



On the “fairness” of search-based software engineering: Investigating the capability of a scalarizing-based function to control the software metrics’ influence on optimization process


Italo Yeltsin  [State University of Ceará | br.yeltsin@gmail.com]

Allysson Alex Araújo  [Federal University of Cariri | allysson.araujo@ufca.edu.br]

Altino Dantas  [Federal University of Goiás | tinodantas@gmail.com]

Pamella Soares  [State University of Ceará | pamella.soares@aluno.uece.br]

Raphael Saraiva  [State University of Ceará | raphael.saraiva@aluno.uece.br]

Jerffeson Souza  [State University of Ceará | jerffeson.souza@uece.br]

Abstract Search-Based Software Engineering (SBSE) aims to transform Software Engineering (SE) problems into search problems by defining a fitness function that guides the search for an optimal or sub-optimal solution. However, designing a fitness function that provides equitable relevance (or weight) to every SE metric associated with an SBSE problem is an assumption that may be challenging. This issue derives from the several properties related to SE metric value domains that can induce the search process to privilege specific metrics over others, misleading to suboptimal outcomes. To deal with this problem, this work proposes a mathematical model based on the scalarization function concept to better control each metric’s relevance in the search process. Our empirical study comprises two computational experiments. The first experiment aimed to evaluate the proposed scalarizing-based approach’s control capability over the SE metrics in a scenario where all metrics should have the same relevance, while the second experiment covers the scenario where metrics do not necessarily should have the same relevance. The results demonstrate the importance of properly considering the impact, nature, and value range of SE metrics in the search process and the effectiveness of the proposed model in controlling SE metric relevance in different scenarios. This research makes three significant contributions. Firstly, we empirically highlight the importance of properly considering the relevance of individual SE metrics in the search process. Secondly, we propose a generic mathematical model based on scalarizing functions to cope with the normalization process and can be applied to a wide range of SBSE problems. Finally, we show that the our scalarizing approach is capable of guiding search-based process not only in the scenario where all metrics relevance must be equal, but also in the variation of the relevance alongside the optimization process, which is quite important for the design of fitness functions in SBSE.

Keywords: *Scalarizing Function, Software Engineering Metrics, Search-Based Software Engineering.*

1 Introduction

Search-Based Software Engineering (SBSE) consists of the application of search-based optimization to Software Engineering (SE) problems (Harman, 2007b; Ouni, 2020). According to Harman et al. (2012), SBSE has two key ingredients: 1) the choice of the representation of the problem and 2) the definition of the fitness function (Harman et al., 2012). The candidate solution representation to the problem defines the search space in which the optimization takes place. Thus, a fitness function is required to differentiate between solutions and measure progress to guide the search-based process. In doing so, many problems in SE also have SE metrics associated with them that naturally form good initial candidates to comprise the fitness function (Harman et al., 2009). Furthermore, the correspondence between metrics and fitness functions means that many metrics can be used as the guiding force behind the search for optimal or near-optimal solutions to several problems (Harman and Clark, 2004).

As widely recognized, SE metrics play an important role in the good software development process (Fenton and Neil, 2000; Singh et al., 2011). In particular, SE measurement is used to assess situations, track progress, evaluate effectiveness, understand whether the requirements are consistent and complete, whether the design is high quality, and more (Fen-

ton and Bieman, 2014). Hence, many intermediate or final software products are developed and measured by several metrics, such as cohesion and coupling for software architecture, cyclomatic complexity for source code, percentage of faults detected and code coverage for software testing, and risk and value for stakeholders for release planning.

When a SE metric becomes part of a fitness function in the context of SBSE, it is no longer seen as a passive quantitative element but, indeed, as a valuable guideline for improvement and change. However, it is worth noting that using SE metrics in SBSE is not a one-size-fits-all solution. Instead, it is essential to carefully select and tailor the metrics used to construct the fitness function to the specific problem at hand. In this sense, an acknowledged challenge in analyzing the SBSE results is the assumption of an adequately weighted aggregate fitness function. That is, a function composed of more than one metric establishes that the optimized objectives are weighted and combined to form a single objective (Chisalita-Cretu, 2014; Augusto et al., 2012). In other words, a fitness function that allows the metrics to have the same relevance and impact in the search process, thus avoiding that one specific metric has undue privilege compared to others.

Furthermore, to allow a fair search-based process, one can highlight the need to conduct an in-depth analysis of the na-

ture and range of value each metric can assume. This awareness during the fitness function design is of particular importance to avoid discrepancies regarding the evaluation of each metric and, consequently, to not fall on misleading results (Chen and Li, 2022). A conventional strategy to deal with this challenge is to normalize the value of each metric using the maximum and minimum (or both) at equal intervals, which can make it feasible that each metric within the aggregated function has the same weight in the search process (Lightner and Director, 1981). However, the maximum or minimum values are only sometimes prior known. Moreover, even if they could be defined, optimization problems are often subject to several constraints, making it challenging to guarantee that these known extreme values are always achievable within the scope of viable solutions. In other words, in SBSE, metrics are often bounded by problem-specific constraints or influenced by the population of solutions in each generation (Simons et al., 2015; Chen and Li, 2022). For example, in bug prioritization, metrics like "importance" and "severity" may not have predefined limits and vary based on the project's history or stakeholder feedback (Dreyton et al., 2015). Similarly, in the software release planning, the "value" and "cost" of features are relative to customer preferences and implementation constraints, which change dynamically based on the available resources and customer requirements at each stage of development (Dantas et al., 2015). These practical scenarios illustrate that metrics can be relative and context-dependent, making it challenging to set static maximum or minimum values prior to the search.

As one can expect, SE problems usually face the need to optimize multiple and conflicting objectives simultaneously (Harman et al., 2009). Typically, as it is not possible to find a solution where all the objectives can reach their individual optimum, we are interested in identifying a set of mathematically equally good solutions, the so-called Pareto optimal solutions (Ruiz et al., 2015). For this reason, multi-objective algorithms that use Pareto-ranking as a selection method, such as NSGA-II (Deb and Kalyanmoy, 2001), have also been widely approached by the SBSE community (Zhang, 2012; Colanzi et al., 2020; Rodriguez and Carver, 2020; Saidani et al., 2020). However, as the number of objectives increases, more of the population becomes non-dominated, and the selective pressure driving the population towards the Pareto set falls rapidly (Purshouse and Fleming, 2003). This means the Pareto dominance relation can not generate enough selection pressure toward the Pareto front. As a result, dominant-based algorithms often improve only the diversity of solutions without improving their convergence toward the Pareto front (Ishibuchi et al., 2009b). In this scenario, the simplest of these non-Pareto methods is also to use the conventional weighted min-max method (Hughes, 2005).

A promising approach for improving the searchability of dominance-based algorithms is using scalarizing fitness functions (Ishibuchi et al., 2009a). Via a scalarizing-based approach, the problem is transformed into a single objective optimization model involving possibly some parameters or additional constraints (Ishibuchi and Nojima, 2007). In most scalarizing functions, the decision maker's preference information is considered to find the most satisfactory solution among the conflicting goal (Miettinen and Mäkelä, 2002a;

Saraiva et al., 2017). Indeed, it was already clarified that better results were obtained for multi-objective problems by multiple runs of single-objective optimizers with scalarizing fitness functions (Hughes, 2005; Ishibuchi et al., 2009b; Kasimbeyli et al., 2019). However, differently from the purpose of these works, we noticed the opportunity of approaching the concept of scalarizing fitness function to normalize and offer a proper relation between the SE metrics during the search process. The significance of this question relies on the fact that we may bring novel light into the fitness function design, which is a key ingredient for SBSE.

In line with the previous motivation, our aim is to investigate the use of a scalarizing fitness function to properly manage the relevance of each SE metric in a search-based process. To this end, we propose a scalarizing-based generic model for the normalization process. Then, we designed an empirical study composed of two computational experiments to evaluate the performance of our proposed approach when compared to 1) aggregate weighted fitness functions in a scenario where all metrics must have the same weight and 2) in the scenarios where not all metrics necessarily should have the same weight. We investigated the Bugs Prioritization (Dreyton et al., 2015) and Software Release Planning (Dantas et al., 2015) problems for the first experiment. Regarding the second experiment, we examined the Multi-Objective Next Release Problem (Zhang et al., 2007) using an adaptation of the proposed mathematical model to iteratively build an approximation of the Pareto Front. In this regard, we intend with this empirical study to demonstrate the feasibility and significance of our approach. Therefore, we do not aim to make broad and sweeping claims about the entire field of SBSE or asserting its position as a superior multi-objective algorithm, but rather to present a specific approach and demonstrate its effectiveness in practical scenarios.

In summary, our research makes three key contributions that enhance the understanding of SE metrics and offer valuable findings to support SBSE practices:

1. Based on evidence-based results, we emphasize the importance of comprehensively understanding the impact, nature, and range of values that a SE metric can assume in the optimization process. This awareness enables practitioners to make informed decisions when selecting and evaluating SE metrics during the fitness function definition;
2. We propose a generic mathematical model that employs a scalarizing function concept to address the normalization process that can be introduced to SBSE problems, even when the metrics do not belong to the ratio scale. While scalarizing functions are not uncommon, the application of such a scalarizing-based model to SBSE problems is a novel aspect of our approach since it dynamically acquires the best and worst values for SE metrics during the search process, which is distinct from traditional scalarization methods that often rely on predefined and static values. This approach allows practitioners to compare and rank different metrics with varying scales and magnitudes;
3. We empirically demonstrate that our proposal can promote an equitable search-based process and achieve

competitive performance not only when each SE metric must carry equal weight but also when varying the relevance. Our proposal is also adaptable to multi-objective approaches, where decision makers aggregate multiple SE metrics into a single objective. Specifically, we integrate a multi-objective implementation with simulated annealing (SA) to optimize multiple conflicting metrics simultaneously, providing a practical and flexible framework for tackling complex SE problems.

This paper is organized as follows. In Section 2, we overview prior work, including the theoretical background and related work. In Section 3, we explain our scalarizing-based approach to SBSE. Then, we report our empirical study in Section 4 while we discuss the threats to validity in Section 5. Finally, we present the final remarks in Section 6.

2 Background and Related Work

In this section, we overview the fundamental concepts and related work. Firstly, we discuss the role of the fitness function in SBSE, including the different optimization approaches that have been addressed. Then, we clarify the SE metrics as a key ingredient in SBSE. Lastly, we address how and why it may be helpful to approach scalarizing functions in SBSE.

2.1 On the role of fitness functions in SBSE

Search-Based Software Engineering (SBSE) refers to the research field where search techniques are approached to solve problems in Software Engineering (SE) (Harman and Jones, 2001). This class of problems is characterized by a search space for candidate solutions from which optimal or approximate solutions are desired and guided by a fitness function. Indeed, SBSE has been explored to solve problems in several stages of the software development process, such as software release planning (Zhang et al., 2014), design tools and techniques (Colanzi et al., 2014), software testing and debugging (Harman, 2007a), and software project management (Ferrucci et al., 2014).

Indeed, SBSE problems involve two or more objectives, which are more or less conflicting. Hence, the SBSE community has approached two alternative strategies in this engineering decision-making: a weighted search (*i.e.*, utility search) or a Pareto search. The former directly searches for a single solution that maximizes the aggregated scalar fitness of the objectives (*e.g.*, by weighted sum), based on a set of weights (also called a weight vector) that reflects relative importance between the objectives (Chen and Li, 2022). In general terms, a hypothetical mono-objective problem is composed of a fitness function as follows:

$$\begin{aligned} &\min f(X) \\ &\text{subject to:} \\ &g_i(X) \quad \text{for } i = 1, \dots, m, \end{aligned}$$

where $f(X)$ is a fitness function responsible for guiding the search process towards an optimal solution; X is a solution to

the problem, and $g_i(X)$ with i ranging from 1 to m represents the constraints of the problem. In turn, an aggregate function can be represented as follows:

$$\text{Fitness}(X) = c_1 \times m_1(X) + c_2 \times m_2(X) + \dots + c_n \times m_n(X),$$

where $m_i(X)$ and c_i , such that $1 \leq i \leq n$, respectively, represents a SE metric in the set of n SE metrics associated with a given optimization problem. The coefficients c_i weights the relevance of the associated metric during the optimization process. This type of function varies according to the relevance of each metric or the use of different aggregation operators, such as product and division. This strategy works when a given coefficient c_i can precisely determine how relevant a given metric is for a fitness function. This issue does not happen due to the diversity of value domains in which the metrics belong (Harman, 2007b). For instance, given metrics $M_x \in [0, 1.0]$ and metric $M_y \in [0, 100]$, even assigning to c_x a value ten times greater than c_y , M_y would have more relevance in the search process because M_y can assume values that can be more than 100 times greater than M_x .

