How to Evaluate Solutions in Pareto-Based Search-Based Software Engineering: A Critical Review and Methodological Guidance

Miqing Li, Tao Chen, *Member, IEEE*, and Xin Yao, *Fellow, IEEE*

Abstract—With modern requirements, there is an increasing tendency of considering multiple objectives/criteria simultaneously in many Software Engineering (SE) scenarios. Such a multi-objective optimization scenario comes with an important issue — how to evaluate the outcome of optimization algorithms, which typically is a set of incomparable solutions (i.e., being Pareto nondominated to each other). This issue can be challenging for the SE community, particularly for practitioners of Search-Based SE (SBSE). On one hand, multi-objective optimization could still be relatively new to SE/SBSE researchers, who may not be able to identify the right evaluation methods for their problems. On the other hand, simply following the evaluation methods for general multi-objective optimization problems may not be appropriate for specific SBSE problems, especially when the problem nature or decision maker's preferences are explicitly/implicitly known. This has been well echoed in the literature by various inappropriate/inadequate selection and inaccurate/misleading use of evaluation methods. In this paper, we first carry out a systematic and critical review of quality evaluation for multi-objective optimization in SBSE. We survey 717 papers published between 2009 and 2019 from 36 venues in seven repositories, and select 95 prominent studies, through which we identify five important but overlooked issues in the area. We then conduct an in-depth analysis of quality evaluation indicators/methods and general situations in SBSE, which, together with the identified issues, enables us to codify a methodological guidance for selecting and using evaluation methods in different SBSE scenarios.

Index Terms—Search-based software engineering, multi-objective optimization, Pareto optimization, quality evaluation, quality indicators, preferences

1 Introduction

In software engineering (SE), it is not uncommon to face a scenario where multiple objectives/criteria need to be considered simultaneously [20], [50]. In such scenarios, there is usually no single optimal solution but rather a set of Pareto optimal solutions (termed a Pareto front in the objective space), i.e., solutions that cannot be improved on one objective without degrading on some other objective. To tackle these multi-objective SE problems, different problem-solving ideas have been brought up. One of them is to generate a set of solutions to approximate the Pareto front. This, in contrast with the idea of aggregating objectives (by weighting) into a single-objective problem, provides different trade-offs between the objectives, from which the decision maker (DM) can choose their favorite solution.

In such Pareto-based optimization, a fundamental issue is to evaluate the quality of solution sets (populations) obtained by computational search methods (e.g., greedy search, heuristics, and evolutionary algorithms) in order to know how well the methods perform. Since the obtained solution sets are typically not comparable to each other with respect to Pareto dominance,[1] how to evaluate/compare them is non-trivial. A straightforward way is to plot the solution sets (by scatter plot) for an intuitive evaluation/comparison. Yet this only works well for the bi-objective case, and when the number of objectives reaches four, it is impossible to show the solution sets by scatter plot. More importantly, visual comparison cannot provide a quantitative comparative result between the solution sets.

Another way to evaluate the solution sets is to report their descriptive statistical results, such as the best, mean, and median values on each objective from each solution set. This has been profoundly used in Search-Based SE (SBSE) [6], [9], [18], [22], [37], [130], [134]. However, some of these statistic indexes may easily give misleading evaluation results. That is, a solution set which is evaluated better than its competitor could be never preferred by the DM

- *Miqing Li is with the School of Computer Science, University of Birmingham, B15 2TT Birmingham, U.K. E-mail: m.li.8@bham.ac.uk.*
- *Tao Chen is with the Department of Computer Science, Loughborough University, LE11 3TU Loughborough, U.K. E-mail: t.t.chen@lboro.ac.uk.*
- *Xin Yao is with the Shenzhen Key Laboratory of Computational Intelligence (SKyLoCI), Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, P. R. China, and also with the CERCIA, School of Computer Science, University of Birmingham, B15 2TT Birmingham, U.K. E-mail: xiny@sustc.edu.cn.*

Manuscript received 17 Feb. 2020; revised 29 Sept. 2020; accepted 17 Oct. 2020. Date of publication 9 Nov. 2020; date of current version 16 May 2022. (Corresponding author: Tao Chen and Xin Yao.) Recommended for acceptance by S. Nejati. Digital Object Identifier no. 10.1109/TSE.2020.3036108

1. A solution set **A** is said to (Pareto) dominate a solution set **B** if for any solution in **B** there exists at least one solution in **A** dominating it, where the dominance relation between two solutions can be seen as a natural "better" relation of the objective vectors, i.e., better or equal on all the objectives, and better at least on one objective [154].

under any circumstance. This will be explained in detail in the text later (Section 5.1.2).

Generic quality indicators, which is arguably the most straightforward evaluation method that maps a solution set to a real number that indicates one or several aspects of the set's quality, have emerged in the fields of evolutionary computation and operational research [10], [62], [116], [154]. Today analyzing and designing quality indicators has become an important research topic. There are hundreds of them in literature [75], with some measuring closeness of the solution set to the Pareto front, some gauging diversity of the solution set, some considering a comprehensive evaluation of the solution set, etc. The SBSE community benefits from this prosperity. A common practice in SBSE is to use some well-established quality indicators, such as hypervolume (*HV*) [153] and inverted generational distance (*IGD*) [26], to evaluate the obtained solution sets. However, some indicators may not be appropriate when it comes to practical SE optimization scenarios. For example, since the Pareto front of a practical SBSE problem is typically unavailable, indicators that require a reference set that well represents the problem's Pareto front may not be well suited [75], such as *IGD*.

More importantly, specific SBSE problems usually have their own nature and requirements. Simply following indicators that were designed for general Pareto-based optimization may fail to reflect these requirements. Take the software product line configuration problem as an example. In this problem, the objective of a product's correctness is always prioritized above other objectives (e.g., cost and richness of features). Equally rating these objectives by using generic indicators like *HV* (which in fact has been commonly practiced in the literature [64], [93], [118], [120]) may return the DM meaningless solutions, i.e., invalid products with good performance on the other objectives. This situation also applies to the test case generation problem, where the DM may first favor the full code coverage and then others (e.g., low cost).

Moreover, some SBSE problems may associate with the DM's explicit/implicit assumptions or preferences between the objectives. It is expected for researchers to select indicators bearing these assumptions/preferences in mind. For instance, in many SE scenarios, the DM may prefer well-balanced trade-off solutions (i.e., knee points on the Pareto front) between conflicting objectives. An example is that when optimizing the conflicting non-functional quality of a software system (e.g., latency and energy consumption), knee points are typically the most preferred solutions, as in such case, it is often too difficult, if not impossible, to explicitly quantify the relative importance between objectives. Under this circumstance, quality indicators that treat all points on the Pareto front equally (such as *IGD*) may not be able to reflect this preference, despite the fact that they have been frequently used in such scenarios [37], [89].

Finally, the study of quality indicator selection itself in multi-objective optimization is in fact a non-trivial task. Each indicator has its own specific quality implication, and the variety of indicators in literature can easily overwhelm the researchers and practitioners in the field. On the one hand, an accurate categorization of quality indicators is of high importance. Failing to do so can easily result in a misleading understanding of search algorithms' behavior, see [74]. On the other hand, even under the same category, different indicators are of distinct quality implications, e.g., *IGD* prefers uniformly distributed solutions and *HV* is in favor of knee solutions. A careful selection needs to be made to ensure the considered quality indicators to be in line with the DM's preferences. In addition, many quality indicators involve critical parameters (e.g., the reference point in the *HV* indicator). It remains unclear how to properly set these parameters under different circumstances, particularly in the presence of the DM's preferences.

Given the above, this paper aims to systematically survey and justify some of the overwhelming issues when evaluating solution sets in SBSE, and more importantly, to provide a systematic and methodological guidance of selecting/using evaluation methods and quality indicators in various Pareto-based SBSE scenarios. Such a guidance is of high practicality to the SE community, as research from the well-established community of multi-objective optimization may still be relatively new to SE researchers and practitioners. This is, to the best of our knowledge, the first work of its kind to specifically target the quality evaluation of solution sets in SBSE based on a theoretically justifiable methodology.

It is worth mentioning that recently there are some attempts from the perspective of empirical studies to provide guidelines for quality indicator selection [4], [138]. Wang *et al.* [138] proposed a practical guide for SBSE researchers based on the observations from experimental results in three SBSE real-world problems. Ali *et al.* [4] significantly extended that work and provided a set of guidelines based on the observations from experimental results in nine SBSE problems from industrial, real-world and open-source projects. However, observations drawn from an empirical investigation on specific SBSE scenarios may not be generalizable. Indeed, different DMs may prefer different trade-offs between objectives, even for the same optimization problem, as nondominated solutions are in essence incomparable. Observations obtained on one (or some) scenario(s) is therefore difficult to be transferred into other scenarios. As a result, a general and theoretically sound guidance based upon the DM's preferences is needful since the fundamental goal of multi-objective optimization is to supply the DM a set of solutions which are the most consistent with their preferences.

For the rest of the paper, we start by providing some background knowledge of multi-objective optimization and quality evaluation (Section 2). Then, we conduct a systematic survey of the SE problems that involve Pareto-based search (hence termed Pareto-based SBSE problems) across all phases in the classic Software Development Life Cycle (SDLC) [112], along with their problem nature, the DM's preferences, the quality indicators and evaluation methods used (Sections 3 and 4). The survey has covered 717 searched papers published between 2009 and 2019, on 36 venues from seven repositories, leading to 95 prominent primary studies in the SBSE community. This is followed by a critical review on the evaluation method selection and use in those primary studies, based on which we identify five important issues that have been significantly overlooked (Section 5). Then, we carry out an in-depth analysis of frequently-used quality indicators in the area (Section 6), in order to make it clear which indicators fit in which situation.

Next, to mitigate the identified issues in the future work of SBSE, we provide a methodological guidance and procedure of selecting, adjusting, and using evaluation methods in various SBSE scenarios (Section 7). The last three sections are devoted to threats to validity, related work, and conclusion, respectively.

2 Preliminaries on Multi-Objective Optimization

Multi-objective optimization is an optimization scenario that considers multiple objectives/criteria simultaneously. Apparently, when comparing solutions[2] in multi-objective optimization, we need to consider all the objectives of a given optimization problem. There are two commonly used terms to define the relations between solutions, Pareto dominance and weak Pareto dominance.

Without loss of generality, let us consider a minimization scenario. For two solutions $\mathbf{a}, \mathbf{b} \in Z$ ($Z \subset \mathbb{R}^m$, where m denotes the number of objectives), solution $\mathbf{a}$ is said to *weakly dominate* $\mathbf{b}$ (denoted as $\mathbf{a} \preceq \mathbf{b}$) if $\mathbf{a}_i \leq \mathbf{b}_i$ for $1 \leq i \leq m$. If there exists at least one objective j on which $\mathbf{a}_j < \mathbf{b}_j$, we say that $\mathbf{a}$ *dominates* $\mathbf{b}$ (denoted as $\mathbf{a} \prec \mathbf{b}$). A solution $\mathbf{a} \in Z$ is called *Pareto optimal* if there is no $\mathbf{b} \in Z$ that dominates $\mathbf{a}$. The set of all Pareto optimal solutions of a multi-objective optimization problem is called its *Pareto front*.

The above relations between solutions can immediately be extended to between sets. Let $\mathbf{A}$ and $\mathbf{B}$ be two solution sets.

Relation 1. *[Dominance between two sets [154]] We say that* $\mathbf{A}$ *dominates* $\mathbf{B}$ *(denoted as* $\mathbf{A} \prec \mathbf{B}$*) if for every solution* $\mathbf{b} \in \mathbf{B}$ *there exists at least one solution* $\mathbf{a} \in \mathbf{A}$ *that dominates* $\mathbf{b}$.

Relation 2. *[Weak Dominance between two sets [154]] We say that* $\mathbf{A}$ *weakly dominates* $\mathbf{B}$ *(denoted as* $\mathbf{A} \preceq \mathbf{B}$*) if for every solution* $\mathbf{b} \in \mathbf{B}$ *there exists at least one solution* $\mathbf{a} \in \mathbf{A}$ *that weakly dominates* $\mathbf{b}$.

We can see that the *weak dominance* relation between two sets does not rule out their equality, while the *dominance* relation does but it also rules out the case that there exist same solutions with respect to the two sets. Thus, we may need another relation to define that $\mathbf{A}$ is *generally* better than $\mathbf{B}$.

Relation 3. *[Better relation between two sets [154]] We say that* $\mathbf{A}$ *is better than* $\mathbf{B}$ *(denoted as* $\mathbf{A} \lhd \mathbf{B}$*) if for every solution* $\mathbf{b} \in \mathbf{B}$ *there exists at least one solution* $\mathbf{a} \in \mathbf{A}$ *that weakly dominates* $\mathbf{b}$*, but there exists at least one solution in* $\mathbf{A}$ *that is not weakly dominated by any solution in* $\mathbf{B}$.

The *better* relation $\lhd$ reflects the most general assumption of the DM's preferences to compare solution sets. However, the *better* relation may leave many solution sets incomparable since it is very likely that there exist some solutions being nondominated with each other in the sets. As typically the size of the Pareto front of a multi-objective optimization problem can be prohibitively large or even infinite, a solution set that can well represent the Pareto front is preferred, especially when the DM's preferences are unavailable. This leads to four quality aspects that we often care about [75] — Convergence, how close the solution set is to the Pareto front; Spread, how large the region that the set covers is; Uniformity, how evenly the solutions are distributed in the set; Cardinality, how many (unique) nondominated solutions are in the set. Over the last three decades, numerous quality evaluation methods have been developed for these four aspects. Among them, quality indicators are the most popular ones [75]. They typically map a solution set to a real number that indicates one or more of the four quality aspects, defining a total order among solution sets for comparison.

3 Review Methodology

Despite that we have randomly witnessed several inappropriate evaluations of solution sets through our own experiences working in SBSE, the key challenge in this work remains *to systematically understand what the trends of issues on the way to evaluate solution sets in the SBSE community are, if any*, so that a clarified guidance can be drawn thereafter. To this end, we at first conduct a systematic literature review covering the studies published between 2009 and 2019. A reason that we consider this period is that one of the most well-known SBSE surveys (Harman *et al.* [50]) has covered the SBSE work from 1976 to 2008, and we try to cover the period that has not been reviewed by that work. In addition, since 2010 or so, there is a rapidly increasing interest in using Pareto-based optimization techniques to deal with SBSE. Given these two reasons, we have chosen 2009 as the starting year of our review. Having said that, we do not aim to provide a complete review on all parts of the SBSE work, but specifically on the aspects related to the major trends of evaluating solution sets.

Our review methodology follows the best practice of a systematic literature review for software engineering [60], consisting of search protocol, inclusion/exclusion criteria, formal data collection process, and pragmatic classification. Specifically, the review aims to answer three research questions (RQs):

- *RQ1:* What evaluation methods have been used to evaluate solution sets in SBSE? (*What*)
- *RQ2:* What are the reasons and practice of using the generic quality indicators? (*Why* and *How*)
- *RQ3:* In what domain and context the evaluation methods have been used? (*Where*)

3.1 Overview of the Literature Review Protocol

As shown in Fig. 1, our literature review protocol obtains inputs from various sources via automatic search, which we will elaborate into details in Section 3.2. This gives us 3,156 studies including duplication. We then removed any duplicated studies by automatically matching their titles,[3] leading to 717 *searched studies*. Next, we filtered the searched studies by reading through their titles and abstracts using two simple filtering criteria:

2. For simplicity, we refer to an objective vector as a *solution* and the outcome of a multi-objective optimizer as a *solution set*.

3. Patents, citation entries, inaccessible papers, and any other non-English documents were also eliminated.

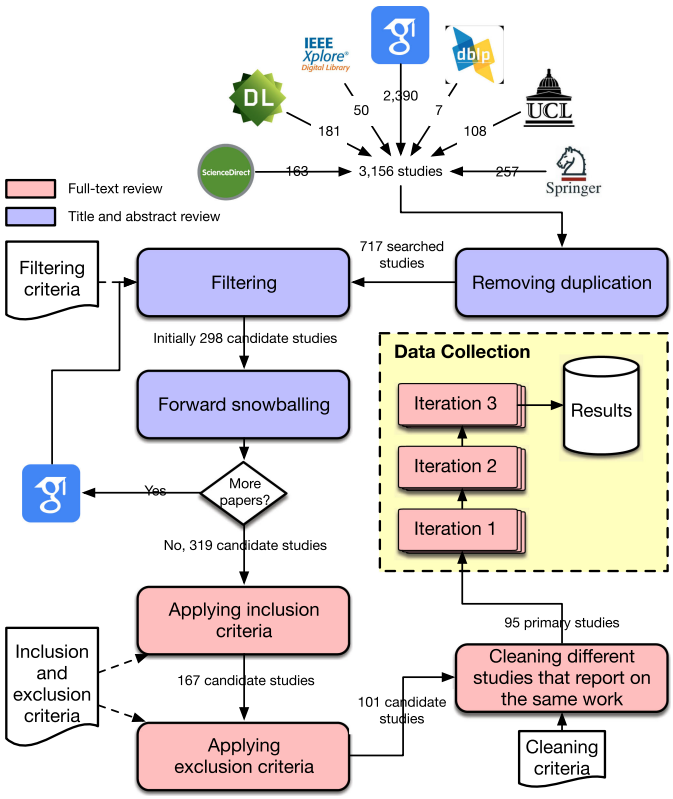


Fig. 1. Systematic literature review protocol.

- The paper is not relevant to SBSE.
- The paper does not investigate or compare multi-objective search/optimization.

A study was ruled out if it meets any of the above two filtering criteria. The aim of filtering is to reduce the found studies to a much smaller and more concise set, namely the *candidate studies*. As can be seen, the process resulted in 298 candidate studies prior to manual search. Starting from the 298 studies, we adopted an iterative forward snowballing as suggested by Felizardo *et al.* [33], where the newly included studies (after filtering) were placed into the next snowballing round. The reason why we did not do backward snowballing is because we have set strict time scale on the studies searched within the last decade, and thus backward snowballing would too easily violate such a requirement of timeliness. To avoid a complicated process, we relied on Google Scholar as the single source for forward snowballing, as it returns most of the searched results as shown in Fig. 1 and has been followed by software engineering surveys [42]. This snowballing stopped when no new studies can be found and it eventually led to 319 candidate studies, upon which the procedure for the full-text review begins.

At the next stage, we reviewed all the 319 studies and temporarily included studies using the inclusion criteria from Section 3.3, which resulted in 167 candidate studies. We then applied the exclusion criteria (see Section 3.3) to extract the temporarily included studies, leading to 101 candidate studies. By using the cleaning criteria specified in Section 3.3, a further cleaning process was conducted to prune different studies that essentially report on the same work, e.g., conference extended journal papers. All the processes finally produced 95 *primary studies* for data analysis and collection.

On these 95 primary studies, we conducted systematic and pragmatic data collection via three iterations, whose details are given in Sections 3.4 and 3.5. The summarized results were reported thereafter.

3.2 Scope and Search String

From 12th to 19th Aug 2019, we conducted an automatic search over a wide range of scientific literature sources, including ACM Library, IEEE Xplore, ScienceDirect, SpringerLink, Google Scholar, DBLP and the SBSE repository maintained by the CREST research group at UCL.[4]

We used a search string that aims to cover a variety of problem nature and application domains with respect to multi-objective optimization. Synonyms and keywords were properly linked via logical operators (AND, OR) to build the search term. The final search string is shown as below:

> *("multi objective" OR "multi criteria" OR "Pareto based" OR "non dominated" OR "Pareto front") AND "search based software engineering" AND optimization*

We conducted a full-text search on ACM Library, IEEE Xplore, ScienceDirect, SpringerLink, and Google Scholar, but rely on searching the title only for DBLP and UCL's SBSE repository, due to their restricted feature. Since DBLP's search feature cannot handle the whole search string, we paired each term in the first bracket with "`search based software engineering`" to run the search independently and collect all results returned. We omitted "`optimization`" as it rarely appears together with "`search based software engineering`" in the title, and having it along would produce many irrelevant results. Due to the similar reason, for the UCL's SBSE repository, we searched each term from the first bracket independently, as it is known that all the studies in this source are SBSE related.

On all the sources except DBLP and UCL's SBSE repository, we tried two versions of the search string: one with a hyphen between the commonly used terms (e.g., "`multi(-)objective`") and another without. The returned results with the highest number of items were used.[5] In particular, when searching on UCL's SBSE repository, the results of these two versions were combined, for example, "`multi objective`" and "`multi-objective`" would lead to different results. We recorded all the results returned under semantically equivalent terms.

3.3 Inclusion, Exclusion, and Cleaning Criteria

For all the candidate studies identified, we first extract the primary studies by using the inclusion criteria as below; studies meeting all of the criteria were temporarily chosen as the primary studies:

4. http://crestweb.cs.ucl.ac.uk/resources/sbse_repository
5. This is because their results are not mutually exclusive, e.g., on Google Scholar, "`multi objective`" would also return all the studies that contain "`multi-objective`" but not the other way around.

1) The study primarily focuses on (or has a section that discusses) a Pareto-based multi-objective solution to the SBSE problem. This means we do not consider papers that utilize the multi-objective treatment that relies on objective aggregation (e.g., weighted sum), unless they have explicitly compared the solution against a Pareto-based multi-objective solution. This is reasonable as if a clear aggregation of objectives can be defined, then there would be almost no need to select quality indicators but rely on the said aggregation to obtain a utility value for comparison.
2) The study explicitly or implicitly discusses (or at least makes assumptions about) the DM's preferences/contextual information between the objectives for the SBSE problem in hand. By implicit discussion, we mean that the study does not clearly state the assumptions, but such assumptions can be easily interpreted from the paper. For example, in the software product line configuration problem, some studies do not explicitly study the assumptions, but the number of valid products (one objective to be optimized) is solely used as an indicator to compare the peer approaches, which gives a clear indication that it is more important than the other objectives. Note that this also includes the assumption of no preferences/contextual information.
3) The SBSE problem in hand can be framed into at least one phase of the classic SDLC [112].
4) The study uses at least one search algorithm to solve the problem.
5) The study includes quantitative experimental results with clear instructions on how the results were obtained.
6) The study uses at least one method to evaluate the experimental results.

