	• Only outperformed by SPEA2 and NSGA-II in terms of S • Ranking positions are still low with respect to HV

Table 8 shows the most relevant aspects related to the behaviour of the different algorithms when dealing with highly dimensional objective spaces.

7 Discussion of Results

Due to the particular nature of the problem, further discussion of the outcomes is still required, not only from the evolutionary perspective, but also in terms of its applicability in tool support and the software architect expectations. Therefore, there are other additional aspects from the experimentation to be observed that can provide relevant information about both the architectural specification and the decision support process. With this aim, the number of non-dominated solutions, the execution time or the dependencies among metrics have been studied. These factors directly affect the number of solutions given to the software architect or the sort of returned architectural solutions.

7.1 On Applicability in Tool Support

Having a wide variety of alternative solutions to choose from is not an ideal scenario for the software architect, even more so if only a few of them could serve to meet his expectations in terms of diversity. In such a case, the number of solutions could be useless to the expert, who would be unable to handle them all during the decision process. This matter is likely to require some further external post-processing to select the most representative solutions. Figure 6 shows the distribution of returned solutions per number of objectives. More precisely, the Y axis represents the mean number of solutions

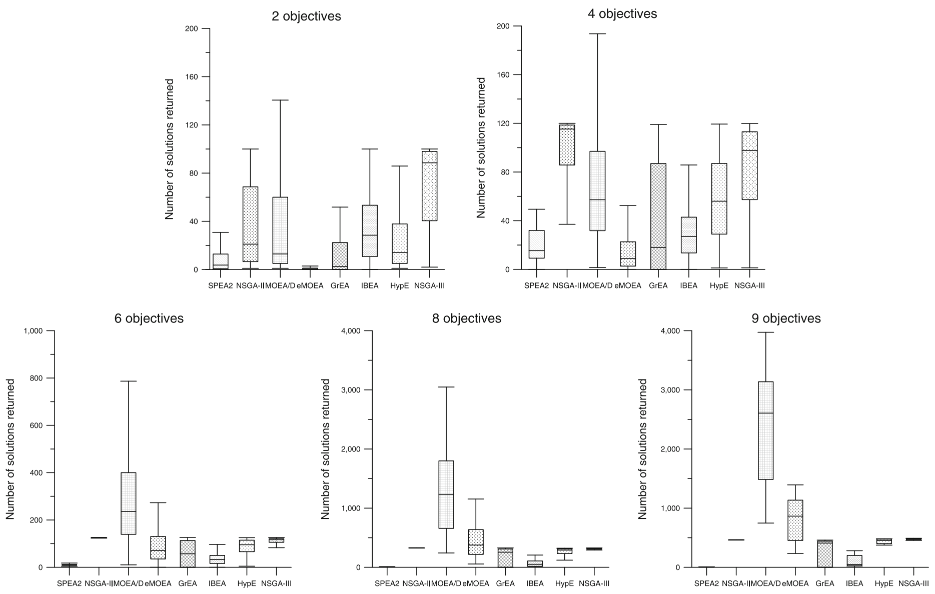


Fig. 6 Distribution of the number of non-dominated solutions composing the Pareto set

found by each algorithm, i.e. the number of non-dominated solutions composing the Pareto set, considering all the possible combinations of design metrics over all the problem instances. As can be seen, the number of objectives has a clear impact on the number of solutions returned. It can be noted that SPEA2 establishes a fixed size for the external population, whereas MOEA/D and ϵ-MOEA do not constrain the size of their archive of solutions. Since NSGA-II, GrEA, IBEA, HypE and NSGA-III do not have any external population, the maximum number of solutions is given by the corresponding population size (see Table 3).

If 2 or 4 objectives are considered, SPEA2 and NSGA-II usually obtain a number of solutions close to the maximum allowed. These algorithms also present some variability in the size of the Pareto set. NSGA-II even shows some difficulties to control such a variability. Having to optimise 6 or more objectives, SPEA2 is unable to fill its external archive with only non-dominated solutions, facing difficulties to find interesting architectures. Finding a more representative set of solutions than the one provided by SPEA2 could benefit the decision making process. NSGA-II always reaches the maximum limit allowed with difficulties to reject equivalent solutions along the search process. MOEA/D provides an excessive number of final solutions for any design problem. Besides, preserving the variety of these solutions is not guaranteed since the maximum number of solutions associated to each search direction is not restricted. The same behaviour could be expected for ϵ-MOEA, but the use of the ϵ-dominance criteria to manage the addition of solutions into the archive clearly serves to implicitly limit its size.

Depending on the combination of objectives, GrEA, IBEA, HypE and NSGA-III can generate as many solutions as the prefixed maximum. As the number of objectives increases, IBEA and GrEA return a lower number of solutions. In contrast, HypE and NSGA-III tend to get closer to their respective maximum limit. In this sense, many-objective approaches provide a better control than NSGA-II and MOEA/D. Nevertheless, only SPEA2 permits setting up the estimated number of solutions that should be returned after the search process.

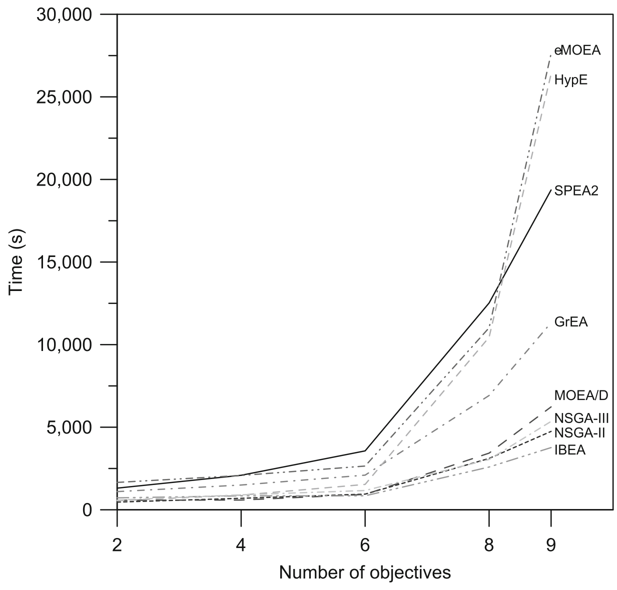


Fig. 7 Average execution time of the evolutionary algorithms

Hence, it is a useful mechanism for the expert, who might be interested in strictly determining the number of solutions that he really wants to check. Notice that GrEA, SPEA2, IBEA and ϵ-MOEA might finish the search without generating any architectural solutions that fulfil the constraints (see Fig. 6). These algorithms have some difficulties in dealing with invalid solutions when optimising the most complex problem instances and certain combinations of objectives. Setting a larger number of iterations would be a first attempt to address this situation. The ability to handle constraints is a relevant factor with respect to its generic applicability to the decision support process.

Another important factor that may affect the usability of search-based approaches is the execution time. Figure 7 shows the scalability of the average execution time of each algorithm in relation to the number of objectives. It can be observed that dealing with many design metrics influences SPEA2, GrEA, ϵ-MOEA and HypE. The density estimator based on clustering proposed by SPEA2, the computation of distance metrics in GrEA, and the hypervolume estimation required by HypE are operations that clearly affect the performance of these algorithms. Besides, SPEA2, GrEA and ϵ-MOEA present some difficulties when removing invalid solutions because they have to execute a large number of attempts during the mutation phase, especially when dealing with the most complex problem instances. Since this kind of instances tends to comprise more interconnected classes, identifying independent components becomes a harder work. In this sense, software engineers should consider if getting high quality solutions could make up for the amount of time spent on the process. On the one hand, ϵ-MOEA seems to be a better choice than HypE or NSGA-II if the architect is mostly interested in high quality solutions. On the other hand, NSGA-II might provide good enough solutions for most of the objective combinations, its execution time being steady and suitable. It is worth mentioning that the computational cost of NSGA-III has proven to be quite similar to NSGA-II, concluding that its execution time has not been significantly influenced by the adjustments proposed to deal with many-objective optimisation.

7.2 On the Influence of Metrics

Analysing whether two metrics, i.e. objectives, are highly competitive is an important factor to properly face the multi-objective optimisation, since it might determine much of the complexity of problem to be solved. Regarding the problem under study, this could lead to situations in which architects might presume beforehand that some specific metrics are closely related, mostly because they apparently arrange the architectural design in a similar way.

Table 9 presents the range of objective values of the final solutions considering all the bi-objective problems. The symbol ▲ indicates that the metric values should be maximised, whereas ▼ stands for minimisation. A unique reference PF has been constructed per problem instance, which is composed of all the solutions that are still not dominated by any other found after 30 executions of all the algorithms for a given problem instance. Then, the minimum and maximum bounds have been calculated in order to determine the extent of each metric range. A value $x_{i,j}$ represents the range of values achieved by final solutions when a measure j is optimised jointly with the measure i. For example, looking at the joint optimisation of icd and erp ($x_{2,1}$), it can be observed that icd varies in the range [0.0, 0.7] when combined with erp, whereas the range of values obtained would vary from 0.4 to 0.8 if optimised together with the metric ins ($x_{3,1}$). Consequently, ins is less opposite to icd than erp, since higher values of icd can be achieved even when ins is highly optimised.

Notice that combinations like $icd - erp$, $cb - gcr$ or $enc - abs$ have a wide range of values for both measures. Therefore, when a solution reaches a near-optimal value for one measure, the obtained value for the other measure is very poor. In contrast, there are other combinations in which one measure has been highly optimised regardless of the other measure being considered, such as when *cl* is chosen. The type of problem instance and the possibility of creating architectural solutions with different number of components allow not exceeding the configured critical link threshold. In such a case, a search process should be capable of finding solutions that reach good values for both objectives at the same time, making the MOP resolution slightly simpler. This would also explain why some algorithms are tied (see Section 6.1).

The choice of design metrics also guides the search for certain types of architectural specifications. For example, a complex scenario is developed regarding the modularity criteria, where a trade-off between coupling and cohesion would be expected. In this sense, metrics like icd or ins imply searching for solutions comprising of more components as a way to

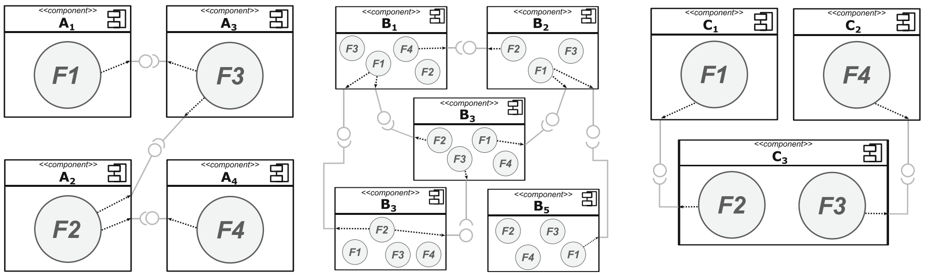
Table 9 Range of absolute objective values for non-dominated solutions considering all the combinations of 2 measures

	$icd^\blacktriangle$	$erp^\blacktriangledown$	$ins^\blacktriangledown$	$enc^\blacktriangle$	$cs^\blacktriangledown$	$cl^\blacktriangledown$	$gcr^\blacktriangledown$	$abs^\blacktriangle$	$cb^\blacktriangle$
$icd^\blacktriangle$	–	[0.0,178.0]	[0.1,0.6]	[0.9,1.0]	[0.0,0.2]	[0.0,0.0]	[1.0,1.4]	[0.1,0.9]	[0.3,1.0]
$erp^\blacktriangledown$	[0.0,0.7]	–	[0.1,0.5]	[0.5,1.0]	[0.0,0.3]	[0.0,0.0]	[1.0,1.7]	[0.1,0.9]	[0.1,1.0]
$ins^\blacktriangledown$	[0.4,0.8]	[0.0,84.0]	–	[0.7,1.0]	[0.0,0.1]	[0.0,0.0]	[1.0,8.3]	[0.3,0.9]	[0.1,1.0]
$enc^\blacktriangle$	[0.5,0.7]	[0.0,161.0]	[0.1,0.5]	–	[0.0,1.0]	[0.0,0.0]	[1.0,4.0]	[0.1,0.9]	[0.0,1.0]
$cs^\blacktriangledown$	[0.3,0.8]	[0.0,258.0]	[0.1,0.2]	[0.6,1.0]	–	[0.0,0.0]	[1.0,7.4]	[0.5,0.9]	[0.1,1.0]
$cl^\blacktriangledown$	[0.6,0.8]	[0.0,0.0]	[0.1,0.2]	[0.9,1.0]	[0.0,0.1]	–	[1.0,1.2]	[0.8,0.9]	[0.6,1.0]
$gcr^\blacktriangledown$	[0.3,0.7]	[0.0,6.0]	[0.1,0.6]	[0.5,1.0]	[0.0,0.2]	[0.0,0.0]	–	[0.3,0.9]	[0.0,1.0]
$abs^\blacktriangle$	[0.0,0.7]	[0.0,34.0]	[0.1,0.7]	[0.2,1.0]	[0.0,0.2]	[0.0,0.0]	[1.0,3.70]	–	[0.1,1.0]
$cb^\blacktriangle$	[0.2,0.7]	[0.0,560.0]	[0.1,0.5]	[0.4,1.0]	[0.0,0.2]	[0.0,1.0]	[1.0,22.0]	[0.1,0.9]	–

improve their cohesion. If the analysability is considered, then the architectures obtained would be composed of smaller components, each providing a well defined set of services. On the contrary, erp or enc look for a low coupling, so the solution structure is characterised by the creation of architectures with only two or three large components. Here, minimising the interactions among software artefacts would benefit the overall security and performance of the resulting system, although an excessive encapsulation of functionalities could harm its reusability and analysability, as observed when abs or cb were considered. Metrics like cs or cb counteract this effect and propitiate the generation of solutions comprised of a larger number of components.

