2016 IEEE/ACM 38th IEEE International Conference on Software Engineering

A Practical Guide to Select Quality Indicators for Assessing Pareto-Based Search Algorithms in Search-Based Software Engineering

Shuai Wang[1], Shaukat Ali[1], Tao Yue[1,2], Yan Li[3], Marius Liaaen[4]
[1]Simula Research Laboratory, Oslo, Norway, [2]Department of Informatics, University of Oslo, Oslo, Norway,
[3]Beihang University, Beijing, China, [4]Cisco Systems, Oslo, Norway
{shuai, shaukat, tao}@simula.no, lyadeng79@gmail.com, marliaae@cisco.com

ABSTRACT
Many software engineering problems are multi-objective in nature, which has been largely recognized by the Search-based Software Engineering (SBSE) community. In this regard, Pareto-based search algorithms, e.g., Non-dominated Sorting Genetic Algorithm II, have already shown good performance for solving multi-objective optimization problems. These algorithms produce Pareto fronts, where each Pareto front consists of a set of non-dominated solutions. Eventually, a user selects one or more of the solutions from a Pareto front for their specific problems. A key challenge of applying Pareto-based search algorithms is to select appropriate quality indicators, e.g., hypervolume, to assess the quality of Pareto fronts. Based on the results of an extended literature review, we found that the current literature and practice in SBSE lacks a practical guide for selecting quality indicators despite a large number of published SBSE works. In this direction, the paper presents a practical guide for the SBSE community to select quality indicators for assessing Pareto-based search algorithms in different software engineering contexts. The practical guide is derived from the following complementary theoretical and empirical methods: 1) key theoretical foundations of quality indicators; 2) evidence from an extended literature review; and 3) evidence collected from an extensive experiment that was conducted to evaluate eight quality indicators from four different categories with six Pareto-based search algorithms using three real industrial problems from two diverse domains.

CCS Concepts
• Software and its engineering~Search-based software engineering

Keywords
Quality Indicators, Multi-objective Software Engineering Problems, Pareto-based Search Algorithms, Practical Guide.

1 INTRODUCTION
Many software engineering problems are multi-objective in nature and can be formulated as multi-objective optimization problems (MOPs) [1, 2, 4, 5, 8]. A substantial number of works in Search-Based Software Engineering (SBSE) has shown the capability to solve MOPs using Pareto-based search algorithms that are based on Pareto optimality theory [4, 6, 7, 40-60]. A Pareto-based search algorithm produces a Pareto front consisting of a set of non-dominated solutions (i.e., solutions with equivalent quality) from which users can select one or more solutions for their specific needs. It is therefore important to assess the quality of Pareto fronts produced by these algorithms to determine the applicability of SBSE for solving MOPs [2, 7, 28].

To evaluate the quality of Pareto fronts, the existing works in SBSE have applied several quality indicators, e.g., hypervolume (HV), Epsilon (ϵ), Generalized Spread (GS), Generational Distance (GD), and Pareto Front Size (PFS). These quality indicators are further classified into different categories, e.g., ϵ and GD are defined to measure the *Convergence* between solutions produced by search algorithms and optimal solutions, and GS is defined to measure the *Diversity* of solutions in a Pareto front [7, 9, 26, 65]. However, based on the improved literature review that we conducted by extending the one reported in [13], we discovered that the current literature of SBSE lacks a practical guide to select quality indicators for different software engineering applications. More specifically, the current literature lacks the evidence for selecting quality indicators for the following three cases. First, there is no evidence to show whether it matters to select a particular quality indicator within the same category (e.g., *Convergence*). For example, if the Pareto front produced by Non-dominated Sorting Genetic Algorithm II (NSGA-II) has a better value of GD (*Convergence*) than the one produced by Improved Strength Pareto Evolutionary Algorithm (SPEA2), there is no evidence that we can observe the same phenomenon for ϵ (*Convergence*). Second, there is no evidence whether it matters to select a particular quality indicator from different categories. For example, using GD from *Convergence* and GS from *Diversity*, there is no evidence to show that the same trend of performance can be observed for NSGA-II and SPEA2. Finally, there is no evidence whether computation time can be used as a criterion for selecting quality indicators. It is important to note that the existing works usually chose a subset of the existing quality indicators without proper justification and in most cases, HV was selected because of its popularity [48, 49, 51-56].

In this paper, we propose a practical guide to select quality indicators for assessing Pareto-based search algorithms in SBSE using the following theoretical and empirical methods: 1) Theoretical foundations of quality indicators, 2) The extended literature review, and 3) An extensive experiment. An overview of the approach that we used to derive the guide is shown in Figure 1.

As shown in Figure 1, first, we studied the theoretical foundations of the quality indicators and conducted an extended literature review based on [13] from the existing literature (*Step 1*). In the second step (*Step 2* in Figure 1), we conducted an extensive experiment with eight quality indicators that have been applied in

ICSE'16, May 14–22, 2016, Austin, TX, USA.
 ISBN 978-1-4503-3900-1/16/05$15.00
DOI: http://dx.doi.org/10.1145/2884781.2884880

the existing works [7, 12, 40-60]. These quality indicators are classified into four different categories, i.e., *Convergence*: GD, Euclidean Distance (ED), ϵ; *Diversity*: PFS and GS; *Combination of convergence and diversity* (*Combination*): Inverted Generational Distance (IGD), HV and *Coverage*: Coverage (C) [9]. We evaluate these quality indicators together with six commonly-used Pareto-based search algorithms (e.g., NSGA-II) by employing three industrial software engineering MOPs based on our long-term collaboration from communication and subsea oil&gas domains. From the communication domain, we selected two problems on testing Video Conferencing Systems (VCSs) [10] including: test suite minimization problem (*TM*) with four objectives and test case prioritization problem (*TP*) with four objectives. From the subsea oil&gas domain, we selected one requirements allocation problem (*RA*) with three objectives [11].

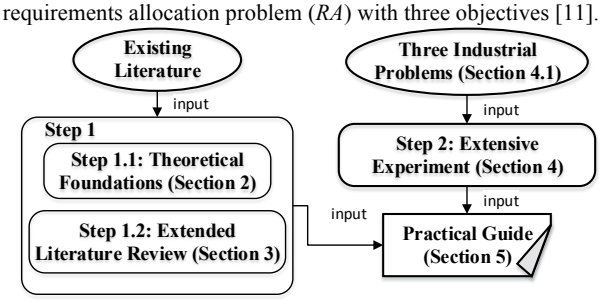

Figure 1 Approach for developing the practical guide

The remainder of the paper is organized as: Section 2 provides theoretical foundations of Pareto optimality, Pareto-based search algorithms, and quality indicators followed by presenting the extended literature review (Section 3). Section 4 presents the extensive experiment and Section 5 provides the practical guide. Last, Section 6 concludes the paper and sketches the future work.

2 THEORETICAL FOUNDATIONS

This section presents theoretical foundations for Pareto optimality theory (Section 2.1), six Pareto-based search algorithms (Section 2.2) and eight quality indicators (Section 2.3).

2.1 Pareto Optimality Theory

Multi-objective optimization is usually based on the Pareto optimality theory [9, 12, 13, 24, 26], which aims to balance several trade-off objectives and produces a number of solutions with equal quality. Pareto optimality defines ***dominance*** to compare solutions, i.e., a solution A is said to dominate another solution B, if for all objectives, A is no worse than B at the same time at least one objective exists that A is better than B.

Formally speaking, suppose there are m objectives $O = \{o_1, o_2, \dots, o_m\}$ to be optimized (let's say minimization), which can be defined as a set of objective functions $F = \{f_1, f_2, \dots, f_m\}$. Solution A dominates solution B (i.e., $A \succ B$) if and only if $\forall i \in \{1,2,\dots,m\}, f_i(A) \leq f_i(B)\ and\ \exists j \in \{1,2,\dots,m\}, f_j(A) < f_j(B)$. The set of solutions that cannot be dominated by others are considered as equally viable, which is named as ***Pareto front***. An ***optimal Pareto front*** (also called ***true Pareto front***) includes all non-dominated solutions that exist in a given search space for a problem, while a Pareto front obtained by a particular search algorithm is usually named as a ***computed Pareto Front*** [9, 26].

Thus, multi-objective search algorithms based on the Pareto optimality theory aim at exploring a given search space and outputs a Pareto front with the aim to provide users with a number of alternative solutions, from which, users can choose the most appropriate solution based on their specific requirements.

2.2 Six Pareto-based Search Algorithms

Search algorithms aim at mimicking natural phenomenon (e.g., bird flocking) to search optimal solutions for optimization problems [17]. To apply search algorithms, fitness functions should be defined to assess the quality of obtained solutions. This paper selects six Pareto-based search algorithms, which are classified into three categories (Table 1) [17]. Notice that our goal is to cover a representative set of existing search algorithms rather than limiting to commonly used ones (e.g., NSGA-II and SPEA2). To achieve this, the six algorithms were chosen systematically for covering at least one algorithm per category [17].

Table 1. Classification of search algorithms

Algorithm Category			Algorithm
Evolutionary Algorithms (EAs)	GAs	Sorting-Based	NSGA-II
		Cellular-Based	MOCell
	Strength Pareto EA		SPEA2
	Evolution Strategies		PAES
Swarm Algorithm	Particle Swarm Theory		SMPSO
Hybrid Algorithm	Cellular GA + differential evolution		CellDE

NSGA-II sorts the population into several non-dominated fronts using a ranking algorithm followed by selecting individuals from these non-dominated fronts and generates new population by applying selection, crossover and mutation operators [4, 5, 18]. Moreover, NSGA-II defines a metric called *crowding distance* to measure the distance between an individual solution and the others. If two individual solutions are in the same non-dominated front, the solution with a higher value of the crowding distance is selected. The aim for *crowding distance* indicator is to maximize the diversity of the outputted non-dominated solutions.