The use of normalization could tackle such a problem. By previously knowing the extreme values for M_x and M_y , one could normalize each metric in the $[0, 1]$ interval. For some metrics, extreme values can even be deduced by analyzing their formula, which makes normalization applicable. That is, SBSE problems are known for their restrictions that can make solutions that reach extreme metric values (max or min) unfeasible. To exemplify this scenario, Figure 1 shows an example of two solutions within a feasible search space S according to two hypothetical metrics, $M_1(X)$ and $M_2(X)$.

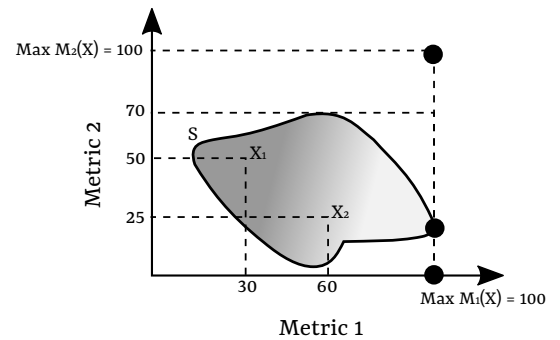


Figure 1. Comparison between two solutions within the search space.

Regarding the example in Figure 1, when comparing X_1 and X_2 using the evaluation function $f(X) = \frac{M_1(X)}{100} + \frac{M_2(X)}{100}$ that normalizes the values over the maximum reachable values by each metric (as illustrated in each axis), $f(X)$ would indicate that X_2 is superior in comparison to X_1 given that $f(X_2) = 0.85$ is greater than $f(X_1) = 0.80$. This information leads to a wrong interpretation of the outcomes because, according to viable solution space, 100 is not the greatest value for $M_2(X)$. The fairest way to normalize would be using the maximum value according to the problem's constraints for $M_2(X)$, which is 70. This way a more suitable function $f_2(X) = \frac{M_1(X)}{100} + \frac{M_2(X)}{70}$, where $f_2(X_1) = 1.014$ and $f_2(X_2) = 0.95$, indicating that X_1 is actually a more valuable solution.

On the other hand, multi-objective optimization problems consist of a set of solutions that satisfy the given constraints. Moreover, this set must be composed of solutions that offer the best trade-offs among the objectives to be optimized regarding the problem being tackled. We exemplify a multi-objective model below:

$$\text{minimize } f_i(X) \quad 1 \leq i \leq n \quad (1)$$

subject to:

$$g_k(X) \quad 1 \leq k \leq c, \quad (2)$$

where $f_i(X)$ is a fitness function of the problem that evaluates a given solution X belonging to a set of solutions. The expression 1 represents the objectives to be minimized, whose amount is n . In turn, if one needs to maximize an objective, it can be achieved by multiplying the function that describes the objective by -1. In addition, Equation 2 represents the constraints of the problem evaluated by each function $g_k(X)$.

It is worth noticing that most multi-objective algorithms approach the dominance criteria to find the front of solutions (Deb and Kalyanmoy, 2001). As depicted in the Equation 3, it is possible to define that a X_1 solution dominates a X_2 solution if, and only if, X_1 is better or equal in all objectives and strictly better in at least one objective, that is:

$$\forall i | 1 \leq i \leq n : f_i(X_1) \leq f_i(X_2) \quad (3)$$

$$\exists k | 1 \leq k \leq n : f_k(X_1) < f_k(X_2) \quad (4)$$

Multi-objective algorithms aim to identify a set of non-dominated solutions, also called a front of solutions. A set of solutions is deemed a Pareto Front when it consists of all non-dominated solutions within a feasible region of the search space. Chen and Li (2022) clarify that when no preferences or weights are given for the objectives, the Pareto search is the most appropriate option. This is because it can reveal the entire Pareto front of solutions, a diverse set that can be examined without prior knowledge of the preferences. However, a weighted search may be appropriate when clear preferences can be elicited, articulated, or assumed. In summary, multi-objective algorithms are used to identify non-dominated solutions or fronts of solutions, such as the Pareto front. The choice between a Pareto search and a weighted search depends on whether or not there are clear preferences for the objectives. The Pareto search could be more suitable for a scenario where the decision maker may decide among different non-dominated solutions *a posteriori*, while a weighted search is preferable when clear preferences can be articulated *a priori*.

2.2 Software Engineering metrics as a key ingredient in SBSE

Software Engineering (SE) metrics continuously apply measurement techniques to improve the software development process and its products (Samli et al., 2020). Through suitable metrics, software engineers may assess the development

progress and the quality of the artifacts in order to answer several questions, such as the program size, estimating cost and duration of some work, the estimated cost of fixing a bug, which method would be the most appropriate to develop a given project, etc (Malhotra, 2015).

We may categorize two significant perspectives on software development. The first relates to the *process* and refers to how the product is developed. The second perspective is concerned with the *product*, that is, the output of the process (which can be documents, source code, and other artifacts produced during the software life cycle). In some cases, one process may use a product from another process. For example, the software design process may produce an architectural document as an output that the software engineers can further approach during the implementation process. Both process and product perspectives are entangled by several metrics composed of internal and external attributes.

In summary, internal attributes are associated with the internal structure of the software, such as size, coupling, and complexity. In contrast, external attributes are associated with external factors, such as understandability, testability, and maintainability. The difference between metrics and attributes is that metrics are used to measure attributes. As exemplified in Figure 2, the maintainability (external attribute) is associated with the following internal attributes: depth of inheritance, cyclomatic complexity, program size, number of error messages, and user manual size.

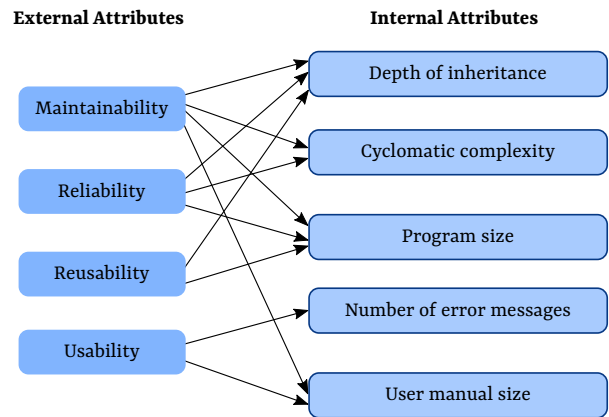


Figure 2. Relationship between internal and external attributes in SE metrics (Sommerville, 2015).

The SE metrics (both for process and product) have also been widely addressed by the SBSE community, since the own metrics are used as part of the fitness function that will guide the search process. Figure 3 exemplifies two SBSE problems and their respective metrics. On the left side, the process perspective is represented by the Next Release Problem (NRP) (Bagnall et al., 2001), which aims to select the set of requirements that must be implemented in the next release of the software and seeks to maximize the satisfaction of a company's most important customers (external attribute). Therefore, a release must be subject to the sum of the costs of implementing the requirements (internal attribute) not exceeding the budget (internal attribute). On the right side, the product perspective is represented by the problem of refactoring programs to conform more closely to a given design quality model (O'Keefe and Cinnéide, 2008). To this end,

the authors propose to select a sequence of refactorings that provide better code regarding flexibility, reusability, and understandability (external attributes). These measurements belong to a set of metrics (Bansiya and Davis, 2002), which relates through weighting the metrics of internal attributes to measurements of external attributes.

Furthermore, each SE metric could be measured in scales that follow measurement standards to determine characteristics related to metric values, such as the unit of measurement (Wang, 2003). The scales are usually classified into five types: nominal, interval, ordinal, ratio, and absolute. *Nominal scales* are characterized by an empirical relationship system that consists only of different classes, without the possibility of indicating order (Fenton, 1991). Unlike the nominal scale, the *ordinal scale* features sortable classes, considering a particular attribute. Operations such as addition and subtraction are not supported since these classes are only for sorting. For example, an estimated time cost for a feature to be implemented could be calculated as follows:

$$Cost(x) = \begin{cases} high & \text{if } x \leq 8 \\ average & \text{if } 2 \leq x < 8 \\ down & \text{if } x < 2. \end{cases} \quad (5)$$

where x is the cost in weeks to implement the feature. This ordinal scale can be used when the real values of a given attribute are not precisely known. For example, we can clarify this issue in the case of cost estimation in development companies. The implementation deadline for a particular feature could range according to the developer's experience.

The *interval scale* has the ordering property similar to the ordinal scale. In this type of scale, there is the possibility of computing the distance between intervals, thus allowing operations such as addition and subtraction. However, the interval scale does not admit the property of the ratio between its values. For example, suppose the existence of a SE metric $M(x)$, whose domain of its image belongs to the interval $[-\infty, +\infty]$. Considering that $M(x_1) = -100$ and $M(x_2) = 32$, the value produced by the ratio between $M(x_1)$ and $M(x_2)$ cannot state, for example, how many times $M(x_1)$ is greater than $M(x_2)$. A case of interval scale may occur in SBSE when there is a subtraction between metrics in the fitness function, which can lead to values for positive and negative ranges.

In turn, the *ratio scale* is characterized by the “zero” element, which means the total absence of an attribute that a metric belongs to this scale measures. This scale maintains the ordering and size property between ranges and allows any arithmetic operation (Fenton, 1991). Defect density for defect-based software quality and failure coverage for software testing can exemplify software metrics related to a ratio scale. Lastly, the *absolute scale* has properties similar to the ratio scale but is limited to the count of attributes its metric measures. For instance, Lines of Code (LoC) is an absolute scale for measuring the number of lines. However, LoC is not an absolute measure scale of the software size, as there are several ways to measure software magnitudes, such as the number of methods or classes, for instance. As one can see, several possibilities exist regarding using software metrics in SBSE. Search-based approaches have been applied to a wide range of software engineering processes,

including software design (Sharma and Sharma, 2023), software testing (Maashi, 2022), software refactoring (Mohan and Greer, 2018), and software project management (Roque et al., 2016). These applications leverage optimization techniques like evolutionary approaches (e.g., genetic programming) and bio-inspired algorithms (e.g., ant-colony optimization) to address complex SE challenges (Colanzi et al., 2020).

2.3 Conceptualizing scalarizing functions for optimization problems

Scalarizing functions are usually approached to solve problems whose character is multi-objective or multi-criteria (Kasimbeyli et al., 2019). In this sense, a scalarizing-based proposal can convert a multi-objective optimization problem to a mono-objective. Consequently, it is feasible to solve this problem through mono-objective optimization techniques (Ishibuchi et al., 2009a).

According to Miettinen and Mäkelä (2002a), most scalarization functions use at least one of the following concepts: *ideal objective vector* or *nadir objective vector*. The first one consists of a vector $z^* = f^* = [f_1^*, f_2^*, \dots, f_n^*]^T$ of the best values for each objective. Suppose a function f_i should be maximized. In that case, its value f_i^* or z_i^* is the maximum value it can reach within the set of feasible solutions. As for minimization, the ideal value would be the lowest possible value that a solution within the set of viable solutions can achieve. A single solution is likely to achieve the optimal value for all functions in multi-objective problems with conflicting objectives (Deb and Kalyanmoy, 2001).