The trade-off among these design criteria becomes a great challenge when 8 or 9 objectives have to be simultaneously optimised. In this case, low and medium values have been obtained for icd, especially if SPEA2, GrEA, IBEA and NSGA-III are executed. Similarly, cb and abs are difficult to optimise. IBEA can only provide solutions comprising components with unbalanced sizes even when cs is considered. Focusing on abs, it is worth mentioning that only NSGA-II, ϵ-MOEA and HypE reach values good enough for all the problem instances. The cs measure is usually cancelled out due to the presence of others like erp, gcr or enc. In this sense, the optimisation of these metrics seems to be harder, which makes the evolutionary process oriented towards them. As a result, these architectural solutions sometimes contain at least one large component. It should be noted that including erp, gcr and abs within the set of objectives leads to more variable outcomes over the different problem instances. In fact, these metrics are based on specific analysis information extracted from the original design, such as the number of abstract classes or the number and types of interrelationships. Finally, similar results are obtained for ins and enc in all the problem instances, meaning that these metrics are less conflicting.

To illustrate how the combination of metrics can influence the quality of the architectural description from the architectural perspective, the solutions returned by NSGA-II for the *Datapro4j* problem instance are scrutinised and discussed. Figure 8a depicts the original system architecture, which is comprised of 4 components ($A_1, \ldots, A_4$) whose functionalities will be referred here from $F1$ to $F4$. Notice that, according to the problem statement, the functionality of a component is represented as a group of highly interconnected classes, whose relationships are originally specified in the input class diagram. If cs and cb are considered together, one of the most interesting solutions returned by the algorithm is formed by 5 components ($B_1, \ldots, B_5$) with similar sizes (see Fig. 8b). The evaluation mechanism classifies all the components as noncritical in terms of their sizes, and both objectives are highly optimised. However, the set of classes inside each component does not represent a comprehensive functionality, but a combination of unrelated classes. The reason for this is that the optimisation process is directed towards the optimal size balance without considering



(a) Sketched original architecture (b) Sketched solution for $cs - cb$ (c) Sketched solution for $icd - erp$

Fig. 8 Returned solutions by NSGA-II for *Datapro4j* problem instance

whether the classes allocated in the same component are related. As a result, functionalities are distributed among different components, decreasing their cohesion and increasing the overall coupling. On the contrary, when icd and erp are the considered metrics, the algorithm found one solution with 3 components very similar to the original proposal (see Fig. 8c). Only the third component, C_3, comprised the classes related to $F2$ and $F3$, being catalogued as critical according to the cs metric. Finally, the joint optimisation of the 4 metrics leads to the discovery of solutions very close or even equal to the original architecture. Considering icd and erp as objective functions has served to properly identify the different functionalities, while cs and cb have encouraged the creation of two medium-sized components, one that contains the classes associated with $F2$ and another specifying $F3$. It should be noted that other problem instances could require different metrics, or even a greater number of objectives, in order to discover the most appropriate system architecture to the engineer. Nevertheless, in the cases in which the software architect is not confident enough with the intended architecture, considering the execution of different combinations of metrics could be appropriate. In short, the selection of metrics made by the architect is a key factor to determine the structure of the returned architectural solutions, whereas the achieved level of optimisation for the selected objectives is more related to the used algorithm.

7.3 On the Selection of the Evolutionary Algorithm

The selection of the evolutionary approach is not a trivial task for the user, not least if he needs to be aware of the impact that the diverse characteristics of each algorithm may have on searching for the expected architectonical solutions. SPEA2 shows a low performance in its search for high quality solutions, even though its diversity preservation technique has shown to be a very effective mechanism to provide a great variety of architectural solutions. On the other hand, NSGA-II provides a valuable scalability with respect to all the considered optimisation problems. Besides, regarding the diversity of solutions, this algorithm behaves more variably. Nevertheless, its scalable execution time and the absence of parameters are appealing features to select this algorithm as a generic choice.

The decomposition approach proposed by MOEA/D is an interesting characteristic to promote the diversity of solutions, but it works better with fewer than 6 objectives. Besides, the exploration of many search directions could lead to an excessive number of returned solutions, their diversity not always being guaranteed. These circumstances, along with the observed difficulties in generating high quality solutions, imply that the expert faces a complex task when looking for the final solution among all the alternatives given by the algorithm. Similarly, NSGA-III looks for the diversity of solutions, performing better than MOEA/D in highly dimensional objective spaces. In addition, this many-objective approach is capable of keeping most of the beneficial characteristics of its predecessor.

Concerning the rest of algorithms and considering the different families, it can be noted that GrEA is overtaken by ϵ-MOEA, whereas HypE shows a better behaviour than IBEA. One observed weakness of GrEA is that it suffers from an excessive convergence and also has some difficulty in removing invalid solutions during the search process. As a result, this algorithm seems to be unable to discover any competitive software architecture for the most complex systems. Something similar happens when ϵ-MOEA is executed over certain combinations of measures. Then, the algorithm could be more sensible to the selection of the ϵ values, whose configuration might be a laborious task for the expert. Nevertheless, ϵ-MOEA provides the best trade-off between convergence and diversity of solutions for a large number of metrics.

In general, IBEA is unable to reach a reasonable trade-off between the set of metrics, whose cardinality has an impact on its behaviour respecting the diversity of solutions. On the contrary, HypE, whose performance improves as the number of objectives increases, is able to find non-dominated solutions for the totality of the considered problems, showing an appropriate explorative capability. The only drawback of HypE, which is also observed in ϵ-MOEA, is its highly computational cost, especially if more than 6 metrics participate in the optimisation problem.

A better knowledge about the specific behaviour of each algorithm can determine its applicability to a particular design scenario. Even so, some general guidelines for the selection of the evolutionary approach are inferred from this study. In case the architect is requested to consider a specific set of metrics, the results in Section 6 would allow him to identify the best algorithm for that specific combination. Additionally, if a general choice is preferred, NSGA-II and ϵ-MOEA have demonstrated to have the best performance in general terms, so the final decision should be made considering other additional factors like the total number of metrics to be optimised or time requirements.

8 Threats to Validity

In order to precisely designate the design concepts and terms that lay the foundation for the proposed approach, the ISO Std. 25000 has been followed as a reference to correctly define the non-functional requirements that guide the search process. Additionally, the OMG standard UML 2 has served to provide a rigorous notation for the specification of the underlying design models at all the abstraction levels, as well as to ensure the semantic validity of quality metrics. Using a standard proposal like UML 2 brings the optimisation approach closer to the everyday realm of the software architect.

Internal validity refers to those aspects of the experimentation that cannot ensure the causality between the hypothesis and the obtained results. In SBSE, these aspects are related to the algorithm's set-up and its parametrisation. The use of default parameter values could produce bias on the results, since the original algorithms were usually tested over real optimisation problems, even though the nature of our problem is quite different. Here, the guidelines provided by their corresponding authors have been followed to configure the algorithms. Nevertheless, some preliminary experiments were performed to check their suitability. If required, some modifications to the original proposal implementations were made to adapt them to our specific combinatorial problem, e.g. the required constraint handling techniques were included in all the algorithms. These changes can affect the performance of the algorithms, but they are necessary to deal with the defined search problem and would affect all the algorithms in a similar way. The experimentation has been designed to avoid any bias due to the intrinsic randomness of the evolutionary approaches. So, every algorithm has been executed 30 times. Additionally, a hardware platform specifically dedicated to experimentation purposes has served to assure that all the algorithms have been executed under the same conditions, meaning that their execution time is also comparable. Additionally, well-known quality indicators like hypervolume and spacing have been used to evaluate the evolutionary performance of the algorithms. Non-parametric statistical tests at a significance level of 95 % were executed to draw precise conclusions in the light of the comparative results of the algorithms under study.

External validity is related with the generalisation of the experimental results. On the one hand, a set of real-world software systems has been considered as problem instances, covering different sizes and complexities. In this paper, the evolutionary search has been performed without any prior knowledge about the original design. It is worth mentioning that any additional analysis information would help to obtain more accurate outcomes from the discovery process. On the other hand, the application of this work to other industrial domains might imply adapting the proposed approach to their specific requirements. The selected design metrics are closely related to maintainability, which is clearly an important characteristic in the design of component-based software architectures. Nevertheless, notice that addressing different design tasks based on other architectural models (e.g. service-oriented architectures, grid-based platforms, etc.) would probably require the evaluation of specific quality criteria.

9 Concluding Remarks

This paper presents a comparative study on the performance of different multi- and many-objective evolutionary algorithms for software architecture discovery, an abstract problem that demands diverse decisions to be made according to a number of requirements. The proposed experimental framework explores the ability of these algorithms to simultaneously deal with different metrics related to maintainability at the architectural level, and put forth some ideas about how they should selected and combined by the software architect. Existing dependencies among metrics have shown to be an important factor that could subsequently impact on both the complexity of the optimisation problem to be solved and the type of the architectural solutions returned by the search process. On the one hand, metrics like icd, ins, erp or gcr are more related to the allocation of classes within components on the basis of their relationships. On the other hand, aspects like avoiding critical components or balancing their sizes might be considered as secondary goals, since they are not capable of identifying the system functionalities.

The results obtained have shown that many-objective evolutionary algorithms provide an interesting alternative to deal with such a complex combinatorial problem. On the one hand, NSGA-II is the only multi-objective approach that achieves competitive results for some objective combinations with respect to many-objective approaches like ϵ-MOEA and HypE. Nevertheless, these algorithms provide a greater performance in terms of the expected trade-off between convergence and diversity when highly dimensional objective spaces need to be considered, as it is often the case of software design optimisation. On the other hand, the set of metrics selected to guide the search becomes a less influential factor on the performance of the algorithms as the number of objectives increases.

An in-depth analysis of the strengths of each algorithm has been presented and extensively discussed. The number of returned solutions or the execution time required to perform the search constitute complementary and valuable aspects to be considered in order to take a justified decision about the selection of a specific algorithm. Additionally, the guidelines provided in this study are essential to take a step further in the automatic recommendation of the evolutionary approach that best fits into a particular architectural problem. As a future work, hyper-heuristic approaches based on the information extracted from this analysis will serve as a basis for the development of decision support tools for software engineers. Finally, many-objective evolutionary optimisation is an open field of research, which makes it interesting to continue exploring some emerging approaches (He et al. 2014; Wang et al. 2014).

Acknowledgments Work supported by the Spanish Ministry of Science and Technology, project TIN2011-22408, the Spanish Ministry of Economy and Competitiveness, project TIN2014-55252-P, and FEDER funds. This research was also supported by the Spanish Ministry of Education under the FPU program (FPU13/01466).

Table 10 Results of the Cliff's Delta test for hypervolume (*n*=negligible, *s*=small, *m*=medium, *l*=large) ($\alpha = 0.05$)

Algorithm	SPEA2	NSGA-II	MOEA/D	ϵ-MOEA	GrEA	IBEA	HypE	NSGA-III
SPEA2	–	−1.00 (*l*)	−1.00 (*l*)	−1.00 (*l*)	−0.60 (*l*)	0.90 (*l*)	−1.00 (*l*)	−0.54 (*l*)
NSGA-II	0.90 (*l*)	–	0.14 (*n*)	−0.88 (*l*)	0.84 (*l*)	0.90 (*l*)	−0.06 (*n*)	0.90 (*l*)
MOEA/D	0.90 (*l*)	−0.14 (*n*)	–	−0.88 (*l*)	0.78 (*l*)	0.90 (*l*)	−0.20 (*s*)	0.90 (*l*)
ϵ-MOEA	0.90 (*l*)	0.88 (*l*)	0.88 (*l*)	–	0.90 (*l*)	0.90 (*l*)	0.88 (*l*)	0.90 (*l*)
GrEA	0.60 (*l*)	−0.90 (*l*)	−0.84 (*l*)	−1.00 (*l*)	–	0.60 (*l*)	−0.96 (*l*)	0.60 (*s*)
IBEA	−1.00 (*l*)	−1.00 (*l*)	−1.00 (*l*)	−1.00 (*l*)	−0.60 (*l*)	–	−1.00 (*l*)	−1.00 (*l*)
HypE	0.90 (*l*)	0.06 (*n*)	−0.20 (*s*)	−0.88 (*l*)	0.88 (*l*)	0.90 (*l*)	–	0.90 (*l*)
NSGA-III	0.54 (*l*)	−1.00 (*l*)	−1.00 (*l*)	−1.00 (*l*)	−0.60 (*l*)	0.900 (*l*)	−1.00 (*l*)	–

Appendix

Tables 10 and 11 present the results of the Cliff's Delta test for hypervolume (HV) and spacing (S), respectively. The value of a cell, $x_{i,j}$, represents the effect size and its interpretation when comparing the algorithm i against the algorithm j for the 9-objective optimisation problem in terms of the specific quality indicator. Table 12 shows the results of the Friedman and the Holm tests for all possible combinations of 2 objectives. Tables 13, 14, 15, 16 report the results for the 4-objective problems, whereas the 6-objective problems are shown from Table 17, 18, 19. In addition, Table 19 contains the results obtained from 8-objective and 9-objective problems. For each of them, the best rankings for the two quality indicators, HV and S, are shown in bold typeface, and their cells are shaded in gray colour when significant differences exist. The critical value, according to the F-Distribution with 6 and 54 degrees of freedom, i.e. the p-value, is 2.2720. Since the Holm test is performed if H_0 is rejected, Tables 12–19 show these ranking values in italic typeface when the corresponding algorithm lies below its critical threshold, i.e. its performance according to the column-specific indicator is worse than that provided by the best algorithm.