Subsequently, studies meeting any of the exclusion criteria below are filtered out from the temporary primary studies:

1) The study neither explicitly nor implicitly mentions SBSE, where the computational search is the key.
2) The study is not "highly visible" or widely followed. We used the citation information from Google Scholar as a single metric to (partially) assess the impact of a study.[6] In particular, we follow a pragmatic strategy that: a study has 5 citations per year from its year of publication is counted in, e.g., a 2010 study would expect to have at least 45 citations.[7] The only exception is for the work published in the year of writing this article (i.e., 2019), we consider those that were published for shorter than 6 months and have been cited by more than once, together with the pre-press ones that have not yet been given an issue number regardless of their citation counts. The reasons behind this setting are three-folds:

(a) We do not attempt to provide a comprehensive survey on the whole SBSE field, but rather to gather the major trends on how solution sets are evaluated, which can at least provide some sources for detailed analysis and discussion. Therefore, some metrics are required to ensure a trade-off between the trend coverage and a reasonably required effort for detailed data collections. This is similar to a sampling of the literature with the aim to gather the "representative" samples. This approach was adopted by many works, such as [40] where they used the citation count from Google Scholar as a threshold to select studies for review, as we did in this work.

(b) It is not uncommon to see that software engineering surveys are conducted using some metrics to measure the "impact" of a work. For example, some restrict their work only at what the authors believe to be premium venues [42], others use a threshold on the impact factors of the published journals, e.g., Cai and Card [12] use 0.55 and Zou *et al.* [155] use 2.0. In our case, it may not be a best practice to apply a metric at the venue level as the SBSE work is often multi-disciplinary (as we will show in Table 2) — it is difficult to quantify the "impact" across communities. We, therefore, have taken a measurement at the paper level based on the citation counts from Google Scholar, which has been used as the sole metric to differentiate between the studies in some prior work [27], [40], [42].

(c) Indeed, there is no rule to set the citation threshold. The settings in this work were taken from the (rounded) average figure within the population of the candidate studies. These may seem very high at the first glance probably due to two reasons: (i) by publication date, we meant the official date that the work appears on the publisher's webpage (for journal work, this means it has been given an official issue number). Yet, it is not uncommon that many studies are made citable as pre-prints before the actual publication, e.g., ICSE often has around 6 months gap between notification and official publication, and there is an even larger gap for some journals. This has helped to accumulate citations. (ii) Google Scholar counts the citations made by any publicly available documents and self-citation, which can still be part of the impact but implies their citation count may be higher than those purely made by peer-reviewed publications. Nevertheless, this could indeed pose a threat of construct validity, which we will discuss in Section 8.

TABLE 1
Data Collection Items

ID	Item	Questions
I_1	Author(s)	N/A
I_2	Year	N/A
I_3	Title	N/A
I_4	Venue (journal or conference)	N/A
I_5	Citation count	N/A
I_6	Indicator and method	RQ1
I_7	Stated quality aspects	RQ2
I_8	Reference point/front	RQ2
I_9	# objectives	RQ3
I_{10}	SBSE problem	RQ3
I_{11}	DM's preferences and contextual information	RQ3

6. Admittedly, there is no metric that is able to well quantify the impact of a paper. Nevertheless, the citation count can indicate something about a paper, e.g., its popularity.
7. All the citations were counted by 23rd Nov 2019.

3) The study is a short paper, i.e., shorter than 8 pages (double column) or 15 pages (single column).
4) The study is a review, survey, or tutorial.
5) The study is published in a non-peer-reviewed public venue, e.g., arXiv.

Finally, if multiple studies of the same research work are found, we applied the following cleaning criteria to determine if they should all be considered. The same procedure is applied if the same authors have published different studies for the same SBSE approach, and thereby only significant contributions are analyzed for the review.

- All studies are considered if they report on the same SBSE problem but different solutions.
- All studies are considered if they report on the same SBSE problem and solutions but have different assumptions about the DM's preference, nature of the problem, or new findings of the problem.
- When the above two points do not hold, only the latest version or the extended journal version is considered.

3.4 Data Items Analysis and Classification Strategy

The items to be collected when reviewing the details of the primary studies have been shown in Table 1. We now describe their design rationales and the procedure to extract and classify the data from each item.

The data for I_1 to I_5 is merely used as the meta-information of the primary studies. I_6, which answers *RQ1*, is the key item of our review. The evaluation method(s) used can be easily identified in a study, most commonly at the *Experiment* section. In general, apart from identifying the evaluation methods used in each study, we also seek to classify them into the following four categories:

- *Generic Quality Indicator:* This refers to indicators that are designed to evaluate the quality of solution sets for generic multi-objective optimization problems (e.g., *HV*, *IGD* and *Spread*), as documented by Li and Yao [75]. Formally, a quality indicator is a metric that maps a set of solutions (i.e., solution vectors) to a real number that indicates one or several aspects of the solution set quality [74], [75], e.g., to indicate how close the set is to the Pareto front and how evenly solutions are distributed in the set.
- *Solution Set Plotting (SSP):* This is a straightforward way to evaluate solution sets — visualizing the results by plotting them.
- *Descriptive Objective Evaluation (DOE):* This resorts to the direct statistical results of objective values, e.g., the best/mean/median of the solution set on each objective.
- *Problem Specific Indicator (PSI):* This refers to indicators that are not used for generic multi-objective optimization problems, but specifically designed for a given SBSE problem.

I_7 is heavily relevant to I_6, but requiring more detailed inspection to the studies. By this means, we aim to collect information about when a generic quality indicator is used, what quality aspect the study seeks to evaluate by it (for *RQ2*), which is the key reason of why such an indicator is chosen. I_7 is classified based on the four quality aspects of a solution set as concluded by Li and Yao [75], i.e., Convergence, Spread, Uniformity, and Cardinality. For each study, we first looked for the section where the generic quality indicators are explained, if no information found, we then searched for every place where the generic quality indicators are mentioned. We classify each indicator into the quality aspects based on whether their keywords have been clearly mentioned, otherwise, the indicator is marked as *Unknown* under I_7 of a study.

For I_8, we wish to understand how the generic quality indicators are used, as some of them requiring a reference point (e.g., *HV*) or a reference Pareto front (e.g., *GD* and

TABLE 2
The Reviewed Studies Counts and Venues

Journal	Search	Candidate	Primary
ACM Transactions on Software Engineering and Methodology	15	10	8
Elsevier Information and Software Technology	65	33	6
Elsevier Applied Soft Computing	17	4	0
Springer Automated Software Engineering	16	6	2
IEEE Transactions on Software Engineering	79	41	9
Springer Empirical Software Engineering	37	17	3
Elsevier Future Generation Computing Systems	3	3	1
Springer Soft Computing	11	4	0
IEEE Transactions on Evolutionary Computation	13	3	2
IEEE Transactions on Services Computing	6	3	3
Elsevier Journal of Systems and Software	70	35	8
Elsevier Information Sciences	18	7	3
Springer Requirements Engineering	5	3	1
Springer Software Quality Journal	12	7	2
Wiley Software Testing, Verification and Reliability	4	2	1
Wiley Software: Practice and Experience	8	4	1
Springer Software and Systems Modeling	2	1	1
IEEE Transactions on Systems, Man, and Cybernetics	2	2	1
Conference, Symposium and Congress			
IEEE/ACM Conference on Software Engineering	30	12	8
Springer Symposium on Search Based Software Engineering	110	33	4
IEEE Congress on Evolutionary Computation	21	8	0
IEEE/ACM Conference on Automated Software Engineering	21	8	5
ACM Conference and Symposium on the Foundations of Software Engineering	13	3	0
ACM Genetic and Evolutionary Computation Conference	57	33	10
IEEE Conference on Software Testing, Verification and Validation	20	11	3
ACM Symposium on Software Testing and Analysis	8	5	3
IEEE/ACM Conference on Empirical Software Engineering and Measurements	1	0	0
ACM Systems and Software Product Line Conference	11	3	1
IEEE Conference on Web Services	2	2	1
IEEE Conference on Software Maintenance	15	1	1
IEEE Conference on Software Maintenance and Reengineering	4	3	1
IEEE Workshop on Combining Modelling and Search-Based Software Engineering	5	3	1
IEEE Conference on Requirements Engineering	7	3	1
ACM Conference on Performance Engineering	2	2	2
IEEE/ACM Conference on Program Comprehension	3	2	1
IEEE Conference on Software Architecture	4	2	1
Total	717	319	95

IGD) in order to be used correctly (for *RQ2*). This is again following a similar procedure to that of I_7; when no such information can be found for an indicator that requires a reference, we marked *Unknown* under the indicator for I_8 of the study.

To answer *RQ3*, I_9 is rather straightforward, and understanding it can help us to know whether some evaluation methods are used appropriately, as some of them have limitations in terms of the number of objectives to be optimized. I_{10} is also relatively easy to identify, most commonly from the *Introduction* section and we classify the Pareto-based SBSE problems into the SDLC phases in a classic waterfall model according to [112]. Note that we choose this model by no mean to rely on its usefulness, but only because it is one of the oldest models which consists of very generic phases that allow us to showcase the categories of SBSE problems.

Finally, to complete *RQ3*, I_{11} is crucial as it enables us to assess whether the evaluation methods are used correctly, given the DM's preferences over the objectives and/or the contextual information, which is one of the core initiatives of this paper. To classify the preferences and contextual information, we followed a pragmatic classification coding:

- *Contextual information:* Every problem has its own nature and characteristics; there is no exception for Pareto-based SBSE problems. In general, such nature and characteristics of the problem form the contextual information, which is precise, clear, and explicitly stated as a fact in a study. For example, in the software product line configuration problem, many studies state that there is no doubt that the correctness objective has higher priority than any others, as an invalid product has no value in practice.
- *DM's preferences:* The DMs often have preferences over certain objectives or are able to provide information about their relative importance and expectation. This may be for example, "`objective A is preferred as long as objective B has at least reached b`"; or well-balanced solutions (a.k.a. knee solutions) are preferred. When the DM's preferences are aligned with contextual information, they are indeed similar. However, the key difference is that the contextual information is clear, and it is a hard requirement that is well acknowledged on the given Pareto-based SBSE problem regardless of whether the DM explicitly states it or not. In contrast, the DM's preferences are often vague and imprecise, and it cannot be generalized to all the scenarios of the given Pareto-based SBSE problem.
- *Not specified:* When neither of the above categories can be applied, the preference and contextual information is marked as *Not specified*.

Extracting the data for I_{11} focuses on understanding exactly what DM's preferences and contextual information are assumed in each study. This was achieved by inspecting the sections relevant to *Problem Statement* and *Approach Design*. If no information can be found, we then looked for insights from the *Experiment* section. For example, when a single objective, which belongs to part of the search, is explicitly discussed and used to compare the peer approaches in the evaluation, it often reflects the assumptions of contextual information and/or DM's preferences in the study.

3.5 Data Collection

For each primary study identified, the data items from Table 1 were collected and classified based on the coding from Section 3.4. The first two authors of this paper reviewed the primary studies independently. The data and classification extracted by one author were checked by the other. Disagreements and discrepancies were resolved by discussing between the two authors or by consulting an additional researcher.

Following the strategy recommended in a recent survey [155], we adopted three iterations for the data collection process, which are detailed as below:

Iteration 1: This iteration aims to conduct an initial data collection to summarize the data and perform preliminary classification. During the process, a notable difficulty between the authors was that the evaluations using descriptive statistics and problem-specific indicators are hard to be distinguished. This is due to the fact that most of them are not clearly stated in the studies and there is a wide variety of problem-specific indicators across all Pareto-based SBSE problems (we found 34 of them in our review). Therefore, any study, which the authors suspected that these two types of methods might have been used but could not be certain, was placed into a *bin* for further investigation in the next iteration. There were 26 studies in the *bin* when this iteration finished.

Iteration 2: In this iteration, the two authors checked the data and classification from each other to ensure consistency. A study was discussed during the process if either author has any concern about the data extracted. Any unresolved studies from the *bin* were also checked by the other author again. Particularly, for each study in the *bin*, a common agreement on the descriptive statistics and problem-specific indicators used was reached via either discussion between the authors or counseling external researchers. Further reading to understand the nature of an evaluation method (for problem-specific indicators) was conducted when necessary. Apart from this, other major discussions raised were concerned with certain generic quality indicators, due to two reasons: (i) some studies have indeed used generic quality indicators, but the actual name of the indicator is missing, although some detailed calculation has been provided, e.g., [48], [145]; (ii) some other studies have used completely different names to refer to the same quality indicator (or even invented their own name), e.g., [69], [147]. These cases require both the authors to thoroughly inspect the detailed calculation of those indicators before reaching an agreement. Overall, a total of 60 studies were discussed in this iteration.

Iteration 3: The process of the final iteration is similar to that of *Iteration 1*, but its goal is to eliminate any typo, missing labels, and errors. The extracted data for 11 studies, which contain errors, were corrected during the process.

4 Results of the Review

A breakdown of the studies identified with respect to the venues where they were published has been shown in

TABLE 3
Acronyms of the Evaluation Methods

Acronym	Full Name	Acronym	Full Name	Acronym	Full Name
AS [63]	Attainment Surface	IGD^+ [56]	Inverted Generational Distance$^+$	CI [87]	Contribution Indicator
$\mathcal{C}$ (CS)	$\mathcal{C}$ Metric	GD [133]	Generational Distance	$Spread$ (Δ) [29]	Spread
SP [121]	Spacing	NFS	Nondominated Front Size	ϵ [154]	ϵ-Indicator
IGD [26]	Inverted Generational Distance	HV [153]	Hypervolume	ED	Euclidean Distance
ER	Error Rate	SSP	Solution Set Plotting	DOE	Descriptive Objective Evaluation

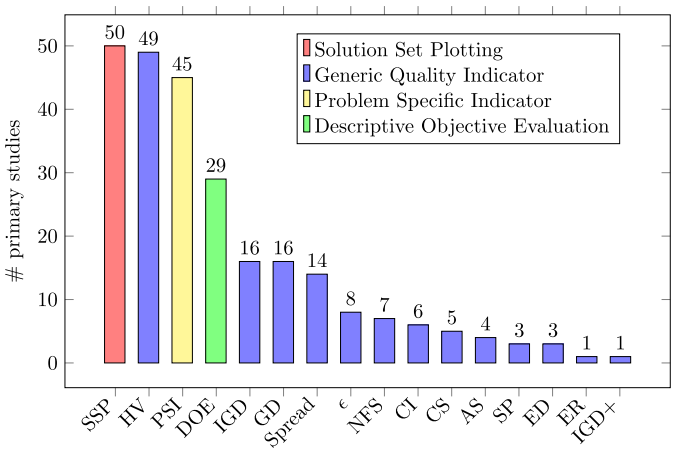


Fig. 2. Usage of evaluation methods in primary studies.

TABLE 4
Descriptive Objective Evaluation (DOE) Methods Used

DOE	Used in
Mean Fitness Value (MFV)	**2:** [76] [137]
Analytic Hierarchy Process (AHP)	**2:** [125] [126]
Mean, best, worst, median and/or statistical result of each objective for solutions in the population*	**27:** [88] [9] [137] [65] [6] [21] [134] [37] [58] [30] [82] [108] [142] [69] [114] [129] [128] [118] [34] [105] [7] [107] [76], [123] [1], [135], [141]
Best of one objective over the population while another is below certain thresholds*	**1:** [66]

*The mean of all repeated runs are reported. Note that a study could involve more than one DOE form.

Table 2. As can be seen, the studies come from a wide range of conferences and journals, which are all respectful.[8] It is worth noting that the results do not only include studies published in software engineering venues, but also those published in service, system and cloud engineering conferences/journals as well as those in the computational intelligence venues, as long as they are related to problems in the software engineering domain and comply with the inclusion/exclusion criteria.

Next, we report on the results collected from our systematic literature review, which would further motivate the remaining of our work.

4.1 RQ1: What Evaluation Methods?

The usage of evaluation methods has been presented in Fig. 2 along with the details for every single primary study presented in Table A1 (at appendix), which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSE.2020.3036108. As can be seen, a total of 13 generic quality indicators have been used in the primary studies (i.e., HV, IGD, GD, $Spread$, ϵ, NFS, CI, CS, AS, SP, ED, ER and IGD^+). Explanations of all the acronyms can be found in Table 3. In particular, HV, IGD and GD are the top three most widely used generic quality indicators across almost all the SBSE problems, due presumably to their popularity as well as "inertia" (i.e., researchers tend to use indicators which were used before even though they are not the best fit) [75].

8. The raw data and minutes recorded during the discussion in the data collection process are publicly available at: https://github.com/taochen/sbse-qi.

There are also 45 primary studies using PSI; for example, MoJoFM [65] is a commonly used symmetric indicator for the software modularization problem, which aims to compare two resulted partitions of classes (i.e., two solutions), thus inapplicable to other optimization scenarios. Note that since the PSI is highly domain-dependent and they are not explicitly designed for evaluating solution sets, we do not specify the usage details for every single one of them. We do however present which particular PSI is used under which context, as shown in Table 8 which we will elaborate in Section 4.3.3. In summary, we found a total of 34 different PSI over all the Pareto-based SBSE problems.

Apart from the quality indicators, SSP and DOE have also been overwhelmingly used by 50 and 29 primary studies respectively to evaluate solution sets. For SSP, we found only two sub-types: the Parallel Coordinate plot which shows the solutions' objective values upon n parallel lines, where n is the number of objectives; and the Scatter Plot that plots the solutions in the objective space. DOE involves more diverse forms, as shown in Table 4, including 27 cases to compare the mean, best, worst, median, or statistical results of each objective in the evaluation along with the remaining five cases that use other three forms.

4.2 RQ2: Why and How Generic Quality Indicators are Used?

As shown in Table 5, for those primary studies that made use of generic quality indicators, most commonly there is a clear statement about what quality aspect(s) they selected an indicator for, as appeared in 73 cases. This is also the reason and evidence that these studies used to justify their choices. However, there is still a considerable amount of cases (59) that were marked as *Unknown*, i.e., no clear and explicit rationale of the choice has been provided. For the reference front

TABLE 5
Summary of How Generic Quality Indicators are Used to Evaluate Solution Sets in the Primary Studies

Indicator	Stated Quality Aspects to Measure	Reference Point or Front
HV	Unknown (25), $Q_1 \cup Q_2 \cup Q_3$ (15), $Q_1 \cup Q_2$ (3), Q_1 (3), $Q_2 \cup Q_3$ (2)	Unknown (23), Worst values (10), Nadir point (9), Boundary (6)
IGD	$Q_1 \cup Q_2 \cup Q_3$ (9), Q_1 (4), Unknown (2), $Q_1 \cup Q_2$ (1)	Best Pareto front found (14), Unknown (2)
$IGD+$	$Q_1 \cup Q_2 \cup Q_3$ (1)	Unknown (1)
GD	Unknown (10), Q_1 (6)	Best Pareto front found (10), Unknown (6)
$Spread$	$Q_2 \cup Q_3$ (9), Q_2 (5)	N/A
ϵ-Indicator	Q_1 (5), Unknown (2), $Q_1 \cup Q_2 \cup Q_3$ (1)	N/A
NFS	Unknown (4), $Q_2 \cup Q_3$ (3)	N/A
CI	Unknown (5), Q_1 (1)	N/A
CS	Unknown (4), Q_1 (1)	N/A
AS	Unknown (4)	N/A
SP	Q_3 (2), $Q_2 \cup Q_3$ (1)	N/A
ED	Unknown (3)	N/A
ER	Q_1 (1)	Unknown (1)

Q_1=Convergence; Q_2=Spread; Q_3=Uniformity; Q_4=Cardinality. The number of primary studies is shown within the brackets.

TABLE 6
Summary of the # Objectives Under Which the Generic Quality Indicators, SPI, SSP and DOE are Used

Method	# Objectives
HV	2 (19), 3 (24), 4 (9), 5 (11), 6 (1), 7 (1), 8 (1), 9 (2)
IGD	2 (4), 3 (5), 4 (2), 5 (5), 8 (1), 15 (1)
$IGD+$	2 (1), 3 (1), 4 (1)
GD	2 (6), 3 (9), 4 (2), 5 (2)
$Spread$	2 (5), 3 (3), 4 (2), 5 (7)
ϵ-Indicator	2 (1), 3 (4), 5 (4)
NFS	2 (5), 3 (2), 5 (2)
CI	2 (3), 3 (4), 4 (1), 5 (1)
CS	2 (3), 4 (1), 5 (1)
AS	2 (2), 3 (2)
SP	4 (1), 5 (1), 6 (1), 7 (1), 8 (1), 9 (1)
ED	2 (2), 3 (1)
ER	5 (1)
SSP	2 (30), 3 (19), 4 (6), 5 (2), 7 (1), 8 (1)
DOE	2 (4), 3 (13), 4 (5), 5 (5), 7 (1), 9 (1)
PSI	2 (17), 3 (14), 4 (4), 5 (5), 6 (1), 7 (1), 8 (1), 9 (2), 15 (1), >100 (2)

The bracket shows the number of problems under which the a pair of objective number and evaluation method is considered. Note that a study may consider problems with different # objectives.

used for GD and IGD, whilst most of the cases the best Pareto front found by all algorithms (i.e., nondominated solutions of the set consisting of the solutions produced by all algorithms) is used, many still do not explicitly declare such information. On the reference point used by HV, we see a diverse way of obtaining such a point, including using the worst objective value of all the solutions found, the boundary of the optimization problem in SBSE, and the nadir point from the Pareto front of all the solutions found.