Multi-objective Cellular (MOCell) is based on the cellular model of GAs (cGAs) with an assumption that an individual only interacts with its neighbours during the search process [19]. Moreover, MOCell stores a set of obtained non-dominated individual solutions in an external archive. After each generation, MOCell replaces a fixed number of solutions randomly chosen from the population by selecting the same number of solutions from the archive until the termination conditions are met. Such replacement only takes place when newly generated solutions from the population are worse than the ones in the archive.

Improved Strength Pareto Evolutionary Algorithm (SPEA2) calculates the fitness value for each solution by summing up a strength raw fitness based on the objective functions and density estimation [20]. The density estimation measures the distance between a solution and its nearest neighbours for maximizing the diversity. SPEA2 stores a fixed number of best solutions into an archive by applying selection, crossover and mutation operators. Then, a new population is created by combining solutions from the archive and the non-dominated solutions of the original population. If the combined non-dominated solutions are greater than the maximum size of the population, the solution with the minimum distance to any other solution is selected by applying a truncation operator to calculate the distances to its neighbourhood.

Pareto Archived Evolution Strategy (PAES) applies the dynamic mutation operator for exploring the search space and manages to find optimal solutions [21]. PAES also stores the obtained non-dominated solutions into an archive and newly generated solutions can be added into the archive if they are better than existing solutions by calculating objective functions.

The Speed-constrained Multi-objective Particle Swarm Optimization (SMPSO) is a biological metaheuristic inspired by the social foraging behaviour of animals such as bird flocking [17, 22]. SMPSO selects the best solutions by calculating the crowding distance and stores the selected individual solutions in an archive. SMPSO takes advantage of mutation operators to accelerate the

speed of convergence and adapts the velocity constriction mechanism to avoid the explosion of swarms [22].

The Differential Evolution (DE) algorithm is considered as another kind of EA, which generates solutions by applying recombination, mutation and selection operators [17]. DE calculates the weighted difference between two randomly selected parent solutions and integrates obtained weighted different parts into a third parent solution for generating an offspring. CellDE is a hybrid metaheuristic algorithm using MOCell as a search engine and replacing the typical selection, crossover and mutation operators for GAs with the recombination mechanism of DE [23]. CellDE takes the advantage of cellular GA and DE with good diversity and convergence [23].

2.3 Quality Indicators

To assess the quality of Pareto fronts produced by algorithms, a certain number of quality indicators has been proposed and applied by the existing works, e.g., Generational Distance (GD) [29], Inverted Generation Distance (IGD) [9], Hypervolume (HV) [28], Epsilon (ϵ) [26], Generalized Spread (GS) [27]. Based on the existing literature [7, 9, 13, 26], we selected the most commonly used eight quality indicators and classified them into four categories based on their definitions, *convergence*, *diversity*, *combination of convergence and diversity* (*combination*), and *coverage* (shown in Table 2).

As discussed in Section 2.1, the optimal Pareto front and a computed Pareto front obtained by an algorithm are referred as PF_o and PF_c, respectively. It is worth mentioning that obtaining PF_o for an optimization problem is infeasible in practice due to the limited time or resources [9, 27]. Thus, when applying quality indicators to evaluate Pareto-based search algorithms, a reference Pareto front (PF_{ref}) is often computed to represent the optimal Pareto front (PF_o). Suppose n number of search algorithms produce n computed Pareto fronts: PF_{c1}, PF_{c2},…, PF_{cn}, PF_{ref} is then a union of all the non-dominated solutions from these n computed Pareto fronts, which can be calculated as: $PF_{ref} = \bigcup_{s_i \in \bigcup_{k=1}^{n} PF_{ck}} (\nexists\, s_j \in \bigcup_{k=1}^{n} PF_{ck})(s_j \succ s_i)$. Table 2 also denotes whether calculating a particular quality indicator requires a reference Pareto front (*Require* column). We detail each quality indicator as follows.

Table 2. Categories of quality indicators

Category	Name	Brief Description	Require
Convergence	GD	The Euclidean distance between solutions in PF_c and the nearest solutions in PF_o	Reference Pareto front
	ED	Euclidean distance between the ideal solution and the closest solution in PF_c	Ideal Solution
	ϵ	Smallest distance for transferring every solution in PF_c to dominate PF_o	Reference Pareto front
Diversity	GS	The extent of spread for PF_c	Reference Pareto front
	PFS	Number of solutions included in PF_c	None
Combination	IGD	Euclidean distance between solutions in PF_o and the nearest solutions in PF_c	Reference Pareto front
	HV	The volume covered by solutions in PF_c	Reference Point
Coverage	C	The dominance between PF_c and PF_o	Reference Pareto front

Convergence*. 1) *Generational Distance (GD) is defined to measure how far are the solutions that exist in PF_c from the nearest solutions in PF_o [29], which can be calculated using the formula: $GD = \frac{\sqrt{\sum_{i=1}^{|PF_c|} d(s_i, PF_o)^2}}{|PF_c|}$, where $|PF_c|$ is the cardinality of PF_c (i.e., the number of solutions included in PF_c) and $d(s_i, PF_o)$ refers to the minimum Euclidean distance from the solution s_i in PF_c to PF_o, i.e., the Euclidean distance between s_i in PF_c and the nearest solution in PF_o. A value of 0 for GD indicates that PF_c and PF_o are the same, i.e., all the obtained solutions by a search algorithm are optimal.

2) *Euclidean Distance from the Ideal Solution (ED)* measures the Euclidean distance between the ideal solution and the closest solution in PF_c [32]. The ideal solution s_{ideal} is created by including all the optimal values for each objective (e.g., minimum values for a minimization problem) obtained from all the non-dominated solutions in PF_c. ED can be calculated as $ED = d(s_{ideal}, PF_c)$ (i.e., the shortest Euclidean Distance from s_{ideal} to PF_c) and a value of 0 for ED indicates that the computed Pareto front includes the ideal solution.

Notice that the main difference between ED and GD is that ED focuses on the shortest Euclidean distance between the computed Pareto front and the ideal solution, while GD aims at measuring the average Euclidean distance between solutions in the computed Pareto front and the optimal Pareto front.

3) *Epsilon (ϵ)* measures the shortest distance used to transform every solution in PF_c to dominate PF_o [33, 34]. Suppose a solution s can be represented as $s = (f_1, f_2, \dots, f_m)$ where m is the number of objectives and f_i is the function value for the objective i. Thus, ϵ can be calculated as: $\epsilon(PF_c) = inf_{\epsilon \in \mathbb{R}} \{\forall (s' \in PF_o)\ \exists s \in PF_c : s \prec_\epsilon s'\}$ where $inf_{\epsilon \in \mathbb{R}}$ refers to the infimum for ϵ and $s \prec_\epsilon s'$ means the solution s' in PF_o dominates the solution s in PF_c with a distance of ϵ, i.e., $s \prec_\epsilon s'$ if and only if $\forall (1 \le i \le m): f_{i_s} \prec \epsilon + f_{i_{s'}}$. Notice that a lower value of ϵ denotes that the computed Pareto front is closer to the optimal Pareto front.

Diversity*. 4) *Generated Spread (GS) is defined to extend the quality indicator *Spread* (which only works for two-objective problems) and measure the extent of spread for the solutions in PF_c [9, 35, 36]. GS can be computed using the formula: $GS = \frac{\sum_{k=1}^{m} d(e_k, PF_c) + \sum_{s \in PF_c} |d(s, PF_c) - \overline{d}|}{\sum_{k=1}^{m} d(e_k, PF_c) + |PF_c| * \overline{d}}$, where $(e_1, e_2, \dots, e_m)$ refers to m extreme solutions for each objective in PF_o (m is the number of objectives). An extreme solution for a particular objective means a solution from PF_o that achieves the optimal value for the objective on the basis of sacrificing other objectives. $d(e_k, PF_c)$ refers to the shortest Euclidean distance between the extreme solution e_k and PF_c. $d(s, PF_c)$ means the shortest Euclidean distance between the solution s in PF_c and the other solutions in PF_c, while $\overline{d}$ is the mean value of $d(s, PF_c)$ for all the solutions in PF_c. A lower value of GS shows that the solutions have a better distribution in PF_c.

5) *Pareto front size (PFS)* measures the number of solutions that are included in PF_c, i.e., $|PF_c|$ [7]. It is used to reflect diversity on the basis that users can have more options to choose and thus a higher value of PFS shows a more diverse computed Pareto front.

Combination*. 6) *Inverted Generational Distance (IGD) measures the shortest Euclidean distance from each solution in PF_o to the closest solution in PF_c [29]. IGD can be calculated as: $IGD = \frac{\sqrt{\sum_{j=1}^{|PF_o|} d(s_j, PF_c)^2}}{|PF_o|}$, where $|PF_o|$ refers to the number of solutions in the optimal Pareto front and $d(s_j, PF_c)$ refers to the minimum Euclidean distance from the solution s_j in PF_o to the computed Pareto front. Notice a lower value of IGD means the computed Pareto front is closer to the optimal Pareto front.

7) *Hypervolume (HV)* is defined to measure the volume in the objective space that is covered by PF_c [9]. *HV* can be calculated by $HV = volume\ (\bigcup_{i=1}^{PF_c} v_i)$ and for each solution $s_i \in PF_c$, v_i means the diagonal corners of the hypercube between the solution s_i and a reference point that is a vector of worst objective function values. The reference point is created by including all the worst values for each objective, which can be obtained from all the non-dominated solutions in PF_c (e.g., maximum value for a minimization problem). Notice that a higher value of HV demonstrates a better performance of the computed Pareto front.

Coverage*. 8) *Coverage (C) measures the dominance between PF_c and PF_o [37, 38], i.e., the number of solutions in PF_o that are covered by PF_c. It can be calculated using $C = \frac{|\bigcup_{s \in PF_c} s \in PF_o|}{|PF_o|}$. Notice that C

ranges from 0 to 1 and a higher value of C is preferable, as it signifies that the computed Pareto front consists of more optimal solutions.