Table 11 Results of the Cliff's Delta test for spacing (*n*=negligible, *s*=small, *m*=medium, *l*=large) ($\alpha = 0.05$)

Algorithm	SPEA2	NSGA-II	MOEA/D	ϵ-MOEA	GrEA	IBEA	HypE	NSGA-III
SPEA2	–	0.32 (*s*)	0.89 (*l*)	0.87 (*l*)	0.87 (*l*)	0.90 (*l*)	0.85 (*l*)	0.68 (*l*)
NSGA-II	−0.32 (*s*)	–	0.90 (*l*)	0.90 (*l*)	0.88 (*l*)	0.90 (*l*)	0.87 (*l*)	0.88 (*l*)
MOEA/D	−0.98 (*l*)	−1.00 (*l*)	–	−0.36 (*m*)	0.64 (*l*)	0.90 (*l*)	−0.10 (*n*)	−0.88 (*l*)
ϵ-MOEA	−0.96 (*l*)	−1.00 (*l*)	0.35 (*m*)	–	0.78 (*l*)	0.90 (*l*)	0.26 (*s*)	−0.82 (*l*)
GrEA	−0.94 (*l*)	−0.96 (*l*)	−0.64 (*l*)	−0.78 (*l*)	–	0.58 (*l*)	−0.76 (*l*)	−0.82 (*l*)
IBEA	−1.00 (*l*)	−1.00 (*l*)	−1.00 (*l*)	−1.00 (*l*)	−0.58 (*l*)	–	−1.00 (*l*)	−1.00 (*l*)
HypE	−0.92 (*l*)	−0.94 (*l*)	0.10 (*n*)	−0.26 (*s*)	0.75 (*l*)	0.90 (*l*)	–	−0.64 (*l*)
NSGA-III	−0.68 (*l*)	−0.96 (*l*)	0.79 (*l*)	0.82 (*l*)	0.81 (*l*)	0.90 (*l*)	0.64 (*l*)	–

Table 12 Average rankings for 2-objective problems obtained from the Friedman test ($\alpha = 0.05$)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
icd-erp	2.80	3.20	**1.30**	4.40	*5.35*	**2.20**	3.35	3.80	*4.70*	*5.80*	*7.75*	*7.50*	*4.10*	4.90	*6.65*	4.20
icd-gcr	4.00	**2.00**	**1.60**	4.20	*5.20*	2.80	3.50	*6.10*	3.80	*5.80*	*7.60*	*7.10*	3.60	3.50	*6.70*	4.50
icd-cs	*5.10*	3.65	2.45	2.60	3.85	**2.50**	4.70	*7.20*	3.85	5.05	*7.10*	*5.30*	**2.05**	4.00	*6.90*	*5.70*
icd-cl	*5.45*	*5.20*	**2.30**	*5.20*	3.90	4.60	2.50	5.20	3.90	*5.20*	*7.05*	**2.60**	3.35	*5.20*	*7.55*	2.80
icd-ins	*5.15*	3.10	2.90	3.70	*4.95*	**3.00**	**1.90**	*6.90*	4.25	5.00	*6.60*	3.10	2.50	4.20	*7.75*	*7.00*
icd-enc	*5.60*	2.40	**1.70**	4.30	3.80	**1.90**	2.80	*7.25*	4.20	*5.55*	*7.20*	*7.10*	3.30	3.30	*7.40*	4.20
icd-abs	2.70	3.20	**1.10**	*5.20*	*4.80*	**2.30**	*5.30*	3.95	*4.70*	*6.30*	*7.25*	*7.55*	2.80	4.60	*7.35*	2.90
icd-cb	3.90	**1.70**	2.30	3.80	*4.70*	1.80	*5.80*	*7.30*	3.50	*5.90*	*7.10*	4.10	**1.60**	4.20	*7.10*	*7.20*
erp-gcr	3.75	*5.20*	**2.75**	*5.80*	4.05	*4.60*	2.90	*5.90*	5.30	*5.30*	*7.30*	2.30	3.25	*5.40*	*6.70*	**1.50**
erp-cs	4.00	**2.80**	**1.70**	4.15	*4.40*	3.60	4.20	*6.75*	*6.25*	5.45	*7.40*	5.55	2.25	4.20	*5.80*	3.50
erp-cl	**3.20**	4.80	**3.20**	4.80	**3.20**	4.80	**3.20**	4.80	5.60	4.80	*7.55*	4.35	3.20	4.80	*6.85*	**2.85**
erp-ins	3.70	**2.20**	**2.25**	4.45	4.45	3.85	2.85	*5.65*	*5.10*	4.85	*7.65*	*5.90*	3.25	4.25	*6.75*	*4.85*
erp-enc	*3.60*	**2.50**	**1.35**	3.50	*3.80*	2.60	*3.80*	*6.20*	*6.20*	*6.10*	*7.35*	*6.15*	*3.60*	4.75	*6.30*	4.20
erp-abs	3.60	4.90	**1.60**	**3.30**	*4.40*	4.25	3.15	4.80	*5.90*	4.60	*7.60*	6.00	3.25	3.65	*6.50*	4.50
erp-cb	*4.20*	2.70	**1.50**	3.90	*4.60*	**2.20**	*4.40*	4.20	3.85	*5.55*	*7.45*	*7.35*	3.20	4.20	*6.80*	*5.90*
gcr-cs	4.20	**2.95**	**1.80**	4.25	3.95	3.35	3.85	*6.05*	*6.20*	5.25	*6.80*	4.85	2.45	3.65	*6.75*	5.65
gcr-cl	3.70	4.90	**3.20**	4.90	**3.20**	4.90	**3.20**	4.90	5.50	4.90	*7.15*	3.80	**3.20**	4.90	*6.85*	**2.80**
gcr-ins	4.70	**1.90**	**2.40**	4.20	4.80	3.70	2.80	*5.75*	4.20	*5.85*	*7.70*	*6.55*	2.80	4.35	*6.60*	3.70
gcr-enc	*4.60*	3.60	**1.30**	3.90	*3.95*	3.80	*3.95*	4.40	*5.70*	*6.50*	*7.00*	*6.40*	*3.50*	**3.20**	*6.00*	4.20
gcr-abs	4.95	5.00	**2.20**	**3.00**	4.70	3.50	2.95	5.20	4.40	4.85	*7.40*	*6.00*	3.00	4.45	*6.40*	4.00
gcr-cb	4.70	**2.60**	**2.25**	4.10	3.35	3.70	3.90	3.40	4.05	5.35	*7.25*	*7.05*	3.20	2.70	*7.30*	*7.10*
cs-cl	3.50	4.85	**2.55**	4.85	3.15	4.15	4.70	4.85	5.05	4.85	*7.05*	4.40	3.45	4.85	*6.55*	**3.20**
cs-ins	*4.45*	3.90	**1.65**	4.40	3.60	4.05	*5.75*	*6.00*	*5.95*	*5.70*	*6.65*	*5.90*	2.05	3.50	*5.90*	**2.55**

Table 12 (continued)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
cs-enc	*4.80*	3.30	**1.70**	3.60	3.60	**2.05**	*5.10*	*6.75*	*5.35*	*5.10*	*6.95*	*6.50*	2.80	4.00	*5.70*	4.70
cs-abs	*4.35*	*5.10*	**1.80**	3.10	4.10	**1.70**	*5.25*	*7.25*	4.30	*4.75*	*7.40*	*6.90*	2.00	3.10	*6.80*	4.10
cs-cb	*5.05*	5.30	**2.05**	**3.15**	2.80	3.25	*5.50*	5.30	4.55	*5.30*	*6.70*	5.30	2.85	3.45	*6.50*	4.95
cl-ins	3.55	4.75	2.60	5.25	**2.55**	4.80	5.10	5.25	*5.15*	4.85	*7.40*	**2.65**	2.60	4.70	*7.05*	3.75
cl-enc	*6.05*	*5.05*	2.55	*5.05*	**2.40**	*5.05*	*5.55*	*5.05*	4.70	*5.05*	*6.35*	*4.25*	2.60	*5.05*	*5.80*	**1.45**
cl-abs	*5.25*	*5.15*	**1.70**	4.15	*4.65*	*5.15*	2.70	5.15	3.85	4.45	*7.35*	*5.15*	3.40	4.35	*7.10*	**2.45**
cl-cb	4.75	*5.80*	**2.60**	2.75	3.40	**2.35**	3.45	*5.80*	4.70	*5.15*	*7.10*	*5.80*	3.30	3.90	*6.70*	4.45
ins-enc	4.50	2.50	**2.45**	2.70	3.75	**2.50**	3.20	*6.25*	5.20	*5.95*	*7.15*	*6.70*	2.70	3.30	*7.05*	*6.10*
ins-abs	*4.80*	2.80	2.70	3.40	*5.20*	**1.40**	**1.30**	*5.75*	*4.20*	5.30	*7.00*	*7.35*	3.40	4.00	*7.40*	*6.00*
ins-cb	3.40	**2.20**	**1.75**	3.40	3.75	2.30	3.70	*5.35*	*6.10*	*6.30*	*7.35*	*7.45*	3.10	3.30	*6.85*	*5.70*
enc-abs	*4.45*	**1.90**	**1.10**	3.10	2.30	3.70	*4.80*	*5.85*	*5.10*	5.30	*7.20*	*7.35*	*4.20*	3.80	*6.85*	*5.00*
enc-cb	*5.30*	*5.20*	**1.40**	2.80	2.80	**1.40**	*4.90*	*6.85*	*4.35*	*5.10*	*7.20*	*6.60*	3.60	2.40	*6.45*	*5.65*
abs-cb	*4.80*	3.00	**1.10**	4.70	2.90	**2.40**	*3.60*	3.80	*5.25*	*6.30*	*7.40*	*7.45*	*3.80*	5.10	*7.15*	3.25
cl-cb	4.75	*5.80*	**2.60**	2.75	3.40	**2.35**	3.45	*5.80*	4.70	*5.15*	*7.10*	*5.80*	3.30	3.90	*6.70*	4.45
ins-enc	4.50	2.50	**2.45**	2.70	3.75	**2.50**	3.20	*6.25*	5.20	*5.95*	*7.15*	*6.70*	2.70	3.30	*7.05*	*6.10*
ins-abs	*4.80*	2.80	2.70	3.40	*5.20*	**1.40**	**1.30**	*5.75*	*4.20*	5.30	*7.00*	*7.35*	3.40	4.00	*7.40*	*6.00*
ins-cb	3.40	**2.20**	**1.75**	3.40	3.75	2.30	3.70	*5.35*	*6.10*	*6.30*	*7.35*	*7.45*	3.10	3.30	*6.85*	*5.70*
enc-abs	*4.45*	**1.90**	**1.10**	3.10	2.30	3.70	*4.80*	*5.85*	*5.10*	5.30	*7.20*	*7.35*	*4.20*	3.80	*6.85*	*5.00*
enc-cb	*5.30*	*5.20*	**1.40**	2.80	2.80	**1.40**	*4.90*	*6.85*	*4.35*	*5.10*	*7.20*	*6.60*	3.60	2.40	*6.45*	*5.65*
abs-cb	*4.80*	3.00	**1.10**	4.70	2.90	**2.40**	*3.60*	3.80	*5.25*	*6.30*	*7.40*	*7.45*	*3.80*	5.10	*7.15*	3.25