4.3 RQ3: What Context?

4.3.1 Number of Objectives

Table 6 shows the number of objectives considered by the evaluation methods. On the generic quality indicators, we see that IGD has been used for the highest number, i.e., 15 objectives, and all of them (except SP and ER) have ever been used on the bi-objective cases, which is the minimum number required to build a Pareto front. Whilst most of the generic quality indicators have been used under the bi- and tri-objectives cases, a considerable amount of them have been used on the objective number over three.

As for PSI, DOE, and SSP, we can observe that they are used on a relatively wider range of objective numbers as compared with most of the generic quality indicators.

4.3.2 Pareto-Based SBSE Problems

From Table 7, it is clear that our systematic review has revealed 21 distinct Pareto-based SBSE problems, which are spread across all the six common phases in the SDLC.[9] Notably, we can see that certain problems have attracted more attention than the others, as evidenced by the much higher number of primary studies included, such as the software product line and the white/black-box test case generation problems. Among others, the software testing phase, as well as the deployment and maintenance phase contain much more diverse problems than the other SDLC phases. This is probably because the nature of those problems, which are usually in later phases of the SDLC, fits the requirements of search-based optimization well.

Indeed, some of the Pareto-based SBSE problems can arguably fit into more than one phase of the SDLC; but in this work, we classify those problems according to which phases can be better matched with the detailed formalization of the problem and the hypotheses that the authors made. Further, certain problems in the deployment and maintenance phase are not classic software engineering problems (e.g., resource management and service composition); however, they have recently attracted more and more attention from software engineering researchers and have been increasingly considered as important issues in the software engineering domain [49].

4.3.3 Assumptions on Preferences and Contextual Information of Problems and Their Evaluation Methods

In Table 8, we summarize the assumptions on the DM's preferences and contextual information about the objectives for each Pareto-based SBSE problem reviewed, and the evaluation methods used to compare different solution sets under these contexts. As we can see, there are 17 cases, covering 25 primary studies, have made assumptions on DM's preferences. The contextual information, in contrast, has been used in five cases over 20 primary studies. The others do not clearly state the assumptions in this regard and hence noted as *Not specified*. One of the most notable observations is that a particular SBSE problem may have multiple, distinct assumptions about the DM's preferences and contextual information. In fact, most of the problems have more than one assumption on the preferences/contextual information, and particularly, the project scheduling problem and software configuration/adaptation problem involve up to four different types of assumptions.

9. Note that [16] studies three different problems.

TABLE 7
Pareto-Based SBSE Problems in Different SDLC Phases

SDLC Phase	SBSE Problem	Description	Primary Studies
Planning	Effort Estimation	Optimize, e.g., accuracy and confidence interval, by changing the number of measured samples.	**2:** [88] [115]
	Project Scheduling	Optimize, e.g., duration and cost, by assigning employee into the tasks of a software project.	**7:** [126] [125] [114] [44] [35] [24] [16]
Requirement Analysis	Requirement Assignment	Optimize, e.g., completeness and familiarity, by assigning requirements to different stakeholders' for their reviews.	**1:** [76]
	Next Release Problem	Optimize, e.g., robustness and cost, by selecting stakeholders' requirements in the next release of software.	**8:** [36] [31] [48] [69] [147] [148] [16] [149]
Design	Software Modeling and Architecting	Optimize, e.g., cohesion and coupling, by modeling the object-oriented concept of the software and its architecture using standard notations.	**7:** [109] [9] [129] [128] [51] [11] [59]
	Software Product Line	Optimize, e.g., correctness and richness, by finding the concrete products from the feature model.	**15:** [118] [119] [120] [54] [52] [93] [144] [137] [16] [45] [143] [66] [113] [77] [53]
Implementation	Library Recommendation	Optimize, e.g., library linked-usage and semantic similarity, by prioritizing the libraries that meet the required functionality to be used in the codebase.	**1:** [99]
	Program Improvement	Optimize, e.g., execution time and number of instructions, by producing semantically preserved software code.	**1:** [141]
	Software Modularization	Optimize, e.g., modularization quality, cohesion and coupling, by placing different classes of code into different clusters.	**5:** [65] [6] [108] [37] [7]
Testing	Code Smell Detection	Optimize, e.g., coverage of bad examples and detection of good examples, by identifying the code and modules that could potentially cause issues.	**1:** [80]
	Defect Prediction	Optimize, e.g., effectiveness and cost, by adjusting the source code components to be predicted by the model.	**4:** [15] [14] [23] [92]
	Test Case Prioritization	Optimize, e.g., coverage of the code and cost of test, by ordering the test cases to be tested.	**7:** [5] [103] [83] [123] [32] [106] [127]
	White Box Test Case Generation	Optimize, e.g., coverage of the code and cost of test, by identifying the test cases, inputs and test suits based on internal information of the software.	**9:** [2] [145] [150] [82] [58] [34] [100] [57] [102]
	Black Box Test Case Generation	Optimize, e.g., length of the inputs, distance to the ideal inputs, and cost of test, by identifying the test cases, inputs and test suits without internal information about the software	**3:** [124] [107] [3]
Deployment and Maintenance	Resource Management	Optimize, e.g., response time and cost, by changing the supported software and hardware resources such as in the Cloud environment.	**3:** [39] [68] [17]
	Software Configuration and Adaptation	Optimize, e.g., response time and energy consumption, by changing software specific configurations, structure and connectors at design time or runtime.	**7:** [105] [43] [84] [67] [1] [19] [13]
	Program Manipulation	Optimize, e.g., response time and memory consumption, by changing the parametrized variables within the program code.	**1:** [142]
	Service Composition	Optimize, e.g., latency and cost, by mapping the concrete services into abstract services within a workflow.	**4:** [135] [134] [131] [21]
	Log Template Identification	Optimize, e.g., the frequency and specificity of the log message matched to a log template.	**1:** [86]
	Workflow Scheduling	Optimize, e.g., makespan and energy consumption, by assigning activities into a given application workflow.	**1:** [30]
	Software Refactoring	Optimize, e.g., number of defects found and semantics, by changing the design model or the program code.	**9:** [94] [91] [95] [96] [97] [98] [81] [90] [89]

This reflects the fact that many problems are complex and the actual preferences can be situation-dependent. Yet, it does bring the requirements that all those situations need to be catered for. In contrast, problems such as effort estimation and requirement assignment have assumed only one type of preferences/contextual information, which implies a relatively more straightforward selection and use in the quality evaluation process on the solution sets.

5 Issues on Quality Evaluation in Pareto-Based SBSE

Based on our systematic literature review, this section provides a systematic analysis of five issues of quality evaluation, classified into two categories, from state-of-the-art Pareto-based SBSE work.

5.1 Problematic Use of Illustrative and Descriptive Statistic Evaluation Methods

As shown in Fig. 2, Tables 8 and A1 (at appendix), available in the online supplemental material, there exist many SBSE studies, particularly in early days, that relied on plotting the solution set returned (SSP) and/or on reporting some DOE results to reflect the quality of solution sets. Despite these two methods being simple to apply, they may easily lead to inaccurate evaluations and conclusions.

TABLE 8
Assumptions of DM's Preferences and Contextual Information in the Pareto-Based SBSE Problems and the Corresponding Evaluation Methods Used

SBSE Problem	Assumptions	Evaluation Methods
Effort Estimation	Not specified [88] [115]	DOE, SSP, CI, GD, HV
Project Scheduling	Not specified [16] [126]	SSP, DOE, $Spread$, GD, IGD, NFS, HV
	(P) Prefer solutions that favor certain objectives using Analytic Hierarchy Process [125]	DOE, CS, SSP, GD, SP, $Spread$, HV
	(P) Prefer knee solutions [114] [44] [35]	SSP, CI, GD, DOE, HV
	(P) Prefer widely distributed solutions [24]	AS, HV
Requirement Assignment	Not specified [76]	DOE, SSP, HV
Next Release Problem	Not specified [36] [31] [48] [69] [147] [16] [149]	SSP, DOE, AS, NFS, GD, CI, $Spread$, HV, % of included requirements⋆
	(P) Prefer extreme solutions [148]	SSP
Software Modeling and Architecting	Not specified [109] [9] [129]	DOE, SSP, SP, HV, % of within-range solutions⋆, % of equivalent solutions⋆
	(P) Prefer knee solutions [59]	average correction⋆, manual correction⋆, recall⋆, precision⋆
	(P) Prefer solutions that favor certain objectives as ranked by users [128]	DOE
	(P) Prefer solutions that meet preferences in, e.g., the requirement documentations or the goal model [51] [11]	SSP
Software Product Line	Not specified [143]	HV, IGD^+, SSP
	(C) Prefer solutions that favor correctness objective over the others [118] [119] [120] [54] [52] [93] [144] [53] [16] [45] [66] [113] [77]	CS, $Spread$, NFS, SSP, ϵ-indicator, IGD, HV, DOE, number of required evaluations to find a valid solution⋆, full coverage ratio⋆, % of valid solutions⋆
	(P) Prefer balanced solutions [137]	DOE, SSP
Library Recommendation	Not specified [99]	GD, $Spread$, HV, SSP, accuracy⋆, precision⋆, recall⋆
Program Improvement	**(C)** Prefer the program validity [141]	DOE, SSP
Software Modularization	**(C)** Prefer solutions that favor modularization quality objective over the others [65] [6] [108]	DOE, SSP, MoJoFM⋆
	(P) Prefer knee solutions [37] [7]	DOE, IGD, HV, precision⋆, recall⋆, manual precision⋆, difficulty to perform task by human⋆, possibility of manually fix the bug in solution by human⋆, possibility of manually adapt the solution by human⋆
Code Smell Detection	Not specified [80]	IGD, HV, precision⋆, recall⋆
Defect Prediction	Not specified [15] [14] [23] [92]	SSP, HV, $ACC⋆$, $P^⋆_{opt}$, precision⋆, recall⋆, AUC⋆, cost of code inspection⋆
Test Case Prioritization	Not specified [5] [103] [83] [123] [32] [106] [127	SSP, DOE, CS, ED, $Spread$, GD, ϵ-indicator, IGD, HV, NFS, $APFD⋆$, % of detected faults⋆
White Box Test Case Generation	Not specified [2] [145] [82] [34] [102] [100]	DOE, SSP, CI, GD, NFS, HV, AS, total coverage⋆
	(P) Prefer solutions that favor certain objectives as ranked by users, i.e., reference point [57] [58]	ED, SSP, $R\text{-}HV$, Average number of solutions in the region of interest⋆
	(C) Prefer solutions that favor coverage objective over the others [150]	SSP, HV
Black Box Test Case Generation	Not specified [107] [3]	DOE, HV, GD, $Spread$
	(C) Prefer solutions that favor coverage objective over the others [124]	SSP, p-measure⋆
Resource Management	Not specified [39]	IGD, HV
	(P) Prefer knee solutions [68] [17]	SSP, CS, GD, elasticity⋆
Software Configuration and Adaptation	Not specified [105] [43]	DOE, GD, ϵ-indicator, IGD, HV, SSP
	(P) Prefer solutions that meet preferences form the natural descriptions from the stakeholders [84] [67] [1]	DOE, SSP, expected value of total perfect information⋆
	(P) Prefer knee solutions [19]	SSP, ED, HV, % of valid solutions⋆
	(P) Prefer robust solutions around a given region [13]	SSP, modified ϵ-indicator and IGD according to problem nature
Program Manipulation	Not specified [142]	DOE, AS, CI, HV, SSP
Service Composition	Not specified [135] [21]	DOE, HV ϵ-indicator
	(P) Prefer extreme solutions [134] [131]	DOE, SSP, IGD, HV, coefficient of variation of objective values⋆
Log Template Identification	**(P)** Prefer knee solutions [86]	SSP, precision⋆, recall⋆, f-measure⋆
Workflow Scheduling	Not specified [30]	DOE, SSP, HV
Software Refactoring	Not specified [94] [91] [95] [96] [97] [98] [81] [90]	SSP, IGD, precision⋆, recall⋆, defect correction ratio⋆, reused refactoring⋆, usefulness by human⋆, % of fixed code smells⋆, code change score⋆, manual precision⋆, quality gains⋆, medium value of refactoring⋆
	(P) Prefer knee solutions [89]	SSP, IGD, precision⋆, recall⋆, manual precision⋆, quality gain⋆, defect correction ratio⋆, number of suggested refactoring⋆, usefulness by human⋆

All problem specific indicators are listed in full and marked as ⋆.
***(P)** denotes DM's preferences; **(C)** denotes contextual information.*

5.1.1 ISSUE I: Inadequacy of Solution Set Plotting (SSP)

A straightforward way to evaluate/compare the quality of solution sets returned by search algorithms is to plot solution sets and judge intuitively how good they are. Such visual comparison is among the most frequently used methods in SBSE, but it may not be very practical in many cases.

First, it cannot scale up well — when the number of objectives is larger than three, the direct observation of solution sets (by scatter plot) is unavailable. Second, the visual comparison fails to quantify the difference between solution sets. Finally, when an algorithm involves stochastic elements, different runs usually result in different solution sets. So, it may not be easy to decide which run should be considered. Printing the solution sets obtained in all the runs can easily clutter the picture. As such, plotting solution sets does not suffice to the quality evaluation in Pareto-based SBSE, despite the fact that it has been used solely to compare solution sets in a considerable amount of the primary studies, e.g., [11], [44], [51], [68], [84], [148], as shown in Table A1 (at appendix), available in the online supplemental material. Nevertheless, it is worth mentioning that SSP is useful as an extra evaluation method in addition to quality indicators, particularly in bi- and tri-objective cases. This will be discussed in the guidance section (Section 7) later on.

5.1.2 ISSUE II: Inappropriate Use of Descriptive Objective Evaluation (DOE)

Many Pareto-based SBSE studies evaluate solution sets by DOE — statistical objective values in the obtained solution set(s). For example, as it can be seen in Table 4, the mean objective value was considered in [6], [7], [9], [102], [105], [107], [114], [118], [128], [129]; the median value in [6], [37], [58]; the best value in [9], [21], [30], [65], [82], [88], [100], [108], [134], [135], [141], [142]; the worst value in [9], [134]; the statistical significance of the differences between distinct solution sets' objective values in [1], [76], [123], [137]. Such DOE measures need to be used in line with the DM's preference. For example, comparing the best value of each objective can well evaluate solution sets if the DM prefers the extreme points (solutions), but may not be well-suited when balanced points are wanted, which, unfortunately, was practiced in some studies such as [137] shown in Table 8. Worse still, many DOE measures may give a misleading evaluation, including those comparing the mean, median, and worst values of each objective and comparing statistically significant differences on each objective. That is to say, by a DOE measure a solution set is evaluated better than another set, but in fact, the latter is always preferred by the DM under any circumstances. Fig. 3 gives such an example (minimization) with respect to calculating the mean of each objective. As shown, the mean of the solution set **A** on either objective f_1 or f_2 is 5, larger than that of the solution set **B** (4), thus **A** being regarded as inferior to **B**. Yet, **A** will always be favored by the DM since there is one solution in **A** better than any solution in **B**.

On the other hand, recalled from Table 4, some work in the primary studies considers selecting one particular solution (by using a decision-making method) from the whole solution set produced by the Pareto-based search for comparison. For example, the studies in [76], [137] considered Mean Fitness Value (MFV) and the studies in [125], [126] considered Analytic Hierarchy Process (AHP) [41]. However, one question is that if we know clear weighting between objectives of the DM (thus being able to take only one solution from the whole solution set into account), why not directly integrate this information into the problem model, thus converting a multi-objective problem into an easier single-objective problem in the first place.

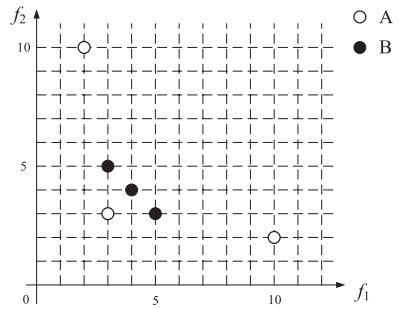


Fig. 3. An example that comparing the mean on each objective fails to reflect the quality of solution sets. In this minimization problem, solution set **A** dominates solution set **B** (i.e., any solution in **B** is dominated by at least one solution in **A**), thus always being favored by the DM. Yet, the mean of **A** on either objective f_1 or f_2 is 5, larger than that of **B** (4); thus **A** is regarded as inferior to **B**.

5.2 Problematic Use of Generic Quality Indicators

As disclosed in Tables 6, 8 and A1 (at appendix), available in the online supplemental material, it has been commonly seen in Pareto-based SBSE studies that select or use quality indicators that cannot accurately reflect the quality of solution sets. This is virtually because the SBSE researchers/practitioners may not be very clear about indicators' behavior, role, and characteristics. This leads them either to fail to select appropriate indicators to evaluate the generic quality of solution sets, or to fail to align the considered indicators with the DM's preferences or the problem's contextual information.

5.2.1 ISSUE III: Confusion of the Quality Aspects Covered by Generic Quality Indicators

As mentioned, the generic quality of a solution set in Pareto-based optimization can be interpreted as how well it represents the Pareto front. It can be broken down into four aspects: convergence, spread, uniformity, and cardinality [75]. It is expected that when the DM's preferences are unknown *a priori*, an indicator (or a combination of indicators) can cover all the four quality aspects since a solution set with these qualities can well represent the Pareto front and have a great probability of being preferred by the DM.

Unfortunately, as shown in Tables 6 and A1 (at appendix), available in the online supplemental material, in SBSE many studies only consider part of these quality aspects. For example, the studies in [48], [69] used the convergence indicator GD [133] as the sole indicator to compare the solution sets. The study in [145] considered both PFS [52] and CI which however are merely for convergence and cardinality. In addition, some indicators were used to evaluate

certain quality aspect(s) of solution sets which, unfortunately, were not designed for, as shown in [74]. For example, the indicator $\mathcal{C}$ [153], designed for convergence evaluation, was considered for evaluating spread in [138]. SP [121], which can only reflect the uniformity of solution sets, was used to evaluate the diversity (i.e., both spread and uniformity) in [109]. PFS, which counts nondominated solutions in the set, was placed into the category of diversity indicators in [52], [138], and ϵ-indicator, which is able to reflect all the quality aspects of solution sets, was placed into the category of convergence indicators in [138].

As can be seen from Tables 5 and 6, some indicators were used incorrectly, For example, the indicator *Spread* (i.e., Δ in [29]) as well as its variants (e.g., GS [151]), which is only effective in the bi-objective case, was frequently used in optimization problems of SBSE with three or more objectives, such as in [3], [52], [99], [118], [120], [125], [126]. Another example is the setting of the critical parameter reference point in the HV indicator, which has experienced various versions. For example, some studies set it to the worst value obtained for each objective during all runs [34], [76], [93], [105], [125], [126], [131]; some did it to precisely the boundaries of the optimization problem in SBSE [30], [54], [144]; some did it to the nadir point of the Pareto front [103]. The first two settings may overemphasize the boundary solutions (as the reference point may be far away from the set to be evaluated), while the last one may lead to the boundary solutions to contribute nothing to the HV value.

It is worth mentioning that as usually the problem's Pareto front in SBSE is unavailable, for indicators which need the Pareto front for reference, a common practice is to collect the nondominated set of all the solutions produced as an estimated Pareto front, as we have shown in Table 5. However, different indicators have different sensitivity to this practice [75]. For example, IGD and *Spread* require the Pareto front consisting of uniformly-distributed points, while HV and ϵ-indicator do not [75]. Therefore, IGD and *Spread* may not be very suitable in SBSE, despite the fact they were frequently used, e.g., in [31], [35], [52], [114], [125], [126], [147].

5.2.2 ISSUE IV: Oblivion of Context Information

As shown in Table 8, in Pareto-based SBSE, many studies compare solution sets without bearing in mind the contextual information with respect to the considered optimization problem. They typically adopt commonly-used quality indicators to directly evaluate the set of all the solutions obtained, although some of these solutions may never or rarely be of interest to the DM. Fig. 4 shows such an example, under a scenario of optimizing the code coverage and the cost of testing time on the software test case generation problem, borrowed from [74]. As can be seen, the set **B** is evaluated better than the set **A** by all eight commonly used quality indicators (GD [133], ED [25], ϵ-indicator [154], GS [151], PFS [52], IGD [26], HV [153] and $\mathcal{C}$ [153]) in SBSE [138]. However, depending on the context (as shown in Table 8), the DM might first favor the full code coverage and then possible low cost [150]. This will lead to set **A** to be of more interest, as it has the solution (450,1.0) that achieves full coverage and lower cost than the one in **B** (500,1.0).

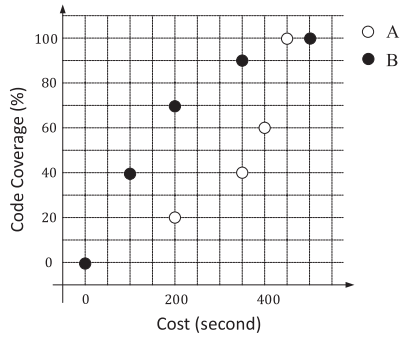


Fig. 4. An example where lack of considering contextual information may give unwanted evaluation results [74] . Considering two solutions sets (**A** and **B**) for optimizing the code coverage and the cost of testing time on the software test case generation problem, where $\mathbf{A} = \{(200, 0.2), (350, 0.4), (400, 0.6), (450, 1.0)\}$ and $\mathbf{B} = \{(0, 0), (100, 0.4), (200, 0.7), (350, 0.9), (500, 1.0)\}$. **B** is evaluated better than **A** on eight frequently-used indicators: $GD(\mathbf{B}) = 0.02 < GD(\mathbf{A}) = 0.26, ED(\mathbf{B}) = 0.5 < ED(\mathbf{A}) = 0.89, \epsilon(\mathbf{B}) = 0.1 < \epsilon(\mathbf{A}) = 0.3, GS(\mathbf{B}) = 0.15 < GS(\mathbf{A}) = 0.46, PFS(\mathbf{B}) = 5 > PFS(\mathbf{A}) = 4, IGD(\mathbf{B}) = 0.02 < IGD(\mathbf{A}) = 0.27, HV(\mathbf{B}) = 0.77 > HV(\mathbf{A}) = 0.43, \mathcal{C}(\mathbf{B}) = 0.8 > \mathcal{C}(\mathbf{A}) = 0.25$. However, the DM may be more interested in **A** (specifically solution (450,1.0)) if they favor the full code coverage and then possible low cost.