On the other hand, the *nadir objective vector* consists of the vector of the worst values for each objective $z^{nad} = f = [f_1^{nad}, f_2^{nad}, \dots, f_n^{nad}]^T$ considering only the Pareto Front. Thus, for a function f_i where maximization is aimed, its value z_i^{nad} consists of the slightest value reached for this fitness function within the set of the Pareto Front; and, in the case of minimization, the highest value. Unlike z^* , z^{nad} cannot be obtained by maximizing or minimizing each function individually. However, z^{nad} can be obtained through the derivation of z^* using the *payoff* table method (Miettinen, 1999). From these vectors, each objective can be normalized as follows:

$$f_i^{norm} = \frac{f_i - f_i^{nad}}{f_i^* - f_i^{nad}}, \quad (6)$$

where the closer f_i is to the best value, the closer to 1 f_i^{norm} is. Otherwise, it would be closer to zero. Since each objective will be normalized in the $[0, 1]$ range, it is possible to make a mono-objective function that relates to the normalization of each objective.

In addition to the conventional scalarizing functions, which utilize the concepts of the *ideal objective vector* and the *nadir objective vector* to normalize and aggregate fitness functions, a subfield of research has emerged over the past decades known as preference-based multi-objective optimization (Thiele et al., 2009). This subfield aims to provide a more direct approach to incorporating user preferences into the optimization process by allowing users to

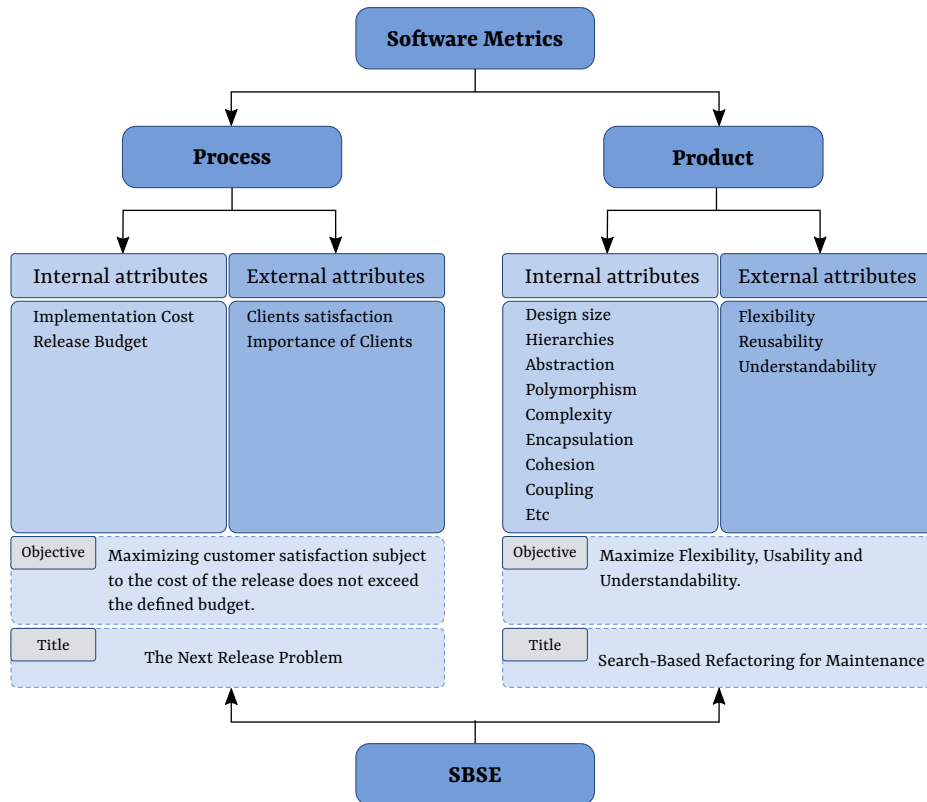


Figure 3. Examples of SE metrics in the context of SBSE (Malhotra, 2015).

define their “ideal point” in the objective space, and optimization algorithms then strive to reach that user-defined point. This approach has gained prominence in practice due to its ability to address the challenges associated with different ranges of objective functions. One well-known approach within preference-based multi-objective optimization is the Reference Point method (Wierzbicki, 1980). This method is rooted in the decision-makers need to express their preferences in solving multi-criteria problems. It involves the use of aspiration levels for each criterion to be optimized. These aspiration levels can be linked to a scalarizing function, thereby enabling the decision-maker to effectively weigh their preferences regarding the different objectives of the problem. As a result, the scalarizing function guides the search process, distinguishing solutions based on the aspiration levels distributed among the objectives. Several reference point-based scalarization functions have been proposed in the literature, such as the *Step Method* (STEM), the *Satisfying trade-off method* (STOM), and the naive method (GUESS) (Miettinen and Mäkelä, 2002a). These scalarization functions offer flexibility in how user preferences are incorporated into the optimization process.

In the context of SBSE, the Reference Point method was explored by Saraiva et al. (2017) to capture the decision maker’s preferences in the software release planning process. In this approach, human interaction occurs after the multi-objective optimization process has generated a Pareto front. At this stage, the decision maker is tasked with selecting the most suitable solution. The scalarizing function plays a crucial role in determining which solution aligns best with the user-defined aspiration levels. These levels define the relative importance of each objective to the decision-maker.

The outcomes reported in this work underscore the significance of providing software developers with the capability to interact and influence the search-based optimization process, aligning with the principles of preference-based multi-objective optimization. By acknowledging and incorporating the advancements in preference-based multi-objective optimization, our research seeks to contribute to the broader spectrum of methodologies available for addressing the challenges associated with SE metrics in SBSE.

3 Designing a scalarizing-based approach to SBSE

Our proposed mathematical model follows the concept of the scalarizing function (Miettinen and Mäkelä, 2002b). The major difference concerns the normalized values in the fitness function. While in a scalarizing model, these values are obtained through exact optimization methods (or payoff table methods), our approach relies on dynamically acquiring the values as the search process performs. Our approach is similar to the one proposed by Chang et al. (2001) in the sense that the best values used to normalize the multiple criteria are also obtained during the optimization process. However, our work formalizes a generic model that can be applied to any mathematical multi-criteria problem, even the ones where the metrics do not belong to the ratio scale. Indeed, the proposal by Chang et al. (2001) relies on the constraint of the investigated metrics having the neutral element (zero), which gives meaning to how the model normalizes by using the ratio of the value being evaluated over the max value. However, when at least one of the approached software metrics

can reach negative values where the minimum value is unknown, their normalization strategy would not be suitable.

As introduced in Section 2.3, the *ideal objective vector* (z^*) and *nadir objective vector* (z^{nad}) represent the best and the worst value, respectively. Analogously to the z_i^* aim, we propose the $best_{ij}$ variable representing the best value for a SE metric i until the iteration j of a given search-based algorithm. On the other hand, similar to z_i^{nad} , we propose the $worst_{ij}$ value to represent the worst value for a SE metric i until the iteration j . In light of previous arguments, our proposed mathematical model consists of:

$$\max \sum_{m_i(X) \in M} \alpha_i \times \left(\frac{m_i(X) - worst_{ij}}{best_{ij} - worst_{ij}} \right) \quad (7)$$

subject to:

$$g_k(X) \quad k = 1, 2, 3, \dots, m, \quad (8)$$

where $M = \{m_1(X), m_2(X), \dots, m_n(X)\}$ indicates the set of SE metrics associated with the problem. Each SE metric $m_i(X)$ has a scalar α_i , representing how relevant the SE metric i is to the search process. The variables $worst_{ij}$ and $best_{ij}$ represent the worst and best value for each metric m_i until the search iteration j , respectively. The $g_k(X)$ function indicates a constraint from the set of m constraints of the problem. Thus, for a given solution, the values of each metric are normalized following the interval $[0,1]$ according to the $worst_{ij}$ and $best_{ij}$. The closer to its worst found value the metric $m_i(X)$ is, the closer to 0 the portion of m_i is in the overall sum, making m_i less relevant. The same portion can tend to 1 as the closer $m_i(X)$ is from the best value. Finally, each metric m_i has a specific weight α_i that represents its relevance to the search process. However, considering a context where all software metrics have the same relevance, one can use $\alpha_1 = \alpha_2 = \dots = \alpha_n$.

In summary, our proposed scalarization model offers benefits in multi-objective optimization scenarios where pre-knowing the exact weights of metrics is not feasible. By dynamically adjusting the relevance of metrics during the search process, the model removes the need for users to define weights a priori. This flexibility allows for the exploration of a broader solution space and adapts to changing requirements as the optimization unfolds. In real-world SBSE applications, where exact weight configurations are often unknown or may evolve, this adaptability ensures that the optimization process remains efficient and relevant to the user's evolving goals. This feature makes the scalarization model especially valuable in scenarios requiring ongoing adjustment and refinement of metric importance.

As we can notice, our proposed scalarization method shares similarities with multi-objective evolutionary algorithms, such as MOEA/D and its variants (Zhang and Li, 2007). These algorithms aim to explore the Pareto Front by searching for different weight vectors to optimize multiple objectives simultaneously. MOEA/D, in particular, decomposes a multi-objective optimization into a number of scalar optimization subproblems, each optimizing a different objective (Zhang and Li, 2007). What sets it apart is that each subproblem is optimized using information from its neighboring subproblems, which allows MOEA/D to achieve lower

computational complexity at each generation. While our approach is based on the scalarization function, it effectively involves the exploration of different weight configurations to steer the search process, aligning with the fundamental idea behind MOEA/D. This similarity underscores the goal of efficiently exploring the Pareto Front in multi-objective SBSE. Our research aims to offer a different perspective by formalizing a scalarization model that can be applied to a wide range of SBSE problems, providing an alternative and complementary approach to existing optimization algorithms.

4 Empirical Study

This empirical study was designed in the form of two computational experiments. The **Experiment #1** aimed to evaluate the control capability of the proposed scalarizing-based approach over the relevance of the SE metrics during the search process in the scenario where all metrics should have the same weight. In other words, we compared the results of our proposal with two SBSE solutions (Dreyton et al., 2015; Dantas et al., 2015) where the authors applied aggregated fitness functions. On the other hand, the **Experiment #2** focused on the efficacy in a scenario where all SE metrics do not necessarily should have the same relevance during the search process. Hence, we also evaluate the capability of the mathematical model using different weight configurations other than just the equality one. We gathered the solutions generated from the different configurations and selected the non-dominated solutions. As a result, we generated an approximation of a Pareto Front that we compared with the NSGA-II (as a benchmark) in one of the most studied SBSE problems: the Multi-objective Next Release Problem (MONRP) (Zhang et al., 2007; Rahimi et al., 2023).

Therefore, our intention was not to conduct an exhaustive exploration of all possible SBSE domains. Instead, our focus was on providing a proof of concept and demonstrating the feasibility of our approach in scenarios that are representative of different stages of the software development lifecycle. This choice allowed us to evaluate the approach under feasible conditions while managing computational resources.

Moreover, knowing that metaheuristics are widely known for being stochastic, distinct executions would probably return different solutions making necessary an accurate analysis to evaluate their performance. Hence, the descriptive statistic with central tendency and variation measures were necessary, being complemented with the use of statistical tests (Arcuri and Briand, 2014). To deal with this issue, the evaluated algorithms were executed 30 times. At the end of all executions, the average and standard deviation from the analyzed variables were collected. In addition, statistical tests were applied to the collected outcomes; they were: the Mann-Whitney U-Test to evaluate the strength of evidence against the null hypotheses (Mann and Whitney, 1947) and the \hat{A}_{12} Vargha-Delaney statistics' to assess the effect size between the variable of two distinct populations (Vargha and Delaney, 2000).

4.1 Experiment #1: On the control capability in a scenario where all SE metrics should have the same relevance during the search process

4.1.1 Experimental design

As previously discussed, this experiment aims to evaluate the application of the proposed mathematical model based on scalarizing function as an alternative to the aggregated functions widely applied in SBSE works. Therefore, this experiment assesses the proposed approach's capability to control the relevance of the SE metrics where all of them should have the same relevance during the optimization process. For this first experiment, we established the following hypotheses:

- **Null hypothesis - H_{10}** : the proposed scalarizing-based approach **has not** a better performance concerning the capability to control SE metrics' egalitarian relevance during the search process;
- **Alternative hypothesis - H_{11}** : the proposed scalarizing-based approach **has** a better performance concerning the capability to control SE metrics egalitarian relevance during the search process;
- **Null hypothesis - H_{20}** : the overall average value between the SE metrics **is not** improved by using our scalarizing-based approach;
- **Alternative hypothesis - H_{21}** : the overall average value between the SE metrics **is improved** by using our scalarizing-based approach.