Table 13 Average rankings for 4-objective problems obtained from the Friedman Test ($\alpha = 0.05$) (cont'd)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
icd-erp-gcr-cs	2.10	3.00	**1.00**	*6.00*	*4.20*	3.40	*4.00*	**2.40**	*5.75*	*6.30*	*7.75*	*7.40*	*4.60*	5.00	*6.60*	2.50
icd-erp-gcr-cl	2.40	2.90	**1.40**	*5.60*	*4.80*	3.20	3.50	3.10	*5.75*	*6.70*	*7.65*	*7.30*	3.90	4.70	*6.60*	**2.50**
icd-erp-gcr-ins	2.90	**2.10**	**1.30**	*5.80*	*4.50*	2.70	2.00	2.40	*6.20*	*6.90*	*7.60*	*7.30*	*4.90*	4.70	*6.60*	4.10
icd-erp-gcr-enc	3.20	2.90	**1.10**	*5.80*	*4.10*	2.80	2.30	**1.50**	*6.40*	*7.10*	*7.60*	*7.50*	*4.70*	*4.60*	*6.60*	3.80
icd-erp-gcr-abs	*3.70*	**1.30**	**1.10**	*5.20*	*4.10*	3.00	1.90	2.70	*6.70*	*7.20*	*7.60*	*7.60*	*4.30*	*4.20*	*6.60*	*4.80*
icd-erp-gcr-cb	*4.40*	1.90	**1.10**	*5.20*	*4.80*	3.00	1.90	**1.90**	*5.10*	*6.80*	*7.80*	*7.60*	*4.30*	*5.00*	*6.60*	*4.60*
icd-erp-cs-cl	2.40	2.70	**1.20**	*5.70*	*4.20*	**2.50**	*5.30*	4.65	*5.50*	*6.45*	*7.70*	*7.20*	3.10	3.80	*6.60*	3.00
icd-erp-cs-ins	*3.60*	2.20	**1.00**	*5.10*	*5.10*	2.50	2.80	**1.90**	*5.00*	*7.20*	*7.70*	*7.60*	*4.20*	*5.20*	*6.60*	4.30
icd-erp-cs-enc	*5.40*	**1.40**	**1.00**	4.70	2.70	*4.30*	*4.40*	2.15	*5.10*	*6.85*	*7.70*	*7.50*	3.10	*5.30*	*6.60*	3.80
icd-erp-cs-abs	*5.60*	**1.00**	**1.10**	*3.60*	*4.10*	*3.80*	1.90	2.20	*5.70*	*7.20*	*7.50*	*7.50*	3.40	*5.60*	*6.70*	*5.10*
icd-erp-cs-cb	*5.30*	**1.20**	**1.60**	*5.00*	*4.90*	3.30	1.90	2.10	*4.45*	*6.70*	*7.75*	*7.60*	3.40	*6.10*	*6.70*	*4.00*
icd-erp-cl-ins	3.35	2.60	**1.50**	*5.60*	*5.10*	3.50	2.45	**1.80**	*5.20*	*6.10*	*7.75*	*7.80*	4.00	4.00	*6.65*	*4.60*
icd-erp-cl-enc	*3.50*	2.60	**1.00**	*6.10*	*4.30*	3.10	3.00	**2.30**	*6.35*	*7.10*	*7.65*	*7.40*	*3.60*	4.80	*6.60*	2.60
icd-erp-cl-abs	*4.00*	**1.30**	**1.00**	*5.30*	*4.60*	3.70	2.00	1.80	*6.45*	*7.40*	*7.75*	*7.60*	*3.60*	*4.20*	*6.60*	*4.70*
icd-erp-cl-cb	*5.20*	**1.70**	**1.30**	*5.20*	*4.90*	2.90	2.20	2.00	*5.15*	*6.80*	*7.70*	*7.70*	2.80	*5.20*	*6.75*	*4.50*
icd-erp-ins-enc	*3.90*	2.00	**1.20**	*5.10*	*4.30*	3.60	1.90	**1.30**	*6.05*	*7.05*	*7.65*	*7.65*	*4.40*	*5.20*	*6.60*	*4.10*
icd-erp-ins-abs	*4.20*	**1.00**	**1.50**	3.30	*4.70*	*3.70*	1.50	2.50	*6.25*	*6.95*	*7.65*	*7.65*	3.60	*5.50*	*6.60*	*5.40*
icd-erp-ins-cb	*5.00*	**1.20**	**1.60**	4.30	*5.10*	3.60	1.60	1.80	*4.90*	*6.75*	*7.75*	*7.75*	3.40	*5.90*	*6.65*	*4.70*
icd-erp-enc-abs	*4.40*	**1.10**	**1.40**	*4.00*	3.70	3.40	1.60	3.00	*6.60*	*7.15*	*7.80*	*7.65*	3.90	*4.60*	*6.60*	*5.10*
icd-erp-enc-cb	*5.60*	**1.00**	**1.30**	*3.80*	*4.40*	*4.00*	1.70	2.00	*5.30*	*7.20*	*7.80*	*7.60*	3.30	*5.60*	*6.60*	*4.80*
icd-erp-abs-cb	*5.70*	**1.00**	1.60	2.80	*4.60*	*4.10*	**1.40**	2.50	*5.40*	*7.30*	*7.80*	*7.60*	3.00	*5.60*	*6.50*	*5.10*
icd-gcr-cs-cl	3.50	3.10	**1.40**	4.50	*5.00*	**2.30**	*4.90*	*6.00*	3.75	*6.00*	*7.65*	*7.40*	3.00	4.00	*6.80*	2.70

Table 13 (continued)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
icd-gcr-cs-ins	3.90	2.10	**1.40**	*4.60*	*5.20*	3.20	3.40	**1.60**	*4.40*	*7.10*	*7.70*	*7.60*	3.40	*5.60*	*6.60*	4.20
icd-gcr-cs-enc	*5.00*	**1.50**	**1.30**	*4.80*	*3.90*	3.90	*4.50*	1.60	*4.55*	*7.20*	*7.75*	*7.60*	2.40	*5.30*	*6.60*	*4.10*
icd-gcr-cs-abs	*5.60*	**1.10**	**1.00**	*3.90*	*4.50*	3.10	2.00	2.10	*5.35*	*7.20*	*7.75*	*7.60*	3.20	*6.10*	*6.60*	*4.90*
icd-gcr-cs-cb	*5.40*	**1.20**	**2.20**	*4.90*	4.80	3.60	2.70	2.50	4.05	*6.35*	*7.65*	*7.65*	2.30	*4.50*	*6.90*	*5.30*
icd-gcr-cl-ins	*4.20*	2.10	**1.70**	*5.40*	*5.10*	2.80	2.60	**1.90**	4.00	*6.60*	*7.60*	*7.60*	4.10	*4.90*	*6.70*	*4.70*
icd-gcr-cl-enc	*4.50*	2.30	**1.10**	*6.20*	*4.30*	3.80	2.80	**1.80**	*5.10*	*7.30*	*7.60*	*7.40*	*4.00*	4.40	*6.60*	2.80
icd-gcr-cl-abs	*4.10*	**1.50**	**1.00**	*5.60*	*5.00*	3.20	2.00	2.40	*6.45*	*7.30*	*7.65*	*7.60*	3.20	3.40	*6.60*	*5.00*
icd-gcr-cl-cb	*4.70*	1.90	**1.40**	*5.00*	*5.40*	3.00	2.50	**1.50**	4.00	*6.80*	*7.55*	*7.60*	3.40	*4.50*	*7.05*	*5.70*
icd-gcr-ins-enc	*5.00*	**1.30**	**1.20**	*5.00*	*4.60*	*3.80*	2.00	1.80	*5.20*	*7.30*	*7.60*	*7.40*	*3.80*	*5.20*	*6.60*	*4.20*
icd-gcr-ins-abs	*4.50*	**1.10**	1.60	3.50	*5.30*	3.40	**1.40**	2.50	*5.90*	*7.40*	*7.60*	*7.60*	3.10	*5.60*	*6.60*	*4.90*

Table 14 Average rankings for 4-objective problems obtained from the Friedman Test ($\alpha = 0.05$) (cont'd)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
icd-gcr-ins-cb	4.80	**1.20**	**1.50**	4.60	5.40	3.50	2.20	2.00	4.65	6.90	7.75	7.60	3.00	5.80	6.70	4.40
icd-gcr-enc-abs	5.00	**1.00**	1.60	4.10	3.80	3.50	**1.40**	3.30	6.65	7.20	7.65	7.60	3.30	4.10	6.60	5.20
icd-gcr-enc-cb	5.60	**1.20**	**1.20**	4.90	4.60	3.30	2.70	1.90	4.90	7.10	7.80	7.60	2.60	5.40	6.60	4.60
icd-gcr-abs-cb	5.60	**1.10**	1.60	3.30	4.50	3.50	**1.40**	2.30	5.40	7.30	7.80	7.60	3.10	5.60	6.60	5.30
icd-cs-cl-ins	5.00	3.50	**2.30**	4.60	5.20	**1.40**	2.70	6.40	3.60	6.15	7.60	7.25	2.60	4.10	7.00	2.60
icd-cs-cl-enc	5.2	**1.80**	**1.80**	4.40	4.00	1.90	4.50	6.95	3.80	6.50	7.40	7.05	2.40	4.00	6.90	3.40
icd-cs-cl-abs	4.40	3.00	**1.00**	5.90	5.60	3.10	3.60	**1.50**	4.90	7.30	7.55	7.20	2.00	5.40	6.95	2.60
icd-cs-cl-cb	4.80	2.90	**2.00**	4.60	5.40	**1.50**	2.70	5.70	3.90	6.50	7.60	7.10	2.60	4.10	7.00	3.60
icd-cs-ins-enc	5.50	**1.70**	2.20	5.20	4.50	3.10	3.20	2.10	4.45	7.10	7.45	7.60	**1.90**	4.80	6.80	4.40
icd-cs-ins-abs	5.00	**1.60**	1.60	4.70	5.20	4.20	**1.50**	2.10	5.25	6.75	7.50	7.45	2.90	6.70	7.05	2.50
icd-cs-ins-cb	4.10	3.10	2.60	5.30	5.20	**2.30**	4.75	4.20	4.05	6.90	7.45	6.70	**1.20**	4.60	6.65	2.90
icd-cs-enc-abs	5.60	**1.00**	**1.30**	3.60	4.10	4.30	2.40	2.00	5.90	7.10	7.70	7.40	2.30	6.50	6.70	4.10
icd-cs-enc-cb	5.20	**1.70**	2.40	5.40	4.90	3.10	2.80	2.10	4.35	7.15	7.75	7.65	**1.90**	4.50	6.70	4.40
icd-cs-abs-cb	5.60	**1.40**	**1.10**	4.50	4.50	4.50	2.20	2.30	5.35	6.85	7.50	7.75	2.70	6.40	7.05	2.30
icd-cl-ins-enc	5.50	**1.90**	**2.00**	5.50	4.50	2.40	2.50	3.30	3.95	7.20	7.60	7.70	3.20	4.60	6.75	3.40
icd-cl-ins-abs	4.30	2.20	1.60	6.00	5.40	3.50	**1.40**	**1.90**	5.30	7.40	7.65	7.60	3.50	4.50	6.85	2.90
icd-cl-ins-cb	4.40	3.20	2.70	5.40	5.60	**1.70**	**2.20**	2.00	3.75	6.75	7.65	7.45	2.80	4.70	6.90	4.80
icd-cl-enc-abs	4.50	**1.70**	**1.30**	5.80	4.20	3.60	1.70	2.20	6.40	7.40	7.55	7.50	3.40	4.40	6.95	3.40
icd-cl-enc-cb	4.90	2.70	**2.00**	5.80	5.10	2.20	2.60	**2.10**	4.40	6.95	7.75	7.65	2.60	5.00	6.65	3.60
icd-cl-abs-cb	5.40	**1.60**	**1.30**	4.90	4.70	3.80	2.10	2.00	5.30	7.25	7.40	7.65	2.60	5.60	7.20	3.20
icd-ins-enc-abs	5.10	**1.20**	1.80	3.90	4.60	3.50	**1.20**	2.40	5.85	7.15	7.70	7.65	3.00	6.20	6.75	4.00
icd-ins-enc-cb	5.40	1.70	2.50	5.00	4.80	2.90	2.20	**1.60**	4.85	7.30	7.75	7.60	**1.90**	4.80	6.60	5.10

Table 14 (continued)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
icd-ins-abs-cb	*5.60*	**1.10**	2.00	*3.70*	*4.60*	*4.10*	**1.20**	2.20	*5.35*	*6.70*	*7.45*	*7.70*	2.80	*6.60*	*7.00*	*3.90*
icd-enc-abs-cb	*5.60*	**1.20**	2.30	3.60	*4.40*	*3.80*	**1.60**	2.10	*5.55*	*7.10*	*7.70*	*7.40*	2.10	*6.30*	*6.75*	*4.50*
erp-gcr-cs-cl	3.15	4.35	**1.70**	4.35	*4.60*	2.75	*4.55*	*7.20*	*5.05*	5.25	*7.65*	5.15	2.80	4.25	*6.50*	**2.70**
erp-gcr-cs-ins	*4.25*	**2.40**	**1.45**	*5.30*	*4.85*	3.60	3.50	4.10	*5.65*	*6.40*	*7.55*	*6.50*	2.45	4.40	*6.30*	3.30
erp-gcr-cs-enc	*4.10*	**2.00**	**1.00**	*5.50*	*3.90*	3.60	*4.80*	4.10	*6.30*	*6.80*	*7.30*	*7.00*	2.50	4.10	*6.10*	2.90
erp-gcr-cs-abs	*3.50*	**2.10**	**1.00**	*5.00*	*4.20*	2.20	*3.50*	*5.30*	*5.95*	*6.30*	*7.65*	*7.20*	*3.70*	3.10	*6.50*	*4.80*
erp-gcr-cs-cb	*3.70*	2.30	**1.20**	*5.10*	*4.60*	**2.10**	2.80	4.00	*4.60*	*6.60*	*7.80*	*7.50*	*4.70*	*5.60*	*6.60*	2.80
erp-gcr-cl-ins	4.15	*4.40*	**2.25**	*4.75*	3.20	**1.70**	2.75	*4.85*	*5.60*	*6.25*	*7.60*	*6.20*	3.85	*4.25*	*6.60*	3.60
erp-gcr-cl-enc	*5.35*	*6.55*	**1.10**	*4.80*	3.30	1.80	*5.65*	*5.50*	*5.25*	*5.95*	*7.10*	*5.80*	2.25	3.80	*6.00*	**1.80**
erp-gcr-cl-abs	*4.10*	4.75	**1.35**	**2.70**	*4.00*	2.80	3.05	*5.80*	*6.05*	*5.70*	*7.55*	*6.20*	3.40	3.55	*6.50*	4.50
erp-gcr-cl-cb	3.40	3.00	**1.10**	4.90	*5.30*	2.90	3.50	**2.70**	*5.60*	*6.80*	*7.80*	*7.40*	2.70	5.40	*6.60*	2.90