Similar observations have been seen in the optimal product selection in software product line [16], [45], [52], [66], [77], [113], [119], [120], [132] where the correctness of configurations is regarded as one objective and equally rated as other objectives (e.g., richness of features and cost). This may lead to an invalid product to be evaluated better than a valid product if the former performs better in other objectives, which is apparently of no value to the DM. In addition, in many SBSE problems, cost could be an objective to minimize, but solutions with zero cost are trivial, e.g., the solution with zero cost and zero coverage in Fig. 4. However, these solutions may largely affect the evaluation results. Therefore, it is necessary to remove solutions that would never be interested by the DM before the evaluation, which, unfortunately, has been rarely practiced in Pareto-based SBSE.

5.2.3 ISSUE V: Noncompliance of the DM's Preferences

Although every quality indicator is designed to reflect certain quality aspect(s) of solution sets (i.e., convergence, spread, uniformity, cardinality, or their combination), they do have their own implicit preferences. For example, the indicators HV and IGD, both designed to cover all of the four quality aspects, have rather distinct preferences. HV prefers knee points of a solution set, while IGD is in favor of a set of uniformly distributed solutions. Therefore, it is important to select indicators whose preferences are in line with the DM's. Neglecting this can lead to misleading evaluation results. Fig. 5 gives such an example — when preferring knee points, considering the indicator IGD could return a misleading result. That is, the set having knee points is evaluated worse than that having no knee point. Similar observations also apply to the indicators GD and CI, as shown in the same figure.

Unfortunately, as what has been revealed in Table 8, such misuse of indicators is not uncommon in the SBSE community. For example, preferring knee points yet using IGD

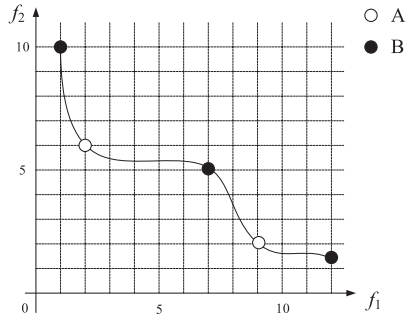


Fig. 5. An example that preferring knee points while using the indicator IGD (as well as GD and CI) can lead to misleading results. Consider two solution sets ($\mathbf{A}$ and $\mathbf{B}$) for a bi-objective minimization scenario, where $\mathbf{A} = \{(2,6),(9,2)\}$ are the two knee points of the Pareto front, and $\mathbf{B} = \{(1,10),(7,5),(12,1.5)\}$ are three well-distributed non-knee points on the Pareto front. Apparently, if the DM prefers knee points then solutions in $\mathbf{A}$ will certainly be selected. Yet, $\mathbf{A}$ is evaluated worse than (or as equal as) $\mathbf{B}$ by IGD, GD and CI: $IGD(\mathbf{A}) = 2.154 > IGD(\mathbf{B}) = 1.433$, $GD(\mathbf{A}) = GD(\mathbf{B}) = 0$, and $CI(\mathbf{A}) = 0.4 < CI(\mathbf{B}) = 0.6$. In contrast, the indicator HV can reflect this preference, $\mathbf{A}$ being evaluated better than $\mathbf{B}$: $HV(\mathbf{A}) = 71.0 > HV(\mathbf{B}) = 45.5$ (the reference point is (13,11)).

in [37], [89]; preferring knee points yet using GD and CI in [35], [114]; and preferring extreme solutions yet using HV and IGD in [131], [134]. HV can be somehow in favor of extreme solutions if the reference point is set far away from the considered set, but IGD certainly does not prefer extreme solutions. Therefore, it is of high importance to understand the behavior, role, and characteristics of the considered indicators, which may not be very clear to the community. In the next section, we will detail widely used quality evaluation methods in the area (as well as other useful indicators) and explain the scope of their applicability.

6 Revisiting Quality Evaluation for Pareto-Based Optimization

In Pareto-based optimization, the general goal for the algorithm designer is to supply the DM a set of solutions from which they can select their preferred one. In general, the actual preferences can be either articulated by the DM or derived from the contextual information of the problem, which may differ depending on the situation. Having said that, the Pareto dominance relation is apparently the foremost criterion in any case, provided that the concept of optimum is solely based on the direct comparison of solutions' objective values (other than on other criteria, e.g., robustness and implementability with respect to decision variables). That is to say, the DM would never prefer a solution to the one that dominates it.

As discussed in Section 2, the *better* relation ($\lhd$) represents the most general and weakest form of superiority between two sets. That is, for two solution sets $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A} \lhd \mathbf{B}$ indicates that $\mathbf{A}$ is at least as good as $\mathbf{B}$, while $\mathbf{B}$ is not as good as $\mathbf{A}$. It meets any preference potentially articulated by the DM. If $\mathbf{A} \lhd \mathbf{B}$, then it is always safe for the DM only to consider solutions in $\mathbf{A}$. Apparently, it is desirable that a quality evaluation method is able to capture this relation; that is to say, for any two solution sets $\mathbf{A}$ and $\mathbf{B}$, if $\mathbf{A} \lhd \mathbf{B}$, then $\mathbf{A}$ is evaluated better than $\mathbf{B}$. Unfortunately, there are very few quality evaluation methods holding this property. HV is one of them [152]. There is a weaker property called being *Pareto compliant* [62], [154], which is more commonly used in the literature. That is, a quality evaluation method is said to be *Pareto compliant* if and only if "at least as good" in terms of the dominance relation implies "at least as good" in terms of the evaluation values (i.e., $\forall \mathbf{A}, \mathbf{B} : \mathbf{A} \preceq \mathbf{B} \Rightarrow I(\mathbf{A}) \leq I(\mathbf{B})$, where I is the evaluation method, assuming that the smaller the better). Despite the relaxation, many quality indicators are not Pareto compliant, including widely used ones, such as GD, IGD, *Spread*, GS, and SP. Pareto compliant indicators are mainly those falling into the category of evaluating convergence of solution sets (e.g., $\mathcal{C}$ and CI) and the category of evaluating comprehensive quality of solution sets (e.g., HV, ϵ-indicator, IPF [10], $R2$ [46], and PCI [72]). DCI [70] is the only known diversity indicator compliant with Pareto dominance when comparing two sets. In addition, some non-compliant indicators can become Pareto compliant after some modifications. For example, GD and IGD can be transformed into two Pareto compliant indicators (called GD^+ and IGD^+) if considering "superiority" distance instead of euclidean distance between points [56]. Overall, it is highly recommended to consider (at least) Pareto compliant quality indicators to evaluate solution sets; otherwise, it may violate the basic assumption of the DM's preferences. That is, recommending the DM a solution set $\mathbf{B}$ over $\mathbf{A}$, where each solution in $\mathbf{B}$ is inferior to or can be replaced by (in the case of equality) some solution in $\mathbf{A}$. This is what the DOE evaluation method that compares the mean on each objective does in the example of Fig. 3.

Now, one may ask why not directly use the *better* relation to evaluate solution sets. The reason is that the *better* relation may leave many solution sets incomparable since in most cases there exist some solutions from different sets being nondominated to each other. Therefore, we need stronger assumptions about the DM's preferences, which are reflected by quality evaluation methods. However, stronger assumptions (than the *better* relation) cannot guarantee that the favored set (under the assumptions) is certainly preferred by the DM, as in different situations the DM indeed may prefer different trade-offs between objectives. Consequently, it is vital to ensure the considered evaluation methods in line with the DM's explicit or implicit preferences.

Back to the example in Fig. 4 where optimizing the objectives code coverage and cost of testing time, essentially these two solution sets are not comparable with respect to the *better* relation despite the fact that most solutions in $\mathbf{A}$ are dominated by some solution in $\mathbf{B}$. As stated, the DM may be more interested in full code coverage and then possible lower cost, thus preferring $\mathbf{A}$ to $\mathbf{B}$. However, the considered eight indicators fail to capture this information and give opposite results. This clearly indicates the importance of understanding quality evaluation methods (including what kind of assumptions they imply). Next, we will review several quality evaluation methods which are commonly used in the SBSE community (as we have shown in Table 2) and at the same time are very representative to reflect certain aspect(s) of solution set quality.

6.1 Descriptive Objective Evaluation (DOE)

As stated before, the DOE methods evaluate a solution set (or several sets obtained by a search algorithm in multiple

runs) by directly reporting statistical results of objective values of its solutions, such as the mean, median, best, worst, and statistical significance (in comparison with other sets). Unfortunately, such methods are rarely being Pareto compliant and unlikely to be associated with the DM's preferences. However, an exception is the method that considering the best value of some objective(s) in a solution set, since it is Pareto compliant and able to directly reflect the DM's preferences in the case that they prefer extreme solutions. Overall, the DOE methods are not recommended, unless the DM explicitly expresses their preferences in line with them.

6.2 Contribution Indicator (CI)

The CI indicator [87], which was designed to compare the convergence of two solution sets, has been frequently used in SBSE, e.g., in [35], [114], [115], [142], [145], [149]. CI calculates the ratio of the solutions of a set that are not dominated by any solution in the other set. Formally, given two sets $\mathbf{A}$ and $\mathbf{B}$,

$$CI(\mathbf{A},\mathbf{B}) = \frac{|\mathbf{A}\cap\mathbf{B}|/2 + |\mathbf{A}_{\prec B}| + |\mathbf{A}_{\not\preceq\cap\not\succ B}|}{|\mathbf{A}\cap\mathbf{B}| + |\mathbf{A}_{\prec B}| + |\mathbf{A}_{\not\preceq\cap\not\succ B}| + |\mathbf{B}_{\prec A}| + |\mathbf{B}_{\not\preceq\cap\not\succ A}|}, \tag{1}$$

where $\mathbf{A}_{\prec B}$ stands for the set of solutions in $\mathbf{A}$ that dominate some solution of $\mathbf{B}$ (i.e., $\mathbf{A}_{\prec B} = \{\mathbf{a}\in\mathbf{A}|\exists\mathbf{b}\in\mathbf{B}:\mathbf{a}\prec\mathbf{b}\}$), and $\mathbf{A}_{\not\preceq\cap\not\succ B}$ stands for the set of solutions in $\mathbf{A}$ that do not weakly dominate any solution in $\mathbf{B}$ and also are not dominated by any solution in $\mathbf{B}$ (i.e., $\mathbf{A}_{\not\preceq\cap\not\succ B} = \{\mathbf{a}\in\mathbf{A}|\nexists\mathbf{b}\in\mathbf{B}:\mathbf{a}\preceq\mathbf{b}\vee\mathbf{b}\prec\mathbf{a}\}$).

The CI value is in the range of [0,1]. A higher value is preferable. It is apparent that $CI(\mathbf{A},\mathbf{B}) + CI(\mathbf{B},\mathbf{A}) = 1$. A clear strength of the indicator CI is that it holds the *better* relation,[10] i.e., if $\mathbf{A}\lhd\mathbf{B}$ then $CI(\mathbf{A},\mathbf{B}) > CI(\mathbf{B},\mathbf{A})$. Moreover, if $\mathbf{A}\prec\mathbf{B}$, then $CI(\mathbf{B},\mathbf{A}) = 0$. In addition, apart from comparing the convergence of solution sets, CI can reflect their cardinality to some extent. A set having a larger number of solutions is likely to be favored by the indicator.

A clear weakness of CI is that it relies completely on the dominance relation between solutions, thus providing little information about to what extent one set outperforms another. Moreover, they may leave many solution sets incomparable if all solutions from the sets are nondominated to each other. This may happen frequently in many-objective optimization, where more objectives are considered.

There is another well-known dominance-based quality indicator (called $\mathcal{C}$ or CS) [153], used in e.g., [5], [17], [125], [146]. It measures the proportion of solutions in a set that is weakly dominated by some solution in the other set; in other words, the percentage of a set that is *covered* by its opponent. The details of the indicator $\mathcal{C}$ can be found in [75]. $\mathcal{C}$ tends to be more popular in the multi-objective optimization community, despite sharing the above strengths and weaknesses with CI. Finally, it is worth mentioning that despite only partially reflecting the convergence of solution sets, such dominance-based indicators are useful since most problems in SBSE are combinatorial ones, where the size of the Pareto front may be relatively small and it is likely to have comparable solutions (i.e., dominated/duplicate solutions) from different sets [75].

10. Note that it is not unusual that binary indicators (i.e., those directly comparing two sets) holds the *better* relation [75].

6.3 Generational Distance (GD)

As one of the most widely used convergence indicators in SBSE (used in e.g., [2], [5], [17], [35], [99], [105], [114], [115], [125], [147]), GD [133] is to measure how close the obtained solution set is from the Pareto front. Since the Pareto front is usually unknown *a priori*, a reference set, $\mathbf{R}$, which consists of nondominated solutions of the collection of solutions obtained by all search algorithms considered, is typically used to represent the Pareto front in practice. Formally, given a solution set $\mathbf{A} = \{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_n\}$, GD is defined as

$$GD(\mathbf{A}) = \frac{1}{n}\left(\sum_{i=1}^{n}(\min_{\mathbf{r}\in\mathbf{R}} d(\mathbf{a}_i,\mathbf{r}))^p\right)^{1/p}, \tag{2}$$

where $d(\mathbf{a}_i,\mathbf{r})$ means the euclidean distance between $\mathbf{a}_i$ and $\mathbf{r}$, and p is a parameter determining what kind of mean of the distances is used, e.g., the quadratic mean and arithmetic mean.

The GD value is to be minimized and the ideal value is zero, which indicates that the set is precisely on the Pareto front. In the original version, the parameter p was set to 2. Unfortunately, this would make the evaluation value rather sensitive to outliers and also affected by the size of the solution set (when $N\to\infty$, $GD\to 0$ even if the set is far away from the Pareto front [122]). Setting $p = 1$ has now been commonly accepted.

Compared to those dominance-based convergence indicators (e.g., CI and $\mathcal{C}$), GD is more accurate in terms of measuring the closeness of solution sets to the Pareto front due to it considering the distance between points. However, a clear weakness of GD is not being Pareto compliant [61], [154]. This is very undesirable since GD, as a convergence indicator, fails to provide reliable evaluation results with respect to the weakest assumption of the DM's preferences. A simple example was given in [154]: consider two solution sets $\mathbf{A} = \{(2,5)\}$ and $\mathbf{B} = \{(3,9)\}$ on a bi-objective minimization scenario, where the reference set is $\mathbf{R} = \{(1,0),(0,10)\}$. Clearly, $\mathbf{A}$ dominates $\mathbf{B}$, but GD returns an opposite result: $GD(\mathbf{A}) = \sqrt{26} > GD(\mathbf{B}) = \sqrt{10}$. Recently, a modified GD was proposed to overcome this issue, called GD^+ [56], where the euclidean distance between $\mathbf{a}_i$ and $\mathbf{r}$ in Equation (2) is modified by only considering the objectives where $\mathbf{r}$ is superior to $\mathbf{a}_i$. Specifically,

$$d^+(\mathbf{a}_i,\mathbf{r}) = \left(\sum_{j=1}^{m}(\max\{\mathbf{a}_{ij}-\mathbf{r}_j,0\})^2\right)^{1/2}, \tag{3}$$

where m denotes the number of objectives, and $\mathbf{a}_{ij}$ denotes the value of solution $\mathbf{a}_i$ on the jth objective. This modification makes the indicator compliant with Pareto dominance. Going back to the above example, now we have the evaluation results of $\mathbf{A}$ better than $\mathbf{B}$ ($GD^+(\mathbf{A}) = 2 < GD^+(\mathbf{B}) = 3$. Finally, note that for both GD and GD^+, normalization of solution sets is needed as their calculation involves objective blending [75].

6.4 Spread (Δ)

The indicator *Spread* (aka Δ) [29] and its variants [125], [151] have been commonly adopted to evaluate the diversity (i.e., spread and uniformity) of solution sets in the field, e.g., in [31], [52], [99], [118], [120], [125], [126], [147]. Specifically, the indicator Δ of a solution set $\mathbf{A}$ (assuming the set only consisting of nondominated solutions) in a bi-objective scenario is defined as follows.

$$\Delta(\mathbf{A}) = \frac{d_{upper} + d_{bottom} + \sum_{i=1}^{n-1} |d_i - \overline{d}|}{d_{upper} + d_{bottom} + (n-1)\overline{d}}, \tag{4}$$

where n denotes the size of $\mathbf{A}$, d_i $(i = 1, 2, \ldots n-1)$ is the euclidean distance between consecutive solutions in the $\mathbf{A}$, and $\overline{d}$ is the average of all the distances d_i. d_{upper} and d_{bottom} are the euclidean distance between the two extreme solutions of $\mathbf{A}$ and the two extreme points of the Pareto front, respectively.

A small Δ value is preferred, which indicates a good distribution of the set in terms of both spread and uniformity. When $\Delta = 0$ means that solutions in the set are equidistantly spaced and their boundaries reach the Pareto front extremes.

A major weakness of Δ (including its variants) is that it only works reliably on bi-objective problems as where nondominated solutions are located consecutively on either objective. With more objectives, the neighbor of a solution on one objective may be far away on another objective [71]. This issue applies to any distance-based diversity indicator [75]. For problems with more than two objectives, region division-based diversity indicators are more accurate [75]. They typically divide the space into many equal-sized cells and then consider cells instead of solutions (e.g., counting the number of these cells). This is based on the fact that a set of more diversified solutions usually populate more cells. However, such indicators may suffer from the curse of dimension as they typically need to record information of every cell. In this regard, the diversity indicator *DCI* [70] may be a pragmatic option since its calculation only involves non-empty cells, thus independent of the number of cells (linearly increasing computational cost in objective dimensionality).

In addition, another indicator *Spacing* (aka *SP*) [121] has also been used to evaluate the diversity of solution sets in e.g., [109], [125]. However, this indicator can only reflect the uniformity (not spread) of solution sets [75].

6.5 Nondominated Front Size (*NFS*)

Used in e.g., [52], [103], [145], the *NFS* (also called Pareto Front Size, *PFS*) is to simply count how many nondominated solutions are in the obtained solution set. However, this indicator may not be very practical as in many cases all solutions in the obtained set are nondominated to each other, particularly in many-objective optimization. In addition, as by definition duplicate solutions are nondominated to each other, a set full of duplicate solutions would be evaluated well by *NFS* if there is no other solution in the set dominating them.

As such, a measure that only considers unique nondominated solutions which are not dominated by any other set seems more reasonable. Specifically, one can consider the ratio of the number of such solutions in each set to the size of the reference set (which consists of unique nondominated solutions of the collections of solutions obtained by the algorithms). In other words, we quantify the contribution of each set to the combined nondominated front of all the sets. Formally, let $\mathbf{A_{unf}}$ be the unique nondominated front of a given solution set $\mathbf{A}$ (i.e., $\mathbf{A_{unf}} \subseteq \mathbf{A} \wedge \mathbf{A_{unf}} \preceq \mathbf{A} \wedge \forall \mathbf{a_i} \in \mathbf{A_{unf}}, \nexists \mathbf{a_j} \in \mathbf{A_{unf}}, \mathbf{j} \neq \mathbf{i} : \mathbf{a_j} \preceq \mathbf{a_i}$). Then, the indicator, denoted as Unique Nondominated Front Ratio (*UNFR*), is defined as

$$UNFR(\mathbf{A}) = \frac{|\mathbf{a} \in \mathbf{A_{unf}} | \nexists \mathbf{r} \in \mathbf{R_{unf}} : \mathbf{r} \prec \mathbf{a}|}{|\mathbf{R_{unf}}|}, \tag{5}$$

where $\mathbf{R_{unf}}$ denotes the reference set which consists of the unique nondominated solutions of the collections of all solutions produced.

The *UNFR* value is in the range of [0,1]. A high value is preferred. Being zero means that for any solution in $\mathbf{A}$ there always exists some solution better in the other sets. Being one means that for any solution in the other sets there always exists some solution in $\mathbf{A}$ better than (or at least equal to) it (i.e., the reference set is precisely comprised by solutions of $\mathbf{A}$). In addition, *UNFR* is Pareto compliant.

6.6 Inverted Generational Distance (*IGD*)

IGD [26] is a well-known indicator in the field (e.g., in [5], [13], [32], [37], [39], [43], [52], [80], [89], [91], [126], [131]). As the name suggests, *IGD*, an inversion of *GD*, is to measure how close the Pareto front is to the obtained solution set. Formally, given a solution set $\mathbf{A}$ and a reference set $\mathbf{R}$, *IGD* is calculated as

$$IGD(\mathbf{A}) = \frac{1}{|\mathbf{R}|} \sum_{\mathbf{r} \in \mathbf{R}} \min_{\mathbf{a} \in \mathbf{A}} d(\mathbf{r}, \mathbf{a}), \tag{6}$$

where $d(\mathbf{r}, \mathbf{a})$ is the euclidean distance between $\mathbf{r}$ and $\mathbf{a}$. A low *IGD* value is preferable.

IGD is capable of reflecting the quality of a solution set in terms of all the four aspects: convergence, spread, uniformity, and cardinality. However, a major weakness of *IGD* is that the evaluation results heavily depend on the behavior of its reference set. A reference set of densely and uniformly distributed solutions along the Pareto front is required; otherwise, it could easily return misleading results [75]. This is particularly problematic in SBSE since the reference set is created normally from the collection of all the obtained solutions; its distribution cannot be controlled.