To evaluate the hypotheses H_{10} e H_{11} , we propose a new experiment metric called *Absolute Difference Average* (ADA) which represents the average of the absolute difference between the normalized metrics:

$$ADA(X) = \frac{2 \times \sum_{i=1}^{n-1} \sum_{j=i+1}^n |(\|m_i(X)\| - \|m_j(X)\|)|}{n(n-1)}, \quad (9)$$

where, given a set of SE metrics $M = \{m_1, m_2, \dots, m_{n-1}, m_n\}$ associated with a problem P whose X is a solution. $\|m_i(X)\|$ is the normalized value of $m_i \in M$ concerning its worst and best-estimated value in the scenario where m_i is a fitness function, and the constraints of the problem are taken into account. For instance, to obtain the best and worst estimated value, a search algorithm can be executed twice using m_i as a fitness function: the first execution would maximize m_i , while the second would minimize it. At the end of these steps, the normalized $m_i(x)$ is given by $\|m_i(X)\| = \frac{m_i(X) - worst_i}{best_i - worst_i}$. Therefore, the closer to zero ADA value of a solution is, the more similar the relevance between the software engineering metrics was in the search process. For instance, an ADA = 0.5 indicates that, on average, the solutions differ in relevance by 50% from each other.

Regarding the hypotheses H_{20} and H_{21} , we also designed a new experiment metric called *Normalized Metrics Average*

(NMA) to support the analysis. Our purpose with this experiment metric is to verify whether the scalarizing-based approach can reach superior average values regarding the normalized values of each SE metric when compared to another fitness function (in our case, the canonical one to the problem under study). Thus, NMA can be formulated as follows:

$$NMA(X) = \sum_{i=1}^n \frac{\|m_i(X)\|}{n}. \quad (10)$$

Analyzing the NMA in conjunction with MDA, one can argue if the proposed approach produces lower values of MDA with similar values of NMA, this result would be indicative that the proposed approach can produce fairer results in terms of relevance (explained by the MDA) without compromising the quality of the optimization (explained by the NMA). Therefore, lower values of NMA would indicate that the scalarizing-based approach guided the process towards a solution more distant from the optimal solution in order to active justice between the SE metrics which is not something desired. For greater values of NMA and still considering lower values of MDA, would be an indicative the proposed approach has control over SE metrics producing fairer results and improving the guidance of the process to near optimal solutions.

Having the experiment metrics introduced, we may advance to formalize the experiment variables. As *independent variables*, we have the fitness function and the number of metrics. In this case, the fitness function can have, as variant characteristics in its metrics, the presence of normalization and the type of operator used to aggregate the metrics. In addition, one can highlight the impact of the number of SE metrics comprising a fitness function on the dependent variable. Then, regarding the *dependent variables*, we approached our proposed experiment variables: *Absolute Difference Average* (ADA) and *Normalized Metrics Average* (NMA).

As we previously mentioned, this first experiment has two different SBSE problems as study objects; they are: 1) the fitness function formulated by Dreyton et al. (2015) to the Bug Prioritization Problem (BPP) and 2) the fitness function designed by Dantas et al. (2015) to the Release Planning Problem (RPP). In summary, the BPP consists of finding the best ordering of bugs to be fixed in open-source development. In turn, the RPP deals with the need to select which requirements must be implemented alongside different software releases. Given the space constraints, we cannot discuss each mathematical model in-depth. For more details, we recommend reading the original papers (Dreyton et al., 2015; Dantas et al., 2015). However, for the sake of clarity, we will overview below the main idea behind each investigated fitness function, including the evaluated SE metrics (which is the most important issue for our study).

Firstly, Dreyton et al. (2015) proposed a mono-objective model to the BPP, which consists of:

$$\begin{aligned} &\text{maximize} && (\alpha \times \text{relevance}(P) + \beta \times \text{importance}(P) \\ &&& - \gamma \times \text{severity}(P)), \\ &\text{subject to:} && pos(P, b_i) < pos(P, b_j), \text{ if } b_i \prec b_j \text{ and } b_j \in P, \end{aligned} \quad (11)$$

where α , β , and γ are weights used to weigh each function according to the faced scenario. The only constraint addressed refers to the technical precedence between bugs (a bug fix depends on the fix of another one). In this case, they proposed that $B = \{b_i \mid i = 1, 2, 3, \dots, N\}$ is a set of reported bugs in which N represents the number of bugs available to be prioritized. The solution representation is a vector of elements of B with a specific ordering, $P = \{p_j \mid j = 1, 2, 3, \dots, M\}$, where M is a parameter defined that represents the number of bugs to be presented as a solution. The *relevance*(P) represents the overall relevance of P according to the community that maintains the project. The *importance*(P) prioritizes bugs with great priority value considering the opinion of the person responsible for bug triage. The *severity*(P) prioritizes the correction of the bugs assigned as severe by users.

The proposed mono-objective model to the RPP designed by Dantas et al. (2015) consists of:

$$\begin{aligned} & \text{maximize} \quad \sum_{i=1}^N y_i \times (\text{value}_i \times (P - x_i + 1) - \text{risk}_i \times x_i), \\ & \text{subject to:} \quad \sum_{i=1}^n \text{cost}_i \times f_{i,q} \leq s_q, \forall q \in 1, 2, \dots, P, \end{aligned} \quad (12)$$

where $y_i \in \{0, 1\}$ is 1 when the requirement r_i was selected to some release, that is, $x_i \neq 0$, and 0 otherwise. Hence, $R = \{r_i \mid i = 1, 2, 3, \dots, N\}$ is the set of requirements available to be allocated for the releases $K = \{k_j \mid j = 1, 2, 3, \dots, P\}$, where N and P are the numbers of requirements and releases. The solution representation is a vector $S = \{x_1, x_2, \dots, x_N\}$ where $x_i \in \{0, 1, \dots, P\}$, such that if $x_i = 0$ the requirement r_i is not allocated to any release; otherwise the requirement is allocated for a release k_p , being $x_i = p$. The *value* _{i} contains the weighted sum of the importance assigned by a client c_j for a requirement r_i . Moreover, the *risk* _{i} is defined as the risk associated with a requirement r_i caused by a supposed implementation postponement. The only constraint concerns the release cost, do not exceed the budget value. In other words, each requirement r_i has a cost *cost* _{i} , and each release k_p has a budget value s_q . In short, the more the requirement with the greatest *value* and *risk* values are implemented in the first releases, the greater the fitness function result is.

In addition, Dantas et al. (2015) also included in their approach the possibility of influencing the optimization process in the light of the subjective preferences of a decision maker (DM). In this case, they suggested an importance level (assigned by the DM to the requirements) that penalizes possible solutions according to the importance level of each preference that is not satisfied (the higher the number of not-satisfied preferences, the higher the penalty value). However, we have decided to exclude this component from our analysis because it goes beyond our experimental scope and does not jeopardize our research objective.

4.1.2 Experimental setup

To evaluate our proposed scalarizing-based approach, we selected the Genetic Algorithm (GA) as the mono-objective search algorithm for this experiment. The GA was chosen due to its popularity and effectiveness in SBSE (Harman,

2011). Moreover, limiting our study to one search optimizer simplifies the evaluation and comparison process, ensuring a clear and focused assessment of our proposed scalarizing-based approach. Given the GA's stochastic nature, we collected the evaluated metrics derived from 30 executions for each instance of each problem. For the BPP, we explored the *dataset_inst100* provided by Dreyton et al. (2015), which comprises 100 bugs reported in the software repository of the Kate Editor. Regarding the RPP, we investigated two instances (also evaluated by Dantas et al. (2015)): *dataset-1* (representing a word processor software containing 50 requirements) and *dataset-2* (representing a software development planning tool with 25 requirements). All instances investigated in this experiment are based on real-world data.

Aiming to get the best and worst values for comparison purposes, we ran the GA 30 times for each SE metric, having each one as the mono-objective to be maximized, collecting the best value of all runs if the metric is to be maximized and the worst otherwise. Similarly, we ran the minimization process for each metric, getting the best value if the metric was to be minimized and the worst one otherwise. Finally, we use these values to calculate our experiment metrics, MDA and NMA.

Moreover, many different parameters must be chosen when applying an SBSE technique. Unfortunately, it is not possible to find an optimal parameter setting. As stated by Arcuri and Fraser (2011), "for any problem, an algorithm is good at solving, you can always find another problem for which that algorithm has worse performance than another algorithm". We present our list of empirically obtained parameters and settings for each investigated problem in Table 1. The selection of parameters in our study was made to provide a balanced and consistent evaluation. The parameters were empirically obtained based on prior research (Arcuri and Fraser, 2013) and careful consideration of the problem domain. For instance, the number of generations was set to 1000 because it represents a trade-off between the computational effort and the convergence of the algorithms.

Table 1. Genetic Algorithm's parameters and general settings

Parameter	Bug Prioritization Problem	Release Planning Problem
Crossover Operator	Order One	Single Point
Mutation Operator	Rank Swap Mutation	Random Resetting
Crossover Rate	90%	90%
Mutation Rate	20%	1%
Population Size	100	100
Number of Generations	1000	1000
Weight functions	$\alpha = \beta = \gamma = 1$	

Regarding the evolutionary operators, Order One, Single Point, and Random Resetting are widely adopted in the SBSE literature. In the specific case of the Mutation Operation for the BPP, however, we had to develop a new operator called Rank Swap Mutation, which is inspired by the well-known Swap Mutation. This adaptation was required because, different from the original Swap Mutation, the used operation is more suitable for partial ordering where the solution size can vary. Given the paper's space constraint, we will not discuss the implementation details. However, all information is openly available in our supporting repository¹.

¹<https://github.com/ItaloYeltsin/SelectionFactor>

4.1.3 Results and analyses

As previously introduced, the primary objective of this first experiment is to validate different hypotheses related to the capability of our scalarizing-based approach to control the relevance of the SE metrics' toward a fair search-based optimization process. In other words, we want to investigate how to provide a fair condition to the search process to achieve balanced results for each SE metric to be optimized when using the same weight configuration and, consequently, to avoid misinterpretation of the results.

Table 2 shows the SE metrics' values reached from our scalarizing-based approach and the canonical fitness functions for the RPP (using *dataset-1* and *dataset-2*). For each dataset and each metric, we present the average results obtained by the scalarizing-based approach (and its normalized result), the canonical function (and its normalized result), and the overall best and worst values achieved throughout the 30 times GA executions. It is also important to highlight that *Risk* is a metric to be minimized and *Value* a metric to be maximized. We may observe a reduction of 47,26% of *Risk* for a loss of 5.79% in *Value* when using the proposed mathematical model in comparison to the canonical one considering the instance *dataset-1*. In regard to the canonical function, the reason why *Value* metric was so privileged about *Risk* can be explained by how greater the *Best Overall* of *Value* (24404) over the *Worst Overall Risk* (1085). Such a difference leads the search process to tend to solutions that maximize the first metric given that the second one has not enough room to penalize the fitness function. The same does not occur in the proposed mathematical model for the reason that both metrics are normalized in the $[0, 1]$ interval.

Regarding the *dataset-2*, we verified the reduction of *Value* is 51.17% for a loss of 8.71% in *Value*. The same behavior can be observed for *dataset-2* concerning the overall values, where *Value* has a *Best Overall* of 38420 and *Risk* a *Worst Overall* 704. Evaluating the normalized value in the light of the best and worst values for the *dataset-1*, there is a gain of 23% of *Risk* for a loss of 5.64% of *Value*. In addition, there is a gain of 27.01% in *Risk* for an 8.59% loss in *Value* on the *dataset-2*.