Table 15 Average rankings for 4-objective problems obtained from the Friedman Test ($\alpha = 0.05$) (cont'd)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
erp-gcr-ins-enc	3.50	**2.40**	**1.20**	*5.90*	*4.00*	3.80	2.00	2.70	*6.60*	*6.30*	*7.60*	*7.50*	*4.50*	3.90	*6.60*	3.50
erp-gcr-ins-abs	3.40	**1.90**	1.90	*4.90*	*4.40*	2.20	**1.30**	4.50	*6.30*	*5.70*	*7.60*	*7.60*	*4.50*	2.80	*6.60*	*6.40*
erp-gcr-ins-cb	*4.20*	**1.60**	**1.10**	*5.20*	*4.20*	3.10	2.30	2.40	*5.70*	*6.45*	*7.80*	*7.65*	*4.10*	*6.10*	*6.60*	3.50
erp-gcr-enc-abs	*4.50*	3.00	**1.00**	4.60	2.90	**3.00**	*3.70*	3.50	*6.30*	*6.60*	*7.40*	*7.00*	*3.80*	4.00	*6.40*	4.30
erp-gcr-enc-cb	*4.10*	**1.20**	**1.00**	*5.30*	*4.00*	3.60	2.40	1.90	*6.60*	*6.70*	*7.80*	*7.60*	*3.50*	*5.90*	*6.60*	*3.80*
erp-gcr-abs-cb	*4.60*	**1.10**	**1.10**	*4.40*	*4.70*	3.20	2.10	2.50	*5.80*	*7.20*	*7.80*	*7.60*	3.30	*5.10*	*6.60*	*4.90*
erp-cs-cl-ins	2.85	3.40	**1.55**	*5.10*	4.10	**1.60**	*4.25*	*6.00*	*5.50*	*5.85*	*7.60*	*6.85*	3.55	4.20	*6.60*	3.00
erp-cs-cl-enc	*5.35*	*6.55*	**1.10**	*4.80*	3.30	1.80	*5.65*	*5.50*	*5.25*	*5.95*	*7.10*	*5.80*	2.25	3.80	*6.00*	**1.80**
erp-cs-cl-abs	3.40	**2.10**	**1.00**	4.60	*4.50*	**2.10**	*4.90*	*5.15*	*5.35*	*6.15*	*7.55*	*7.30*	2.80	3.50	*6.50*	*5.10*
erp-cs-cl-cb	*4.70*	2.60	**1.50**	*5.10*	*4.70*	**1.60**	3.80	*4.30*	3.90	*6.80*	*7.80*	*7.70*	3.00	*5.10*	*6.60*	2.80
erp-cs-ins-enc	*4.95*	**1.60**	**1.10**	*4.70*	*4.75*	3.90	3.15	2.00	*5.85*	*6.90*	*7.30*	*7.60*	2.60	*5.40*	*6.30*	3.90
erp-cs-ins-abs	*4.80*	**1.50**	**1.10**	*4.30*	*4.40*	2.50	2.10	3.30	*5.20*	*7.10*	*7.60*	*7.60*	*4.20*	4.00	*6.60*	*5.70*
erp-cs-ins-cb	*5.50*	**1.70**	**1.40**	*5.10*	*4.60*	3.20	2.00	2.20	*4.70*	*6.70*	*7.80*	*7.70*	3.40	*6.50*	*6.60*	2.90
erp-cs-enc-abs	*5.40*	**1.00**	**1.00**	3.00	2.60	*3.90*	*4.00*	*3.70*	*5.90*	*6.40*	*7.60*	*7.40*	3.00	*5.80*	*6.50*	*4.80*
erp-cs-enc-cb	*5.40*	**1.30**	**1.00**	3.90	*4.00*	*4.00*	*3.60*	2.60	*4.60*	*7.10*	*7.80*	*7.60*	3.10	*6.30*	*6.50*	3.20
erp-cs-abs-cb	*5.30*	**1.30**	**1.10**	3.10	*4.50*	3.50	2.50	2.30	*5.50*	*7.15*	*7.80*	*7.65*	2.70	*6.10*	*6.60*	*4.90*
erp-cl-ins-enc	*3.60*	2.80	**1.10**	*6.00*	*3.60*	3.30	2.50	**1.20**	*6.35*	*7.15*	*7.65*	*7.55*	*4.60*	*4.00*	*6.60*	*4.00*
erp-cl-ins-abs	3.70	2.20	2.00	4.10	*4.40*	2.40	**1.50**	4.20	*5.65*	*6.90*	*7.60*	*7.60*	*4.50*	**2.20**	*6.65*	*6.40*
erp-cl-ins-cb	*4.90*	**1.60**	**1.30**	*5.20*	*4.90*	3.10	2.20	1.70	*5.50*	*6.60*	*7.70*	*7.80*	2.80	*5.00*	*6.70*	*5.00*
erp-cl-enc-abs	*4.60*	3.20	**1.00**	4.40	2.80	**2.90**	*4.75*	4.00	*5.95*	*6.50*	*7.60*	*7.10*	2.90	3.60	*6.40*	4.30
erp-cl-enc-cb	*5.10*	**1.10**	**1.00**	*5.10*	*4.20*	3.20	2.90	2.40	*5.95*	*7.00*	*7.75*	*7.60*	2.50	*4.70*	*6.60*	*4.90*
erp-cl-abs-cb	*4.90*	**1.20**	**1.10**	*4.50*	*4.80*	2.80	2.60	2.40	*5.90*	*6.95*	*7.80*	*7.65*	2.30	*4.80*	*6.60*	*5.70*
erp-ins-enc-abs	*4.60*	**1.30**	1.50	3.20	3.80	3.40	**1.50**	3.00	*6.35*	*7.10*	*7.75*	*7.60*	3.90	*5.40*	*6.60*	*5.00*

Table 15 (continued)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
erp-ins-enc-cb	*5.60*	**1.10**	**1.50**	*3.60*	*4.40*	*4.30*	1.50	2.10	*5.10*	*7.10*	*7.80*	*7.60*	3.50	*5.70*	*6.60*	*4.50*
erp-ins-abs-cb	*5.60*	**1.00**	1.60	3.30	*4.40*	*3.70*	**1.50**	2.60	*5.30*	*7.30*	*7.80*	*7.60*	3.20	*5.70*	*6.60*	*4.80*
erp-enc-abs-cb	*5.30*	**1.00**	**1.50**	2.80	4.10	*4.40*	2.40	2.90	*5.90*	*6.70*	*7.75*	*7.60*	2.40	*5.20*	*6.65*	*5.40*
gcr-cs-cl-ins	2.75	2.90	**1.65**	*5.55*	*4.75*	**1.70**	*4.70*	*5.85*	*4.45*	*5.95*	*7.75*	*7.25*	3.35	4.20	*6.60*	2.60
gcr-cs-cl-enc	*4.95*	*4.60*	**1.00**	*4.60*	2.90	2.60	*6.85*	*6.90*	*4.90*	*6.00*	*7.00*	*6.30*	2.70	3.40	*5.70*	**1.60**
gcr-cs-cl-abs	*4.30*	3.25	**1.15**	*4.40*	*5.10*	**1.30**	*4.00*	*5.90*	*4.85*	*6.65*	*7.65*	*7.10*	2.35	3.10	*6.60*	*4.30*
gcr-cs-cl-cb	*4.90*	**2.10**	**1.50**	*5.00*	*5.00*	2.50	3.50	3.50	*4.15*	*5.95*	*7.60*	*7.65*	2.50	*4.70*	*6.85*	4.60
gcr-cs-ins-enc	*5.20*	1.70	**1.00**	*4.50*	*4.80*	*4.70*	*3.70*	**1.40**	*4.85*	*6.80*	*7.55*	*7.60*	2.60	*5.90*	*6.30*	3.40
gcr-cs-ins-abs	*4.70*	**1.30**	**1.30**	*4.50*	*5.20*	2.40	2.30	3.00	*5.05*	*6.90*	*7.75*	*7.60*	3.10	*4.50*	*6.60*	*5.80*
gcr-cs-ins-cb	*5.40*	**1.70**	**1.20**	*5.00*	*4.60*	4.30	2.40	2.10	*4.60*	*6.70*	*7.80*	*7.80*	3.40	*6.20*	*6.60*	2.20
gcr-cs-cl-ins	2.75	2.90	**1.65**	*5.55*	*4.75*	**1.70**	*4.70*	*5.85*	*4.45*	*5.95*	*7.75*	*7.25*	3.35	4.20	*6.60*	2.60
gcr-cs-cl-enc	*4.95*	*4.60*	**1.00**	*4.60*	2.90	2.60	*6.85*	*6.90*	*4.90*	*6.00*	*7.00*	*6.30*	2.70	3.40	*5.70*	**1.60**
gcr-cs-cl-abs	*4.30*	3.25	**1.15**	*4.40*	*5.10*	**1.30**	*4.00*	*5.90*	*4.85*	*6.65*	*7.65*	*7.10*	2.35	3.10	*6.60*	*4.30*
gcr-cs-cl-cb	*4.90*	**2.10**	**1.50**	*5.00*	*5.00*	2.50	3.50	3.50	*4.15*	*5.95*	*7.60*	*7.65*	2.50	*4.70*	*6.85*	4.60
gcr-cs-ins-enc	*5.20*	1.70	**1.00**	*4.50*	*4.80*	*4.70*	*3.70*	**1.40**	*4.85*	*6.80*	*7.55*	*7.60*	2.60	*5.90*	*6.30*	3.40
gcr-cs-ins-abs	*4.70*	**1.30**	**1.30**	*4.50*	*5.20*	2.40	2.30	3.00	*5.05*	*6.90*	*7.75*	*7.60*	3.10	*4.50*	*6.60*	*5.80*
gcr-cs-ins-cb	*5.40*	**1.70**	**1.20**	*5.00*	*4.60*	4.30	2.40	2.10	*4.60*	*6.70*	*7.80*	*7.80*	3.40	*6.20*	*6.60*	2.20

Table 16 Average rankings for 4-objective problems obtained from the Friedman test ($\alpha = 0.05$)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
gcr-cs-enc-abs	*5.75*	3.05	**1.00**	**2.40**	2.80	4.00	*3.65*	3.35	*5.80*	*7.00*	*7.50*	*7.30*	3.10	5.20	*6.40*	3.70
gcr-cs-enc-cb	*5.50*	**1.50**	**1.20**	*4.10*	*4.40*	*4.00*	*3.90*	3.85	*4.25*	*6.85*	*7.75*	*7.50*	2.60	*4.90*	*6.40*	3.30
gcr-cs-abs-cb	*5.20*	**1.30**	**1.00**	2.90	*4.00*	3.90	2.60	2.20	*5.40*	*7.30*	*7.70*	*7.60*	3.30	*5.90*	*6.80*	*4.90*
gcr-cl-ins-enc	*4.50*	2.20	**1.00**	*6.20*	3.10	4.00	2.40	**1.60**	*6.00*	*6.70*	*7.60*	*7.60*	*4.80*	*4.70*	*6.60*	3.00
gcr-cl-ins-abs	*4.30*	**2.40**	2.00	4.20	*4.80*	2.60	**1.20**	3.00	*5.75*	*6.85*	*7.65*	*7.65*	*3.70*	3.10	*6.60*	*6.20*
gcr-cl-ins-cb	*4.30*	2.40	**1.50**	*5.70*	*5.10*	*4.00*	2.70	**1.30**	*4.60*	*6.90*	*7.80*	*7.80*	3.40	*5.20*	*6.60*	2.70
gcr-cl-enc-abs	*5.60*	3.50	**1.00**	4.90	2.70	3.80	*4.10*	**2.90**	*6.10*	*6.20*	*7.20*	*7.20*	3.10	3.10	*6.20*	4.40
gcr-cl-enc-cb	*5.10*	**2.40**	**1.10**	5.00	*4.60*	3.40	3.20	4.10	*4.30*	*6.90*	*7.80*	*7.60*	3.30	3.80	*6.60*	2.80
gcr-cl-abs-cb	*4.60*	**1.70**	**1.00**	4.00	*5.20*	2.60	3.00	2.60	*5.55*	*7.00*	*7.75*	*7.50*	2.20	*5.00*	*6.70*	*5.60*
gcr-ins-enc-abs	*5.20*	**1.10**	1.70	2.90	3.70	3.70	**1.30**	2.80	*6.40*	*7.20*	*7.60*	*7.60*	3.50	*6.10*	*6.60*	*4.60*
gcr-ins-enc-cb	*5.80*	**1.10**	**1.20**	*3.90*	*4.60*	*4.10*	2.20	2.10	*4.80*	*7.10*	*7.80*	*7.60*	3.20	*6.30*	*6.40*	*3.80*
gcr-ins-abs-cb	*5.60*	**1.10**	**1.30**	3.30	*4.40*	3.50	1.80	2.40	*5.40*	*7.70*	*6.95*	*7.70*	3.00	*6.00*	*6.80*	*5.10*
gcr-enc-abs-cb	*5.50*	**1.75**	**1.10**	2.50	*3.90*	4.20	3.25	3.25	*5.60*	*7.20*	*7.75*	*7.40*	2.40	*5.50*	*6.50*	4.20
cs-cl-ins-enc	*5.20*	**1.80**	**1.70**	*5.30*	4.10	2.90	3.00	3.05	4.30	*6.40*	*7.60*	*7.45*	3.50	*4.60*	*6.60*	*4.50*
cs-cl-ins-abs	3.40	*3.90*	**1.40**	*4.80*	*5.60*	**1.30**	3.50	*5.25*	*4.80*	*5.85*	*7.70*	*7.60*	2.80	*4.50*	*6.80*	2.80
cs-cl-ins-cb	3.20	**2.20**	1.70	4.50	*4.90*	2.60	**1.50**	3.30	*5.60*	*7.00*	*7.60*	*7.60*	*4.60*	3.80	*6.90*	5.00
cs-cl-enc-abs	*5.60*	*5.00*	**1.00**	3.60	2.60	2.50	*4.60*	*5.95*	*5.00*	*5.95*	*7.30*	*6.90*	*3.90*	4.30	*6.00*	**1.80**
cs-cl-enc-cb	*4.50*	*4.65*	**1.50**	4.40	*4.70*	**1.80**	*4.00*	*7.15*	4.30	*5.75*	*7.60*	*7.15*	3.20	2.90	*6.20*	2.20
cs-cl-abs-cb	*5.00*	3.40	**1.00**	4.30	*4.40*	2.40	*3.85*	4.30	*5.25*	*6.60*	*7.60*	*7.70*	2.10	*5.30*	*6.80*	**2.00**
cs-ins-enc-abs	*5.60*	**1.30**	1.70	3.10	4.00	*4.30*	**1.60**	1.90	*6.00*	*6.75*	*7.70*	*7.75*	2.80	*6.50*	*6.60*	*4.40*
cs-ins-enc-cb	*5.60*	1.90	**1.50**	*5.10*	*4.20*	3.50	2.30	**1.40**	*4.70*	*6.80*	*7.80*	*7.80*	3.30	*5.60*	*6.60*	3.90
cs-ins-abs-cb	*5.50*	**1.60**	**1.20**	*4.50*	*4.60*	*4.40*	1.90	2.70	*5.45*	*7.00*	*7.75*	*7.80*	2.90	*6.20*	*6.70*	1.80