Consider an example in Fig. 6a, where comparing two solution sets $\mathbf{A}$ and $\mathbf{B}$. The reference set is comprised of all the nondominated solutions, i.e., the three solutions of $\mathbf{A}$ and the two boundary solutions of $\mathbf{B}$. As can be seen, $\mathbf{B}$ performs significantly worse than $\mathbf{A}$ in terms of convergence, with its solutions being either dominated by some solution in $\mathbf{A}$ or slightly better on one objective but much worse on the other objective; thus $\mathbf{B}$ unlikely to be preferred by the DM. However, *IGD* gives an opposite evaluation: $IGD(\mathbf{A}) \approx 2.80 > IGD(\mathbf{B}) \approx 1.08$.

In addition, the way of how the reference set is created makes IGD prefer a specific distribution pattern consistent with the majority of the considered solution sets [75]. In other words, if a solution set is distributed very differently

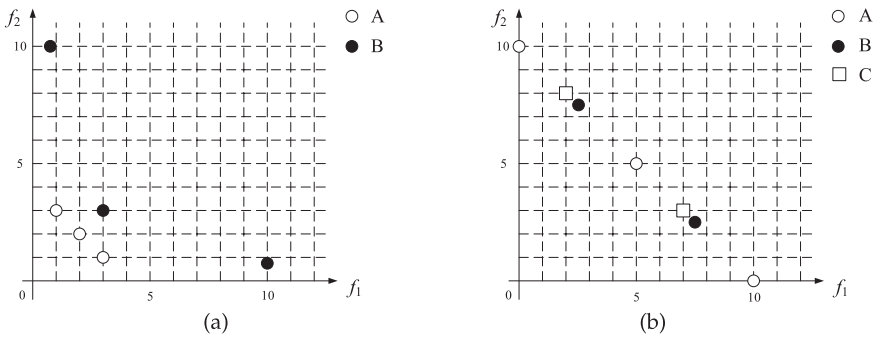


Fig. 6. Two examples that the collection of solution sets as the reference set may lead to misleading evaluations for IGD. (a) For two bi-objective sets $\mathbf{A} = \{(1,3),(2,2),(3,1)\}$ and $\mathbf{B} = \{(0.75,10),(3,3),(10,0.75)\}$, **A** should be highly likely to be preferred to **B** as solutions of **B** are either dominated by some solution in **A** or slightly better on one objective but significantly worse on the other objective. but IGD gives opposite results: $IGD(\mathbf{A}) \approx 2.80 > IGD(\mathbf{B}) \approx 1.08$. (b) For three bi-objective sets $\mathbf{A} = \{(0,10),(5,5),(0,10)\}$, $\mathbf{B} = \{(2.5,7.5),(7.5,2.5)\}$ and $\mathbf{C} = \{(2,8),(7,3)\}$, in general **A** may be likely to be preferred by the DM than **B** and **C** as it provides better spread and cardinality, but IGD gives opposite results: $IGD(\mathbf{A}) \approx 1.82 > IGD(\mathbf{B}) \approx 1.72 > IGD(\mathbf{C}) \approx 1.61$.

from others, then the set is likely to assign a poor IGD value whatever its actual distribution is. Fig. 6b is such an example. When comparing **A** with **B** (the reference set comprised of these two sets), we will have **A** evaluated better than **B** ($IGD(\mathbf{A}) \approx 1.41 < IGD(\mathbf{B}) \approx 2.12$). But if adding another set **C** which has the similar distribution pattern to **B** into the evaluation, and now the reference set is comprised of the three sets, we will have **A** worse than **B** ($IGD(\mathbf{A}) \approx 1.82 > IGD(\mathbf{B}) \approx 1.72$). A potential way to deal with this issue is to cluster crowded solutions in the reference set first and then to consider these well-distributed clusters instead of arbitrarily-distributed points, as did in the indicator PCI [72]. Yet, this could induce another issue — how to properly cluster the solutions in the reference set subject to potentially highly irregular solution distribution.

6.7 Hypervolume (HV)

Like IGD, HV [153] evaluates the quality of a solution set in terms of all the four aspects. Due to its desirable practical usability and theoretical properties, HV is arguably the most commonly used indicator in SBSE, e.g., used in [24], [30], [34], [37], [54], [93], [99], [103], [105], [125], [126], [131], [139], [144]. For a solution set, its HV value is the volume of the union of the hypercubes determined by each of its solutions and a reference point. It can be formulated as

$$HV(\mathbf{A}) = \lambda\Big(\bigcup\nolimits_{\mathbf{a}\in\mathbf{A}}\{\mathbf{x}|\mathbf{a} \prec \mathbf{x} \prec \mathbf{r}\}\Big), \tag{7}$$

where **r** denotes the reference point and λ denotes the Lebesgue measure. A high HV value is preferred.

A limitation of the HV indicator is its exponentially increasing computational time with respect to the number of objectives. Many efforts have been made to reduce its running time, theoretically and practically (see [75] for a summary), which makes the indicator workable on a solution set with more than 10 objectives (under a reasonable set size).

As stated previously, HV is in favor of knee points of a solution set, thus a good choice when the DM prefers knee points of the problem's Pareto front. In addition, the settings of the reference point can affect its evaluation results. Consider the two solution sets **A** and **B** in Fig. 7, where **A** consists of two boundary solutions, and **B** consists of four uniformly distributed inner solutions. When the reference point is set to (6,6) (Fig. 7a), **A** is evaluated worse than **B**. When the reference point is set to (11,11) (Fig. 7b), **A** is evaluated better than **B**. Fortunately, we can make good use of such a behavior of HV to enable the indicator to reflect the DM's preferences. If the DM prefers the extreme points, then a reference point can be set to be fairly distant from the solution sets' boundaries, e.g., doubling the Pareto front's range, namely, $\mathbf{r}_i = nadir_i + l_i$ where $nadir_i$ is the nadir point of the Pareto front (or the reference set, i.e., the combined nondominated front) on its ith objective, and l_i is the range of the Pareto front (or the reference set) on the ith objective. If there is no clear preference from the DM, unfortunately, no consensus regarding how to set the reference point has been reached in the multi-objective optimization field. A common practice is to set it 1.1 times of the range of the combined nondominated front (i.e., $\mathbf{r}_i = nadir_i + l_i/10$). Some recent studies [55] suggested to set it as $\mathbf{r}_i = nadir_i + l_i/h$, where h is an integer subject to $C_{m-1}^{h+m-1} \leq n < C_{m-1}^{h+m}$ (m and n being the number of objectives and the size of the considered set, respectively). In any case, the reference point setting is non-trivial — an appropriate setting needs to consider not only the number of objectives and the size of the solution set, but also the actual dimensionality of the set, its shape, etc.

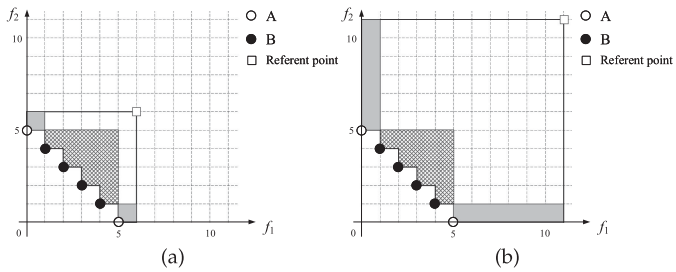


Fig. 7. An example that distinct reference points lead to that HV prefers different solution sets, where the set **A** consists of two boundary solutions ($\mathbf{A} = \{(0,5),(5,0)\}$), and the set **B** consists of four uniformly distributed inner solutions ($\mathbf{B} = \{(1,4),(2,3),(3,2),(4,1)\}$). The grey area $\subset HV(\mathbf{A})$ but $\not\subseteq HV(\mathbf{B})$ and the hatched area $\subset HV(\mathbf{B})$ but $\not\subseteq HV(\mathbf{A})$. In (a) where the reference point is (6,6), **A** is evaluated worse than **B**: $HV(\mathbf{A}) = 11 < HV(\mathbf{B}) = 19$. In (b) where the reference point is (11,11), **A** is evaluated better than **B**: $HV(\mathbf{A}) = 96 > HV(\mathbf{B}) = 94$.

6.8 ϵ-indicator

ϵ-indicator is another well-established comprehensive indicator frequently appearing in SBSE, e.g., [13], [32], [43], [52], [135]. It measures the maximum difference between two solution sets and can be defined as

$$\epsilon(\mathbf{A},\mathbf{B}) = \max_{\mathbf{b}\in\mathbf{B}} \min_{\mathbf{a}\in\mathbf{A}} \max_{i\in\{1\ldots m\}} \mathbf{a}_i - \mathbf{b}_i, \tag{8}$$

where $\mathbf{a}_i$ denotes the objective of $\mathbf{a}$ for the ith objective and m is the number of objectives. A low value is preferred. $\epsilon(\mathbf{A},\mathbf{B}) \leq 0$ implies that $\mathbf{A}$ weakly dominates $\mathbf{B}$. When replacing $\mathbf{B}$ with a reference set that represents the Pareto front, the ϵ-indicator becomes a unary indicator, measuring the gap of the considered set to the Pareto front.

ϵ-indicator is Pareto compliant and user friendly (parameter-free and quadratic computational effort). Yet, the calculation of ϵ-indicator only involves one particular objective of one particular solution in either set (where the maximum difference is), rendering its evaluation omitting the difference on other objectives and other solutions. This may lead to different solution sets having same/similar evaluation results, as reported in [78]. In addition, in some studies [111], ϵ-indicator has been empirically found to behave very similarly as HV in ranking solution sets.

6.9 Summary

Table 9 summarizes the above 12 indicators on several aspects, namely, (i) what kind of quality aspect(s) they are able to reflect, (ii) if they are Pareto compliant, (iii) what we need to take care of when using them, and (iv) what situation they are suitable for. The following guidelines can be derived from the table.

- If the DM wants to know the convergence quality of a solution set to the Pareto front, GD^+ (instead of GD) could be an ideal choice — it is Pareto compliant and the reference set required can be set as the combined nondominated front of all the considered sets, not necessarily a set of uniformly-distributed points. If the DM wants to know the relative quality between two solution sets in terms of the Pareto dominance relation, CI (or $\mathcal{C}$) could be a choice.
- If the DM wants to know the diversity quality (both spread and uniformity) of a solution set, for bi-objective cases, Δ is a good choice; for problems with more objectives, DCI can be used. SP can only reflect the uniformity of a solution set which may not be very useful — uniformly distributed solutions concentrating in a tiny area typically not in the DM's favor.
- The indicator $UNFR$ should replace NFS to measure the cardinality of solution sets.
- Regarding comprehensive evaluation indicators, HV can generally be the first choice, especially when the DM prefers knee points. In addition, if the DM prefers extreme solutions, the reference point needs to be set fairly distant from the solution sets' boundaries. ϵ-indicator is user-friendly, but is less sensitive to solution sets' quality difference than HV since its value only lies upon one particular solution on one particular objective. IGD may not be very practical as it requires a Pareto front representation consisting of densely and uniformly distributed points.

7 Methodological Guidance to Quality Evaluation in Pareto-Based SBSE

In this section, we provide guidance on how to select and use quality evaluation methods in Pareto-based SBSE. As discussed previously, selecting and using quality evaluation methods needs to be aligned with the DM's preferences.[11] A solution set being evaluated better means nothing but its solutions having a bigger chance to be picked out by the DM. However, to different problems or even to the same problem but under different circumstances, the articulation of the DM's preferences may differ. In some cases, the DM is confident to articulate their preferences (or can be easily derived from contextual information); e.g., they see one objective more important than others. In some cases, the DM may experience difficulty in precisely articulating their preferences; e.g., they are only able to provide some vague preference information such as a fuzzy region around one point. In some other cases, the DM's preferences may not be available at all; e.g., when the DM wants to see what the whole Pareto front looks like before articulating their preferences. Therefore, quality evaluation needs to be conducted in accordance with different cases. Next, we consider four general cases of quality evaluation with respect to the DM's preferences.

7.1 When the DM's Preferences Are Clear

The case of the DM's preferences being clear can often fall into two categories over Pareto-based SBSE problems. The first is when relative importance/weighting among the objectives considered can be explicitly expressed and quantified, e.g., in [125]. It is worth noting that the weighting between objectives may not need to be fixed *a priori*. For example, in the case of interactive Pareto-based SBSE for software modeling and architecting problems [128], the DM is asked to explicitly rank the relative importance of the objectives as the search proceeds. Under this circumstance, the sum of the weighted objectives can be used to find the fittest solution from a solution set, and then determine the quality of the set.

The other category concerns when the DM prefers some objective to some other (i.e., a clear priority can be assumed, which is a unique situation that often implies some clear contextual information of a hard requirement under the SBSE problem), or when the DM is only interested in solutions which is up to scratch on some objective (which could be seen as a constraint). This happens frequently in the software product line configuration problem [52], [54], [93], [118], [119], [120], [144], where the correctness of the products (i.e., the feature model's dependency compliance) is always of higher priority than other objectives such as the richness and the cost of the model — only the solutions (products) that achieve full dependency compliance are of

11. For convenience, from this point forwards we use DM's preference information as a general term, which refers to not only the preferences that the DM articulates but also the requirements derived from problem nature and contextual information.

TABLE 9
A Summary of Representative Quality Indicators Used in SBSE, Their Usage Note/Caveats and Applicable Conditions

Indicator	Convergence	Spread	Uniformity	Cardinality	Pareto Compliant	Usage Note/Caveats	Applicable Conditions
CI	−			−	+	(i) not able to distinguish between sets if their solutions are nondominated to each other, which may happen frequently in many-objective optimization; (ii) binary indicator which evaluates relative quality of two sets and cannot be converted into a unary indicator.	(i) when the DM wants to know the relative quality difference (in terms of dominance relation) between two sets, and (ii) when the Pareto front size is relatively small, e.g., on some low-dimensional combinatorial problems.
$\mathcal{C}$ (CS)	−			−	+	(i) not able to distinguish between sets if their solutions are nondominated to each other, which may happen frequently in many-objective optimization; (ii) binary indicator which evaluates relative quality of two sets and cannot be converted into a unary indicator; (iii) removing duplicate solutions before the calculation.	(i) when the DM wants to know the relative quality difference (in terms of dominance relation) between two sets, and (ii) when the Pareto front size is relatively small, e.g., on some low-dimensional combinatorial problems.
GD	+					(i) additional problem knowledge: a reference set that represents the Pareto front (not necessarily a set of uniformly-distributed points); (ii) each objective needs normalization; (iii) may give misleading results due to not holding the Pareto compliance property.	(i) when the DM wants to know how close the obtained sets from the Pareto front, (ii) when the compared sets are nondominated to each other (i.e., no *better* relation between the sets), and (iii) when the Pareto front range can be estimated properly (e.g., no DRS points in the reference set [75]).
GD^+	+				+	(i) additional problem knowledge: a reference set that represents the Pareto front (not necessarily a set of uniformly-distributed points); (ii) each objective needs normalization.	(i) when the DM wants to know how close the obtained sets from the Pareto front.
$Spread$		+	+			(i) additional problem knowledge: extreme points of the Pareto front; (ii) each objective needs normalization; (iii) reliable only on bi-objective problems.	(i) when the DM wants to know the diversity (including both spread and uniformity) of the obtained sets on bi-objective problems, and (ii) when the compared sets are nondominated to each other.
DCI		+	−	−	−	(i) additional problem knowledge: proper setting of the grid division; (ii) M-nary indicator that evaluates relative quality of M sets, but can be converted into a unary one by comparing the obtained set with the Pareto front.	(i) when the DM wants to know the diversity of the obtained sets.
SP			+			(i) each objective needs normalization; (ii) cannot reflect the spread of solution sets.	(i) when the DM wants to know the uniformity of the obtained sets, and (ii) when the compared sets are nondominated to each other.
NFS				+		(i) not able to compare sets as it only counts the number of nondominated solutions in a set.	(i) not reliable when the DM wants to compare sets.
$UNFR$				+	+	None	(i) when the DM wants to compare the cardinality of sets, particularly how much they contribute the combined nondominated front.
IGD	+	+	−	−		(i) additional problem knowledge: a reference set that well represents the Pareto front (i.e., densely and uniformly distributed points); (ii) each objective needs normalization; (iii) may give misleading results due to the lack of Pareto compliance property.	(i) when the DM wants to know how well the obtained sets can represent the Pareto front, (ii) when the compared sets are nondominated to each other, and (iii) when there is a Pareto front with densely and uniformly distributed points.
HV	+	+	−	+	+	(i) additional problem knowledge: a reference point that worse than the nadir point of the Pareto front; (ii) exponentially increasing computational cost in objective dimensionality. (iii) the DM can specify the reference point according to their preference to extreme solutions or to inner ones.	(i) when the DM wants to know comprehensive quality of the obtained sets, especially suitable if the DM prefers knee points of the problem, and (ii) when the objective dimensionality is not very high.
ϵ-indicator	+	+	−	−	+	(i) each objective needs normalization; (ii) binary indicator, but can be converted into a unary indicator by comparing the obtained set with the Pareto front; (iii) differently-performed sets may have the same/similar evaluation results.	(i) when the DM wants to know maximum difference between two solution sets (or the obtained solution set from the Pareto front).

"+" generally means that the indicator can well reflect the specified quality (or meet the specified property). "−" for convergence means that the indicator can reflect the convergence of a set to some extent; e.g., indicators only considering the dominance relation as convergence measure. "−" for spread means that the indicator can only reflect the extensity of a set. "−" for uniformity means that the indicator can reflect the uniformity of a set to some extent; i.e., a disturbance to an equally-spaced set may not certainly lead to a worse evaluation result. "−" for cardinality means that adding a nondominated solution into a set is not surely but likely to lead to a better evaluation result and also it never leads to a worse evaluation result. "−" for Pareto compliance means that the indicator holds the property subject to certain conditions.

interest. This is obvious, as a violation of dependency implies faulty and incorrect configuration, thus valueless in practice. A similar situation applies to the test case generation problem [53], [103], [104], [124], [150] where the DM is typically interested in test suites with full coverage. In addition, the DM may only be interested in solutions that reach a certain level on some objective. For example, in software deployment and maintenance, it is not uncommon to have a statement like "`The software service shall be available for at least 95 percent of the time`". In such a case, it is rather clear that any value of availability less than 95 percent is unacceptable, while anything beyond 95 percent can be considered.

An appropriate way to perform evaluation under the above circumstance is to transfer the DM's preferences into the solution set to be evaluated. This can be done by first removing solutions that are irrelevant from the set. After that, the set of the remaining solutions is evaluated, subject to two situations: if the remaining solutions are of the same value on the objective(s) where the DM articulates their preferences, then the quality evaluation is performed only on the other objectives; otherwise, the evaluation is done on all the objectives. The former has been commonly seen when the DM is only interested in solutions which achieve the best of the objective, such as the solutions with full coverage for the test case generation problem, whereas the latter often applies when the DM is interested in a particular threshold of solution quality on the objective, such as in the software deployment and maintenance case mentioned above, only the solutions with availability values not less than 95 percent would be evaluated.

7.2 When the DM's Preferences Are Vague/Rough

It is not uncommon that there exist important, yet imprecise preferences in the SDLC. In general, they are mainly derived from the non-functional requirements recorded in documentations, notes, and specifications, which are often vague in nature, as in [1], [11], [51], [67], [84]. For example, in the software configuration and adaptation problem, some statements may be rather ambiguous like "`the first objective should be reasonable and the others are as good as possible`". In such a situation, one may not be able to integrate the preferences into the quality evaluation since it is not possible to quantify qualitative descriptions like "reasonable". As such, a safe choice is to treat them as a general multi-objective optimization case (i.e., without specific preferences).

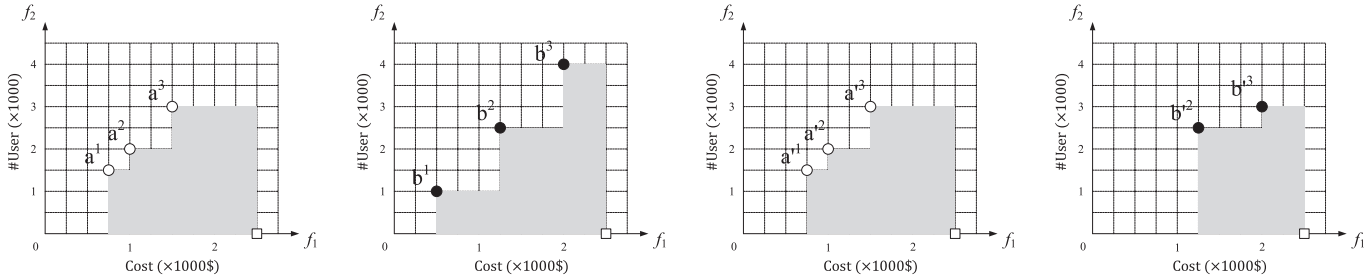
In other situations, the SBSE researchers/practitioners may give some preference information around some values/thresholds on one (or several) objective. This, in contrast to the case of the DM's preferences being clear, allows some tolerances on the specified value/threshold. For example, the software may have a requirement stating that "`the cost shall be low while the product shall support ideally up to 3000 simultaneous users`". This typically happens for SME where the budget of a software project (e.g., money for buying required Cloud resources or the consumption of data centers) is low, and thus it is more realistic to set a threshold point such that certain level of performance (e.g., 3000 simultaneous users) would be sufficient (anything beyond is deemed as equivalent). However, while the requirement gives a clear cap of the best performance expected, it does not constrain on the worst case, implying that it allows tolerances when the 3000 users goal cannot be met.

Despite not impossible, it can be a challenging task to find a quality indicator that is able to reflect such preference information. First, the quality indicator should be capable of accommodating such preference information in the sense that the evaluation results can embody it. Second, the introduction of the preferences should neither compromise the general quality aspect that the indicator reflects, nor violate properties that the indicator complies with (e.g., being Pareto compliant). In this regard, the indicator HV [153] could be a good choice since it (i) can relatively easily integrate the DM's preference information [136], [152] and (ii) can still be Pareto compliant after a careful introduction of the DM's preference information [152].