3 EXTENDED LITERATURE REVIEW

Search algorithms are increasingly becoming an efficient means for solving complex optimization problems in all phases of software development life cycle. This area of research is termed as Search-Based Software Engineering (SBSE) [1, 3, 5, 44]. In the last two decades, research in SBSE has been significantly increased, e.g., the SBSE repository hosted by the CREST centre [14] contains 1389 published papers as of August 18th, 2015.

Initially, SBSE was mainly focused on solving single-objective optimization problems (SOPs) using search algorithms, e.g., Genetic Algorithms. However, many SE problems are multi-objective by nature [2, 4, 7, 8, 13] and thus it is becoming critical for SBSE to deal with multi-objective optimization problems (MOPs), e.g., selecting a subset of test cases from a large number of test cases without significantly decreasing fault detection capability and at the same time achieving high coverage (three-objective test case selection problem) [4]. One approach in SBSE try to solve MOPs by converting them into SOPs by assigning weights to each objective [16]. Such an approach has two key problems: 1) it is not possible to select precise and accurate weights for each objective; 2) several solutions with equivalent quality may be lost due to the conversion. To overcome these problems, Pareto-based search algorithms (e.g., NSGA-II) are increasingly being used to solve MOPs. These algorithms produce Pareto fronts, where each Pareto front contains a set of non-dominated solutions and eventually a user can select one or more solutions [2, 4, 7, 13] for their problems based on their specific requirements. To evaluate the performance of various Pareto-based search algorithms, a number of quality indicators have been proposed, e.g., Hypervolume (HV) for assessing the quality of Pareto fronts produced by search algorithms [26].

A literature review has been conducted in [13] to review the existing works on applying Pareto-based search algorithms for solving SE problems coined as *Pareto-optimal SBSE (POSBSE)* from the following perspectives: 1) algorithms used; 2) number of objectives to optimize; 3) framework to implement algorithms, and 4) quality indicators for evaluating Pareto fronts. The results show that only 51 out of 1101 papers (as of April 2013) have focused on MOPs suggesting that it is still a new area of research in SBSE. In addition, we also observed that only 15 out of these 51 papers have applied standard quality indicators (e.g., HV) to assess the quality of Pareto fronts, which is commonly used in the optimization community [9, 26, 39, 65]. Even for these 15 papers, there is no clear justification for selecting quality indicators. For most of these works, HV was selected as a quality indicator because it was commonly used and it combines convergence and diversity as discussed in Section 2.3 [13].

Since this work focuses on providing a practical guide on how to choose quality indicators, we have followed the same template and extended the review work in [13] by selecting and reviewing the papers from the SBSE repository related with *POSBSE* from April 2013 to August 2015. The goal for this extended literature review is to study which quality indicators have been used and how the quality indicators were chosen in the recent SBSE works. In addition, we also report the number of objectives for optimization, which can be used for proposing the practical guide. Notice that we do not report the applied multi-objective algorithms and the tool for algorithm implementation (as reported in [13]) since they are out of scope of this paper.

From our extended literature review, we observe papers on *POSBSE* have increased from 51 to 73 until August 2015 and Table 3 lists all these incremental works (in total 22). We can see that the works applying the standard quality indicators has increased from 15 reported in [13] (by April 2013) to 28 (August 2015). However, similarly as discussed in [13], our extended literature review also shows that there is still no clear justification or explanation on the selection criteria for choosing quality indicators in various SE applications. For instance, all the works applying HV [40, 43, 48, 49, 51-53] simply justified that HV was selected either because of its definition based on convergence and diversity or just because of its popularity. Furthermore, since quality indicators can be classified into different categories (Section 2.3), there is no evidence to show whether applying a subset of quality indicators (e.g., ED and HV were used in [43] while HV and GD were applied in [48]) is sufficient to evaluate Pareto fronts produced by search algorithms. There are also no practical guides in the existing SBSE literature for choosing quality indicators in different SE applications.

Based on the extended literature review, this paper is the first work in the SBSE community that aims at providing a practical guide for selecting quality indicators when assessing Pareto-based search algorithms. Notice that the eight quality indicators chosen in this paper cover all the quality indicators that have been applied in the recent SBSE works (Table 3).

Table 3. Extended *POSBSE* works in SBSE

Reference	Quality Indicators	# of Objectives
[6]	Not Applied	5
[7]	HV, PFS, Spread, IGD, Epsilon	5
[40]	PFS, HV, Spread	2
[41]	Not Applied	2
[42]	C, GD, IGD, ED	2, 4
[43]	ED, HV	2
[45]	Not Applied	4
[46]	Not Applied	2
[47]	Not Applied	3
[48]	HV, GD	2
[49]	HV	2
[50]	IGD	15
[51]	HV, IGD, C	2
[52]	HV, Spread	2
[53]	HV	2, 4, 6
[54]	HV, GD, IGD, C	2
[55]	HV, GD, IGD, Epsilon	2
[56]	HV, C, ED	5
[57]	Not Applied	2
[58]	Not Applied	4
[59]	Not Applied	2
[60]	Not Applied	3

4 EXPERIMENT

This section presents the extensive experiment we conducted for empirically evaluating the eight quality indicators together with the six Pareto-based search algorithms, which includes: 1) description of the industrial problems (Section 4.1); 2) experiment design (Section 4.2); 3) experiment results (Section 4.3); 4) discussion based on the results (Section 4.4); and 5) discussion related with threats to validity (Section 4.5).

4.1 Description of the Industrial Problems

This section presents three industrial problems from two distinct domains: communication (Section 4.1.1) and subsea oil&gas (Section 4.1.2), as shown in Table 4.

4.1.1 Communication Case Studies

Since 2007, we have established a close collaboration with Cisco, Norway, focusing on improving the cost and effectiveness of testing a variety of Videoconferencing Systems (VCSs) developed by Cisco [10]. The core functionality of a VCS is to establish a videoconference among participants at various physical locations. There is also a possibility of transmitting

presentations in parallel to a videoconference using VCSs. Generally speaking, VCSs aim at offering efficient means to organize high-quality, face-to-face meetings, without requiring physically gathering of participants from different geographic locations. Each VCS has on average three million lines of embedded C code. Notice that for testing *Saturn* (a product line of VCSs focused in this paper), a test case repository has been constructed by Cisco's test engineers with more than 2000 test cases. New test cases are continuously added to this repository.

Table 4. An overview of the industrial problems

Domain	Problem	Objective
Communication	Test Suite Minimization	Test Minimization Percentage (*TMP*)
		Feature Pairwise Coverage (*FPC*)
		Fault Detection Capability (*FDC*)
		Overall Execution Time (*OET*)
	Test Case Prioritization	Prioritized Extent (PE)
		Feature Pairwise Coverage (*FPC*)
		Fault Detection Capability (*FDC*)
		Overall Execution Cost (*OEC*)
Subsea oil&gas	Requirements Allocation	Extent of Assigned Requirements (*ASSIGN*)
		Familiarity of Stakeholders (*FAW*)
		Overall Differences of Workloads (*OWL*)

For testing a new VCS, we learned that two industrial problems are required to be addressed, i.e., 1) ***Cost-effective Test Suite Minimization***: eliminating redundant test cases without significantly reducing the effectiveness (e.g., feature pairwise coverage) at the same time minimizing the cost (execution time) [6, 16]; 2) ***Cost-effective Test Case Prioritization***: prioritizing test cases into an optimal order within a limited test resource budget with the aim to maximize the effectiveness (e.g., fault detection capability) and minimize the cost (e.g., execution time) [15]. Considering the number of test cases is large (i.e., search space is huge) for both industrial problems, each of the two problems can be formulated as a multi-objective optimization problem and applying search algorithms shows promising results as we previously studied [6, 15, 16].

4.1.1.1 Test Suite Minimization Problem

Test suite minimization aims at eliminating redundant test cases, while maximizing the effectiveness and minimizing the cost. To deal with this problem, four cost/effectiveness objectives (Table 4) were defined together with test engineers at Cisco.

a) Test Minimization Percentage (*TMP*) measures the number of test cases that can be minimized as compared with the original test suite; *TMP* can be measured as $TMP = \left(1 - \frac{nt_{minimized}}{nt_{original}}\right) * 100\%$, where $nt_{minimized}$ is the number of minimized test cases and $nt_{original}$ is the number of original test cases;

b) Feature Pairwise Coverage (*FPC*) measures how many pairs of features (testing functionalities) can be covered by the minimized test cases; *FPC* is measured as $FPC = \frac{NumFP_{minimized}}{NumFP_{p_i}}$, where $NumFP_{minimized}$ refers to the number of feature pairs covered by the minimized test cases, while $NumFP_{p_i}$ is the number of feature pairs that should be covered by VCS product p_i;

c) Fault Detection Capability (*FDC*) measures how many test cases can manage to find faults within a specified time period (e.g., one week in the past); *FDC* can be measured as $FDC = \frac{\sum_{i=1}^{nt_{minimized}} SucR_{tc_i}}{nt_{minimized}}$, where $SucR_{tc_i}$ refers to the success rate that a test case tc_i can manage to detect faults in a given time;

d) Overall Execution Time (*OET*) measures how long it takes for executing the minimized test cases. *OET* can be measured as $OET = \sum_{i=1}^{nt_{minimized}} AET_{tc_i}$, where AET_{tc_i} refers to the average historical execution time for test case tc_i.

More details about the objective functions can be found in [6].