Table 16 (continued)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
cs-enc-abs-cb	*5.50*	**1.50**	**1.00**	2.20	3.40	3.70	*4.50*	*5.05*	*5.25*	*6.50*	*7.75*	*7.55*	2.10	*6.10*	*6.50*	3.40
cl-ins-enc-abs	3.80	1.70	1.60	*4.90*	3.40	3.40	**1.40**	**1.50**	*6.20*	*7.45*	*7.75*	*6.85*	*5.20*	*5.20*	*6.65*	*5.00*
cl-ins-enc-cb	*4.40*	2.20	**1.40**	*4.80*	*5.10*	3.30	1.80	**1.20**	*5.00*	*6.85*	*7.65*	*7.75*	*3.90*	*4.20*	*6.75*	*5.70*
cl-ins-abs-cb	*5.50*	1.90	**1.10**	*4.70*	*4.60*	3.20	2.30	**1.30**	*5.30*	*7.00*	*7.65*	*7.80*	*2.70*	*5.60*	2.70	*4.50*
cl-enc-abs-cb	*5.20*	**1.70**	**1.00**	3.00	*3.40*	2.90	*4.00*	*5.85*	*5.10*	*6.75*	*7.65*	*7.30*	*3.20*	*4.60*	*6.45*	3.90
ins-enc-abs-cb	*5.60*	**1.20**	1.80	3.10	*4.40*	*4.20*	**1.40**	2.00	*5.60*	*6.45*	*7.80*	*7.75*	2.80	*6.40*	*6.60*	*4.90*

Table 17 Average rankings for 6-objective problems obtained from the Friedman Test ($\alpha = 0.05$) (cont'd)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
icd-erp-gcr-cs-cl-ins	*4.50*	**1.70**	**1.00**	*5.00*	*4.20*	3.50	2.10	2.50	*6.15*	*7.30*	*7.75*	*7.60*	*3.70*	*5.40*	*6.60*	3.00
icd-erp-gcr-cs-cl-enc	*5.60*	**1.30**	**1.00**	*4.90*	3.60	*4.70*	3.30	2.20	*5.85*	*7.30*	*7.75*	*7.50*	2.30	*5.10*	*6.60*	3.00
icd-erp-gcr-cs-cl-abs	*5.60*	**1.00**	1.60	3.20	3.60	*5.10*	**1.40**	3.10	*6.00*	*7.40*	*7.80*	*7.60*	3.40	*4.60*	*6.60*	*4.00*
icd-erp-gcr-cs-cl-cb	*5.50*	**1.10**	**1.40**	*4.50*	*4.60*	*4.40*	1.70	2.40	*5.40*	*7.40*	*7.80*	*7.60*	3.00	*5.70*	*6.60*	2.90
icd-erp-gcr-cs-ins-enc	*5.50*	**1.10**	1.80	3.00	3.60	*5.30*	**1.50**	3.10	*5.50*	*7.10*	*7.70*	*7.60*	3.80	*5.00*	*6.60*	*3.80*
icd-erp-gcr-cs-ins-abs	*5.60*	**1.00**	2.10	2.20	3.50	*5.00*	**1.10**	3.30	*5.90*	*7.20*	*7.80*	*7.60*	3.40	*6.10*	*6.60*	3.60
icd-erp-gcr-cs-ins-cb	*5.80*	**1.10**	1.70	3.10	3.70	*5.00*	**1.40**	3.20	*5.25*	*7.10*	*7.75*	*7.50*	3.90	*6.00*	*6.50*	3.00
icd-erp-gcr-cs-enc-abs	*5.60*	**1.00**	2.80	2.30	3.20	*5.10*	**1.00**	*3.80*	*6.10*	*7.40*	*7.70*	*7.60*	3.00	*5.10*	*6.60*	*3.70*
icd-erp-gcr-cs-enc-cb	*5.90*	**1.00**	**1.70**	2.80	3.80	*5.20*	**1.70**	3.20	*5.40*	*7.20*	*7.80*	*7.60*	3.40	*5.30*	*6.30*	*3.70*
icd-erp-gcr-cs-abs-cb	*6.10*	**1.00**	2.10	2.00	*4.10*	*5.20*	**1.30**	3.40	*5.70*	*7.10*	*7.80*	*7.60*	2.80	*5.90*	*6.10*	*3.80*
icd-erp-gcr-cl-ins-enc	*4.20*	**1.70**	**1.30**	*5.10*	*3.90*	*4.40*	1.70	2.80	*6.60*	*7.40*	*7.60*	*7.60*	*4.10*	4.20	*6.60*	2.80
icd-erp-gcr-cl-ins-abs	*4.80*	**1.00**	1.70	*3.30*	3.80	*5.00*	**1.30**	*4.30*	*6.70*	*7.40*	*7.60*	*7.60*	3.50	*3.20*	*6.60*	*4.20*
icd-erp-gcr-cl-ins-cb	*5.50*	**1.10**	1.60	*4.00*	4.10	*4.90*	**1.50**	3.10	*5.80*	*7.40*	*7.80*	*7.60*	3.10	*5.40*	*6.60*	2.50
icd-erp-gcr-cl-enc-abs	*5.40*	**1.00**	2.00	*3.30*	3.40	*4.70*	**1.10**	*4.90*	*6.40*	*7.40*	*7.60*	*7.60*	3.50	*3.30*	*6.60*	*3.80*
icd-erp-gcr-cl-enc-cb	*5.60*	**1.00**	**1.50**	*4.20*	4.00	*4.80*	1.60	3.10	*5.80*	*7.30*	*7.80*	*7.60*	3.10	*5.30*	*6.60*	2.70
icd-erp-gcr-cl-abs-cb	*5.90*	**1.00**	1.70	2.40	*4.20*	*5.20*	**1.30**	*3.50*	*5.75*	*7.40*	*7.75*	*7.60*	3.00	*5.10*	*6.40*	*3.80*
icd-erp-gcr-ins-enc-abs	*5.50*	**1.00**	2.60	2.20	3.00	*4.00*	**1.00**	*4.80*	*6.25*	*7.30*	*7.65*	*7.60*	3.40	5.30	*6.60*	*3.80*
icd-erp-gcr-ins-enc-cb	*5.90*	**1.10**	1.90	2.50	3.80	*5.10*	**1.30**	*4.10*	*5.70*	*7.10*	*7.80*	*7.60*	3.30	*5.00*	*6.30*	3.50
icd-erp-gcr-ins-abs-cb	*5.80*	**1.40**	2.40	2.00	*3.80*	*4.80*	**1.10**	3.60	*5.80*	*7.20*	*7.80*	*7.60*	2.90	*5.70*	*6.40*	3.70
icd-erp-gcr-enc-abs-cb	*6.00*	**1.00**	2.50	2.00	*4.00*	*5.00*	**1.10**	*4.20*	*6.00*	*7.20*	*7.80*	*7.60*	2.40	*5.50*	*6.20*	*3.50*
icd-erp-cs-cl-ins-enc	*5.60*	**1.00**	**1.50**	*3.70*	3.90	*5.70*	1.70	2.20	*5.20*	*7.40*	*7.70*	*7.60*	3.80	*4.70*	*6.60*	*3.70*
icd-erp-cs-cl-ins-abs	*5.60*	**1.00**	2.10	2.30	3.70	*5.60*	**1.10**	2.90	*6.05*	*7.40*	*7.65*	*7.60*	3.20	*4.60*	*6.60*	*4.60*

Table 17 (continued)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
icd-erp-cs-cl-ins-cb	*5.60*	**1.10**	**1.70**	3.20	*4.60*	*5.10*	1.70	3.00	*4.90*	*7.40*	*7.80*	*7.60*	3.20	*5.20*	*6.50*	3.40
icd-erp-cs-cl-enc-abs	*5.60*	**1.00**	2.00	2.40	3.00	*6.00*	**1.20**	*3.90*	*6.10*	*7.40*	*7.70*	*7.60*	3.80	*3.70*	*6.60*	*4.00*
icd-erp-cs-cl-enc-cb	*5.80*	**1.00**	**1.50**	3.60	*4.40*	*5.40*	1.60	2.70	*5.10*	*7.40*	*7.80*	*7.60*	3.40	*5.30*	*6.40*	3.00
icd-erp-cs-cl-abs-cb	*5.90*	**1.00**	1.80	2.20	*4.20*	*5.40*	**1.40**	3.20	*5.80*	*7.40*	*7.80*	*7.60*	2.80	*5.20*	*6.30*	*4.00*
icd-erp-cs-ins-enc-abs	*5.60*	**1.00**	3.40	2.00	2.90	*5.10*	**1.00**	*3.80*	*6.10*	*7.40*	*7.70*	*7.60*	2.70	*5.70*	*6.60*	3.40
icd-erp-cs-ins-enc-cb	*5.90*	**1.00**	2.00	2.40	*4.30*	*5.00*	**1.30**	3.20	*5.20*	*7.00*	*7.80*	*7.60*	3.20	*6.30*	*6.30*	3.50
icd-erp-cs-ins-abs-cb	*6.10*	**1.00**	2.70	2.10	*4.10*	*4.90*	**1.10**	3.40	*5.90*	*7.10*	*7.80*	*7.60*	2.20	*6.30*	*6.10*	*3.60*
icd-erp-cs-enc-abs-cb	*6.20*	**1.00**	2.60	2.00	*4.00*	*5.20*	**1.20**	*3.90*	*6.00*	*7.10*	*7.80*	*7.60*	2.20	*5.90*	*6.00*	3.30