To integrate preferences into HV, one approach, called the weighted HV presented in [152], is to interpret the HV value as the volume of the objective space enclosed by the attainment function [28] and the axes. Here, the attainment function is to give for each vector in the objective space the probability that it is weakly dominated by the outcome of the solution set. Then, to give different weights to different regions by a weight distribution function, the weighted HV is calculated as the integral over the product of the weight distribution function and the attainment function [152]. This essentially transforms the preference information into a weight distribution function to unequalize the HV contribution from different regions. However, it is not trivial to construct a weight distribution function that is able to reflect preferences expressed by the SBSE researchers/practitioners. Even for the situation that the preference information is clear (e.g., clear weighting between objectives), the preferences cannot be used as the weight distribution function because of the interaction of the weight distribution function and the attainment function in the calculation.

Another (perhaps more pragmatic) approach is to directly transform the original solutions into new solutions which accommodate the preference information, and then apply HV (or other quality indicators) to the new solutions, provided that such a transformation is in line with the selected indicator. For instance, consider the above example that the cost shall be low while the product shall ideally support up to 3000 users. Let us say that there are two solutions $\mathbf{a} = (1500, 3000)$ and $\mathbf{b} = (2000, 4000)$ obtained for this problem. Solution $\mathbf{a}$ has a lower cost while solution $\mathbf{b}$ can support more users. However, according to the preference information, for the number of users anything beyond 3000 can be deemed as equivalent. As such, the user number of the solution $\mathbf{b}$ can be transformed to 3000, that is, now $\mathbf{b} = (2000, 3000)$, worse than (i.e., dominated by) $\mathbf{a}$. The HV indicator can capture such dominance relation information — a dominated solution is always evaluated worse by HV than one dominating it. Next, we look at a case study based on this example to see how such transformation affects the evaluation results.

Consider a situation of designing a product with the requirements that "`the cost shall be low while the product should be able to support at least 1500`



(a) $HV(\mathbf{A}) = 4,375,000$ (b) $HV(\mathbf{B}) = 4,625,000$ (c) $HV(\mathbf{A}') = 4,375,000$ (d) $HV(\mathbf{B}') = 3,375,000$

Fig. 8. HV comparison of with/without integrating the DM's preferences "`the cost shall be low while the product should be able to support at least 1500 simultaneous users and ideally reach 3000 users`" into solutions on the basis of transformation function Equation (9). Without considering the preferences ((a) and (b)), the solution set $\mathbf{B}$ is evaluated by HV better than $\mathbf{A}$ as it can provide more diverse solutions. Yet, when considering the preferences, the solution $\mathbf{b}^1$ in $\mathbf{B}$ will be no interest to the DM (thus discarded) as its supported user number is less than 1500, and the number of users that $\mathbf{b}^3$ supports can be regarded down to 3000 as 3000 is the best expected value. After this transformation ((c) and (d)), the set $\mathbf{B}'$ is evaluated significantly worse than $\mathbf{A}'$.

`simultaneous users and ideally reach 3000 users`". As can be seen, the first objective cost is a normal one (i.e., the lower the better), while for the second objective the number of simultaneous users, there are two types of preferences: clear one and vague one. The statement "`support at least 1500 users`" is a clear one, which means the product is useless if it cannot support 1,500 users. The statement "`ideally reach 3000 users`" is a vague one, which implies that despite the threshold, it is acceptable to support less users and it will have the same level of satisfaction even if more users are supportable. As such, we can have the following transformation function.

$$\mathbf{a}'_2 = \begin{cases} 3000, & \mathbf{a}_2 > 3000 \\ \mathbf{a}_2, & 1500 \leq \mathbf{a}_2 \leq 3000 \\ \text{to disgard } \mathbf{a}, & \mathbf{a}_2 < 1500 \end{cases}, \tag{9}$$

where $\mathbf{a}'_i$ denotes the transformed value of solution $\mathbf{a}$ on the ith objective.

Now let us assume two solution sets $\mathbf{A} = \{\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3\}, \mathbf{B} = \{\mathbf{b}^1, \mathbf{b}^2, \mathbf{b}^3\}$ obtained by two search algorithms, where $\mathbf{a}^1 = (750, 1500), \mathbf{a}^2 = (1000, 2000), \mathbf{a}^3 = (1500, 3000), \mathbf{b}^1 = (500, 1000), \mathbf{b}^2 = (1250, 2500), \mathbf{b}^3 = (2000, 4000)$, shown in Figs. 8a and 8b. We want to evaluate and compare them under the circumstances with/without the preference information given above to see how transferring preferences into solutions affects the evaluation results. As seen in Figs. 8a and 8b, without considering the preferences $\mathbf{A}$ is evaluated worse than $\mathbf{B}$ ($HV(\mathbf{A}) = 4,375,000 < \mathbf{B} = 4,625,000$. This makes sense as the solutions of $\mathbf{B}$ spread more widely than those of $\mathbf{A}$. Yet, when considering the preferences of the DM (transferred by Equation (9)), while the set $\mathbf{A}$ stays unchanged ($\mathbf{A}' = \mathbf{A}$), the solution $\mathbf{b}^1$ will be discarded and the solution $\mathbf{b}^3$ will become $(2000, 3000)$. As a result, $\mathbf{A}'$ is evaluated significantly better than $\mathbf{B}'$ ($HV(\mathbf{A}') = 4,375,000 > \mathbf{B}' = 3,375,000$), as shown in Figs. 8c and 8d. This shows that the integration of the DM's preferences can completely change the evaluation results between solution sets.

7.3 When the DM's Interest Is in Some Specific Parts of the Pareto Front

Sometimes, the DM may be more interested in specific part/solutions of the Pareto front than others. Knee points are certainly among such solutions, preferred in many situations, e.g., in [17], [19], [35], [37], [44], [68], [89], [114], [137]. Knee points are points on the Pareto front where a small improvement on one objective would lead to a large deterioration on at least one other objective. They represent "good" trade-offs between conflicting objectives, thus naturally more of interest to the DM. For example, on the cloud autoscaling problem [17], where different cloud tenants (users) may introduce conflicting objectives due to the interference and shared infrastructure. From the perspective of the cloud vendor, ensuring fairness among tenants of the same class is often the top priority and thus the knee solutions are more of interest. As we explained previously, HV is a good choice in such a situation, alongside other indicators like the ϵ-indicator [154], IGD^+ [56] and PCI [72], whereas unfortunately IGD [26] is not one of them, despite being widely used, e.g., in [37], [89].

Another relatively common situation is that the DM may be more interested in the extreme solutions (e.g., in [131], [134], [148]), namely, solutions achieving the best on one objective or another. For example, for the service composition problem [134], one may prefer the extreme solutions around the edges, e.g., those with low latency but high cost, or vice versa. For this situation, HV can also be a viable solution. As shown previously (Section 5.7), setting the reference point fairly distant from the combined nondominated solution set gives the extreme solutions bigger weighting on the evaluation results. Besides, one may directly compare solution sets through their best value on the corresponding objective(s). Such a DOE measure, in contrast to HV which provides comprehensive evaluation results, returns the objective values which are straightforward for the DM to understand.

7.4 When the DM's Preferences Are Completely Unavailable

As can be seen in Table 8, the majority of studies in Pareto-based SBSE effectively do not involve any preference. For this situation, a solution set that well represents the whole Pareto front is preferred. As aforementioned, the "representation" can be broken down to the quality aspects convergence, diversity (i.e., spread and uniformity), and cardinality. Naturally, it

is expected to consider quality indicators which (together) are able to cover all of them.

In general, there are two ways to implement that in practice. One is to consider several indicators, each responsible for one specific aspect. For example, GD^+ [56] is for a solution set's convergence, Δ [29] for diversity (under the bi-objective circumstance), and $UNFR$ for cardinality. The other one is to consider a comprehensive indicator to evaluate all the aspects. Such indicators include HV, IGD, and ϵ-indicator. Today, there is a tendency to use comprehensive indicators. Numerous recent studies used HV and IGD. However, as explained previously, IGD may not be an ideal indicator in Pareto-based SBSE as a Pareto front representation with densely and uniformly distributed points is usually unavailable in practice.

In addition, when using comprehensive indicators we suggest to consider multiple differently-behaving indicators if applicable. Each indicator has its own (explicitly or implicitly) preferences. A solution set evaluated better on an indicator is often evaluated better as well on another similar indicator; this means nothing but the set favored under this type of preference. When a solution set is evaluated better on all of the considered indicators whose preferences are quite different, then that set certainly has a higher chance to be chosen by the DM. Unfortunately, many comprehensive indicators behave similarly as HV [78], [111], namely, preferring knee points of the Pareto front rather than a set of uniformly-distributed solutions on the Pareto front, such as $R2$ [46] and ϵ-indicator (except IGD which, however, is not applicable typically). Therefore, as a supplement to HV, considering a quality indicator that can well evaluate the diversity of a solution set sounds reasonable. In this regard, the indicator Δ ($Spread$) [29] may be chosen for the bi-objective case and DCI [70] for the more-objective case.

7.5 Aided Evaluation Methods

The above are four general cases of solution sets' quality evaluation on the basis of the DM's preferences. On top of those, there exist some quality indicators for specific SBSE scenarios (see Table 8), which we call problem-specific indicators (PSI). For example, in the library recommendation [99], top-k accuracy, precision, and recall on history datasets are commonly used PSI for evaluating recommendation systems. For another example, in the software modularization problem the PSI MoJoFM [65], derived from the MoJo distance, compares a produced solution to a given "golden rule" solution, which naturally represents the DM's preference over the objectives such as cohesion and coupling [140]. Another example can be seen in the test case generation problem where some works use multi-objectivization to improve the code coverage [100], [101], [102]. They reformulate the coverage criterion as a many-objective optimization problem, where the objectives to be optimized are different coverage targets (e.g., branches in [100]), but only the total coverage of all test cases in the produced test suite is of interest. In this case, such a total coverage criterion can be regarded as a problem-specific indicator, as it does reflect the quality of a solution set in this specific problem but not directly involved in the search-based optimization. Overall, such PSI indicators not only represent more "accessible" quality evaluation of solution sets (i.e., how they perform under the practical problem background), but also usually imply some preferences from the DM. Therefore, it is highly recommended to include them in the evaluation if existing.

Nevertheless, it is worth noting that PSI indicators usually need to work together with generic quality indicators (e.g., those in Table 9) to provide reliable evaluations since they may be irrelevant to Pareto-based optimization (e.g., only focusing on particular objectives in evaluation). For example, the study [106] mainly relies on APFD, the average percentage of fault detected, to evaluate the solution set of prioritized test cases. Indeed, APFD is a frequently used PSI in the test case prioritization, but it can only reflect the rate of fault detected, not the reliance of test cases, both of which are the objectives to be optimized for the problem.

In addition, plotting representative solution sets (SSP) is also desirable as an auxiliary evaluation, as it empowers the SBSE researchers/practitioners to get a sense of what the solution set looks like. This is very helpful not only for solution set comparison, but also for the DM to understand the problem and then perhaps to refine their preferences further.

To use SSP, for an algorithm involving stochastic elements we suggest plotting the solution set in a particular run which corresponds to the evaluation result (obtained by a comprehensive quality indicator, e.g., HV) that is the closest to the median value in all the runs. Alternatively, for optimization problems with two and three objectives, median attainment surfaces [38], [63], [79] can be used to visualize the performance of the algorithm with respect to all the runs (which have already been adopted in the literature [24], [34], [142], [148]). For problems with more objectives, the parallel coordinates plot (instead of the scatter plot) is a helpful tool, which can reflect the convergence and diversity of a solution set to some extent [73]. It has started to be used recently, e.g., in [53], [89], [91], [143], [144].

7.6 A General Procedure

Based on the above, we now are in a good position to provide a general procedure of how to evaluate solution sets in Pareto-based SBSE in Fig. 9. At first, we suggest conducting some screening (**P1** in the figure) to filter out trivial solutions in the considered solution sets according to the nature of the optimization problem in SBSE. The trivial solutions can be seen as those which are straightforward to obtain and would never be of interest to the DM, but may affect the evaluation result. e.g., the solution with zero cost and zero coverage in the example of Fig. 4.

After the filtering, it comes to evaluating solution sets according to the DM's preference information (**D1**). If there is no preference information available at all (e.g., the effort estimation and test case prioritization problem), we suggest to consider quality indicators that together are able to accurately reflect all the quality aspects (**D2**–**D5**). For example, one can consider separately evaluating distinct quality aspects of solution sets, e.g., GD^+ for convergence (together with CI if willing to know the dominance relation between sets), DCI for diversity (when involving ≥ 3 objectives), and $UNFR$ for cardinality. Alternatively, one can consider evaluating the comprehensive quality of solution sets, e.g., HV in most cases; or even some mix (e.g., HV plus $UNFR$) if it requires understanding on some specific quality aspects on top of solution sets' general quality. Note that the cardinality of solution sets tends to have more

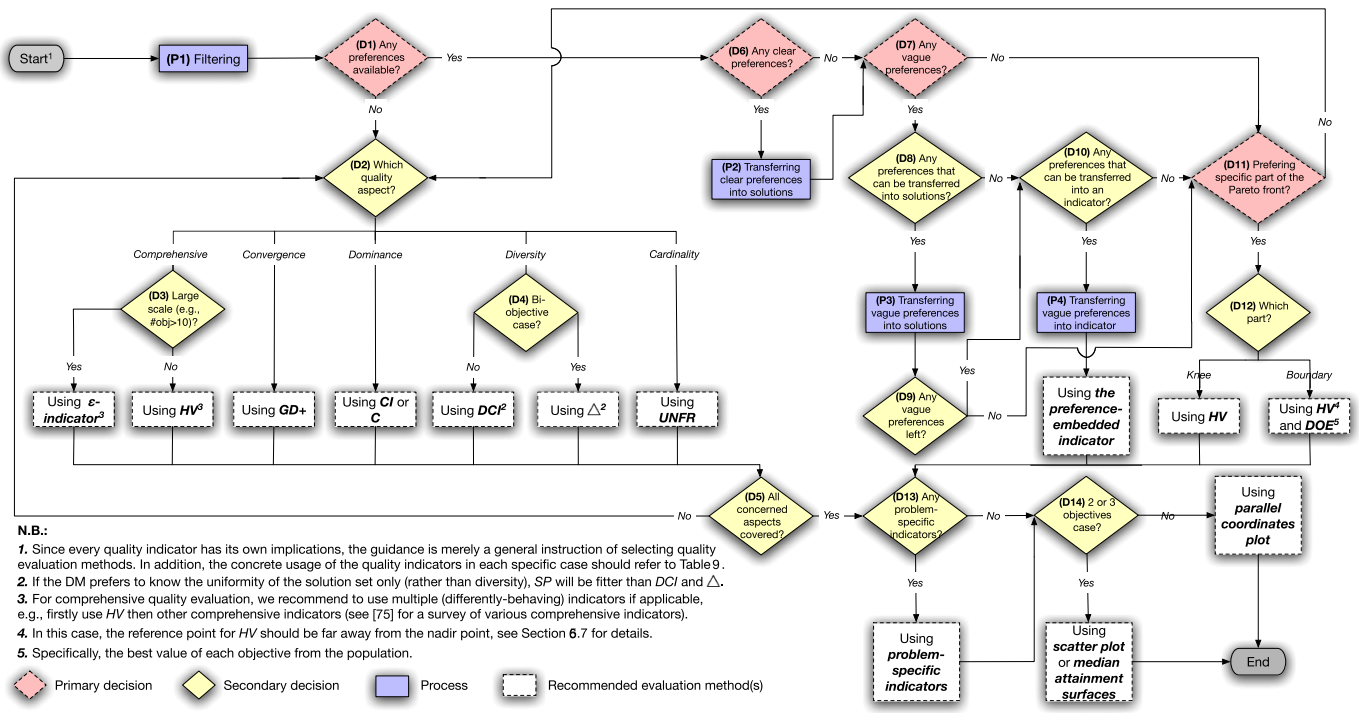


Fig. 9. General procedure of quality evaluation in Pareto-based SBSE.

weight in the SBSE area than some other areas such as evolutionary computation, since many SBSE problems are combinatorial ones, where their Pareto front size may be relatively small and it is likely to have comparable solutions (e.g., dominated/duplicate solutions) from different sets. In any case, whatever indicators considered, the way of using them needs to comply with their usage note and caveats (see Table 9).

If there is preference information available, we recommend first to see whether (part of) it belongs to clear preferences (**D6**); if so, such as the software product line configuration problem, one can transfer the clear preferences into the solutions (**P2**), as what we discussed in Section 7.1. It is necessary to note that sometimes after the transfer there is only one objective left to be considered (e.g., in the example of Fig. 4). In this case, the best value on that remaining objective represents the quality of the solution set.

After considering clear preferences, one needs to see whether there exist some vague preferences (**D7**). if the answer is yes, e.g., the software configuration and adaptation problem, then we transfer those that are transferable (**D8, P3, D9**), e.g., the example in Fig. 8 and resolution from Section 7.2. After that, we recommend to check whether the rest preferences can be transferred into an indicator (**D10**); if so, one can then transfer them (**P4**), e.g., transferring certain preferences into a weight distribution function in the weighted *HV* [152], and use that indicator to evaluate the solution sets as shown in Section 7.2.

When the preference information cannot be accommodated into an indicator, the next step is to check if the DM prefers a specific part of the Pareto front (**D11**), such as the software modularization and service composition problem. As we discussed in Section 7.3, if one prefers knee points on the Pareto front (i.e., well-balanced solutions between conflicting objectives), then *HV* is a good option. If one prefers boundary solutions, then *HV* with an unusual configuration of its reference point can be used, alongside with reporting the best value of relevant objective(s) in the population.

Note that the DM may present several types of preference information. For example, one may specify a clear threshold on one objective and at the same time be interested in knee points on the Pareto front — a typical case when non-functional requirements of the software are involved. Another example has been seen in the situation of Fig. 8, where the DM's preferences contain both clear and vague information. In addition, it is necessary to mention that there do exist some situations where the DM's preferences cannot be quantified/transferred properly, e.g., the DM may state like "`the cost should be reasonable`". In such a situation, we suggest proceeding to **D2** — the general multi-objective optimization case (without specific preferences).

After going through all possible cases of the DM's preferences, it comes to check the last two quality evaluation methods, problem-specific evaluation, i.e., *PSI* (**D13**), and solution set plotting, i.e., *SSP* (**D14**). These are two very helpful methods in reflecting the solution set's quality that may have not been captured by generic quality indicators.

8 Threats to Validity

Threats to construct validity can be raised by the research methodology, which may not serve the purpose of surveying the evaluation methods for Pareto-based optimization in existing SBSE studies. We have mitigated such threats by following the systematic review protocol proposed by Kitchenham *et al.* [60], which is a widely recognized search methodology for conducting a survey in the SE research. Another threat is related to the citation count used in the exclusion criteria. Indeed, it is difficult to set a threshold for such, as the citation count itself cannot well reflect the impact of work. Since there is no metric that suffices to do

so, in this work we used the citation count and set a threshold by averaging the candidate studies. It is however worth noting that we do not seek to provide a comprehensive review over the entire SBSE field, but to capture major trends on the evaluation of solution sets, which can at least provide some sources for analyzing and building the methodological guidance. Therefore it is necessary to reach a trade-off between the trend coverage and the efforts required for detailed data collections of the studies.

Threats to internal validity may be introduced by having inappropriate classification and interpretation of the SBSE papers, their implied preferences, and used quality indicators/evaluation methods. We have limited this by conducting three iterations of study reviews by the first two authors. Error checks and investigations were also conducted to correct any issues found during the search procedure. The key issues identified have also been resolved among the first two authors or by counseling external researchers.

Threats to external validity may restrict the generalizability of the proposed guidance and considered cases. We have mitigated such by conducting the survey more widely and deeply: it covers 717 searched papers published between 2009 and 2019, on 36 venues from seven repositories; while at the same time, extracting 95 prominent primary studies following the exclusion and inclusion procedures. This has included 21 most noticeable SBSE problems that spread across the whole SDLC. The extracted assumptions of the DM's preferences, together with rigorous analyses of the 12 representative quality indicators (i.e., either used widely in SBSE or proposed herein for a more accurate evaluation), have provided rich sources for us to establish a general methodological guidance for the community.

Finally, although our guidance has been designed in a way that it aims to cover a wide range of SBSE problems, it is always possible that there are situations which we have unfortunately missed; for example, the behavior of solutions in the decision space, e.g., their diversity and robustness. As different settings of parameters (i.e., decision variables) may lead to similar/same solutions' quality (e.g., multiple points in the decision map to a single point in the objective space), the DM of course likes those which are easier to implement. Therefore, a set of diverse solutions in the decision space are preferred, providing more options for the DM. Another aspect that the DM may consider is robustness [13], which is related to how fast the quality of solutions degrades when varying their parameters (decision variables). This issue is particularly important in an uncertain environment where the solution may not be able to be deployed accurately and/or the objective functions estimated may be of a margin of error. Therefore, robust solutions are preferred to sensitive ones even if their quality is slightly lower in some circumstances. Overall, in those cases, an evaluation of solution sets' quality both in the decision space and in the objective space is needed.

9 Related Work

Various surveys on SBSE (e.g., [47], [50], [85]) reveal intense interests in developing computational search methods for complex optimization problems in SE. Some of them focus on or are relevant to Pareto-based multi-objective optimization. For example, Sayyad *et al.* [117] performed a brief literature review of SBSE studies that used Pareto-based evolutionary algorithms for multi-objective optimization problems; Boussaïd *et al.* [8] conducted a comprehensive survey on search-based model-driven engineering and classified relevant search algorithms into single- and multi-objective ones; Ramírez *et al.* [110] reviewed SBSE studies on a subarea of multi-objective optimization, many-objective optimization, where the number of objectives is larger than 3. In general, these papers concentrate on the development of search algorithms for Pareto-based multi-objective optimization problems; very few touch on the quality evaluation of the results obtained by search algorithms until recently.