Table 2. Average of the non-normalized, normalized, worst, and best value for each metric associated to the Release Planning Problem (RPP)

	Functions	Value	Risk
dataset-1	Scalarizing	22368.8±537.74	280.97±21.9
	Normalized Scalarizing	0.9166±0.022	0.741±0.0202
	Canonical	23745.73±394.99	532.77±26.35
	Normalized Canonical	0.973±0.0162	0.509±0.0243
	Best Overall	24404	0
	Worst Overall	0	1085
dataset-2	Scalarizing	34599.3±976.78	181.43±23.39
	Normalized Scalarizing	0.9006±0.0254	0.7423±0.0332
	Canonical	37902.13±348.3	371.6±19.11
	Normalized Canonical	0.9865±0.0091	0.4722±0.0271
	Best Overall	38420	0
	Worst Overall	0	704

By observing the outcomes in Table 3 concerning the BPP (using the instance *dataset_inst100*), we may conclude that the scalarizing approach presented an improvement of 30.10% on the metrics *Relevance* for a loss of 7.14% in *Importance* and a raise of 18.68% in *Risk*. Regarding the nor-

malized values, we noticed a relative gain 23.73% in *Relevance* for a loss of 7.78% in *Importance* and a gain of 5.58% in *Risk*. These results indicate an excellent gain for *Relevance* for a lesser loss in the remainder of the metrics, *Importance* and *Risk*, considering both gross and normalized values.

In addition, a behavior can be observed from *RPP* to *BPP* with respect to the metrics *Importance* and *Severity* when compared to *Relevance*. As the first two SE metrics can reach greater overall values, they have more weight in the canonical fitness function which leads the search process to be guided toward solutions that favor these metrics.

Table 3. Average of the non-normalized, normalized, worst, and best value for each metric associated to the Bug Prioritization Problem

Instance	Functions	Importance	Relevance	Severity
dataset_inst100	Scalarizing	351.21 ±9.86	13.98 ±0.31	111.38 ±5.14
	Normalized Scalarizing	0.7376 ±0.0284	0.8375 ±0.0238	0.8769 ±0.0151
	Canonical	378.21 ±6.88	10.85 ±0.43	92.54 ±3.32
	Normalized Canonical	0.8154 ±0.0198	0.6002 ±0.033	0.9323 ±0.0098
	Best Overall	441.7	16.08	70.15
	Worst Overall	93.5	2.95	408.05

Given the previous findings, we may advance to discuss the experiment metrics (ADA and NMA) and provide another important layer of analysis. To this end, we present in Table 4 the ADA and NMA results calculated from the normalized metrics' values for each problem (and each evaluated instance). Consequently, we may contrast the results obtained by the scalarizing function with the canonical one in terms of a) the average of the absolute difference between the normalized values and b) the normalized metric average. In addition, Table 4 also shows the obtained effect size values (using the Vargha-Delaney \hat{A}_{12} statistics) for each SE metric when comparing the scalarizing approach (1) regarding the canonical function (2). To this analysis, we approached the following symbol convention considering a confidence level of 99% ($p - value < 0.01$): Δ means that one did not yield a *significantly* higher average to 2; \blacktriangle means that 1 yielded a statistically *significantly* higher average than 2; ∇ means that 1 has not a *significantly* smaller average than 2 and, finally, \blacktriangledown means that 1 has a *significantly* smaller average than 2.

Table 4. Average values of the SE metrics and results of the statistical tests for ADA and NMA

Problem	Instance	Function	ADA	\hat{A}_{12}	NMA	\hat{A}_{12}
Bug Prioritization Problem	data_inst100	Scalarizing	0.093±0.0247		0.8173±0.0088	
		Canonical	0.2214±0.0202	0 ∇	0.7826±0.011	0.9977 \blacktriangle
Release Problem	dataset-1	Scalarizing	0.1756±0.0384		0.8288±0.0088	
		Canonical	0.4641±0.0277	0 ∇	0.741±0.0153	1 \blacktriangle
	dataset-2	Scalarizing	0.1583±0.0555		0.8214±0.0102	
		Canonical	0.5144±0.0231	0 ∇	0.7293±0.0166	1 \blacktriangle

As we can see in Table 4, the scalarizing-based approach has a lower ADA value when compared to the canonical function of the BPP. This result demonstrates that the BPP metrics differ by 9% from each other in terms of normalized values against 22.14% using the canonical function. Also, the statistical tests demonstrate significant differences with high magnitude in favor of the scalarizing function. The zero effect size for ADA indicates that none of the 30 GA executions

using the proposed model returned an ADA value greater than some other ADA using the canonical function. Moreover, the results for the RPP are quite similar to the BPP. Hence, the ADA value achieved by the scalarizing function is significantly lower (with zero effect size) than using the canonical function. This lower ADA value means that the average difference between each RPP metric decreased from 46.41% to 17.56% for *dataset-1*, and 51.44% to 15.83% for *dataset-2*.

Regarding the NMA analysis of the BPP, we observed a 3,47% relative gain proportional to the maximum value (which is 1). The obtained effect size indicates that none of the executions using the proposed model had an NMA value smaller than some other NMA value using the canonical function. By analyzing the RPP results, we verify a gain of 8.78% and 9.21% for *dataset-1* and *dataset-2*, respectively. This finding is also statistically covered since, for both instances, there is an effect size of 1.

Aiming to supplement the previous discussion, we provide in Figure 4 an important comparative analysis between the reached NMA values using the canonical function and the scalarizing function for RPP (upper side) and BPP (bottom side). We also provide a reference line that indicates the average of the normalized values for each metric. As one can see, there is a considerable similarity between the normalized value of the metrics using the scalarizing approach in the case of BPP. This balance does not cause a loss in the average of the normalized values but a rise since the less favored metric in the canonical approach, *Severity*, has a gain visually greater than the loss in *Importance* and *Relevance*. On the other hand, the metric *Importance* is less favored in the scalarizing approach, having the greatest loss in its normalized value. The reason why this behavior happens can be associated with the fact that the search space does not favor well-feasible and equally balanced solutions where the *Importance* outperforms in terms of normalized values. Moreover, we may notice that this behavior is also quite similar in the context of RPP. In particular, it is even more evident since, for both instances (*dataset-1* and *dataset-2* of RPP), the balances between the metrics are visually better. The difference between the reference lines representing the *NMA* is even sharper when comparing the canonical function to the scalarizing one.



Figure 4. NMA values for RPP and BPP.

Given the aforementioned findings, we conclude that the null hypotheses H_{10} and H_{20} can be rejected for all evaluated problems and instances. In addition, we can state the validity of the alternative hypotheses H_{11} and H_{21} . This conclusion indicates that the scalarizing approach has better control over the relevance of the SE metrics influence during the search-based process. This conclusion further demonstrates that our scalarizing approach is capable of producing a better average of the optimized values within the feasible solution space, particularly in scenarios where all metrics are assigned equal weight.

4.2 Experiment #2: On the efficacy in a scenario where all SE metrics do not necessarily should have the same relevance during the search process

4.2.1 Experimental design

While the previous experiment focused on scenarios where all SE metrics are given equal importance, it is also common in SBSE to encounter situations where the relevance of each metric varies during the search process. To address this issue, we adapted our scalarizing-based model to handle multi-objective context. Specifically, we aimed to develop a solution capable of constructing a near-optimal approximation of the Pareto Front, which would demonstrate the flexibility and robustness of our scalarizing approach across different weight configurations. In line with this assumption, we implemented the *Multi-Objective Scalarization Function Based Algorithm* (MOSFBA), a multi-objective algorithm grounded in our scalarizing function, designed to accommodate varying levels of metric relevance.

Hence, our second experiment aims to assess the quality of solutions generated by MOSFBA when using our proposed model with weight configurations where not all weights are equal. Such an assessment is achieved by evaluating the efficacy of MOSFBA on generating a Pareto front having as a benchmark the well-known Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb and Kalyanmoy, 2001). It is also important to highlight that this experiment does not aim to prove that MOSFBA outperforms NSGA-II. Instead, we want to exclusively show that the proposed model is capable of guiding the search process toward good solutions in a multi-objective scenario where the relevance of all metrics not necessarily must be equal.

In particular, NSGA-II was selected as the baseline algorithm due to its extensive application and proven effectiveness in multi-objective optimization within various SBSE contexts (Zhang et al., 2007; Rahimi et al., 2023; Colanzi et al., 2020). Its widespread adoption and well-documented performance in the literature made it an ideal reference point for evaluating our approach. While preference-based algorithms and MOEA/D share relevant optimization strategies with our method, especially regarding weight configurations and preference handling, we prioritized NSGA-II to maintain consistency and clarity in the experimental setup. Not comparing with MOEA/D and preference-based algorithms does not affect the contribution or validity of our work, as the primary aim of this study is to demonstrate the feasibility and ef-

fectiveness of our scalarization-based approach, rather than asserting its superiority over other multi-objective optimization algorithms. Therefore, excluding MOEA/D does not compromise the validity of the results. Moreover, the choice of NSGA-II as a baseline allows us to clearly showcase the benefits and potential of our method in a controlled experimental framework. In other words, this contribution strengthens SBSE research by introducing a validated scalarization-based approach, offering a practical and effective alternative for addressing multi-objective optimization problems in SE.

In the light of Equation 7, a Pareto Front may be iteratively constructed from the weight variation of $(\alpha_1, \alpha_2, \dots, \alpha_n)$. Following this idea, Algorithm 1 describes the MOSFBA approach, including how the weights are generated and how the solution set is built. In summary, we designed this algorithm to vary the weights according to a variation (Δ) and a maximum number (max) that each weight α_i can reach. Once a weight set is generated, a coupled metaheuristic (e.g., Simulated Annealing) is executed using our proposed mathematical model and the generated weight setting. At each loop of line 6, from line 7 to 16, a combination of weights is generated. In line 21, the meta-heuristic is executed with the configuration of weights at each loop, returning a solution s that will be added (line 22) to the Pareto front if it meets the dominance criteria presented in line 22.

As we can see, the concept behind the MOSFBA is that each parameter configuration leverages the optimization process toward different places on the Pareto front. As depicted in Figure 5, $\alpha_{i,j}$ is a variation of the weight α_i and the red curved line represents the Pareto Front of a hypothetical problem, which aims to minimize f_1 and f_2 .

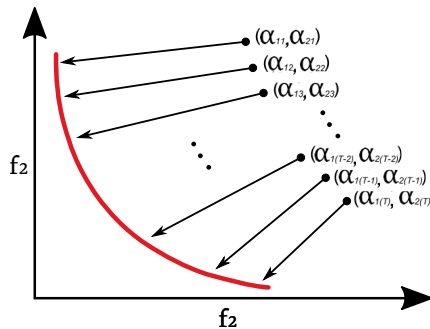


Figure 5. Pareto front construction for different weight variations.

Given that MOSFBA has the variation of weights applied to our proposed model as its main component in the guidance to build its solution set, probing MOSFBA efficacy implies that our scalarizing-based model is capable of providing good solutions in the scenario where the weights or value of the importance of each metric can vary. Consequently, the relevance of the involved metrics is set to not be equal in the search process. In that regard, we formulated the following hypotheses for our second experiment:

- **Null hypothesis - H_{10} :** the MOSFBA **has no** comparable efficacy on providing a good front of solutions having the NSGA-II as baseline comparison;
- **Alternative hypothesis - H_{11} :** the MOSFBA **has** comparable efficacy on providing a good front of solutions having the NSGA-II as baseline comparison;

Algorithm 1: Multi-Objective Scalarization Function Based Algorithm (MOSFBA)

Input: Δ , max

Output: $p \leftarrow$ Approximated Pareto Front of Solutions

```

1 begin
2   foreach  $\alpha_i \in \{\alpha_1, \alpha_2, \dots, \alpha_n\}$  do
3      $\alpha_i \leftarrow 0$ ;
4   end
5   int pos  $\leftarrow n$ ;
6   while true do
7      $\alpha_{pos} \leftarrow \alpha_{pos} + \Delta$ ;
8     while  $\alpha_{pos} > max$  do
9        $\alpha_{pos} \leftarrow 0.0$ ;
10      pos  $\leftarrow pos - 1$ ;
11      if pos < 0 then
12        break;
13      else
14         $\alpha_{pos} \leftarrow \alpha_{pos} + \Delta$ ;
15      end
16    end
17    if pos < 0 then
18      break;
19    else
20      solution s;
21      s  $\leftarrow$ 
22        executeMetaheuristic( $\alpha_1, \alpha_2, \dots, \alpha_n$ );
23      if isNotDominatedByAnySolutionFrom(s,
24        p) then
25        add s to p;
26      end
27    end
28    pos  $\leftarrow n$ ;
29  end
30 end

```

- **Null hypothesis - H_{20} :** the proposed mathematical model **is not capable of** guiding the search to provide good solutions in scenarios where **not all** the metrics to be optimized must have the same relevance.
- **Alternative hypothesis - H_{21} :** the proposed mathematical model **is capable of** guiding the search to provide good solutions in scenarios where **not all** the metrics to be optimized must have the same relevance.