4.1.1.2 Test Case Prioritization Problem

Test case prioritization aims to cost-effectively prioritize a set of test cases within a limited budget of available test resources [15]. In our case, test resources refer to correct software versions deployed on hardware since test cases aim at testing different software versions. Notice that it may take different time to allocate particular test resources for a specific test case. Similarly, to address such a test prioritization problem, we defined four cost-effectiveness measures as below (shown in Table 4).

a) Prioritized Extent (*PE*) measures the number of test cases that can be prioritized within the test resource budget; *PE* can be measured as $PE = \frac{nt_{prioritized}}{nt}$, where $nt_{prioritized}$ refers to the number of prioritized test cases within available test resources while nt is the total number of test cases to be prioritized;

b) Feature Pairwise Coverage (*FPC*) measures how many pairs of features can be covered by the prioritized test cases; *FPC* can be measured as $FPC_{s_k} = \frac{NumFP_{prioritized}}{NumFP}$, where $NumFP_{prioritized}$ is the number of feature pairs covered by the prioritized test cases and $NumFP$ refers to the total number of feature pairs that should be tested;

c) Fault Detection Capability (*FDC*) measures the fault detection capability of the prioritized test cases. It is measured as $FDC = \frac{\sum_{i=1}^{nt_{prioritized}} SucR_{tc_i}}{nt_{prioritized}}$, where $SucR_{tc_i}$ refers to the success rate that a test case tc_i can manage to find faults in a given time;

d) Overall Execution Cost (*OEC*) measures how long it takes to setup the required test resources and execute the prioritized test cases. It can be calculated as $OEC = \sum_{i=1}^{nt_{prioritized}} (AET_{tc_i} + TTR_{tc_i})$, where AET_{tc_i} refers to the average historical execution time for test case tc_i and TTR_{tc_i} is the total time for allocating relevant test resources for tc_i. More details can be found in [15].

4.1.2 Subsea Oil&Gas Case Study

Subsea production systems (SPSs) are large-scale, heterogeneous and highly-hierarchical Cyber-Physical Systems (CPSs) that control and monitor physical processes such as oil and gas production platforms [30, 31]. SPSs manage the exploitation of oil and gas production fields by integrating hundreds of hardware components and software.

At the early phase of developing such a large scale CPS, a large number of requirements are required to be inspected by different stakeholders from different organizations or departments of the same organization. These requirements have different characteristics such as various extents of importance to an organization, complexity, and dependencies between each other, thereby requiring different effort (workload) to inspect [25]. Therefore, one practical challenge has been identified, i.e., ***requirements allocation*** that aims to maximize stakeholders' familiarities to the assigned requirements and at the same time balance the overall workload of each stakeholder. The problem has been formulated as a multi-objective optimization problem in our previous works [11, 25].

4.1.2.1 Requirements Allocation Problem

To address this problem, three cost-effectiveness measures have been defined in [25].

a) *ASSIGN* represents the extent of assigning all the requirements to stakeholders. It can be measured as *ASSIGN* $= \frac{\sum_{i=1}^{n_{St}} n_{AR_i}}{n_R}$, where n_{AR_i} returns the number of requirements assigned to the i_{th} stakeholder, n_R is the total number of

requirements and n_{St} is the total number of stakeholders;

b) *FAM* denotes the overall familiarity of the stakeholders to requirements allocated to them. It can be measured as $FAM = \frac{\sum_{i=1}^{n_{St}} \sum_{j=1}^{n_{AR_i}} \frac{FM_{ji} - FM_{min}}{(FM_{max} - FM_{min})}}{\sum_{k=1}^{n_{St}} n_{AR_k}}$, where FM_{ji} represents the familiarity value of stakeholder S_j for requirement R_i and all familiarity values range from FM_{min} to FM_{max};

c) *OWL* represents the overall differences of workloads of the stakeholders and it can be measured as $OWL = \frac{\sum_{j=1}^{n_{St}-1} \sum_{k=j+1}^{n_{St}} abs(WL_j - WL_k)}{n_{St}*(n_{St}-1)}$, where WL_j computes the workload for all the requirements assigned to i_{th} stakeholder based on their complexity, dependency index, and importance: $WL_i = \frac{\sum_{j=1}^{n_{AR_i}} ((\frac{CM_j - CM_{min}}{(CM_{max} - CM_{min})} + \frac{DP_j - DP_{min}}{(DP_{max} - DP_{min})} + \frac{IM_j - IM_{min}}{(IM_{max} - IM_{min})})/3)}{n_{AR_i}}$, in which, CM_j, DP_j and IM_j represent the complexity, dependency and importance of the requirement j_{th} respectively.

4.2 Experiment Design

In this section, we present the experiment design from these aspects: 1) Research questions (Section 4.2.1); 2) Case studies of the three industrial problems (Section 4.2.2); 3) Experiment tasks performed and statistical tests (Section 4.2.3); 4) Evaluation metric and algorithm parameter settings (Section 4.2.4).

4.2.1 Research Questions

Our goal is to provide a practical guide for the SBSE community to select quality indicators for SE applications. To meet our objective, we define the following research questions:

RQ1: Does it matter to select a particular quality indicator within each category?

Answering this research question helps us to define a guide for selecting quality indicators within the same category for assessing Pareto-based search algorithms in different SE contexts.

RQ2: Does it matter to select quality indicators from different categories?

Answering this research question helps to define a guide for selecting quality indicators from different categories for assessing Pareto-based search algorithms in different SE contexts.

RQ3: Does it matter to take calculation time into account when selecting quality indicators?

Answering this research question helps us to study whether calculation time can be an additional layer to define a guide for selecting quality indicators.

4.2.2 Case Studies

In Section 4.1, we introduced three industrial problems: 1) test suite minimization problem (TM), 2) test case prioritization problem (TP), and 3) requirements allocation problem (RA). We detail the case studies used for each problem as below.

TM: We chose four VCSs from *Saturn*: C20, C40, C60 and C90 (Table 5). There are 169 features (testing functionalities) in *Saturn* and each VCS includes a subset of these features. Moreover, C20, C40, C60 and C90 include 17, 25, 32 and 43 features respectively and 138, 167, 192 and 230 test cases relevant for testing these VCSs, respectively [6]. Notice that each feature can be tested by at least one test case and each test case can be used to test at least one feature. Each test case tc_i has a success rate for execution ($SucR_{tc_i}$) for calculating *FDC*, an average execution time (AET_{tc_i}) for measuring *OET*. In general, for *Saturn*, each feature is associated with 5-10 test cases and each test case tc_i is associated with 1-5 features with $SucR_{tc_i}$ ranging from 50% to 100%, and AET_{tc_i} ranging from 2 to 60 minutes.

TP: We chose a testing cycle that includes 257 test cases for testing 53 functionalities (features) (Table 5). Each feature can be tested by at least one test case and one test case can be executed for testing one or more features. Each test case tc_i has a success rate for execution ($SucR_{tc_i}$) ranging from 50% to 100%, an average execution time AET_{tc_i} ranging from 2 to 60 minutes, and time for allocating relevant test resources (TTR_{tc_i}) ranging from 1 to 30 minutes. Moreover, there are 59 available test resources used to setup the test environment (e.g., correct software) for executing such 257 test cases. Notice that each test resource can be allocated for executing one or more test cases and execution of each test case requires one or more test resources.

RA: We selected a real-world case study (i.e., a large-scale CPS) including 287 requirements and identified 10 stakeholders who are responsible for reviewing and checking these requirements (Table 5). Each requirement R_i has three attributes that include: complexity CM ranging from 0 to 9, dependency index DP from 0 to $n_R - 1$ (n_R is the number of requirements) and importance IM ranging from 0 to 9. Each stakeholder S_j has one attribute, i.e., familiarity for a specific requirement R_i (FM_{ji}) ranging from 0 to 9 in our case.

It is worth mentioning that all the three industrial problems are complex based on our previous works [6, 11, 15, 16, 25] since random search (a baseline) has been compared with search algorithms and results consistently show that search algorithms significantly outperform random search.

4.2.3 Experiment Tasks and Statistical Tests

Experiment Tasks: We first perform a task by running the six algorithms for the case studies in each industrial problem (as shown in Figure 2). Thus, for each case study in an industrial problem, 100 values are obtained for each quality indicator in terms of each search algorithm.

Table 5 An overview of the experiment design

RQs	Experiment Tasks		Statistical Tests	Algorithms	Quality Indicators		Problems	Case Studies	
1	T_1	T_{11}: Compare each pair of the algorithms by analyzing the values of the quality indicators within category	Vargha and Delaney statistics Mann-Whitney U test	NSGA-II SPEA2 MOCell CellDE SMPSO PAES	Convergence	GD	TM TP RA	TM	C20
						ED			C40
		T_{12}: Analyze the correlations of each pair of the quality indicators within category	Kendall Rank Test			ϵ			C60
					Diversity	GS			C90
2	T_2	T_{21}: Compare each pair of the algorithms by analyzing the values of the quality indicators across categories	Vargha and Delaney statistics Mann-Whitney U test			PFS		TP	C20
					Combination	IGD			C40
		T_{22}: Analyze the correlations for each pair of the quality indicators across categories	Kendall Rank Test			HV			C60
					Coverage	C			C90
3	T_3: Compare the quality indicators in terms of computation time		Average Value Mann-Whitney U test					RA	CPS

For each industrial problem:
  For each case study:
    1: Run the six Pareto-based search algorithms with certain number of times (100 in our case);
    For each time of run:
      2: Obtain a reference Pareto front by combining Pareto fronts produced by the six algorithms;
      3: Calculate the values for the eight quality indicator so that each algorithm has eight values associated with each quality indicator;
    End;
    4: Obtain the 100 values for each quality indicator for each algorithm;
  End;
End.

Figure 2 Pseudo code for obtaining quality indicator values

We defined a corresponding task as shown in Table 5 to answer each research question based on the obtained quality indicator values. The T_1 task is divided into T_{11} and T_{12}, where T_{11} compares each pair of the search algorithms using each quality indicator within the same category for each case study. The T_{12} task answers RQ1 from another perspective, where we study the correlations of two quality indicators within category by ignoring the differences of the search algorithms. T_2 is also divided into two tasks. T_{21} compares each pair of the algorithms by analyzing the values of quality indicators across categories. T_{22} studies the correlations between the quality indicators across categories. Last, in T_3, we compare the quality indicators based on their calculation time to answer RQ3.