Table 18 Average rankings for 6-objective problems obtained from the Friedman Test ($\alpha = 0.05$) (cont'd)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
icd-erp-cl-ins-enc-abs	*5.50*	**1.00**	2.00	2.50	3.30	*5.50*	**1.10**	*4.30*	*6.15*	*7.40*	*7.75*	*7.60*	3.60	3.20	*6.60*	*4.50*
icd-erp-cl-ins-enc-cb	*5.60*	**1.00**	1.60	2.50	*4.30*	*5.40*	**1.40**	2.90	*5.20*	*7.40*	*7.80*	*7.60*	3.50	*5.20*	*6.60*	*4.00*
icd-erp-cl-ins-abs-cb	*5.60*	**1.00**	2.00	2.00	*4.00*	*5.80*	**1.00**	*3.50*	*6.00*	*7.40*	*7.80*	*7.60*	3.00	*4.40*	*6.60*	*4.30*
icd-erp-cl-enc-abs-cb	*5.80*	**1.10**	1.90	1.90	3.60	*5.80*	**1.10**	*4.40*	*6.00*	*7.40*	*7.80*	*7.60*	3.40	*4.00*	*6.40*	*3.80*
icd-erp-ins-enc-abs-cb	*6.00*	**1.00**	2.90	2.00	*3.80*	*4.90*	**1.00**	*4.10*	*6.00*	*7.20*	*7.80*	*7.60*	2.30	*6.10*	*6.20*	3.10
icd-gcr-cs-cl-ins-enc	*5.60*	**1.40**	**1.10**	3.60	*4.10*	*5.10*	2.40	2.30	*4.95*	*7.40*	*7.75*	*7.60*	3.50	*5.00*	*6.60*	3.60
icd-gcr-cs-cl-ins-abs	*5.60*	**1.10**	1.80	2.60	*4.20*	*5.20*	**1.20**	3.30	*5.65*	*7.40*	*7.75*	*7.60*	3.20	*4.70*	*6.60*	*4.10*
icd-gcr-cs-cl-ins-cb	*5.60*	**1.00**	**1.60**	3.50	*4.60*	*5.10*	1.80	2.80	*5.00*	*7.30*	*7.80*	*7.70*	3.00	*5.30*	*6.60*	3.30
icd-gcr-cs-cl-enc-abs	*5.60*	**1.00**	2.10	2.40	3.30	*5.70*	**1.00**	*4.20*	*6.10*	*7.40*	*7.70*	*7.60*	3.60	*3.90*	*6.60*	*3.80*
icd-gcr-cs-cl-enc-cb	*5.60*	**1.00**	**1.20**	*4.20*	*4.40*	*5.00*	2.20	2.30	*4.80*	*7.30*	*7.80*	*7.60*	3.40	*5.00*	*6.60*	*3.60*
icd-gcr-cs-cl-abs-cb	*5.60*	**1.00**	1.70	2.50	*4.30*	*5.40*	**1.60**	3.40	*5.70*	*7.40*	*7.80*	*7.60*	2.70	*4.80*	*6.60*	*3.90*
icd-gcr-cs-ins-enc-abs	*5.60*	**1.00**	3.20	2.00	3.40	*4.90*	**1.00**	*3.60*	*6.10*	*7.30*	*7.70*	*7.50*	2.40	*6.20*	*6.60*	*3.50*
icd-gcr-cs-ins-enc-cb	*5.60*	**1.00**	2.30	2.60	4.40	*4.90*	**2.10**	2.70	*5.10*	*6.90*	*7.80*	*7.60*	2.10	*6.10*	*6.60*	*4.20*
icd-gcr-cs-ins-abs-cb	*5.70*	**1.00**	2.60	2.30	*4.10*	*4.90*	**1.10**	3.30	*5.90*	*7.10*	*7.80*	*7.60*	2.30	*6.30*	*6.50*	3.50
icd-gcr-cs-enc-abs-cb	*5.80*	**1.20**	2.50	1.80	4.00	*4.90*	1.80	3.60	*6.00*	*7.10*	*7.80*	*7.60*	**1.70**	*6.30*	*6.40*	3.50
icd-gcr-cl-ins-enc-abs	*5.60*	**1.10**	2.00	2.20	3.40	*5.20*	**1.10**	*4.90*	*6.15*	*7.40*	*7.65*	*7.60*	3.50	3.20	*6.60*	*4.40*
icd-gcr-cl-ins-enc-cb	*5.60*	**1.00**	**1.50**	3.20	*4.20*	*5.40*	1.60	3.00	*5.00*	*7.40*	*7.80*	*7.60*	3.70	*5.00*	*6.60*	3.40
icd-gcr-cl-ins-abs-cb	*5.60*	**1.00**	1.80	2.20	*4.10*	*5.60*	**1.20**	3.30	*5.80*	*7.40*	*7.80*	*7.60*	3.10	*4.60*	*6.60*	*4.30*
icd-gcr-cl-enc-abs-cb	*5.70*	**1.00**	1.60	2.20	3.70	*5.70*	**1.40**	*4.40*	*6.00*	*7.40*	*7.80*	*7.60*	3.30	*4.00*	*6.50*	*3.70*
icd-gcr-ins-enc-abs-cb	*5.70*	**1.00**	3.10	2.00	3.90	*4.70*	**1.40**	*3.80*	*6.00*	*7.00*	*7.80*	*7.60*	1.60	*6.40*	*6.50*	*3.50*
icd-cs-cl-ins-enc-abs	*5.60*	**1.10**	2.00	2.70	*3.60*	*5.90*	**1.00**	3.20	*6.05*	*7.40*	*7.75*	*7.60*	3.40	*4.20*	*6.60*	*3.90*
icd-cs-cl-ins-enc-cb	*5.60*	1.80	2.20	*4.30*	*4.60*	*3.90*	**1.90**	**1.20**	*4.80*	*7.40*	*7.80*	*7.60*	2.50	*5.70*	*6.60*	*4.10*
icd-cs-cl-ins-abs-cb	*5.60*	**1.20**	2.30	2.60	*4.20*	*5.50*	**1.10**	3.20	*5.55*	*7.30*	*7.75*	*7.70*	2.80	*5.50*	*6.70*	3.00

Table 18 (continued)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
icd-cs-cl-enc-abs-cb	*5.60*	**1.00**	1.90	2.70	*4.00*	*5.60*	**1.30**	2.60	*5.90*	*7.40*	*7.80*	*7.60*	2.90	*5.00*	*6.60*	*4.10*
icd-cs-ins-enc-abs-cb	*5.60*	**1.00**	3.10	2.00	*4.00*	*4.70*	**1.30**	3.20	*5.90*	*6.90*	*7.80*	*7.40*	1.70	*6.70*	*6.60*	*4.10*
icd-cl-ins-enc-abs-cb	*5.60*	**1.40**	2.00	1.80	*3.90*	*5.90*	**1.00**	3.40	*5.80*	*7.40*	*7.75*	*7.60*	3.30	*4.60*	*6.65*	3.90
erp-gcr-cs-cl-ins-enc	*5.50*	**1.30**	**1.10**	*4.50*	3.20	*5.50*	2.20	2.60	*6.15*	*7.30*	*7.65*	*7.60*	3.60	*4.90*	*6.60*	2.30
erp-gcr-cs-cl-ins-abs	*5.50*	**1.00**	**1.30**	*4.80*	3.70	*3.80*	1.90	*3.60*	*6.00*	*7.30*	*7.80*	*7.60*	3.20	*3.90*	*6.60*	*4.00*
erp-gcr-cs-cl-ins-cb	*5.50*	**1.60**	**1.60**	4.00	4.20	*5.10*	1.70	2.80	*5.80*	*7.35*	*7.80*	*7.65*	2.80	*5.50*	*6.60*	2.00
erp-gcr-cs-cl-enc-abs	*5.30*	**1.00**	**1.00**	*3.30*	2.60	*5.90*	3.60	*3.70*	*6.25*	*7.10*	*7.55*	*7.50*	3.20	*4.20*	*6.50*	*3.30*
erp-gcr-cs-cl-enc-cb	*5.60*	**1.30**	**1.00**	2.40	*3.80*	*5.10*	3.00	3.40	*5.80*	*7.40*	*7.80*	*7.60*	2.50	*5.00*	*6.50*	2.40

Table 19 Average rankings for 6-, 8- and 9-objective problems obtained from the Friedman Test ($\alpha = 0.05$)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
erp-gcr-cs-cl-abs-cb	*5.50*	**1.30**	**1.30**	2.70	*4.10*	*5.10*	2.80	2.20	*5.50*	*7.40*	*7.80*	*7.60*	2.40	*5.50*	*6.60*	*4.20*
erp-gcr-cs-ins-enc-abs	*5.70*	**1.00**	2.60	2.10	2.80	*5.10*	**1.00**	*3.70*	*6.05*	*7.40*	*7.65*	*7.60*	*3.70*	*5.60*	*6.50*	*3.50*
erp-gcr-cs-ins-enc-cb	*6.10*	**1.10**	1.90	2.80	3.30	*5.10*	**1.50**	3.40	*5.50*	*7.20*	*7.80*	*7.60*	3.80	*5.80*	*6.10*	3.00
erp-gcr-cs-ins-abs-cb	*5.80*	**1.00**	2.00	2.30	3.50	*5.10*	**1.40**	3.20	*5.60*	*7.30*	*7.80*	*7.60*	3.50	*5.80*	*6.40*	*3.70*
erp-gcr-cs-enc-abs-cb	*5.70*	**1.10**	**1.50**	2.20	3.00	*5.20*	2.70	3.20	*5.90*	*7.10*	*7.80*	*7.60*	2.90	*5.90*	*6.50*	3.70
erp-gcr-cl-ins-enc-abs	*5.10*	**1.00**	2.00	*3.30*	3.20	*5.10*	**1.00**	*5.10*	*6.70*	*6.30*	*7.60*	*7.60*	*3.80*	*4.00*	*6.60*	*3.60*
erp-gcr-cl-ins-enc-cb	*5.70*	**1.10**	1.60	3.60	3.80	*5.40*	**1.50**	3.20	*5.80*	*7.40*	*7.80*	*7.60*	3.30	*5.30*	*6.50*	2.40
erp-gcr-cl-ins-abs-cb	*5.70*	**1.10**	**1.60**	2.60	4.10	*5.50*	1.60	3.30	*5.90*	*7.30*	*7.80*	*7.60*	2.80	*5.40*	*6.50*	3.20
erp-gcr-cl-enc-abs-cb	*5.50*	**1.00**	**1.30**	2.30	3.50	*5.90*	2.20	*3.60*	*6.10*	*7.40*	*7.80*	*7.60*	3.00	*4.80*	*6.60*	3.40
erp-gcr-ins-enc-abs-cb	*6.10*	**1.10**	2.30	1.90	3.60	*5.30*	**1.10**	*4.00*	*6.00*	*7.30*	*7.80*	*7.60*	3.00	*5.50*	*6.10*	3.30
erp-cs-cl-ins-enc-abs	*5.60*	**1.00**	2.20	2.30	2.90	*6.00*	**1.00**	*4.40*	*6.15*	*7.40*	*7.65*	*7.60*	*3.90*	*3.90*	*6.60*	3.40
erp-cs-cl-ins-enc-cb	*5.90*	**1.10**	1.60	3.10	*4.10*	*5.50*	**1.40**	3.00	*5.20*	*7.35*	*7.80*	*7.65*	3.70	*5.40*	*6.30*	2.90
erp-cs-cl-ins-abs-cb	*5.70*	**1.10**	1.80	2.40	4.10	*5.50*	**1.70**	3.20	*5.80*	*7.40*	*7.80*	*7.60*	2.60	*5.00*	*6.50*	*3.80*
erp-cs-cl-enc-abs-cb	*5.60*	**1.10**	**1.30**	2.00	3.00	*5.90*	2.50	3.70	*6.00*	*7.40*	*7.80*	*7.60*	3.20	*4.90*	*6.60*	3.40
erp-cs-ins-enc-abs-cb	*6.20*	**1.00**	2.80	2.00	3.30	*5.10*	**1.10**	*3.90*	*6.00*	*7.10*	*7.80*	*7.60*	2.80	*6.20*	*6.00*	3.10
erp-cl-ins-enc-abs-cb	*5.70*	**1.10**	2.10	1.90	3.10	*6.00*	**1.00**	*4.30*	*6.00*	*7.40*	*7.80*	*7.60*	*3.80*	*4.30*	*6.50*	3.40
gcr-cs-cl-ins-enc-abs	*5.60*	**1.00**	2.10	2.30	2.80	*5.80*	**1.10**	*4.30*	*6.15*	*7.40*	*7.65*	*7.60*	*4.00*	*3.80*	*6.60*	*3.80*
gcr-cs-cl-ins-enc-cb	*5.80*	**1.10**	**1.60**	3.20	3.90	*5.70*	1.60	2.70	*5.10*	*7.30*	*7.80*	*7.60*	3.80	*5.30*	*6.40*	3.10
gcr-cs-cl-ins-abs-cb	*5.90*	**1.00**	2.60	2.00	3.60	*4.90*	**1.20**	*3.80*	*6.00*	*7.20*	*7.80*	*7.60*	2.60	*6.20*	*6.30*	3.30
gcr-cs-cl-enc-abs-cb	*5.70*	**1.10**	**1.30**	2.30	3.20	*5.80*	2.40	3.20	*6.00*	*7.40*	*7.80*	*7.60*	3.10	*4.60*	*6.50*	*4.00*
gcr-cs-ins-enc-abs-cb	*5.90*	**1.00**	2.60	2.00	3.60	*4.90*	**1.20**	*3.80*	*6.00*	*7.20*	*7.80*	*7.60*	2.60	*6.20*	*6.30*	3.30
gcr-cl-ins-enc-abs-cb	*5.70*	**1.00**	1.80	2.00	3.30	*6.00*	**1.20**	*4.30*	*6.00*	*7.40*	*7.80*	*7.60*	3.70	*4.30*	*6.50*	3.40

Table 19 (continued)

Objectives	SPEA2		NSGA-II		MOEA/D		ϵ-MOEA		GrEA		IBEA		HypE		NSGA-III	
	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S	HV	S
cs-cl-ins-enc-abs-cb	*5.70*	**1.30**	2.10	2.40	3.50	*5.60*	**1.20**	3.10	*5.80*	*7.40*	*7.80*	*7.60*	3.40	*5.30*	*6.50*	3.30
icd-erp-gcr-cs-cl-ins-enc-abs	*5.80*	**1.00**	3.20	2.20	2.80	*5.80*	**1.00**	*4.90*	*5.80*	*7.30*	*7.80*	*7.70*	3.00	*3.90*	*6.60*	3.20
icd-erp-gcr-cs-cs-ins-enc-cb	*6.00*	**1.00**	1.90	2.80	*4.20*	*5.60*	**1.30**	3.50	*5.65*	*7.40*	*7.75*	*7.60*	2.90	*5.20*	*6.30*	2.90
icd-erp-gcr-cs-cl-ins-abs-cb	*6.30*	**1.30**	2.40	1.90	*3.80*	*5.70*	**1.10**	3.80	*5.75*	*7.10*	*7.85*	*7.70*	2.70	*5.20*	*6.10*	3.30
icd-erp-gcr-cs-cl-enc-abs-cb	*6.50*	**1.80**	2.20	**1.80**	*3.90*	*5.70*	**1.10**	3.90	*5.95*	*7.40*	*7.75*	*7.60*	2.80	*4.80*	*5.80*	3.00
icd-erp-gcr-cs-ins-enc-abs-cb	*6.60*	2.00	3.40	**1.60**	3.50	*5.10*	**1.10**	3.80	*5.90*	*7.00*	*7.70*	*7.50*	2.00	*6.10*	*5.80*	2.90
icd-erp-gcr-cl-ins-enc-abs-cb	*6.20*	**1.00**	2.40	2.00	3.60	*5.70*	**1.00**	*4.30*	*5.75*	*7.30*	*7.85*	*7.70*	3.00	*4.80*	*6.20*	3.20
icd-erp-cs-cl-ins-enc-abs-cb	*6.60*	**1.10**	2.80	1.90	3.20	*6.00*	**1.00**	*4.20*	*5.95*	*7.40*	*7.75*	*7.60*	3.00	*4.60*	*5.70*	3.20
icd-gcr-cs-cl-ins-enc-abs-cb	*6.30*	**1.30**	2.60	1.90	3.40	*5.80*	**1.10**	*4.10*	*6.00*	*7.40*	*7.80*	*7.60*	2.90	*4.50*	*5.90*	3.40
erp-gcr-cs-cl-ins-enc-abs-cb	*6.70*	1.80	3.00	**1.60**	2.90	*5.80*	**1.00**	4.00	*5.75*	*7.20*	*7.85*	*7.70*	3.10	*4.80*	*5.70*	3.10
icd-erp-gcr-cs-cl-ins-enc-abs-cb	*6.70*	**1.70**	2.90	1.80	3.30	*5.30*	**1.00**	*4.60*	*5.50*	*6.60*	*7.90*	*7.80*	2.80	*5.00*	*5.90*	3.20