With the hypotheses formulated and the information provided at the beginning of this section, we can assert that negating H_{10} implies also negating H_{20} , that is: $\sim H_{10} \implies \sim H_{20}$. In the end, the most important hypothesis for this study is H_{20} , which is directly related to our proposed model.

As the SBSE problem for this second experiment, we addressed the widely known Multi-objective Next Release Problem (MONRP), in which a set of customers with varying requirements are targeted for the next release of an existing software system (Zhang et al., 2007; Rahimi et al., 2023; Pérez-Piqueras et al., 2023). There are different formu-

lations of the MONRP; however, we followed the canonical proposal in which two objectives are considered: maximize customer satisfaction (or total value for the company) and minimize required cost (Zhang et al., 2007). In other words, the most valuable solutions are the ones that present the best trade-offs between a release that most satisfies the clients involved and has the cheapest cost. These objectives are qualified by analyzing the requirements selected regarding the variable $score_i$ that represents the sum of the satisfaction scores assigned by each client to the requirement i . In turn, the cost of each selected requirement i is represented by the variable $cost_i$. This problem can be modeled as follows:

$$\max \sum_{i=1}^n score_i \times x_i \quad (13)$$

$$\min \sum_{i=1}^n cost_i x_i \quad (14)$$

$$\text{subject to:} \quad (15)$$

$$\forall i, j \in \{1, 2, \dots, n-1, n\}, x_j \prec x_i : x_i - x_j < 1, \quad (16)$$

where, x_i is the decision variable that has value $x_i = 1$ if the requirements i was selected to be included in the next release and $x_i = 0$ otherwise. The only constraint of the problem is related to the precedence that can occur between the requirements, which is represented by $x_j \prec x_i$. In this case, x_i has as precedence x_j , and a solution that contains x_i is just feasible if x_j is also included.

To evaluate the quality of the generated solutions, we selected three performance metrics: *Generational Distance* (GD), *Spread* (SP), and *Hypervolume Ratio* (HVR). Briefly, GD measures the average geodesic distance of solutions from a front, P , to the Pareto front P^* . The smaller the GD value, the greater the convergence of P to P^* . In turn, SP measures the spacing between solutions on an approximate Pareto front P , thus also indicating the diversity of solutions on the front. Finally, we used a variation of the Hypervolume called HVR (Deb and Kalyanmoy, 2001), which consists of the ratio of the hypervolume of the analyzed Pareto front over the hypervolume of the Pareto front P^* . Given the space constraints, we cannot discuss each metric in-depth. However, these metrics are widely discussed in the SBSE literature, and we refer readers to (Van Veldhuizen, 1999; Jiang et al., 2014; Wang et al., 2016; Nuh et al., 2021) for further exploration.

4.2.2 Experimental setup

For this second experiment, we utilized the same RP instances (*dataset-1* and *dataset-2*) from Experiment #1. All instances, along with the source code, are available in our supporting repository². To account for the stochastic nature of the algorithms and ensure a fair comparison, each approach was executed 30 times for each instance. The parameters used were empirically determined based on prior research (Arcuri and Fraser, 2013) and carefully adjusted to fit the specific context of the problem. In the case of MOSFBA, we established $\Delta = 10^{-2}$ and $max = 1$ for *dataset-1*, and $\Delta = 10^{-5}$ e $max = 10^{-3}$ for *dataset-2*. This delta

value was established through preliminary experimentation aimed at balancing exploration and convergence. In our experiment, delta was chosen to ensure efficient exploration of different weight configurations without significantly increasing computational effort. However, the optimal choice of delta may vary depending on the problem domain and the desired precision. We recommend users experiment with different delta values to tailor the algorithm's behavior to their specific optimization scenario. Smaller delta values may provide more precise exploration at the cost of longer runtimes, while larger deltas may prioritize efficiency over thoroughness in the search process.

Regarding the NSGA-II configuration, the algorithm was executed within the same required time by the MOSFBA. We defined 1000 individuals per population with the following evolutionary operators: Bitflip Mutation (1% probability) and Single Point Crossover (90% probability). It is worth noticing that MOSFBA has a degree of complexity depending on the number of objectives to be solved and the parameters Δ and max , which are related to the algorithm's accuracy in building the Pareto front. Despite this fact, the executions of the coupled metaheuristic referring to each weight variation are independent of each other; that is, they do not need to wait for information from previous executions to be executed. Given this issue, we noticed a favorable scenario to carry out these executions in parallel with the support of a Graphics Processing Unit (GPU) (Bleiweiss, 2008). This adaptation was viable since there is a set of independent data that can be processed in parallel (the solutions and the weight configurations) and a single instruction (the coupled metaheuristic). To this purpose, we used the Aparapi³, which is an open-source framework for executing native Java code on the GPU. Figure 6 overviews the MOSFBA parallel workflow process using GPU.

As shown in Figure 6 before starting processing on the GPU, the problem and its instance are loaded into the motherboard's RAM. Then, configuration variations are generated using Algorithm 1. After this process, all the mentioned data and the instructions are copied to the GPU's RAM. This instruction consists of the coupled metaheuristic. In this experiment, we choose the Simulated Annealing as the metaheuristic to be coupled in the MOSFBA (and run on the different cores of the video card). This metaheuristic was selected because of its required low level of complexity and the required memory/time (Delahaye et al., 2019; van Laarhoven and Aarts, 1987).

For both dataset instances, regarding the algorithm coupled to the MOSFBA, the same Simulated Annealing configuration was approached: an initial temperature $T_0 = 4000$ and cooling coefficient $\alpha = 0.97$. From then on, the various cores of the video card can process that same instruction in parallel for different weight settings. Each core has a Global Identifier (GI) that is received before the start of the instruction. This GI is used to identify which weight configuration refers to the execution in that core. As shown in Figure 6, the kernel with $GI = 1$ has the configuration of weights W_1 as input and, at the end of the process, returns the solution S_1 referring to these same weights. The GPU does not necessarily

²<https://github.com/ItaloYeltsin/SelectionFactor>

³<https://aparapi.com>

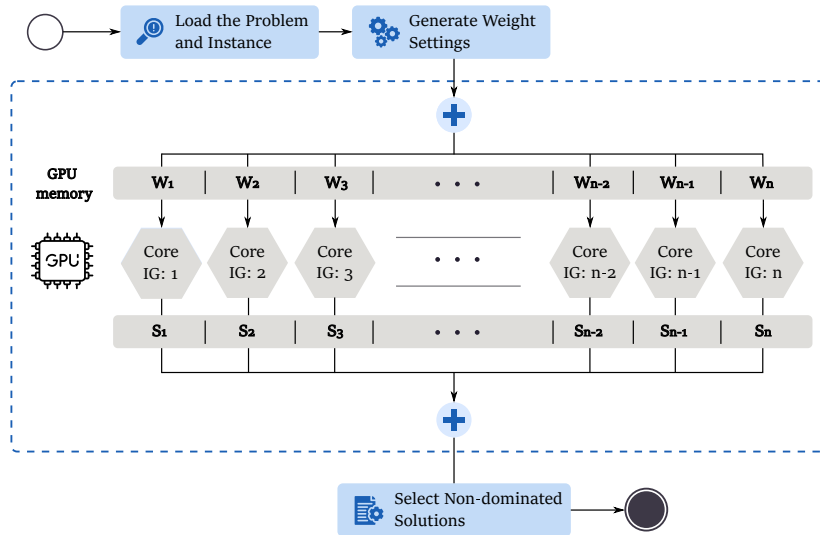


Figure 6. MOSFBA parallel workflow process using GPU.

execute all n weight configurations in parallel (only as much as the number of cores support). In this case, the number of global identifiers refers to the number of times the instruction is executed. Right after finishing its work with a certain GI, a core can subsequently receive another GI and process another configuration of weights referring to this new GI, and so on, until all configurations of weights are processed. After processing all the weights, the solutions generated in the GPU memory are copied to the motherboard's RAM, where the non-dominated solutions are selected, and, finally, the approximated Pareto front obtained by the approach is defined.

4.2.3 Results and analysis

Table 5 presents the average values of *Generational Distance* (GD), *Spread* (SP), and *Hypervolume Ratio* (HVR) obtained by the MOSFBA and NSGA-II regarding the evaluated MONRP datasets. Initially, by analyzing the results derived from the *dataset-1*, we may highlight that MOSFBA presented an increase of 20% in HVR and 97% in GD compared to the NSGA-II. On the other hand, regarding the SP value, one can observe a decrease of 41,88%. By analyzing the results from *dataset-2*, we observe that NSGA-II outperformed MOSFBA in terms of GD and SP. Specifically, NSGA-II achieved a 50% lower GD and a 228.39% higher SP. With a difference of approximately 1.03%, NSGA-II achieved a slightly higher HVR than MOSFBA.

Table 5 also shows the effect size for each metric using the MOSFBA₍₁₎ and NSGA-II₍₂₎, represented by \hat{A}_{12} , followed by one of the possible symbols \triangle , \blacktriangle , ∇ , and \blacktriangledown , of which the first two respectively indicate that the MOSFBA compared to the NSGA-II obtained a **greater** mean without and with a statistical difference, and the last two indicate a **smaller** mean without and with a statistical difference for a confidence level of 99%. MOSFBA results for the *dataset-1* obtained an effect size of 1, indicating that 100% of the times all HVR values for MOSFBA are superior, with a higher mean and statistical difference for the HVR metric; for the GD metric, effect size 0 was obtained, indicating better values of GD in 100% of the times, with a lower mean and statistically significant difference; and effect size 0 with a smaller mean and statistical dif-

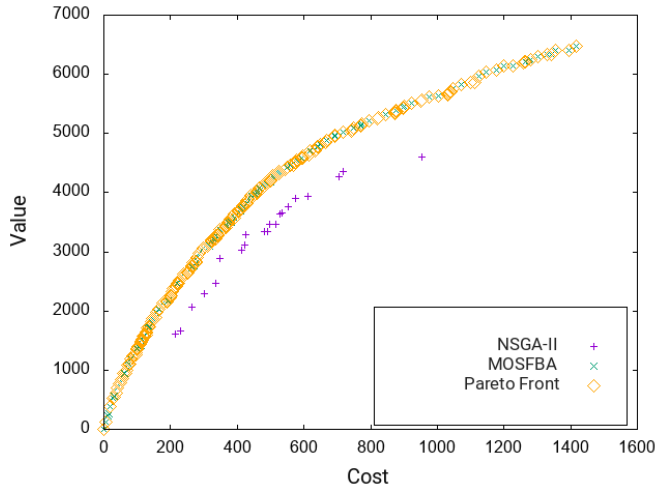
ference for the SP. Still analyzing the *dataset-1*, only 5% of the time on average, the scalarizing approach outperformed the NSGA-II in HVR and for this same metric there is a statistical difference with a lower average. Regarding the GD metric, the proposed approach obtained, on average, higher values in 85.97% of the times with statistical difference and higher mean. In terms of SP, there is an effect size of 0 with a smaller mean and statistical difference.

In order to complement our previous analysis, we present in Figure 7 the Pareto fronts obtained by MOSFBA and NSGA-II when compared to a reference Pareto front (which is denoted by the best non-dominated solutions found by the junction of the fronts obtained by the IBEA2, MOCeII, NSGA-II, and MOSFBA). As shown in Figure 7a, NSGA-II reached a front with the highest value in SP for *dataset-1*. However, this SP value is insignificant because although the solutions are well-spaced, this front does not achieve good values of HVR and GD. In addition, such an SP value does not indicate good diversity and distribution because the MOSFBA front is uniformly distributed under the reference Pareto front, which denotes good coverage (as indicated by the HVR). Regarding Figure 7b, which covers the *dataset-2*, we observe that the Pareto fronts from both NSGA-II and MOSFBA are quite similar to the reference front. However, while both algorithms exhibit similar convergence to the reference, NSGA-II demonstrates superior diversity, as reflected by its higher SP value, in contrast to MOSFBA.