Wang *et al.* [138] proposed a practical guide for SBSE researchers/practitioners to select quality indicators in Pareto-based optimization, on the basis of the results of experimental studies evaluating eight quality indicators in three industrial and real-world problems. They first classified these indicators into four categories, convergence, diversity, combination, and coverage, and then they, based on empirical observations, have drawn several conclusions about the indicator selection. For example, they have concluded that it does matter which indicator to select in the diversity category, but it does not matter which indicator to select within the same convergence or combination category.

Very recently, Ali *et al.* [4] substantially extended Wang *et al.*'s work [138] and provided a set of guidelines drawn from an extensive empirical evaluation in nine SBSE problems from industrial, real-world and open-source projects. From these experiments, they produced 22 observations based on statistical comparisons between six multi-objective evolutionary algorithms. They have claimed that the differences in SBSE problems have high effect on the consistency of quality indicators' evaluation results, whereas the effect of search algorithms is low. A noticeable difference from [138] is that the guidance provided did not build on a classification of indicators.

Li *et al.* [74] conducted a critical review of Wang *et al.*'s work [138]. They argued that some conclusions (e.g., it matters which indicator to select in the category diversity) are actually caused by the inaccurate classification of the considered indicators. More importantly, they argued that even if an accurate classification is made, one still cannot draw any conclusions like it does not matter which indicator to select, whether in the same category or across different categories.

Indeed, as can be seen in Section 6, each quality indicator has its own distinct quality implications. A solution set being evaluated better by an indicator does not mean that it generally has higher quality, but rather that it is preferred under the assumption that the indicator accurately reflects the DM's preferences. However, different DMs may prefer different trade-off solutions between objectives, even for the same problem. For example, for the project scheduling problem, in some scenarios, the DM may prefer knee solutions [35], [44], [114], in some other scenarios, the DM may prefer widely distributed solutions [24], in some other scenarios, the DM may prefer specific solutions relying on the Analytic Hierarchy Process [125]. Consequently, observations on quality indicators drawn from an empirical investigation on specific SBSE scenarios may not be well generalized. This suggests a need of a general, methodological guidance on how to select and use indicators in SBSE.

Such a guidance is not based upon empirical studies on specific problems but upon the fundamental goal of multi-objective optimization — supplying the DM a set of solutions which are the most consistent with their preferences.

It is worth mentioning that a recent survey paper [75] on quality evaluation in multi-objective optimization appeared, albeit not specific for SBSE. It systematically reviewed 100 quality indicators, analyzed correlations between representative indicators, discussed several important issues in designing indicators, and suggested a few future research directions. One key purpose of that work is about indicator design and development, i.e., to inform the researchers and practitioners on what aspects to bear in mind when designing new indicators. In contrast, our work here is about indicators (and other evaluation methods) selection and use, i.e., to guide the researchers and practitioners on how to select/use existing indicators, or even specialize them, for evaluating solution sets in SBSE. Another major difference between the two works is that the work [75] considered the general situation that the DM's preferences are not available, whereas our work considers the situations based precisely upon various DM's preferences.

As such, these two works complement well with each other. If the SBSE researchers/practitioners want to understand correlations between different quality indicators, to know some important issues (e.g., scaling, normalization and effect of dominated/duplicate solutions) when performing quality evaluation for a practical problem, or even by themselves to design/develop new indicators, they can refer to the general survey paper in [75]. In contrast, if the SBSE researchers/practitioners want to select and use existing indicators in various optimization scenarios in SBSE, or to adapt existing indicators to fit explicit/implicit preferences from the DM in the given SBSE problem, this work can be well served.

Overall, in comparison with the existing works, this work presents several additional contributions. First, we conduct a systematic literature review on quality evaluation for Pareto-based optimization in SBSE. Second, we, from that review, present a variety of inappropriate/inadequate selection and inaccurate/misleading use of evaluation methods, and identify five important but overlooked issues. Third, from the perspective of the goal of multi-objective optimization, we discuss the reasons that quality indicators are needed, carry out an in-depth analysis of frequently-used quality indicators in the area, and explain the scope of their applicability. Finally, we provide a methodological guidance and procedure of selecting and using evaluation methods in various SBSE scenarios.

10 Conclusion

The nature of considering multiple (conflicting) objectives in many SBSE problems leads to a link between SE and multi-objective optimization. However, compared to the flourish of the use/design of multi-objective optimizers in SBSE, the evaluation of the optimizers' outcome remains relatively "casual". Existing SBSE researches often work by analogy, namely, following popular (or previously used) quality evaluation methods without considering whether they are truly suitable for their specific situation. In this paper, we have carried out a systematic and critical review of the quality evaluation in Pareto-based SBSE, covering 95 prominent studies published between 2009 and 2019 from 36 venues in seven repositories. We have found that in many studies the selection/use of evaluation methods is not appropriate and can even be misleading, based on which we summarize five critical issues, namely:

- Inadequacy of Solution Set Plotting.
- Inappropriate use of Descriptive Objective Evaluation.
- Confusion of the quality aspects covered by generic quality indicators.
- Oblivion of context information.
- Noncompliance of the DM's preferences.

Through revisiting the pros and cons of widely used quality indicators in SBSE, we have provided a methodological guidance and procedure of selecting, adjusting and using quality evaluation methods on the basis of the following availability/types of the DM's preferences:

- There are clear preferences between the objectives.
- There are vague/rough preferences between the objectives.
- There are preferences on some specific parts of the Pareto front.
- There are no preferences available.

We hope that our guidance would help to mitigate the evaluation issues in future SBSE work, and more importantly, would enable the quality evaluation of solution sets easier, clearer and more accurate for SBSE researchers and practitioners.

Acknowledgments

This work was supported by the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531), and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008). We are grateful to the editor and anonymous reviewers for their constructive comments on the early version of this paper. Miqing Li and Tao Chen contributed equally to this research.

References

[1] H. Abdeen, H. Sahraoui, and C. Debreceni, "Multi-objective optimization in rule-based design space exploration," in *Proc. IEEE/ACM Int. Conf. Autom. Softw. Eng.*, 2014, pp. 289–300.

[2] R. B. Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing advanced driver assistance systems using multi-objective search and neural networks," in *Proc. IEEE/ACM Int. Conf. Autom. Softw. Eng.*, 2016, pp. 63–74.

[3] R. B. Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng.*, 2018, pp. 1016–1026.

[4] S. Ali, P. Arcaini, D. Pradhan, S. A. Safdar, and T. Yue, "Quality indicators in search-based software engineering: An empirical evaluation," *ACM Trans. Softw. Eng. Methodol.*, vol. 29, no. 2, pp. 1–29, 2020.

[5] W. K. G. Assuncao, T. E. Colanzi, S. R. Vergilio, and A. Pozo, "A multi-objective optimization approach for the integration and test order problem," *Inf. Sci.*, vol. 267, no. 2, pp. 119–139, 2014.

[6] G. Bavota, F. Carnevale, A. D. Lucia, M. D. Penta, and R. Oliveto, "Putting the developer in-the-loop: An interactive GA for software re-modularization," in *Proc. Int. Symp. Search Based Softw. Eng.*, 2012, pp. 75–89.

[7] S. Boukharata, A. Ouni, M. Kessentini, S. Bouktif, and H. Wang, "Improving web service interfaces modularity using multi-objective optimization," *Autom. Softw. Eng.*, vol. 26, no. 2, pp. 275–312, 2019.

[8] I. Boussaïd, P. Siarry, and M. Ahmed-Nacer, "A survey on search-based model-driven engineering," *Autom. Softw. Eng.*, vol. 24, no. 2, pp. 233–294, 2017.

[9] M. Bowman, L. C. Briand, and Y. Labiche, "Solving the class responsibility assignment problem in object-oriented analysis with multi-objective genetic algorithms," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, pp. 817–837, Nov./Dec. 2010.

[10] B. Bozkurt, J. W. Fowler, E. S. Gel, B. Kim, M. Köksalan, and J. Wallenius, "Quantitative comparison of approximate solution sets for multicriteria optimization problems with weighted Tchebycheff preference function," *Operations Res.*, vol. 58, no. 3, pp. 650–659, 2010.

[11] S. A. Busari and E. Letier, "RADAR: A lightweight tool for requirements and architecture decision analysis," in *Proc. 39th Int. Conf. Softw. Eng.*, 2017, pp. 552–562.

[12] K.-Y. Cai and D. Card, "An analysis of research topics in software engineering–2006," *J. Syst. Softw.*, vol. 81, no. 6, pp. 1051–1058, 2008.

[13] R. Calinescu, M. Ceska, S. Gerasimou, M. Kwiatkowska, and N. Paoletti, "Designing robust software systems through parametric Markov Chain synthesis," in *Proc. IEEE Int. Conf. Softw. Architecture*, 2017, pp. 131–140.

[14] G. Canfora, A. D. Lucia, M. D. Penta, R. Oliveto, A. Panichella, and S. Panichella, "Defect prediction as a multiobjective optimization problem," *Softw. Testing Verification Rel.*, vol. 25, no. 4, pp. 426–459, 2015.

[15] G. Canfora, A. D. Lucia, M. D. Penta, R. Oliveto, A. Panichella, and S. Panichella, "Multi-objective cross-project defect prediction," in *Proc. IEEE 6th Int. Conf. Softw. Testing Verification Validation*, 2013, pp. 252–261.

[16] J. Chen, V. Nair, R. Krishna, and T. Menzies, "'Sampling' as a baseline optimizer for search-based software engineering," *IEEE Trans. Softw. Eng.*, vol. 45, no. 6, pp. 597–614, Jun. 2019.

[17] T. Chen and R. Bahsoon, "Self-adaptive trade-off decision making for autoscaling cloud-based services," *IEEE Trans. Services Comput.*, vol. 10, no. 4, pp. 618–632, Jul./Aug. 2017.

[18] T. Chen, R. Bahsoon, S. Wang, and X. Yao, "To adapt or not to adapt?: Technical debt and learning driven self-adaptation for managing runtime performance," in *Proc. ACM/SPEC Int. Conf. Perform. Eng.*, 2018, pp. 48–55.

[19] T. Chen, K. Li, R. Bahsoon, and X. Yao, "FEMOSAA: Feature guided and knee driven multi-objective optimization for self-adaptive software," *ACM Trans. Softw. Eng. Methodol.*, vol. 27, no. 2, 2018, Art. no. 5.

[20] T. Chen, M. Li, K. Li, and K. Deb, "Search-based software engineering for self-adaptive systems: Survey, disappointments, suggestions and opportunities," 2020, *arXiv:2001.08236*.

[21] T. Chen, M. Li, and X. Yao, "On the effects of seeding strategies: A case for search-based multi-objective service composition," in *Proc. Genetic Evol. Comput. Conf.*, 2018, pp. 1419–1426.

[22] T. Chen, M. Li, and X. Yao, "Standing on the shoulders of giants: Seeding search-based multi-objective optimization with prior knowledge for software service composition," *Inf. Softw. Technol.*, vol. 114, pp. 155–175, 2019. [Online]. Available: https://doi.org/10.1016/j.infsof.2019.05.013

[23] X. Chen, Y. Zhao, Q. Wang, and Z. Yuan, "MULTI: Multi-objective effort-aware just-in-time software defect prediction," *Inf. Softw. Technol.*, vol. 93, pp. 1–13, 2018.

[24] F. Chicano, F. Luna, A. J. Nebro, and E. Alba, "Using multi-objective metaheuristics to solve the software project scheduling problem," in *Proc. Conf. Genetic Evol. Comput.*, 2011, pp. 1915–1922.

[25] J. L. Cochrane and M. Zeleny, *Multiple Criteria Decision Making*. Columbia, SC, USA: Univ. South Carolina Press, 1973.

[26] C. A. C. Coello and M. R. Sierra, "A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm," in *Proc. Mex. Int. Conf. Artif. Intell.*, 2004, pp. 688–697.

[27] T. E. Colanzi, W. K. G. Assunção, P. R. Farah, S. R. Vergilio, and G. Guizzo, "A review of ten years of the symposium on search-based software engineering," in *Proc. Int. Symp. Search-Based Softw. Eng.*, 2019, pp. 42–57.

[28] V. G. Da Fonseca, C. M. Fonseca, and A. O. Hall, "Inferential performance assessment of stochastic optimisers and the attainment function," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.*, 2001, pp. 213–225.

[29] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[30] J. J. Durillo, V. Nae, and R. Prodan, "Multi-objective energy-efficient workflow scheduling using list-based heuristics," *Future Gener. Comput. Syst.*, vol. 36, no. 3, pp. 221–236, 2014.

[31] J. J. Durillo, Y. Y. Zhang, E. Alba, and A. J. Nebro, "A study of the multi-objective next release problem," in *Proc. Int. Symp. Search Based Softw. Eng.*, 2009, pp. 29–60.

[32] M. G. Epitropakis, S. Yoo, M. Harman, and E. K. Burke, "Empirical evaluation of Pareto efficient multi-objective regression test case prioritisation," in *Proc. Int. Symp. Softw. Testing Anal.*, 2015, pp. 234–245.

[33] K. R. Felizardo, E. Mendes, M. Kalinowski, É. F. de Souza, and N. L. Vijaykumar, "Using forward snowballing to update systematic reviews in software engineering," in *Proc. 10th ACM/IEEE Int. Symp. Empir. Softw. Eng. Meas.*, 2016, pp. 53:1–53:6.

[34] J. Ferrer, F. Chicano, and E. Alba, "Evolutionary algorithms for the multi-objective test data generation problem," *Softw. Pract. Experience*, vol. 42, no. 11, pp. 1331–1362, 2012.

[35] F. Ferrucci, M. Harman, J. Ren, and F. Sarro, "Not going to take this anymore: Multi-objective overtime planning for software engineering projects," in *Proc. 35th Int. Conf. Softw. Eng.*, 2013, pp. 462–471.

[36] A. Finkelstein, M. Harman, S. A. Mansouri, J. Ren, and Y. Zhang, "A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making," *Requirements Eng.*, vol. 14, no. 4, pp. 231–245, 2009.

[37] M. Fleck, J. Troya, M. Kessentini, M. Wimmer, and B. Alkhazi, "Model transformation modularization as a many-objective optimization problem," *IEEE Trans. Softw. Eng.*, vol. 43, no. 11, pp. 1009–1032, Nov. 2017.

[38] C. M. Fonseca and P. J. Fleming, "On the performance assessment and comparison of stochastic multiobjective optimizers," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 1996, pp. 584–593.

[39] S. Frey, F. Fittkau, and W. Hasselbring, "Search-based genetic optimization for deployment and reconfiguration of software in the cloud," in *Proc. 35th Int. Conf. Softw. Eng.*, 2013, pp. 512–521.

[40] W. Fu, T. Menzies, and X. Shen, "Tuning for software analytics: Is it really necessary?," *Inf. Softw. Technol.*, vol. 76, pp. 135–146, 2016.

[41] J. Fülöp, "Introduction to decision making methods," in *Proc. BDEI-3 Workshop*, 2005, pp. 1–15.

[42] M. Galster, D. Weyns, D. Tofan, B. Michalik, and P. Avgeriou, "Variability in software systems —A systematic literature review," *IEEE Trans. Softw. Eng.*, vol. 40, no. 3, pp. 282–306, Mar. 2014.

[43] S. Gerasimou, G. Tamburrelli, and R. Calinescu, "Search-based synthesis of probabilistic models for quality-of-service software engineering (T)," in *Proc. IEEE/ACM Int. Conf. Autom. Softw. Eng.*, 2016, pp. 319–330.

[44] S. Gueorguiev, M. Harman, and G. Antoniol, "Software project planning for robustness and completion time in the presence of uncertainty using multi-objective search based software engineering," in *Proc. Conf. Genetic Evol. Comput.*, 2009, pp. 1673–1680.

[45] J. Guo *et al.*, "SMTIBEA: A hybrid multi-objective optimization algorithm for configuring large constrained software product lines," *Softw. Syst. Model.*, vol. 18, no. 2, pp. 1447–1466, 2019.

[46] M. P. Hansen and A. Jaszkiewicz, "Evaluating the quality of approximations to the nondominated set," Technical University of Denmark, Denmark, Tech. Rep. IMM-REP-1998–7, 1998.

[47] 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," in *Proc. 18th Int. Softw. Product Line Conf.*, 2014, pp. 5–18.

[48] M. Harman, J. Krinke, J. Ren, and S. Yoo, "Search based data sensitivity analysis applied to requirement engineering," in *Proc. Conf. Genetic Evol. Comput.*, 2009, pp. 1681–1688.

[49] M. Harman, K. Lakhotia, J. Singer, D. R. White, and S. Yoo, "Cloud engineering is search based software engineering too," *J. Syst. Softw.*, vol. 86, no. 9, pp. 2225–2241, 2013.

[50] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: Trends, techniques and applications," *ACM Comput. Surv.*, vol. 45, no. 1, 2012, Art. no. 11.

[51] W. Heaven and E. Letier, "Simulating and optimising design decisions in quantitative goal models," in *Proc. IEEE 19th Int. Requirements Eng. Conf.*, 2011, pp. 79–88.

[52] C. Henard, M. Papadakis, M. Harman, and Y. L. Traon, "Combining multi-objective search and constraint solving for configuring large software product lines," in *Proc. IEEE/ACM Int. Conf. Softw. Eng.*, 2015, pp. 517–528.

[53] R. M. Hierons, M. Li, X. Liu, J. A. Parejo, S. Segura, and X. Yao, "Many-objective test suite generation for software product lines," *ACM Trans. Softw. Eng. Methodol.*, vol. 29, no. 1, 2020, Art. no. 2.

[54] R. M. Hierons, M. Li, X. Liu, S. Segura, and W. Zheng, "SIP: Optimal product selection from feature models using many-objective evolutionary optimization," *ACM Trans. Softw. Eng. Methodol.*, vol. 25, no. 2, pp. 1–39, 2016.

[55] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima, "How to specify a reference point in hypervolume calculation for fair performance comparison," *Evol. Comput.*, vol. 26, no. 3, pp. 411–440, 2018.

[56] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, "Modified distance calculation in generational distance and inverted generational distance," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.*, 2015, pp. 110–125.

[57] H. L. Jakubovski Filho, T. N. Ferreira, and S. R. Vergilio, "Preference based multi-objective algorithms applied to the variability testing of software product lines," *J. Syst. Softw.*, vol. 151, pp. 194–209, 2019.

[58] S. Kalboussi, S. Bechikh, M. Kessentini, and L. B. Said, "Preference-based many-objective evolutionary testing generates harder test cases for autonomous agents," in *Proc. Int. Symp. Search Based Softw. Eng.*, 2013, pp. 245–250.

[59] W. Kessentini, H. Sahraoui, and M. Wimmer, "Automated metamodel/model co-evolution: A search-based approach," *Inf. Softw. Technol.*, vol. 106, pp. 49–67, 2019.

[60] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering - A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009.

[61] J. D. Knowles and D. W. Corne, "On metrics for comparing nondominated sets," in *Proc. Congress Evol. Comput.*, 2002, vol. 1, pp. 711–716.

[62] J. D. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," ETH Zurich, Switzerland, Tech. Rep. 214, 2006.

[63] J. D. Knowles, "A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers," in *Proc. 5th Int. Conf. Intell. Syst. Des. Appl.*, 2005, pp. 552–557.

[64] S. Kumar, R. Bahsoon, T. Chen, K. Li, and R. Buyya, "Multi-tenant cloud service composition using evolutionary optimization," in *Proc. 24th IEEE Int. Conf. Parallel Distrib. Syst.*, 2018, pp. 972–979.

[65] A. C. Kumari and K. Srinivas, "Hyper-heuristic approach for multi-objective software module clustering," *J. Syst. Softw.*, vol. 117, pp. 384–401, 2016.

[66] N.-Z. Lee, P. Arcaini, S. Ali, and F. Ishikawa, "Stability analysis for safety of automotive multi-product lines: A search-based approach," in *Proc. Genetic Evol. Comput. Conf.*, 2019, pp. 1241–1249.

[67] E. Letier, D. Stefan, and E. T. Barr, "Uncertainty, risk, and information value in software requirements and architecture," in *Proc. Int. Conf. Softw. Eng.*, 2014, pp. 883–894.

[68] H. Li, G. Casale, and T. Ellahi, "SLA-driven planning and optimization of enterprise applications," in *Proc. Joint Wosp/sipew Int. Conf. Perform. Eng.*, 2010, pp. 117–128.

[69] L. Li, M. Harman, E. Letier, and Y. Zhang, "Robust next release problem: Handling uncertainty during optimization," in *Proc. Conf. Genetic Evol. Comput.*, 2014, pp. 1247–1254.

[70] M. Li, S. Yang, and X. Liu, "Diversity comparison of Pareto front approximations in many-objective optimization," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2568–2584, Dec. 2014.

[71] M. Li, S. Yang, and X. Liu, "Shift-based density estimation for Pareto-based algorithms in many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 348–365, Jun. 2014.

[72] M. Li, S. Yang, and X. Liu, "A performance comparison indicator for Pareto front approximations in many-objective optimization," in *Proc. Genetic Evol. Comput. Conf.*, 2015, pp. 703–710.

[73] M. Li, L. Zhen, and X. Yao, "How to read many-objective solution sets in parallel coordinates," *IEEE Comput. Intell. Mag.*, vol. 12, no. 4, pp. 88–97, Nov. 2017.

[74] M. Li, T. Chen, and X. Yao, "A critical review of "A practical guide to select quality indicators for assessing Pareto-based search algorithms in search-based software engineering": Essay on quality indicator selection for SBSE," in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng.: New Ideas Emerg. Technol. Results*, 2018, pp. 17–20.

[75] M. Li and X. Yao, "Quality evaluation of solution sets in multiobjective optimisation: A survey," *ACM Comput. Surv.*, vol. 52, no. 2, 2019, Art. no. 26.

[76] Y. Li, T. Yue, S. Ali, and L. Zhang, "Zen-ReqOptimizer: A search-based approach for requirements assignment optimization," *Empir. Softw. Eng.*, vol. 22, no. 1, pp. 175–234, 2017.