References

Adra S, Fleming P (2011) Diversity management in evolutionary many-objective optimization. IEEE Trans Evol Comput 15(2):183–195

Aleti A, Buhnova B, Grunske L, Koziolek A, Meedeniya I (2013) Software architecture optimization methods: a systematic literature review. IEEE Trans Softw Eng 39(5):658–683

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), pp 1–10. IEEE

Assunção WKG, Colanzi TE, Vergilio SR, Pozo A (2014) A multi-objective optimization approach for the integration and test order problem. Inf Sci 267:119–139

Bader J, Zitzler E (2011) HypE: an algorithm for fast hypervolume-based many-objective optimization. Evol Comput 19(1):45–76

Bansiya J, Davis CG (2002) A hierarchical model for object-oriented design quality assessment. IEEE Trans Softw Eng 28(1):4–17

Bosch J, Molin P (1999) Software architecture design: evaluation and transformation. In: Proceedings of IEEE conference and workshop on engineering of computer-based systems (ECBS'99), pp 4–10

Boussaïd I, Lepagnot J, Siarry P (2013) A survey on optimization metaheuristics. Inf Sci 237:82–117

Bouwers E, Correia J, van Deursen A, Visser J (2011) Quantifying the analyzability of software architectures. In: Proceedings of the 9th working IEEE/IFIP conference on software architecture (WICSA'11), pp 83–92

Bowman M, Briand LC, Labiche Y (2010) Solving the class responsibility assignment problem in object-oriented analysis with multi-objective genetic algorithms. IEEE Trans Softw Eng 36 (6):817–837

Coello Coello CA, Lamont GB, Van Veldhuizen DA (2007) Evolutionary algorithms for solving multi-objective problems, 2nd edn. Springer

Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, New York

Deb K, Jain H (2014) An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: solving problems with box constraints. IEEE Trans Evol Comput 18(4):577–601

Deb K, Mohan M, Mishra S (2003) Towards a quick computation of well-spread pareto-optimal solutions. In: Evolutionary multi-criterion optimization of LNCS, vol 2632. Springer, pp 222–236

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

del Sagrado J, del Águila IM, Orellana FJ (2015) Multi-objective ant colony optimization for requirements selection. Empir Softw Eng 20(3):577–610

Derrac J, García 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

Dobrica L, Niemela E (2002) A survey on software architecture analysis methods. IEEE Trans Softw Eng 28(7):638–653

Ducasse S, Pollet D (2009) Software architecture reconstruction: a process-oriented taxonomy. IEEE Trans Softw Eng 35(4):573–591

Durillo JJ, Zhang Y, Alba E, Harman M, Nebro AJ (2011) A study of the bi-objective next release problem. Empir Softw Eng 16(1):29–60

Garlan D (2000) Software architecture: a roadmap. In: Proceedings of the 22th international conference of software engineering (ICSE'00), pp 91–101

Grunske L (2006) Identifying "Good" architectural design alternatives with multi-objective optimization strategies. In: Proceedings of the 28th international conference on software engineering (ICSE'06), pp 849–852

Gupta P, Verma S, Mehlawat M (2012) Optimization model of COTS selection based on cohesion and coupling for modular software systems under multiple applications environment. In: Computational science and its applications (ICCSA) of LNCS, vol 7335. Springer, pp 87–102

Hadka D, Reed P (2013) Borg: an auto-adaptive many-objective evolutionary computing framework. Evol Comput 21(2):231–259

Harman M, Mansouri SA, Zhang Y (2012) Search-based software engineering: trends, techniques and applications. ACM Comput Surv 45(1):11:1–61

He Z, Yen G, Zhang J (2014) Fuzzy-based Pareto optimality for many-objective evolutionary algorithms. IEEE Trans Evol Comput 18(2):269–285

ISO (2011a) ISO/IEC 25010:2011(E). Software product Quality Requirements and Evaluation (SQuaRE) - System and software quality models. ISO

ISO (2011b) ISO/IEC/IEEE FDIS 42010/D9. Systems and software engineering - Architecture description

Kalboussi S, Bechikh S, Kessentini M, Ben Said L (2013) Preference-based many-objective evolutionary testing generates harder test cases for autonomous agents. In: Proceedings of 5th symposium on search based software engineering (SSBSE'13), pp 245–250

Khare V, Yao X, Deb K (2003) Performance scaling of multi-objective evolutionary algorithms. In: Evolutionary multi-criterion optimization of lecture notes in computer science, vol 2632. Springer, Berlin, pp 376–390

Koziolek A, Ardagna D, Mirandola R (2013) Hybrid multi-attribute QoS optimization in component based software systems. J Syst Softw 86(10):2542–2558

Krogmann K (2010) Reconstruction of software component architectures and behaviour models using static and dynamic analysis. KIT Scientific Publishing

Li R, Etemaadi R, Emmerich MTM, Chaudron MRV (2011) An evolutionary multiobjective optimization approach to component-based software architecture design. In: Proceedings of the IEEE congress on evolutionary computation (CEC'11), pp 432–439

Luna F, González-Álvarez DL, Chicano F, Vega-Rodríguez MA (2014) The software project scheduling problem: A scalability analysis of multi-objective metaheuristics. Appl Soft Comput 15: 136–148

Lutz R (2001) Evolving good hierarchical decompositions of complex systems. J Syst Archit 47(7):613–634

Martin R (1994) OO design quality metrics - An analysis of dependencies. In: Object-Oriented programming systems, languages and applications (OOPSLA), pp 1–8

Mkaouer MW, Kessentini M, Bechikh S, Deb K, Ó Cinnéide M (2014) High dimensional search-based software engineering: finding tradeoffs among 15 objectives for automating software refactoring using NSGA-III. In: Proceedings of the 16th annual genetic and evolutionary computation conference (GECCO'14), pp 1263–1270

Narasimhan VL, Hendradjaya B (2007) Some theoretical considerations for a suite of metrics for the integration of software components. Inf Sci 177(3):844–864

OMG (2010) Unified modeling language 2.4 superstructure specification. OMG. formal/2010-11-14, http://www.omg.org/spec/UML/2.4/

Ouni A, Kessentini M, Sahraoui H, Hamdi MS (2013) The use of development history in software refactoring using a multi-objective evolutionary algorithm. In: Proceedings of the 15th annual genetic and evolutionary computation conference (GECCO'13), pp 1461–1468

Praditwong K, Harman M, Yao X (2011) Software module clustering as a multi-objective search problem. IEEE Trans Softw Eng 37(2):264–282

Praditwong K, Yao X (2007) How well do multi-objective evolutionary algorithms scale to large problems. In: Proceedings of the IEEE congress on evolutionary computation (CEC'07), pp 3959–3966

Purshouse R, Fleming P (2007) On the evolutionary optimization of many conflicting objectives. IEEE Trans Evol Comput 11(6):770–784

Räihä O (2010) A survey on search-based software design. Comput Sci Rev 4(4):203–249

Räihä O, Koskimies K, Makinen E (2011) Generating software architecture spectrum with multi-objective genetic algorithms. In: Proceedings of the 3th world congress on nature and biologically inspired computing (NaBIC'11), pp 29–36

Ramírez A, Romero JR, Ventura S (2014) On the performance of multiple objective evolutionary algorithms for software architecture discovery. In: Proceedings of the 16th annual genetic and evolutionary computation conference (GECCO'14), pp 1287–1294

Ramírez A, Romero JR, Ventura S (2015) An approach for the evolutionary discovery of software architectures. Inf Sci 305:234–255

Romano J, Kromrey JD, Coraggio J, Showronek J (2006) Appropriate statistics for ordinal level data: should we really be using t-test and cohen's d for evaluating group differences on the nsse and other surveys? In: Annual meeting of the Florida association of institutional research

Sant'Anna C, Figueiredo E, Garcia A, Lucena CJ (2007) On the modularity of software architectures: a concern-driven measurement framework. In: Software architecture of LNCS, vol 4758. Springer, pp 207–224

Sayyad A, Ammar H (2013) Pareto-optimal search-based software engineering (POSBSE): a literature survey. In: 2nd inter. workshop on realizing artificial intelligence synergies in software engineering (RAISE), pp 21–27

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 35th international conference on software engineering (ICSE'13), pp 492–501

Schutze O, Lara A, Coello Coello CA (2011) On the influence of the number of objectives on the hardness of a multiobjective optimization problem. IEEE Trans Evol Comput 15(4):444–455

Szyperski C (2002) Component software: beyond object-oriented programming, 2nd edn. Addison-Wesley, Boston

Ventura S, Romero C, Zafra A, Delgado JA, Hervás C (2008) JCLEC: a Java framework for evolutionary computation. Soft Comput 12(4):381–392

von Lücken C, Barán B, Brizuela C (2014) A survey on multi-objective evolutionary algorithms for many-objective problems. Comput Optim Appl 58(3):707–756

Wagner T, Beume N, Naujoks B (2007) Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In: Evolutionary multi-criterion optimization of LNCS, vol 4403. Springer, pp 742–756

Wang H, Jiao L, Yao X (2014) An improved two-archive algorithm for many-objective optimization. IEEE Trans Evol Comput. To appear

Yang S, Li M, Liu X, Zheng J (2013) A grid-based evolutionary algorithm for many-objective optimization. IEEE Trans Evol Comput 17(5):721–736

Yao X (2013) Some recent work on multi-objective approaches to search-based software engineering. In: Proceedings of the 5th symposium on search based software engineering (SSBSE), pp 4–15

Zhang Q, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731

Zhang Y, Harman M, Lim SL (2013) Empirical evaluation of search based requirements interaction management. Inf Softw Technol 55(1):126–152

Zhou A, Qu B-Y, Li H, Zhao S-Z, Suganthan PN, Zhang Q (2011) Multiobjective evolutionary algorithms: a survey of the state of the art. Swarm Evol Comput 1(1):32–49

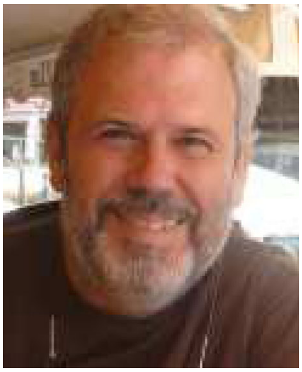
Zitzler E, Künzli S (2004) Indicator-based selection in multiobjective search. In: Parallel problem solving from nature - PPSN VIII of LNCS, vol 3242. Springer, pp 832–842

Zitzler E, Laumanns M, Bleuler S (2004) A tutorial on evolutionary multiobjective optimization. In: Meta-heuristics for multiobjective optimisation of lecture notes in economics and mathematical systems, vol 535. Springer, Berlin, pp 3–37

Zitzler E, Laumanns M, Thiele L (2001) SPEA2: improving the strength Pareto evolutionary algorithm. In: Proceedings of the conference on evolutionary methods for design, optimisation and control with applications to industrial problems, pp 95–100

Aurora Ramírez was born in Córdoba, Spain, in 1989. She received a M.Sc. degree in Computer Science from the University of Córdoba, Spain, in 2012. Since 2012, she has been with the Knowledge Discovery and Intelligent Systems Research Laboratory of the University of Córdoba, where she is currently working towards obtaining a Ph.D., and performing research tasks. Her research interests include the application of evolutionary computation on search-based software engineering and data mining.

José Raúl Romero received his Ph.D. in Computer Science from the University of Málaga, Spain, in 2007. He has worked as an IT consultant for important business consulting and technology companies for several years. He is currently an Associate Professor at the Department of Computer Science of the University of Córdoba, Spain. His current research interests include the industrial use of intelligent systems, search-based software engineering, the use of bio-inspired algorithms for data mining, and model-driven software development and its applications. He has published more than 80 papers in journals and scientific conferences. Dr. Romero is a member of the ACM, and the Spanish Technical Normalization Committee AEN/CTN 71/SC7 of AENOR. He can also be reached at http://www.jrromero.net.



Sebastián Ventura is currently an Associate Professor in the Department of Computer Science and Numerical Analysis at the University of Cordoba, where he heads the Knowledge Discovery and Intelligent Systems Research Laboratory. He received his B.Sc. and Ph.D. degrees in Sciences from the University of Cordoba, Spain, in 1989 and 1996, respectively. He has published more than 150 papers in journals and scientific conferences, and he has edited three books and several special issues in international journals. He has also been engaged in 12 research projects (being the coordinator of four of them) supported by the Spanish and Andalusian governments and the European Union. His main research interests are in the fields of machine learning, data mining, computational intelligence and their applications. Dr. Ventura is a senior member of the IEEE Computer, the IEEE Computational Intelligence and the IEEE Systems, Man and Cybernetics Societies, as well as the Association of Computing Machinery (ACM).