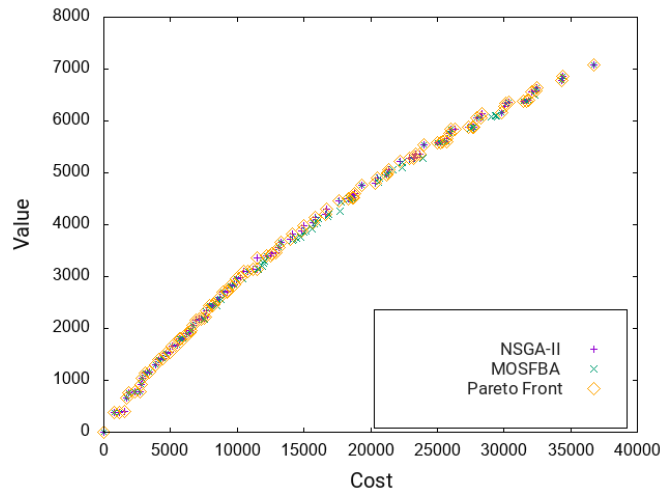
Finally, we may conclude that MOSFBA yielded better results for the *dataset-1* and comparable performance for *dataset-2*. In this case, one can argue that the low performance of the NSGA-II algorithm for the larger instance (*dataset-1*) may be related to the amount of precedence between the requirements depicted in the instance. As explained by del Sagrado et al. (2011), the number of precedences between requirements adds restrictions to the problem making the genetic operators often not be able to obtain valid solutions. This challenge is less pronounced for Simulated Annealing, the metaheuristic used in MOSFBA, as it relies primarily on a mutation operator that causes only minor disturbances.

Table 5. Average, Standard Deviation (SD) and Effect Size values of HVR, GD, and SP obtained by MOSFBA and NSGA-II

Algorithm	Instance	HVR		GD		SP	
		AVG±SD	\hat{A}_{12}	AVG±SD	\hat{A}_{12}	AVG±SD	\hat{A}_{12}
dataset-1	MOSFBA	0.684±2e-04	1▲	3e-04±0	0▼	0.5121±0.0214	0▼
	NSGA-II	0.5668±0.0597		0.0121±0.0232		0.8811±0.0804	
dataset-2	MOSFBA	0.5813±0.002	0.0589▼	2e-04±1e-04	0.8597▲	0.5266±0.0584	0▼
	NSGA-II	0.5873±0.0054		1e-04±3e-04		1.7293±0.1444	



(a) Results considering dataset-1



(b) Results considering dataset-2

Figure 7. Known Pareto Fronts for each dataset and SE metric.

Our results indicate that MOSFBA performed better than NSGA-II for the larger dataset (*dataset-1*), which presented a complex search space due to the numerous constraints generated by the precedences between the requirements. Conversely, NSGA-II slightly outperformed MOSFBA for the smaller dataset (*dataset-2*), which had fewer constraints. However, upon analyzing the solutions generated by both algorithms for *dataset-2*, we found that their performance was comparable (see Figure 7b). As a result, we can reject the null hypothesis H_{10} and accept the alternative hypothesis H_{11} , which states that the MOSFBA has comparable performance to NSGA-II. By rejecting H_{10} , we can also reject H_{20} , which was the main analysis element of the study, and accept the alternative hypothesis H_{21} . Therefore, we can conclude that the proposed mathematical model effectively controls

relevance and guides algorithms to good solutions that are superior (*dataset-1*) or closely comparable (*dataset-2*) to the NSGA-II, considering the scenarios where not all SE metrics should have necessarily the same weight.

5 Threats to Validity

Barros and Dias-Neto (2011) researched the validation of experimental studies in SBSE, listing possible threats that may occur according to four major types: Conclusion, Construction, Internal and External. We discuss below these threats and how they were mitigated in the course of this research.

Among the *conclusion threats*, one can not discard the challenge of dealing with the random variation since stochastic algorithms were used and, consequently, different executions can generate different results. Furthermore, all algorithms run in these two experiments are (or use) metaheuristics. In order to mitigate this threat, 30 executions were made for each algorithm, collecting the average and standard deviation of the metrics analyzed. We also applied statistical tests to the populations of results. In addition, a lack of a more robust performance baseline is another limitation. In Experiment #1, we analyzed two SBSE problems with a constrained number of instances. However, all of them are based on real-world data. Regarding Experiment #2, we approached the Multi-objective Next Release Problem (MONRP) with two real-world instances (varying the number of requirements).

By analyzing the *internal threats*, we may observe the influence of the parameter settings on the algorithms' performance. According to Arcuri and Fraser (2011), it is recommended that a tuning process on the parameters is a way to ensure a fairer comparison. Indeed, parameter definition in SBSE is depicted as an usual challenge (Arcuri and Fraser, 2013). However, as Experiment #1 does not compare algorithms but the influence of using different fitness functions on the result (using the same algorithm with the same parameters), these configurations do not directly influence the dependent variables. Also, in Experiment #2, the parameters were empirically obtained and were carefully defined considering prior research and the problem domain. Moreover, the Simulated Annealing coupled in the MOSFBA is extremely sensitive to instance change, which makes the ideal parameters for *dataset-1* far from the parameters for *dataset-2*. Regarding the clarity of tools and procedures to obtain the instances, all instances are based on real-world data, and the papers that make them available were cited. These instances are also open and available in our supporting repository. We also provide in Section 3 and 4 all details concerning the design and implementation of our approach and experiments. In Experiment #1, we used the Genetic Algorithm (GA) due to

its effectiveness in mono-objective optimization tasks, which aligned with the nature of the problem (Alizadeh et al., 2019; Almeida Neto et al., 2022). For Experiment #2, we selected Simulated Annealing (SA) because of its lower complexity and reduced memory and time requirements (Delahaye et al., 2019; van Laarhoven and Aarts, 1987). The use of different algorithms for each experiment ensured a fair comparison and minimized potential biases between the single-objective and multi-objective setups.

Concerning the *construction threats*, in Experiment #1, there was no need for computational cost evaluation, given that an algorithm was used varying only how the fitness function is calculated. In Experiment #2, both evaluated algorithms were run for identical amounts of time. However, the computational costs arising from using *hardware* (such as RAM, GPU, and CPU) were not considered. Nevertheless, this issue is not covered by our research aim. We also note that while our approach uses simulated global best/worst values to establish a stable baseline for comparison, this simplification may not fully capture the dynamic nature of fitness evaluations in algorithms employing external archives (e.g., hall-of-fame mechanisms). In real-world applications, the evolving nature of these references may lead to discrepancies when comparing archived solutions with current ones, as archived solutions may no longer represent the most recent progress. To address this issue, re-evaluating archived solutions against updated global best/worst values is important to maintain consistency and fairness in fitness comparisons. This re-evaluation ensures that archived solutions reflect the latest state of the algorithm, preventing outdated solutions from unfairly influencing the overall search process. While this approach introduces additional computational overhead, the impact on real-world performance is mitigated through periodic re-evaluations that help maintain the accuracy of the archive. However, this issue does not affect the validity of our experimental setup, as the simulated global values provide a consistent and stable reference for the comparative analysis.

Finally, about *external threats*, while our empirical study focused on three specific SBSE problems (Bugs Prioritization, Release Planning Problem, and Multi-Objective Next Release Problem), these problems were selected with care to represent diverse aspects of SBSE and their suitability for testing our proposed scalarizing-based approach. We considered previously published (Dantas et al., 2015; Dreyton et al., 2015; Karim and Ruhe, 2014) and based on real-world instances of each problem to account for variation within the problem space. This approach allowed us to assess the robustness of our proposed scalarization-based model across different problem instances. In particular, the Genetic Algorithm (GA) was selected as the primary search optimizer in our study due to its widespread adoption, consistency, and benchmarking value in SBSE. GA's scalability, established tuning guidelines, and suitability for complex software optimization problems align with the objectives of our research. While we acknowledge the existence of alternative algorithms, our deliberate choice of GA ensures a focused and fair comparison with our proposed scalarizing-based approach.

Lastly, our aim with our empirical study was not to exhaustively cover all possible SBSE domains but rather to focus

on a diverse set of problems that are representative of different scenarios of the SBSE development lifecycle. In this sense, we emphasize that the goal of our research is not to make sweeping claims about the entire field of SBSE but to introduce and validate our proposed scalarization-based approach. Hence, our findings encourage the SBSE community to further investigate the applicability of our model to various problems.

6 Conclusion

In the Search-Based Software Engineering (SBSE) field, several works propose evaluation functions that utilize a set of Software Engineering (SE) metrics, which vary on different properties, such as the domain to which their values belong. This variability in SE metric values can lead to some metrics being favored over others during the search process without proper acknowledgment of the decision maker. To address this issue, this study proposes a generic mathematical model based on the concept of scalarization function that aims to achieve better control over the relevance of SE metrics during the search process.

The empirical study conducted in this work aimed to investigate two perspectives. In the first perspective, we examined the control capability of our approach in a scenario where all SE metrics should have the same relevance during the search process. Specifically, we sought to understand how this approach influenced the relevance and control of software metrics during the optimization process. To do so, we compared the performance of our proposal with two existing SBSE solutions that utilized aggregated fitness functions. The second perspective focused on evaluating the proposal's efficacy in a scenario where all SE metrics do not necessarily should have the same relevance during the search process. In other words, we explored how our scalarizing-based model could handle various weight configurations beyond equal weights. We compared the performance of our approach enabled by our proposed multi-objective algorithm (MOSFBA) with the well-established NSGA-II.

The results of our empirical study demonstrated the feasibility of our scalarizing-based model as well as its capability to provide isonomic relevance between SE metrics during the search process than the canonical models proposed in previous works addressing the Release Planning Problem and Bug Prioritization Problem. Moreover, the MOSFBA outperformed the NSGA-II for the larger instance of the Multi-Objective Next Release Problem and achieved comparable performance for the smaller instance.

The research makes three significant contributions. First, it highlights the importance of understanding SE metrics comprehensively, including their impact and value range in the search-based optimization process, enabling informed metric selection. Second, it introduces a novel and generic mathematical model that utilizes scalarization function to normalize SE metrics in SBSE, adapting dynamically during the search process, thus enabling comparisons between metrics with varying scales. Lastly, through empirical evidence, we demonstrate that our approach promoted an equitable search-based process and achieved competitive performance when

each SE metric must carry equal weight. These contributions enhance SE metric understanding and offer valuable advances for the design of fitness functions in SBSE.

Future work would evaluate our approach for more SBSE problems, including many-objective scenarios and larger instances. Another avenue could explore direct comparisons with MOEA/D and preference-based algorithms to further validate the scalarization approach and expand its applicability across diverse SBSE problem contexts. Additionally, as the search space constantly changes, regarding fitness function, whenever a solution with the best/worst value is found for a metric, Evolutionary Dynamic Optimization could be explored by adapting our proposed model. Lastly, given the decision maker's freedom to configure parameters that indicate the relevance of each SE metric alongside the optimization process, our scalarizing-based approach could be analyzed in an Interactive Optimization setup.