Statistical Tests: For T_{11} and T_{21}, the Vargha and Delaney statistics and Mann-Whitney U test are used by following the guides reported in [61] to compare the results of the search algorithms based on the quality indicators within or across categories (Table 5). The Vargha and Delaney statistics is used to calculate $\hat{A}_{12}$, which is a non-parametric effect size measure. In our context, $\hat{A}_{12}$ is used to compare the probability of yielding higher values of each quality indicator for two algorithms *A* and *B*. Each pair of algorithms is further compared using the Mann-Whitney U test (*p*-value) to determine the significance of the results with the significance level of 0.05, i.e., there is a significant difference if *p*-value is less than 0.05. For the quality indicators HV, PFS, C (higher value, better performance), *A* significantly outperforms *B* if $\hat{A}_{12}$ is greater than 0.5 and the *p*-value is less than 0.05 while *A* performs significantly worse than *B* if $\hat{A}_{12}$ is less than 0.5 and the *p*-value is less than 0.05. There is no significant difference between *A* and *B* if the *p*-value is greater than 0.05. For the other quality indicators (i.e., GD, ED, epsilon, IGD and GS), vice versa.

To study the correlation between each pair of quality indicators (T_{12} and T_{22}), we choose the Kendall rank correlation coefficient (τ) as shown in Table 5 [64]. The τ value ranges from -1 to 1, i.e., there is a positive correlation if τ is equal to 1 and a negative correlation when τ is -1. A τ close to 0 shows that there is no correlation between two sets of data. Moreover, we also report significance of correlation using *Prob*>|τ |; a value lower than 0.05 means that the correlation is statistically significant. Notice that the test does not require the monotonicity of two data sets, which suits our case. In addition, the Mann-Whitney U test is applied to determine whether there are significant differences for the time to calculate each quality indicator (T_3).

4.2.4 Metric and Parameter Settings

Evaluation Metric: To address RQ1 and RQ2, we define a metric $Matters$ to measure whether two quality indicators can demonstrate the same trend of performance when evaluating the algorithms: $Matters\ (QI_1, QI_2) = \prod_{i=1}^{C(n,2)} matter_i(QI_1, QI_2)$, where n is the number of the algorithms and $C(n,2)$ refers to the number of the algorithm pairs. For instance, there are $C(6,2) = 15$ pairs for the selected six Pareto-based search algorithms in our case. The $matter_i(QI_1, QI_2)$ function denotes whether it matters to choose one of the quality indicators QI_1 and QI_2 when comparing the i_{th} algorithm pair. In other words, $matter_i(QI_1, QI_2)$ is 1 if both QI_1 and QI_2 reveal the same trend of performance for the i_{th} algorithm pair, i.e., it doesn't matter which quality indicator to choose; otherwise, $matter_i(QI_1, QI_2)$ is 0. For example, suppose that HV and IGD are used to compare NSGA-II and SPEA2. If both HV and IGD consistently indicate one of the three cases: 1) NSGA-II performs significantly better than SPEA2; 2) NSGA-II performs significantly worse than SPEA2; and 3) there is no significant difference between NSGA-II and SPEA2, $matter_i(HV, IGD)$ will be 1. Otherwise, if HV and IGD demonstrate different trends of performance for NGSA-II and SPEA2, $matter_i(HV, IGD)$ will be 0. Using $Matters$, RQ1 and RQ2 can be answered in a precise and compact manner. In conclusion, we can say that it does not matter if we choose QI_1 or QI_2 for assessing the algorithms if $Matters\ (QI_1, QI_2) = 1$ for all pairs of the search algorithms (e.g., 15 pairs for the six search algorithms in our case); otherwise it matters and therefore both QI_1 and QI_2 should be applied together if $Matters\ (QI_1, QI_2) = 0$ since there is at least one algorithm pair that the two quality indicators show the different trends of performance.

Parameter Settings: We employ jMetal [9] to encode the three industrial problems together with the implementation of the six selected Pareto-based search algorithms. All the parameters that are used for configuring these algorithms are shown in Table 6, which are suggested as default parameters from the jMetal library [9]. In addition, we set the maximum number of fitness evaluations as 25000, i.e., the search is terminated if the fitness function has been evaluated for 25000 times. Notice that tuning parameters may lead to different performance of search algorithms, but standard parameter settings are usually recommended [61]. Furthermore, all the eight quality indicators mentioned in Section 2.3 are implemented based on jMetal [9]. As suggested in [61], each algorithm was run 100 times to account for random variations.

Table 6. Parameter settings of the search algorithms

Algorithm	Parameter Settings
NSGA-II SPEA2	***Population Size***: 100; ***Selection of Parents***: binary tournament + binary tournament; ***Recombination***: simulated binary, crossover rate = 0.9; ***Mutation***: polynomial, mutation rate = 1.0/n
MOCell	***Population Size***: 100; *Neighbourhood*: 1-hop neighbours (8 surrounding solutions); ***Selection of Parents***: binary tournament + binary tournament; ***Recombination***: simulated binary, crossover rate = 0.9: ***Mutation***: polynomial, mutation rate = 1.0/n; *Archive Size*: 100
PAES	***Mutation***: polynomial, mutation rate = 1.0/n; ***Archive Size***: 100
SMPSO	***Population Size***: 100; ***Mutation***: polynomial, mutation rate = 1.0/n; ***Archive Size***: 100
CellDE	***Population Size***: 100; *Neighbourhood*: 1-hop neighbours (8 surrounding solutions); ***Selection of Parents***: binary tournament + binary tournament; ***Recombination***: differential evolution, crossover rate = 0.9; ***Archive Size***: 100

4.3 Experiment Results

This section presents the results for each research question[1].

RQ1: RQ1 focuses on empirically evaluating the quality indicators within the same category. Table 7 lists the values of

[1] For reproducibility, we make all the data related with the experiment publicly available at: http://zen-tools.com/ICSE2015.html

$Matters$ for the quality indicators within each category by applying the six Pareto-based search algorithms in the four VCS products (case studies) for solving the TM and TP problems and one CPS for solving the RA problem. From Table 7, we can observe that the three quality indicators within *Convergence* (i.e., GD, ED and ϵ) show the same trend of performance when comparing the 15 pairs of the six algorithms in the three industrial problems since all the values for $Matters$ are equal to 1. Similarly, the two quality indicators (i.e., IGD and HV) within *Combination* also demonstrate the same trend of performance when comparing the 15 pairs of the six algorithms. However, we observe that in *Diversity*, GS and PFS show different trends of performance since all the values of $Matters$ are 0. This observation shows that when using GS and PFS for measuring diversity, we may observe different performance of search algorithms for the same problem. Notice that we only have one quality indicator in *Coverage* (Table 5) and thus we cannot calculate $Matters$ within *Coverage*.

Table 7. *Matters* of the quality indicators within each category

Category	Quality Indicators	TM	TP	RA
Convergence	GD and ED	1	1	1
	GD and ϵ			
	ED and ϵ			
Diversity	GS and PFS	0	0	0
Combination	IGD and HV	1	1	1

Moreover, we apply the Kendall rank correlation coefficient test (τ) for studying the correlations between quality indicators within the same category. Table 8 summaries the key results for pairs of quality indicators using the τ test. From Table 8, one can observe that the three quality indicators (ED, GD and ϵ) of *Convergence* have significantly positive correlations for all the case studies of the three industrial problems since all the values of τ are greater than 0 (close to 1) and the *p*-values for *Prob*>|τ| are all less than 0.05 (due to the limited space, we do not report the individual values for τ and *Prob*>|τ|). Therefore, we can conclude that for *Convergence*, all the three quality indicators show the same trend of performance when comparing different algorithms and it therefore does not matter which one to choose.

Table 8. Key results for the correlation analysis using the τ test for the quality indicators within category

Category	Quality Indicators	TM	TP	RA
Convergence	GD and ED	$\tau > 0$ $p < 0.05$	$\tau > 0$ $p < 0.05$	$\tau > 0$ $p < 0.05$
	GD and ϵ			
	ED and ϵ			
Diversity	GS and PFS	$p > 0.05$	$p > 0.05$	$p > 0.05$
Combination	IGD and HV	$\tau < 0$ $p < 0.05$	$\tau < 0$ $p < 0.05$	$\tau < 0$ $p < 0.05$

As for *Diversity*, the results in Table 8 show that there is no significant correlation between GS and PFS since the *p*-values for *Prob*>| τ | are greater than 0.05 and thus GS and PFS have to be selected together when evaluating the diversity of a Pareto front. For *Combination* (i.e., HV and IGD), Table 8 shows that a significantly negative correlation exists between HV and IGD. The correlation is negative since a higher value of HV shows a better Pareto front, which is represented by a lower value of IGD. Thus, for this category, it also does not matter which indicator to choose when assessing the performance of the search algorithms.

Based on the above results, RQ1 can be answered as follows. For *Convergence* and *Combination*, it doesn't matter which quality indicator within the same category to choose; however, it does matter for *Diversity*, i.e., both GS and PFS should be used together when assessing Pareto fronts.

RQ2: RQ2 aims at empirically evaluating the quality indicators across categories. Based on the results of RQ1 (i.e., GD, ED and ϵ have significant correlations, and IGD and HV also have a significant correlation), we chose ϵ and HV for representing the categories of *Convergence* and *Combination*, respectively since the results of GD and ED are consistent with ϵ for *Convergence* and the results of IGD are consistent with HV for *Combination*. Therefore, the five quality indicators from the four different categories (i.e., ϵ, HV, PFS, GS and C) are compared in the three industrial problems (i.e., TM, TP and RA). Table 9 summarizes the key results of $Matters$ when comparing each pair of the five quality indicators in the three industrial problems.