[77] X. Lian, L. Zhang, J. Jiang, and W. Goss, "An approach for optimized feature selection in large-scale software product lines," *J. Syst. Softw.*, vol. 137, pp. 636–651, 2018.

[78] A. Liefooghe and B. Derbel, "A correlation analysis of set quality indicator values in multiobjective optimization," in *Proc. Genetic Evol. Comput. Conf.*, 2016, pp. 581–588.

[79] M. López-Ibánez, L. Paquete, and T. Stützle, "Exploratory analysis of stochastic local search algorithms in biobjective optimization," in *Proc. Exp. Methods Anal. Optim. Algorithms*, 2010, pp. 209–222.

[80] U. Mansoor, M. Kessentini, B. R. Maxim, and K. Deb, "Multi-objective code-smells detection using good and bad design examples," *Softw. Quality J.*, vol. 25, no. 2, pp. 1–24, 2016.

[81] U. Mansoor, M. Kessentini, M. Wimmer, and K. Deb, "Multi-view refactoring of class and activity diagrams using a multi-objective evolutionary algorithm," *Softw. Qual. J.*, vol. 25, no. 2, pp. 1–29, 2015.

[82] K. Mao, M. Harman, and Y. Jia, "Sapienz: Multi-objective automated testing for Android applications," in *Proc. 25th Int. Symp. Softw. Testing Anal.*, 2016, pp. 94–105.

[83] T. Mariani, G. Guizzo, R. S. Vergilio, and T. R. A. Pozo, "Grammatical evolution for the multi-objective integration and test order problem," in *Proc. Genetic Evol. Comput. Conf.*, 2016, pp. 1069–1076.

[84] A. Martens, H. Koziolek, S. Becker, and R. Reussner, "Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms," in *Proc. Joint WOSP/SIPEW Int. Conf. Perform. Eng.*, 2010, pp. 105–116.

[85] P. McMinn, "Search-based software testing: Past, present and future," in *Proc. IEEE 4th Int. Conf. Softw. Testing Verification Validation Workshops*, 2011, pp. 153–163.

[86] S. Messaoudi, A. Panichella, D. Bianculli, L. Briand, and R. Sasnauskas, "A search-based approach for accurate identification of log message formats," in *Proc. 26th Conf. Program Comprehension*, 2018, pp. 167–177.

[87] H. Meunier, E. G. Talbi, and P. Reininger, "A multiobjective genetic algorithm for radio network optimization," in *Proc. Congress Evol. Comput.*, 2000, vol. 1, pp. 317–324.

[88] L. L. Minku and X. Yao, "Software effort estimation as a multiobjective learning problem," *ACM Trans. Softw. Eng. Methodol.*, vol. 22, no. 4, pp. 402–418, 2013.

[89] M. W. Mkaouer, M. Kessentini, S. Bechikh, M. Cinnéide, and K. Deb, "On the use of many quality attributes for software refactoring: A many-objective search-based software engineering approach," *Empir. Softw. Eng.*, vol. 21, no. 6, pp. 2503–2545, 2016.

[90] M. W. Mkaouer, M. Kessentini, S. Bechikh, and K. Deb, "Recommendation system for software refactoring using innovization and interactive dynamic optimization," in *Proc. ACM/IEEE Int. Conf. Autom. Softw. Eng.*, 2014, pp. 331–336.

[91] W. Mkaouer *et al.*, "Many-objective software remodularization using NSGA-III," *ACM Trans. Softw. Eng. Methodol.*, vol. 24, no. 3, pp. 17:1–17:45, 2015.

[92] C. Ni, X. Chen, F. Wu, Y. Shen, and Q. Gu, "An empirical study on Pareto based multi-objective feature selection for software defect prediction," *J. Syst. Softw.*, vol. 152, pp. 215–238, 2019.

[93] R. Olaechea, D. Rayside, J. Guo, and K. Czarnecki, "Comparison of exact and approximate multi-objective optimization for software product lines," in *Proc. 18th Int. Softw. Product Line Conf.*, 2014, pp. 92–101.

[94] A. Ouni, M. Kessentini, and H. Sahraoui, "Search-based refactoring using recorded code changes," in *Proc. Eur. Conf. Softw. Maintenance Reengineering*, 2013, pp. 221–230.

[95] A. Ouni, M. Kessentini, H. Sahraoui, and M. Boukadoum, "Maintainability defects detection and correction: A multi-objective approach," *Autom. Softw. Eng.*, vol. 20, no. 1, pp. 47–79, 2013.

[96] A. Ouni, M. Kessentini, H. Sahraoui, and M. S. Hamdi, "Search-based refactoring: Towards semantics preservation," in *Proc. IEEE Int. Conf. Softw. Maintenance*, 2012, pp. 347–356.

[97] A. Ouni, M. Kessentini, H. Sahraoui, and M. S. Hamdi, "The use of development history in software refactoring using a multi-objective evolutionary algorithm," in *Proc. 15th Annu. Conf. Genetic Evolutionay Comput. Conf.*, 2013, pp. 1461–1468.

[98] A. Ouni, M. Kessentini, H. Sahraoui, K. Inoue, and K. Deb, "Multi-criteria code refactoring using search-based software engineering: An industrial case study," *ACM Trans. Softw. Eng. Methodol.*, vol. 25, no. 3, 2016, Art. no. 23.

[99] A. Ouni, R. G. Kula, M. Kessentini, T. Ishio, D. M. German, and K. Inoue, "Search-based software library recommendation using multi-objective optimization," *Inf. Softw. Technol.*, vol. 83, pp. 55–75, 2017.

[100] A. Panichella, F. M. Kifetew, and P. Tonella, "Reformulating branch coverage as a many-objective optimization problem," in *Proc. IEEE Int. Conf. Softw. Testing, Verification Validation*, 2015, pp. 1–10.

[101] A. Panichella, F. M. Kifetew, and P. Tonella, "Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets," *IEEE Trans. Softw. Eng.*, vol. 44, no. 2, pp. 122–158, Feb. 2018.

[102] A. Panichella, F. M. Kifetew, and P. Tonella, "Incremental control dependency frontier exploration for many-criteria test case generation," in *Proc. Int. Symp. Search Based Softw. Eng.*, 2018, pp. 309–324.

[103] A. Panichella, R. Oliveto, M. D. Penta, and A. D. Lucia, "Improving multi-objective test case selection by injecting diversity in genetic algorithms," *IEEE Trans. Softw. Eng.*, vol. 41, no. 4, pp. 358–383, Apr. 2015.

[104] J. A. Parejo, A. B. Sánchez, S. Segura, A. Ruiz-Cortés, R. E. Lopez-Herrejon, and A. Egyed, "Multi-objective test case prioritization in highly configurable systems: A case study," *J. Syst. Softw.*, vol. 122, pp. 287–310, 2016.

[105] G. G. Pascual, R. E. Lopez-Herrejon, L. Fuentes, and A. Egyed, "Applying multiobjective evolutionary algorithms to dynamic software product lines for reconfiguring mobile applications," *J. Syst. Softw.*, vol. 103, no. C, pp. 392–411, 2015.

[106] D. Pradhan, S. Wang, S. Ali, T. Yue, and M. Liaaen, "REMAP: Using rule mining and multi-objective search for dynamic test case prioritization," in *Proc. IEEE Int. Conf. Softw. Testing, Verification Validation*, 2018, pp. 46–57.

[107] D. Pradhan, S. Wang, T. Yue, S. Ali, and M. Liaaen, "Search-based test case implantation for testing untested configurations," *Inf. Softw. Technol.*, vol. 111, pp. 22–36, 2019.

[108] K. Praditwong, M. Harman, and X. Yao, "Software module clustering as a multi-objective search problem," *IEEE Trans. Softw. Eng.*, vol. 37, no. 2, pp. 264–282, Mar./Apr. 2011.

[109] A. Ramírez, J. R. Romero, and S. Ventura, "A comparative study of many-objective evolutionary algorithms for the discovery of software architectures," *Empir. Softw. Eng.*, vol. 21, no. 6, pp. 2546–2600, 2016.

[110] A. Ramirez, J. R. Romero, and S. Ventura, "A survey of many-objective optimisation in search-based software engineering," *J. Syst. Softw.*, vol. 149, pp. 382–395, 2019.

[111] M. Ravber, M. Mernik, and M. Črepinšek, "The impact of quality indicators on the rating of multi-objective evolutionary algorithms," *Appl. Soft Comput.*, vol. 55, pp. 265–275, 2017.

[112] N. B. Ruparelia, "Software development lifecycle models," *ACM SIGSOFT Softw. Eng. Notes*, vol. 35, no. 3, pp. 8–13, 2010.

[113] T. Saber, D. Brevet, G. Botterweck, and A. Ventresque, "Is seeding a good strategy in multi-objective feature selection when feature models evolve?" *Inf. Softw. Technol.*, vol. 95, pp. 266–280, 2018.

[114] F. Sarro, F. Ferrucci, M. Harman, A. Manna, and J. Ren, "Adaptive multi-objective evolutionary algorithms for overtime planning in software projects," *IEEE Trans. Softw. Eng.*, vol. 43, no. 10, pp. 898–917, Oct. 2017.

[115] F. Sarro, A. Petrozziello, and M. Harman, "Multi-objective software effort estimation," in *Proc. IEEE/ACM 38th Int. Conf. Softw. Eng.*, 2016, vol. 21, pp. 619–630.

[116] S. Sayın, "Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming," *Math. Program.*, vol. 87, no. 3, pp. 543–560, 2000.

[117] A. S. Sayyad and H. Ammar, "Pareto-optimal search-based software engineering (POSBSE): A literature survey," in *Proc. 2nd Int. Workshop Realizing Artif. Intell. Synergies Softw. Eng.*, 2013, pp. 21–27.

[118] A. S. Sayyad, J. Ingram, T. Menzies, and H. Ammar, "Optimum feature selection in software product lines: Let your model and values guide your search," in *Proc. Int. Workshop Combining Modelling Search-Based Softw. Eng.*, 2013, pp. 22–27.

[119] A. S. Sayyad, J. Ingram, T. Menzies, and H. Ammar, "Scalable product line configuration: A straw to break the camel's back," in *Proc. IEEE/ACM Int. Conf. Autom. Softw. Eng.*, 2013, pp. 465–474.

[120] A. S. Sayyad, T. Menzies, and H. Ammar, "On the value of user preferences in search-based software engineering: A case study in software product lines," in *Proc. 35th Int. Conf. Softw. Eng.*, 2013, pp. 492–501.

[121] J. R. Schott, "Fault tolerant design using single and multicriteria genetic algorithm optimization," Master's thesis, Dept. Aeronautics Astronautics, Massachusetts Inst. Technol., Cambridge, MA, 1995.

[122] O. Schutze, X. Esquivel, A. Lara, and C. C. A. Coello, "Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 4, pp. 504–522, Aug. 2012.

[123] S. Segura, R. E. Lopez-Herrejon, and A. Egyed, "Multi-objective test case prioritization in highly configurable systems," *J. Syst. Softw.*, vol. 122, no. C, pp. 287–310, 2016.

[124] A. Shahbazi and J. Miller, "Black-box string test case generation through a multi-objective optimization," *IEEE Trans. Softw. Eng.*, vol. 42, no. 4, pp. 361–378, Apr. 2016.

[125] X. Shen, L. Minku, R. Bahsoon, and X. Yao, "Dynamic software project scheduling through a proactive-rescheduling method," *IEEE Trans. Softw. Eng.*, vol. 42, no. 7, pp. 658–686, Jul. 2016.

[126] X. Shen, L. Minku, N. Marturi, Y. Guo, and Y. Han, "A Q-learning-based memetic algorithm for multi-objective dynamic software project scheduling," *Inf. Sci.*, vol. 428, pp. 1–29, 2018.

[127] S. Y. Shin, S. Nejati, M. Sabetzadeh, L. C. Briand, and F. Zimmer, "Test case prioritization for acceptance testing of cyber physical systems: A multi-objective search-based approach," in *Proc. 27th ACM SIGSOFT Int. Symp. Softw. Testing Anal.*, 2018, pp. 49–60.

[128] C. L. Simons and I. C. Parmee, "Elegant object-oriented software design via interactive, evolutionary computation," *IEEE Trans. Syst. Man Cybern. C*, vol. 42, no. 6, pp. 1797–1805, Nov. 2012.

[129] C. L. Simons, I. C. Parmee, and R. Gwynllyw, "Interactive, evolutionary search in upstream object-oriented class design," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, pp. 798–816, Nov./Dec. 2010.

[130] D. Sobhy, L. L. Minku, R. Bahsoon, T. Chen, and R. Kazman, "Run-time evaluation of architectures: A case study of diversification in IoT," *J. Syst. Softw.*, vol. 159, 2020, Art. no. 110428. [Online]. Available: https://doi.org/10.1016/j.jss.2019.110428

[131] B. Tan, H. Ma, Y. Mei, and M. Zhang, "Evolutionary multi-objective optimization for web service location allocation problem," *IEEE Trans. Services Comput.*, to be published.

[132] T. H. Tan, Y. Xue, M. Chen, J. Sun, Y. Liu, and J. S. Dong, "Optimizing selection of competing features via feedback-directed evolutionary algorithms," in *Proc. Int. Symp. Softw. Testing Anal.*, 2015, pp. 246–256.

[133] D. A. Van Veldhuizen and G. B. Lamont, "Evolutionary computation and convergence to a Pareto front," in *Proc. Late Breaking Papers Genetic Program. Conf.*, 1998, pp. 221–228.

[134] H. Wada, J. Suzuki, Y. Yamano, and K. Oba, "E3: A multiobjective optimization framework for SLA-aware service composition," *IEEE Trans. Services Comput.*, vol. 5, no. 3, pp. 358–372, 2012.

[135] F. Wagner, A. Klein, B. Klopper, F. Ishikawa, and S. Honiden, "Multi-objective service composition with time- and input-dependent QoS," in *Proc. IEEE Int. Conf. Web Services*, 2012, pp. 234–241.

[136] T. Wagner and H. Trautmann, "Integration of preferences in hypervolume-based multiobjective evolutionary algorithms by means of desirability functions," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 688–701, Oct. 2010.

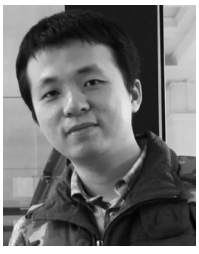
[137] S. Wang, S. Ali, and A. Gotlieb, "Cost-effective test suite minimization in product lines using search techniques," *J. Syst. Softw.*, vol. 103, no. C, pp. 370–391, 2015.

[138] S. Wang, S. Ali, T. Yue, Y. Li, and M. Liaaen, "A practical guide to select quality indicators for assessing pareto-based search algorithms in search-based software engineering," in *Proc. IEEE/ACM 38th Int. Conf. Softw. Eng.*, 2016, pp. 631–642.

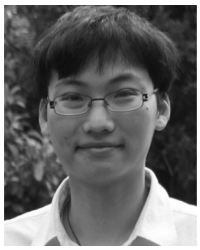
[139] Z. Wang, K. Tang, and X. Yao, "Multi-objective approaches to optimal testing resource allocation in modular software systems," *IEEE Trans. Rel.*, vol. 59, no. 3, pp. 563–575, Sep. 2010.

[140] Z. Wen and V. Tzerpos, "An effectiveness measure for software clustering algorithms," in *Proc. 12th IEEE Int. Workshop Program Comprehension*, 2004, pp. 194–203.

[141] D. R. White, A. Arcuri, and J. A. Clark, "Evolutionary improvement of programs," *IEEE Trans. Evol. Comput.*, vol. 15, no. 4, pp. 515–538, Aug. 2011.
[142] F. Wu, W. Weimer, M. Harman, Y. Jia, and J. Krinke, "Deep parameter optimisation," in *Proc. Annu. Conf. Genetic Evol. Comput.*, 2015, pp. 1375–1382.
[143] Y. Xiang, X. Yang, Y. Zhou, and H. Huang, "Enhancing decomposition-based algorithms by estimation of distribution for constrained optimal software product selection," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 245–259, Apr. 2020.
[144] Y. Xiang, Y. Zhou, Z. Zheng, and M. Li, "Configuring software product lines by combining many-objective optimization and SAT solvers," *ACM Trans. Softw. Eng. Methodol.*, vol. 26, no. 4, 2018, Art. no. 14.
[145] S. Yoo and M. Harman, "Using hybrid algorithm for Pareto efficient multi-objective test suite minimisation," *J. Syst. Softw.*, vol. 83, no. 4, pp. 689–701, 2010.
[146] G. Zhang, Z. Su, M. Li, F. Yue, J. Jiang, and X. Yao, "Constraint handling in NSGA-II for solving optimal testing resource allocation problems," *IEEE Trans. Rel.*, vol. 66, no. 4, pp. 1193–1212, Dec. 2017.
[147] Y. Zhang, M. Harman, and S. L. Lim, "Empirical evaluation of search based requirements interaction management," *Inf. Softw. Technol.*, vol. 55, no. 1, pp. 126–152, 2013.
[148] Y. Zhang, M. Harman, and S. A. Mansouri, "The multi-objective next release problem," in *Proc. Genetic Evol. Comput. Conf.*, 2007, pp. 1129–1137.
[149] Y. Zhang, M. Harman, G. Ochoa, G. Ruhe, and S. Brinkkemper, "An empirical study of meta-and hyper-heuristic search for multi-objective release planning," *ACM Trans. Softw. Eng. Methodol.*, vol. 27, no. 1, 2018, Art. no. 3.
[150] W. Zheng, R. M. Hierons, M. Li, X. H. Liu, and V. Vinciotti, "Multi-objective optimisation for regression testing," *Inf. Sci.*, vol. 334, pp. 1–16, 2016.
[151] 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," in *Proc. IEEE Congress Evol. Comput.*, 2006, pp. 892–899.
[152] E. Zitzler, D. Brockhoff, and L. Thiele, "The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.*, 2007, pp. 862–876.
[153] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms - A comparative case study," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 1998, pp. 292–301.
[154] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, Apr. 2003.
[155] W. Zou, D. Lo, Z. Chen, X. Xia, Y. Feng, and B. Xu, "How practitioners perceive automated bug report management techniques," *IEEE Trans. Softw. Eng.*, vol. 46, no. 8, pp. 836–862, Aug. 2020.



Miqing Li is currently a lecturer at the School of Computer Science, University of Birmingham, U.K. His research is principally on multi-objective optimisation, where he focuses on developing population-based randomised algorithms (mainly evolutionary algorithms) for both general challenging problems (e.g., many-objective optimisation, constrained optimisation, robust optimisation, expensive optimisation) and specific challenging problems (e.g., those in software engineering, system engineering, product disassembly, post-disaster response, neural architecture search, reinforcement learning for games). He has published more than 60 research papers in scientific journals and international conferences. Some of his papers, since published, have been amongst the most cited papers in corresponding journals such as the *IEEE Transactions on Evolutionary Computation*, *Artificial Intelligence*, the *ACM Transactions on Software Engineering and Methodology*, the *IEEE Transactions on Parallel and Distribution Systems*, and the *ACM Computing Surveys*. His work has received the Best Student Paper Award or Best Paper Award nomination in EC mainstream conferences, CEC, GECCO, and SEAL. He is the founding chair of the IEEE CIS' Task Force on Many-Objective Optimisation.



Tao Chen (Member, IEEE) received the PhD degree from the School of Computer Science, University of Birmingham, U.K., in 2016. He is currently a lecturer (assistant professor) in computer science at the Department of Computer Science, Loughborough University, U.K. He has broad research interests on software engineering, including but not limited to performance engineering, self-adaptive software systems, search-based software engineering, data-driven software engineering, and computational intelligence. As the lead author, his work has been published in internationally renowned journals, such as the *IEEE Transactions on Software Engineering*, the *ACM Transactions on Software Engineering and Methodology*, the *IEEE Transactions on Services Computing*, and the *Proceedings of the IEEE*; and top-tier conferences, e.g., ICSE, ASE, and GECCO. Among other roles, He regularly serves as a PC member for various conferences in his fields and is an associate editor for the *Services Transactions on Internet of Things*.

Xin Yao (Fellow, IEEE) received the BSc degree from the University of Science and Technology of China (USTC), Hefei, China, in 1982, the MSc degree from the North China Institute of Computing Technologies, Beijing, China, in 1985, and the PhD degree from the University of Science and Technology of China, China, in 1990. He is a chair professor of computer science at the Southern University of Science and Technology (SUSTech), Shenzhen, China, and a part-time professor of computer science at the University of Birmingham, U.K. His current research interests include evolutionary computation, machine learning, and their real world applications, especially to software engineering. He started his work on search-based software engineering (SBSE) more than a decade ago, including "Coevolving Programs and Unit Tests from Their Specification" at ASE'07 and "Software Module Clustering as a Multi-Objective Search Problem" in March 2011's the *IEEE Transactions on Software Engineering*. His latest work on SBSE includes "Software Effort Interval Prediction via Bayesian Inference and Synthetic Bootstrap Resampling" in January 2019's the *ACM Transactions on Software Engineering and Methodology* and "Synergizing Domain Expertise With Self-Awareness in Software Systems: A Patternized Architecture Guideline" in July 2020's the *Proceedings of the IEEE*. He was a recipient of the Royal Society Wolfson Research Merit Award, in 2012, the IEEE Computational Intelligence Society (CIS) Evolutionary Computation Pioneer Award, in 2013 and the IEEE Frank Rosenblatt Award, in 2020. His work won the 2001 IEEE Donald G. Fink Prize Paper Award, the 2010, 2016, and 2017 the *IEEE Transactions on Evolutionary Computation* outstanding paper awards, the 2011 the *IEEE Transactions on Neural Networks* Outstanding Paper Award, and many other best paper awards at conferences. He was the President of IEEE CIS from 2014 to 2015 and the editor-in-chief of the *IEEE Transactions on Evolutionary Computation* from 2003 to 2008. He was a distinguished lecturer of IEEE CIS.