References

- Alizadeh, V., Fehri, H., and Kessentini, M. (2019). Less is more: From multi-objective to mono-objective refactoring via developer's knowledge extraction. In *2019 19th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pages 181–192. IEEE.
- Almeida Neto, A. D., Pereira, R. L., and De Oliveira, R. C. L. (2022). Application of a genetic algorithm with social interaction to search based testing. In *2022 IEEE Latin American Conference on Computational Intelligence (LACCI)*, pages 1–6. IEEE.
- Arcuri, A. and Briand, L. (2014). A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Software Testing, Verification and Reliability*, 24(3):219–250.
- Arcuri, A. and Fraser, G. (2011). On parameter tuning in search based software engineering. In *International Symposium on Search Based Software Engineering*, pages 33–47. Springer.
- Arcuri, A. and Fraser, G. (2013). Parameter tuning or default values? an empirical investigation in search-based software engineering. *Empirical Software Engineering*, 18:594–623.
- Augusto, O. B., Bennis, F., and Caro, S. (2012). A new method for decision making in multi-objective optimization problems. *Pesquisa Operacional*, 32(2):331–369.
- Bagnall, A., Rayward-Smith, V., and Whittle, I. (2001). The next release problem. *Information and Software Technology*, 43(14):883 – 890.
- Bansiya, J. and Davis, C. G. (2002). A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on software engineering*, 28(1):4–17.
- Barros, M. and Dias-Neto, A. (2011). Threats to validity in search-based software engineering empirical studies. *Relatórios Técnicos do DIA/UNIRIO*, (0006).
- Bleiweiss, A. (2008). Gpu accelerated pathfinding. In *Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, GH '08, pages 65–74, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.
- Chang, C. K., Christensen, M. J., and Zhang, T. (2001). Genetic algorithms for project management. *Annals of Software Engineering*, 11(1):107–139.
- Chen, T. and Li, M. (2022). The weights can be harmful: Pareto search versus weighted search in multi-objective search-based software engineering. *ACM Transactions on Software Engineering and Methodology*.
- Chisalita-Cretu, C. (2014). First results on the evolutionary solution for the strategy-based refactoring set selection problem. *International Journal of Emerging Technology and Advanced Engineering*.
- Colanzi, T. E., Assunção, W. K., Vergilio, S. R., Farah, P. R., and Guizzo, G. (2020). The symposium on search-based software engineering: Past, present and future. *Information and Software Technology*, 127:106372.
- Colanzi, T. E., Vergilio, S. R., Gimenes, I. M., and Oizumi, W. N. (2014). A search-based approach for software product line design. In *Proceedings of the 18th International Software Product Line Conference-Volume 1*, pages 237–241.
- Dantas, A., Yeltsin, I., Araújo, A. A., and Souza, J. (2015). Interactive software release planning with preferences base. In *International Symposium on Search Based Software Engineering*, pages 341–346. Springer.
- Deb, K. and Kalyanmoy, D. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA.
- del Sagrado, J., Águila, I. M., and Orellana, F. J. (2011). Requirements interaction in the next release problem. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '11*, pages 241–242, New York, NY, USA. ACM.
- Delahaye, D., Chaimatanan, S., and Mongeau, M. (2019). Simulated annealing: From basics to applications. *Handbook of metaheuristics*, pages 1–35.
- Dreyton, D., Araújo, A. A., Dantas, A., Freitas, Á., and Souza, J. (2015). Search-based bug report prioritization for kate editor bugs repository. In *International Symposium on Search Based Software Engineering*, pages 295–300. Springer.
- Fenton, N. and Bieman, J. (2014). *Software metrics: a rigorous and practical approach*. CRC press.
- Fenton, N. E. (1991). *Software Metrics: A Rigorous Approach*. Chapman & Hall, Ltd., London, UK, UK.
- Fenton, N. E. and Neil, M. (2000). Software metrics: roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 357–370.
- Ferrucci, F., Harman, M., and Sarro, F. (2014). Search-based software project management. *Software project management in a changing world*, pages 373–399.
- Harman, M. (2007a). Automated test data generation using search based software engineering. In *Second International Workshop on Automation of Software Test (AST'07)*, pages 2–2. IEEE.
- Harman, M. (2007b). The current state and future of search based software engineering. In *2007 Future of Software Engineering*, pages 342–357. IEEE Computer Society.

- Harman, M. (2011). Software engineering meets evolutionary computation. *Computer*, 44(10):31–39.
- Harman, M. and Clark, J. (2004). Metrics are fitness functions too. In *Software Metrics, 2004. Proceedings. 10th International Symposium on*, pages 58–69. IEEE.
- Harman, M. and Jones, B. F. (2001). Search-based software engineering. *Information and Software Technology*, 43(14):833 – 839.
- Harman, M., Mansouri, S. A., and Zhang, Y. (2009). Search based software engineering: A comprehensive analysis and review of trends techniques and applications. *Department of Computer Science, King's College London, Tech. Rep. TR-09-03*.
- Harman, M., Mansouri, S. A., and Zhang, Y. (2012). Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys (CSUR)*, 45(1):1–61.
- Hughes, E. J. (2005). Evolutionary many-objective optimization: many once or one many? In *2005 IEEE congress on evolutionary computation*, volume 1, pages 222–227. IEEE.
- Ishibuchi, H. and Nojima, Y. (2007). Optimization of scalarizing functions through evolutionary multiobjective optimization. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 51–65. Springer.
- Ishibuchi, H., Sakane, Y., Tsukamoto, N., and Nojima, Y. (2009a). Adaptation of scalarizing functions in moea/d: An adaptive scalarizing function-based multiobjective evolutionary algorithm. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 438–452. Springer.
- Ishibuchi, H., Tsukamoto, N., and Nojima, Y. (2009b). Empirical analysis of using weighted sum fitness functions in nsga-ii for many-objective 0/1 knapsack problems. In *2009 11th International Conference on Computer Modelling and Simulation*, pages 71–76. IEEE.
- Jiang, S., Ong, Y.-S., Zhang, J., and Feng, L. (2014). Consistencies and contradictions of performance metrics in multi-objective optimization. *IEEE transactions on cybernetics*, 44(12):2391–2404.
- Karim, M. R. and Ruhe, G. (2014). Bi-objective genetic search for release planning in support of themes. In *Search-Based Software Engineering: 6th International Symposium, SSBSE 2014, Fortaleza, Brazil, August 26-29, 2014. Proceedings 6*, pages 123–137. Springer.
- Kasimbeyli, R., Ozturk, Z. K., Kasimbeyli, N., Yalcin, G. D., and Erdem, B. I. (2019). Comparison of some scalarization methods in multiobjective optimization. *Bulletin of the Malaysian Mathematical Sciences Society*, 42(5):1875–1905.
- Lightner, M. and Director, S. (1981). Multiple criterion optimization for the design of electronic circuits. *IEEE Transactions on Circuits and Systems*, 28(3):169–179.
- Maashi, M. S. (2022). A comprehensive review of software testing methodologies based on search-based software engineering. *Webology (ISSN: 1735-188X)*, 19(1).
- Malhotra, R. (2015). *Empirical research in software engineering : concepts, analysis, and applications*. CRC Press.
- Mann, H. B. and Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60.
- Miettinen, K. (1999). *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media.
- Miettinen, K. and Mäkelä, M. M. (2002a). On scalarizing functions in multiobjective optimization. *OR Spectrum*, 24(2):193–213.
- Miettinen, K. and Mäkelä, M. M. (2002b). On scalarizing functions in multiobjective optimization. *OR spectrum*, 24:193–213.
- Mohan, M. and Greer, D. (2018). A survey of search-based refactoring for software maintenance. *Journal of Software Engineering Research and Development*, 6:1–52.
- Nuh, J. A., Koh, T. W., Baharom, S., Osman, M. H., and Kew, S. N. (2021). Performance evaluation metrics for multi-objective evolutionary algorithms in search-based software engineering: Systematic literature review. *Applied Sciences*, 11(7):3117.
- O’Keefe, M. and Cinnéide, M. □. (2008). Search-based refactoring for software maintenance. *Journal of Systems and Software*, 81(4):502 – 516. Selected papers from the 10th Conference on Software Maintenance and Reengineering (CSMR 2006).
- Ouni, A. (2020). Search based software engineering: challenges, opportunities and recent applications. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 1114–1146.
- Pérez-Piqueras, V., Bermejo, P., and Gámez, J. A. (2023). Fedanrp: a fixed-structure multivariate estimation of distribution algorithm to solve the multi-objective next release problem with requirements interactions. *Engineering Applications of Artificial Intelligence*, 124:106555.
- Purshouse, R. C. and Fleming, P. J. (2003). Evolutionary many-objective optimisation: An exploratory analysis. In *The 2003 Congress on Evolutionary Computation, 2003. CEC’03.*, volume 3, pages 2066–2073. IEEE.
- Rahimi, I., Gandomi, A. H., Nikoo, M. R., and Chen, F. (2023). A comparative study on evolutionary multi-objective algorithms for next release problem. *Applied Soft Computing*, 144:110472.
- Rodriguez, D. V. and Carver, D. L. (2020). Multi-objective information retrieval-based nsga-ii optimization for requirements traceability recovery. In *2020 IEEE International Conference on Electro Information Technology (EIT)*, pages 271–280. IEEE.
- Roque, L., Araújo, A. A., Dantas, A., Saraiva, R., and Souza, J. (2016). Human resource allocation in agile software projects based on task similarities. In *Search Based Software Engineering: 8th International Symposium, SSBSE 2016, Raleigh, NC, USA, October 8-10, 2016, Proceedings 8*, pages 291–297. Springer.
- Ruiz, A. B., Saborido, R., and Luque, M. (2015). A preference-based evolutionary algorithm for multiobjective optimization: the weighting achievement scalarizing function genetic algorithm. *Journal of Global Optimization*, 62(1):101–129.
- Saidani, I., Ouni, A., Chouchen, M., and Mkaouer, M. W. (2020). On the prediction of continuous integration build failures using search-based software engineering. In *Pro-*

- ceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 313–314.
- Samli, R., Aydın, Z. B. G., and Yücel, U. O. (2020). Measurement in software engineering: The importance of software metrics. In *Applications and Approaches to Object-Oriented Software Design: Emerging Research and Opportunities*, pages 166–182. IGI Global.
- Saraiva, R., Araújo, A. A., Dantas, A., Yeltsin, I., and Souza, J. (2017). Incorporating decision maker's preferences in a multi-objective approach for the software release planning. *Journal of the Brazilian Computer Society*, 23(1):11.
- Sharma, D. and Sharma, G. (2023). Systematic literature review of search-based software engineering techniques for code modularization/remodularization. *Computational Intelligence Applications for Software Engineering Problems*, pages 241–266.
- Simons, C., Singer, J., and White, D. R. (2015). Search-based refactoring: Metrics are not enough. In *Search-Based Software Engineering: 7th International Symposium, SSBSE 2015, Bergamo, Italy, September 5-7, 2015, Proceedings 7*, pages 47–61. Springer.
- Singh, G., Singh, D., and Singh, V. (2011). A study of software metrics. *IJCEM International Journal of Computational Engineering & Management*, 11:22–27.
- Sommerville, I. (2015). *Software Engineering*. Pearson, 10th edition.
- Thiele, L., Miettinen, K., Korhonen, P. J., and Molina, J. (2009). A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary computation*, 17(3):411–436.
- van Laarhoven, P. J. M. and Aarts, E. H. L. (1987). chapter Simulated annealing, pages 7–15. Springer Netherlands, Dordrecht.
- Van Veldhuizen, D. A. (1999). *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations*. Air Force Institute of Technology.
- Vargha, A. and Delaney, H. D. (2000). A critique and improvement of the cl common language effect size statistics of mcgraw and wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132.
- Wang, S., Ali, S., Yue, T., Li, Y., and Liaaen, M. (2016). A practical guide to select quality indicators for assessing pareto-based search algorithms in search-based software engineering. In *Proceedings of the 38th international conference on software engineering*, pages 631–642.
- Wang, Y. (2003). The measurement theory for software engineering. In *CCECE 2003 - Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology (Cat. No.03CH37436)*, volume 2, pages 1321–1324 vol.2.
- Wierzbicki, A. P. (1980). The use of reference objectives in multiobjective optimization. In Fandel, G. and Gal, T., editors, *Multiple Criteria Decision Making Theory and Application*, pages 468–486, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Zhang, Q. and Li, H. (2007). Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731.
- Zhang, Y. (2012). Repository of publications on search-based software engineering.
- Zhang, Y., Harman, M., and Mansouri, S. A. (2007). The multi-objective next release problem. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07*, pages 1129–1137, New York, NY, USA. ACM.
- Zhang, Y., Harman, M., Ochoa, G., Ruhe, G., and Brinkkemper, S. (2014). An empirical study of meta-and hyper-heuristic search for multi-objective release planning. *RN*, 14(07).