Table 9. *Matters* of the quality indicators across categories

Category Across	Pair	TM	TP	RA
Convergence and Diversity	ϵ and GS	0	0	0
	ϵ and PFS			
Convergence and Combination	ϵ and HV			
Convergence and Coverage	ϵ and C	1	1	1
Diversity and Combination	GS and HV	0	0	0
	PFS and HV			
Diversity and Coverage	GS and C			
	PFS and C			
Combination and Coverage	HV and C			

From Table 9, we observe that two quality indicators from different categories can result in different trends of performance of the search algorithms except for ϵ and C. In other words, it does not matter which quality indicators to choose in terms of *Convergence* and *Coverage*, but it does matter for the quality indicators from other categories since all values for $Matters = 0$.

Table 10 summarizes the key findings by studying correlations between each pair of quality indicators across categories using the Kendall rank correlation coefficient test (τ). We can see that there is no significant correlation for all the pairs of the quality indicators except for ϵ and C since the *p*-values for *Prob*>| τ | are greater than 0.05. As for ϵ and C, there is a significantly negative correlation since all the values for τ are less than 0 (close to -1) and the *p*-values for *Prob*>| τ | are all less than 0.05. Such a significant correlation is negative since a lower value of ϵ denotes a better Pareto front, which is represented by a higher value of C.

Table 10. Key results for the correlation analysis using the τ test for the quality indicators across categories

Category Across	Pair	TM	TP	RA
Convergence and Diversity	ϵ and GS	$p > 0.05$	$p > 0.05$	$p > 0.05$
	ϵ and PFS			
Convergence and Combination	ϵ and HV			
Convergence and Coverage	ϵ and C	$\tau < 0$ $p < 0.05$	$\tau < 0$ $p < 0.05$	$\tau < 0$ $p < 0.05$
Diversity and Combination	GS and HV	$p > 0.05$	$p > 0.05$	$p > 0.05$
	PFS and HV			
Diversity and Coverage	GS and C			
	PFS and C			
Combination and Coverage	HV and C			

Based on the above results, RQ2 can be answered as follows. It does matter to choose quality indicators across categories except for *Convergence* and *Coverage* when assessing Pareto-based search algorithms in different SE contexts.

RQ3: This research question is designed to compare time to calculate each indicator. Notice that each algorithm is run for 100 times in our case and each run provides a data point for the time. Since six algorithms are selected and three industrial problems are involved including 9 case studies (Table 5), 5400 data points of time can be obtained in total for calculating each quality indicator. We report the average value of these 5400 time data points for each quality indicator (Table 11) and perform the Mann-Whitney U test to determine whether there are significant differences in terms of calculation time for each pair of quality indicators.

Results show that for all the eight quality indicators, PFS takes significantly less time than all the others since all the *p*-values are less than 0.05, which are not reported to save space. Within *Convergence* and *Coverage*, calculating C takes significantly less

time than the others (i.e., GD, ED and ϵ). Within *Combination*, there is no significant difference for calculating HV and IGD. Thus, we can answer RQ3 as follows: there are significant differences in terms of time for calculating quality indicators and thus calculation time can be used as additional criterion in our guide for selecting quality indicators. However, the practical differences for calculating these quality indicators may not be huge since all of them are in a few seconds (Table 11).

Table 11. Average time to calculate each quality indicator

Category	Quality Indicators	Average Time (Seconds)
Convergence	GD	3.57
	ED	1.84
	ϵ	4.86
Diversity	GS	5.41
	PFS	0.35
Combination	IGD	3.76
	HV	3.02
Coverage	C	1.03

4.4 Discussion on Results

For RQ1, we observe that within *Convergence*, all the three quality indicators (i.e., GD, ED and ϵ) show the same trend of performance when comparing Pareto-based search algorithms at the same time there are significantly positive correlations among them. This can be explained from the fact that GD, ED and ϵ are defined to measure the distance of solutions in a computed Pareto front to the optimal solutions (in the optimal Pareto front) though different mathematical formulas are applied (Section 2.3). It is worth mentioning that calculating ED only requires an ideal set of objective values, i.e., the optimal value that each objective can achieve (Section 2.3), while GD and ϵ require obtaining a reference Pareto front (simulating the optimal Pareto front) [39, 65]. In the context of our case studies, we were not aware of ideal objective set beforehand (which is obtained from the reference Pareto front) and the results show that GD, ED and ϵ are equivalent in terms of *Convergence*. However, for certain SE problems, if the ideal set of objective values is known beforehand (e.g., for testing, the maximum value of feature pairwise coverage is 1), ED should be a more accurate quality indicator as compared with GD and ϵ, since these two indicators require a reference Pareto front for representing the optimal Pareto front [39, 65].

As for *Combination*, both HV and IGD are defined to measure: 1) how obtained solutions are close to optimal solutions, and 2) how obtained solutions are distributed in a computed front. Based on the results of the experiment, HV and IGD indicate the same trend of performance when comparing the algorithms. Furthermore, calculating HV requires a reference point (the worst objective set) instead of a reference Pareto front required by IGD (Section 2.3). Therefore, HV is considered as a more accurate quality indicator when such a reference point is known beforehand for certain SE problems, e.g., the minimum number of test cases to be eliminated is 0 for test suite minimization.

However, within *Diversity*, the two quality indicators (i.e., GS and PFS) do not demonstrate the same trend of performance when comparing the search algorithms and the correlation between them is not significant. This can be explained based on the fact that PFS is defined based on the assumption that more solutions included into a Pareto front, more options a user can choose from and thus it reflects a more diverse Pareto front. However, when all the solutions are close to each other in the front, even higher values of PFS cannot necessarily demonstrate a Pareto front with a higher diversity. As compared with PFS, GS is defined to measure how well solutions are distributed in a computed Pareto front (Section 2.3). Thus, GS and PFS provide two different perspectives of measuring diversity, which should be applied together. When calculating GS, it requires calculating a reference Pareto front (Section 2.3) while PFS does not require anything.

As for RQ2, the results show that the quality indicators of *Convergence* show the same trend of performance as the quality indicator of *Coverage*. Such interesting finding can be explained that when a computed Pareto front shares more common solutions with the optimal Pareto front (i.e., the value of C is higher), the computed Pareto front and the optimal solutions should be closer (i.e., the values for GD, ED and ϵ are lower). Meanwhile, significant correlations were observed among these four quality indicators, which provide further evidence for the observation. As for other quality indicators across categories, no such phenomenon were found that denotes that quality indicators can not replace others that belongs to different categories except for the above-mentioned ones (i.e., GD, ED, ϵ and C).

In terms of computation effort (RQ3), the results demonstrate that calculating quality indicators can take significantly different time, e.g., the calculation time for PFS is significantly less than the others (Section 4.3). Based on the theoretical foundations [39, 65], we observe that the computation effort for PFS is linear, whereas the others except HV are in quadratic. As for HV, the calculation time will be increased exponentially with the increase in the number of objectives. When the number of objectives is less (e.g., three and four objectives in our cases), the practical differences are not very huge when looking into the average calculation time for each indicator, e.g., 0.35 seconds for PFS and 4.86 seconds for ϵ (Table 11). Notice that when the number of objectives is greater than 10 (15 as reported in [50]), HV is not applicable since it becomes very expensive to calculate. Based on these observations, calculation time can be an additional criterion for selecting quality indicators.

4.5 Threats to Validity

A threat to *internal validity* is that we have experimented with only one-default configuration setting for algorithm parameters. However, these settings are in accordance with the common guides in the literature [61, 63]. The *conclusion validity* threat in experiments involving algorithms is due to random variations. To address it, we repeated experiments 100 times to reduce the possibility that the results were obtained by chance. We also reported the results using the Vargha and Delaney statistics (to measure the effect size), Mann-Whitney U test (to determine the statistical significance) and the Kendall rank correlation coefficient (to measure the correlations among the indicators).

The observed *construct validity* threat is that the measures used are not comparable across the algorithms. In our context, we used the same stopping criteria for all the algorithms, i.e., the number of fitness evaluations (i.e., 25000). As for *external validity* threat related with generalization of results, three industrial problems were chosen from two different domains (i.e., communication and subsea oil&gas), which cover two different phases of software lifecycle, i.e., requirements and testing. For the test suite minimization problem and test case prioritization problem, four different VCS products with varying complexity were selected for the experiment. For the requirements allocation problem, one large-scale CPS was chosen. Notice that such threat to external validity is common to all empirical studies [3, 62].

5 PRACTICAL GUIDE

This section provides a practical guide (Figure 3) for selecting quality indicators, when assessing Pareto-based search algorithms. As mentioned before, the guide is derived based on: a) the theoretical foundations (*TF*); b) the extended literature review (*LR*); and c) the extensive experiment (*EE*). In Figure 3, we

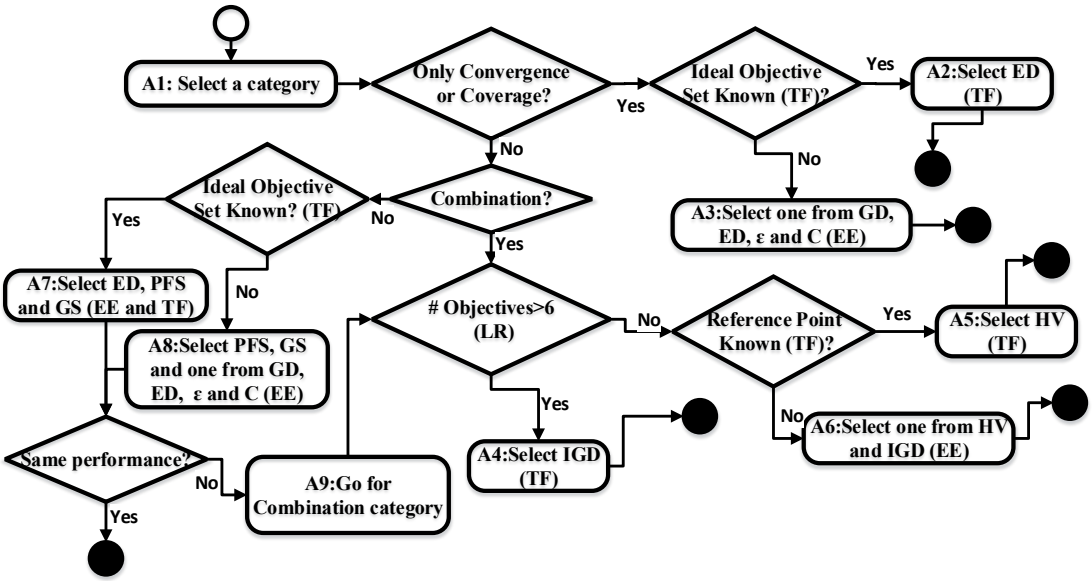

Figure 3 A practical guide for choosing quality indicators

explicitly show this information inside the brackets when recommending a quality indicator to apply.

As shown in Figure 3, a category of quality indicators should be first selected (***A1***):

1) Select *Convergence* or *Coverage* if a user only cares whether obtained solutions are optimal or not. In particular, for some SE problems (e.g., RA in our case), only one optimal solution is required regardless of the diversity of solutions. In this case, a dedicated quality indicator for *Convergence* is recommended. Moreover, if an ideal set of objective values (i.e., optimal objective values) is unknown for a specific SE problem (e.g., RA in our case), any of GD, ED, ϵ and C can be selected (***A3***) based on *EE*, i.e., the results showed that all of them indicate the same trend of performance when comparing algorithms. Otherwise, if an ideal objective set is known before, ED should be selected (***A2***) since it only requires an ideal objective set instead of a whole reference Pareto front based on *TF*.

2) Select *Combination* if a user prefers more diverse solutions to choose from in addition to *Convergence*. For example, a user can choose solutions based on the preference of objectives. In this case, the first selection criterion is based on the number of objectives to be optimized since the computation effort of HV increases exponentially with the number of objectives [39, 65]. From our experiment with three and four objectives (TM, TP and RA problems), there is no significant time difference for calculating HV and IGD, and from the extended literature review (Section 3), we observed that the number of objectives is always less than or equal to six for all the existing works that applied HV. Thus, according to *LR*, we set the threshold of the number of objectives as six for the guide, i.e., when the number of objectives for a problem are more than six, IGD should be selected (***A4***) since the computation effort of IGD is only in quadratic based on *TF*. When the number of objectives is less than or equal to six, HV should be selected (***A5***) if an accurate reference point is known, i.e., the worst values for all the objectives. That is because calculating HV only requires a reference point rather than an entire reference Pareto front (*TF*). Otherwise, if a reference point is not known before (e.g., the RA problem in our case), either HV or IGD can be chosen (***A6***) based on *EE* since HV and IGD indicates the same trend of performance for comparing algorithms.

3) Select *Convergence* or *Coverage* together with *Diversity*. It is also possible to evaluate *Convergence* (*Coverage*) and *Diversity* of Pareto fronts produced by search algorithms separately since dedicated quality indicators for *Convergence* (*Coverage*) and *Diversity* may be more accurate than a combined quality indicator (e.g., HV) [39, 65]. Notice that quality indicators of *Diversity* cannot be applied separately since it does not make sense in realistic situations that users only care about the diversity of solutions without considering they are optimal or not. In this case, we need to learn whether the ideal objective set is known for a SE problem. If the ideal objective set is known, ED from *Convergence* should be applied together with PFS and GS (*Diversity*) (***A7***) based on *TF* and *EE*, i.e., PFS and GS indicate different trends of performance when comparing algorithms. Otherwise, any quality indicator from *Convergence* or *Coverage* can be chosen together with PFS and GS (***A8***) based on *EE*. Notice that PFS and GS should be selected together since they reflect the diversity from two different perspectives, i.e., the number of obtained solutions and distribution of solutions. It is worth mentioning that the quality indicators from *Convergence* and *Diversity* may demonstrate completely different performance of algorithms and thus making it impossible to obtain a definite answer which algorithm is better. In this case, we recommend selecting the quality indicators from *Combination* (***A9***) since applying HV or IGD can tell us which algorithm is better by combining both *Convergence* and *Diversity*.

6 CONCLUSION AND FUTURE WORK

This paper provides a practical guide for the Search-based Software Engineering (SBSE) community to select proper quality indicators when assessing Pareto-based search algorithms in different software engineering applications. The guide is derived from: 1) theoretical foundations of quality indicators; 2) an extended literature review; and 3) an extensive experiment to evaluate eight quality indicators along with six Pareto-based search algorithms using three industrial problems.

In the future, we plan to involve more quality indicators into the guide, which have not been investigated by the SBSE community, e.g., purity [26], dominance ranking [39]. We also plan to employ more industrial problems from other domains with the aim to further improve the proposed practical guide.

ACKNOWLEDGEMENT

This research was supported by the Research Council of Norway (RCN) funded Certus SFI. Shuai Wang is also supported by RFF Hovedstaden funded MBE-CR project. Tao Yue and Shaukat Ali are also supported by RCN funded Zen-Configurator project, the EU Horizon 2020 project funded U-Test, RFF Hovedstaden funded MBE-CR project and RCN funded MBT4CPS project.

REFERENCES

[1] M. Harman, S.A. Mansouri and Y. Zhang, "Search Based Software Engineering: A Comprehensive Analysis and Review of Trends Techniques and Applications", Technical Report TR-09-03, Department of Computer Science, King College London, 2009.

[2] M. Harman, "Making the Case for MORTO: Multi Objective Regression Test Optimization", Proc. of the IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, pp. 111-114, 2011.

[3] S. Ali, L.C. Briand, H. Hemmati, and R. K Panesar-Walawege, "A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation," IEEE Transactions on Software Engineering 36 (6), pp. 742-762, 2010.

[4] S. Yoo, and M. Harman, "Pareto Efficient Multi-Objective Test Case Selection," Proc. of International Symposium on Software testing and analysis (ISSTA), pp. 140-150, 2007.[

[5] Y. Zhang, A. Finkelstein and M. Harman, "Search Based Requirements Optimization: Existing Work and Challenges," Requirements Engineering: Foundation for Software Quality, pp. 88-94, 2008.

[6] S. Wang, S. Ali and A. Gotlieb, "Cost-effective test suite minimization in product lines using search techniques", Journal of Systems and Software, vol (103), 370-391, 2015.

[7] C. Henard, M. Papadakis, M. Harman, and Y.L. Traon, "Combining Multi-Objective Search and Constraint Solving for Configuring Large Software Product Lines", Proc. of the 37th International Conference on Software Engineering (ICSE), 2015.

[8] S. Wang, S. Ali, T. Yue and M. Liaaen, 'UPMOA: An improved search algorithm to support user-preference multi-objective optimization", Proc. of International Symposium on Software Reliability Engineering, pp. 393-404, 2015.

[9] J.J. Durillo, A.J. Nebro, "jMetal: A Java framework for multi-objective optimization," Advances in Engineering Software 42, pp. 760-771. 2011.

[10] S. Wang, S. Ali and A. Gotlieb, and M. Liaaen, "A Systematic Test Case Selection Methodology for Product Lines: Results and Insights From an Industrial Case Study", Empirical Software Engineering Journal, pp. 1-37, 2014.

[11] Y. Li, T. Yue, S. Ali, K. Nie and L. Zhang, "Zen-ReqOptimizer: A Search-based Approach for Requirements Assignment Optimization", Empirical Software Engineering Journal, 2016.

[12] A.S. Sayyad, T. Menzies, H. Ammar, "On the value of user preferences in search-based software engineering: a case study in software product lines", Proc. of the International Conference of Software Engineering (ICSE), 492-501, 2013.

[13] A.S. Sayyad, H. Ammar, "Pareto-Optimal Search-Based Software Engineering (POSBSE): A literature Survey", Proc. of 2nd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, 21-27, 2013.

[14] Y. Zhang and M. Harman and A. Mansouri, "The SBSE Repository: A repository and analysis of authors and research articles on Search Based Software Engineering", CREST Centre, UCL.

[15] S. Wang, D. Buchmann, S, Ali, A. Gotlieb, D. Pradhan and M. Liaaen, "Multi-objective test prioritization in software product line testing: an industrial case study", Proc. of the 18th International Software Product Line Conference, pp. 32-41, 2014.

[16] S. Wang, S. Ali and A. Gotlieb, "Minimizing Test Suites in Software Product Lines Using Weighted-based Genetic Algorithms", Proc. of the Genetic and Evolutionary Computation Conference (GECCO), pp. 1493-1500, 2013.

[17] J. Brownlee, "Clever Algorithms: Nature-Inspired Programming Recipes," *ISBN: 978-1-4467-8506-5, 2011.*

[18] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," IEEE Trans on Evolutionary Computation, 6(2), pp. 182–197, 2002.

[19] A.J. Nebro, J.J. Durillo, F. Luna, B. Dorronsoro and E. Alba, "Design Issues in a Multiobjective Cellular Genetic Algorithm," Evolutionary Multi-Criterion Optimization, pp. 126-140, 2007.

[20] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," Proc. of the EUROGEN 2001-Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems", pp. 95-100, 2001.

[21] J.D. Knowles and D.W. Corne, "Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy," Evolutionary Computation 8(2), 149-172, 2000.

[22] A.J. Nebro, J.J. Durillo, J. Garcia-Nieto, C.A. Coello Coello, F. Luna, and E. Alba, "SMPSO: A new PSO-based Metaheuristic for Multi-objective Optimization," Proc. of the Symposium on Computational Intelligence in Multicriteria Decision-Making (MCDM), pp. 66-73, 2009.

[23] J.J. Durillo, A.J. Nebro, F. Luna, and E. Alba, "Solving Three-objective Optimization Problems using a New Hybrid Cellular Genetic Algorithm," Parallel Problem solving from nature- PPSN X. Lecture notes in computer science (5199), pp. 661-670, 2008.

[24] S. Wang, S. Ali, T. Yue, Ø. Bakkeli, M. Liaaen, "Enhancing Test Case Prioritization in an Industrial Setting with Resource Awareness and Multi-Objective Search", Proc. of the 38th International Conference on Software Engineering, 2016.

[25] T., Yue, and S., Ali, "Applying Search Algorithms for Optimizing Stakeholders Familiarity and Balancing Workload in Requirements Assignment", Proc. of ACM Genetic and Evolutionary Computation Conference (GECCO), pp. 1295-1302, 2014.

[26] J. Knowles, L. Thiele, and E. Zitzler, "A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers," Computer Engineering and Networks Laboratory (TIK), ETH Zurich, TIK Report 214, Feb. 2006.

[27] K., Deb, "Multi-objective optimization using evolutionary algorithms," John Wiley & Sons; 2001.

[28] E., Zitzler, L., Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," IEEE Trans Evol Comput; 3(4), 257–71, 1999.

[29] D.A., Van Veldhuizen, G.B., Lamont, "Multiobjective evolutionary algorithm research: A history and analysis, Tech. Rep. TR-98-03, Dept. Elec. Comput. Eng., Graduate School of Eng., Air Force Inst.Technol., Wright-Patterson, AFB, OH; 1998.

[30] T. Yue, S. Ali and B. Selic, "Cyber-physical system product line engineering: comprehensive domain analysis and experience report", Proc. of International Conference on Software Product Line, pp. 338-347, 2015.

[31] L. Briand, D. Falessi, S. Nejati, M. Sabetzadeh, and T. Yue, "Research-based innovation: A tale of three projects in model-driven engineering," Proc. of ACM/IEEE 15th International Conference on Model Driven Engineering Languages and Systems (MODELS), pp. 793-809, 2012.

[32] J.L. Cochrane and M. Zeleny, "Multiple Criteria Decision Making," University of South Carolina Press, 1973.

[33] C.M., Fonseca, P.J., Flemming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms-part ii: application example," IEEE Trans System, Man, Cybern 28 (1), pp. 38–47, 1998.

[34] M., Tanaka, H., Watanabe, Y., Furukawa, T., Tanino, "GA-based decision support system for multicriteria optimization," Proc. of the IEEE international conference on systems, man, and cybernetics, vol. 2, pp. 1556–1561, 1995.

[35] A.J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J.J. Durillo and A. Beham, "AbYSS: Adapting Scatter Search to Multiobjective Optimization," IEEE Trans Evol Comput 12(4), pp. 439–457., 2008.

[36] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff and E. Tsang, “Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion, Proc. of IEEE Congress on evolutionary computation (CEC), pp. 3234–3241. 2006.

[37] W. K. G. Assunção, T. E. Colanzi, A. T. R. Pozo and S. R. Vergilio, "Establishing Integration Test Orders of Classes with Several Coupling Measures," Proc. of GECCO, Dublin, Ireland, pp. 1867-1874. 2011.

[38] J. T. de Souza, C. L. Maia, F. G. de Freitas and D. P. Coutinho, "The human competitiveness of search based software engineering," Proc. of SSBSE, pp. 143-152, 2010.

[39] M. Li, S. Yang and X. Liu, “Diversity comparison of Pareto front approximations in many-objective optimization”, IEEE Trans Cybern, 44(12), pp. 2568-2584, 2014.

[40] J. M. Chaves-González and M. A. Pérez-Toledano, “Differential Evolution with Pareto Tournament for the Multi-objective Next Release Problem”, Applied Mathematics and Computation, vol. 252, pp. 1-13, 2015.

[41] Amarjeet and J. K. Chhabra, “An Empirical Study of the Sensitivity of Quality Indicator for Software Module Clustering”, Proc. of 7th International Conference on Contemporary Computing (IC3 '14), pp. 206-211, 2014.

[42] W.K.G. Assunção, T.E. Colanzi, S.R. Vergilio and A. Pozo, “A multi-objective optimization approach for the integration and test order problem”, Information Sciences, Vol. 267, pp. 119-139, 2014.

[43] G. Guizzo, T.E. Colanzi and S.R. Vergilio, “A Pattern-Driven Mutation Operator for Search-Based Product Line Architecture Design”, Proc. of the 6th International Symposium on Search-Based Software Engineering (SSBSE), pp. 77-91, 2014.

[44] M. Harman, Y. Jia, J. Krinke W.B. Langdon, J. Petke and Y. Zhang, “Search Based Software Engineering for Software Product Line Engineering: A Survey and Directions for Future Work”, Proc. of the 18th International Software Product Line Conference (SPLC), pp. 5-18, 2014.

[45] V. Hrubá, B. Křena, Z. Letko, H. Pluháčková and T. Vojnar, “Multi-objective Genetic Optimization for Noise-Based Testing of Concurrent Software”, Proc. of the 6th International Symposium on Search-Based Software Engineering (SSBSE), pp. 107-122, 2014.

[46] M.R. Karim and G. Ruhe, “Bi-Objective Genetic Search for Release Planning in Support of Themes”, Proc. of the 6th International Symposium on Search-Based Software Engineering (SSBSE), pp. 123-137, 2014.

[47] L. Li, M. Harman, E. Letier and Y. Zhang, “Robust Next Release Problem: Handling Uncertainty During Optimization”, Proc. of the Conference on Genetic and Evolutionary Computation (GECCO), pp. 1247-1254, 2014.

[48] R.E. Lopez-Herrejon, J. Ferrer, F. Chicano, A. Egyed and E. Alba, “Comparative Analysis of Classical Multi-Objective Evolutionary Algorithms and Seeding Strategies for Pairwise Testing of Software Product Lines”, Proc. of IEEE Congress on Evolutionary Computation (CEC), pp. 387-396, 2014.

[49] F. Luna, D.L. González-Álvarez, F. Chicano and M.A. Vega-Rodríguez, “The Software Project Scheduling Problem: A Scalability Analysis of Multi-objective Metaheuristics”, Applied Soft Computing, Vol. 15, pp. 136-148, 2014.

[50] M.W. Mkaouer, M. Kessentini, S.,Bechikh, K.,Deb and M.Ó. Cinnéide, “High Dimensional Search-based Software Engineering: Finding Tradeoffs among 15 Objectives for Automating Software Refactoring using NSGA-III”, Proc. of the Conference on Genetic and Evolutionary Computation (GECCO), pp. 1263-1270, 2014.

[51] M.W. Mkaouer, M. Kessentini, S. Bechikh, and M.Ó. Cinnéide, “A Robust Multi-objective Approach for Software Refactoring under Uncertainty”, Proc. of the 6th International Symposium on Search-Based Software Engineering (SSBSE), pp. 168-183, 2014.

[52] S. Nejati and L.C. Briand, “Identifying Optimal Trade-offs between CPU Time Usage and Temporal Constraints using Search”, Proc. of the International Symposium on Software Testing and Analysis (ISSTA), pp. 251-261, 2014.

[53] A. Ramrez, J.R. Romero and S. Ventura, “On the Performance of Multiple Objective Evolutionary Algorithms for Software Architecture Discovery”, Proc. of Conference on Genetic and Evolutionary Computation (GECCO), pp. 1287-1294, 2014.

[54] L.S. de Souza, R.B.C. Prudencio and F.A. Barros, “A Comparison Study of Binary Multi-Objective Particle Swarm Optimization Approaches for Test Case Selection”, Proc. of the IEEE Congress on Evolutionary Computation (CEC), pp. 2164-2171, 2014.

[55] A. Sureka, “Requirements Prioritization and Next-Release Problem under Non-Additive Value Conditions”, Proc. of the 23rd Australian Software Engineering Conference (ASWEC), pp. 120-123, 2014.

[56] W.K.G. Assunção, T.E. Colanzi, S.R. Vergilio and A. Pozo, “On the Application of the Multi-Evolutionary and Coupling-Based Approach with Different Aspect-Class Integration Testing Strategies”, Proc. of the 5th International Symposium on Search-Based Software Engineering (SSBSE), pp. 19-33, 2013.

[57] M. Bozkurt, “Cost-aware Pareto Optimal Test Suite Minimisation for Service-centric Systems”, Proc. of the Conference on Genetic and Evolutionary Computation (GECCO), pp. 1429-1436, 2013.

[58] L.C. Briand, Y. Labiche and K. Chen, “A Multi-Objective Genetic Algorithm to Rank State-Based Test Cases”, Proc. of the 5th International Symposium on Search-Based Software Engineering (SSBSE), pp. 66-80, 2013.

[59] M.W. Mkaouer, M. Kessentini, S. Bechikh and D.R. Tauritz, “Preference-based Multi-Objective Software Modelling”, Proc. of 1st International Workshop on Combining Modelling and Search-Based Software Engineering (CMSBSE '13), pp. 61-66, 2013.

[60] A. Ouni, M. Kessentini, H. Sahraoui and M.S. Hamdi, “The Use of Development History in Software Refactoring using A Multi-objective Evolutionary Algorithm”, Proc. of Conference on Genetic and Evolutionary Computation (GECCO), pp. 1461-1468, 2013.

[61] A. Arcuri, and L.C. Briand, “A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering,” Proc. of International Conference on Software Engineering (ICSE), pp. 21-28, 2011.

[62] M.O. Barros and A.C. Dias-Neto, “Threats to Validity in Search-based Software Engineering Empirical Studies”, UNIRIO - Universidade Federal do Estado do Rio de Janeiro0006/2011, 2011.

[63] D.J. Sheskin, “Handbook of Parametric and Nonparametric Statistical Procedures”, 2003.

[64] M. Kendall, "A New Measure of Rank Correlation". Biometrika 30 (1–2): 81–89, 1938.

[65] E. Zizler, J. Knowles and L. Thiele, “Quality Assessment of Pareto Set Approximations,” Multiobjective Optimization Lecture Notes in Computer Science, pp. 373-404, 2008.