Journal of the
Brazilian Computer Society

RESEARCH Open Access

Incorporating decision maker's preferences in a multi-objective approach for the software release planning

Raphael Saraiva[1*], Allysson Allex Araújo[2], Altino Dantas[3], Italo Yeltsin[1] and Jerffeson Souza[1]

Abstract

Background: Release planning (RP) is one of the most complex and relevant activities in the iterative and incremental software development, because it addresses all decisions associated with the selection and assignment of requirements to releases. There are many approaches in which RP is formalized as an optimization problem. In this context, search-based software engineering (SBSE) deals with the application of search techniques to solve complex problems of software engineering. Since RP is a wicked problem with a large focus on human intuition, the decision maker's (DM) opinion is a relevant issue to be considered when solving release planning problem. Thus, we emphasize the importance in gathering the DM's preferences to guide the optimization process through search space area of his/her interests.

Methods: Typically, RP is modelled as a multi-objective problem by considering to maximize overall clients satisfaction and minimize project risk. In this paper, we extend this notion and consider DM's preferences as an additional objective. The DM defines a set of preferences about the requirements allocation which is stored in a preference base responsible for influencing the search process. The approach was validated through an empirical study, which consists of two different experiments, respectively identified as (a) automatic experiment and (b) participant-based experiment. Basically, the former aims to analyze the approach using different search-based algorithms (NSGA-II, MOCell, IBEA, and SPEA-II), over artificial and real-world instances, whereas the latter aims at evaluating the use of the proposal in a real scenario composed of human evaluations.

Results: The automatic experiment points out that NSGA-II obtained overall superiority in two of the three datasets investigated, positioning itself as a superior search technique for scenarios with few number of requirements and preferences, while IBEA showed to be better for larger ones (with more requirements and preferences). Regarding the participant-based experiment, it was found that two thirds of the participants evaluated the preference-based solution better than the non-preference-based one.

Conclusions: The results suggest that it is feasible to investigate the approach in a real-world scenario. In addition, we made available a prototype tool in order to incorporate the human's preferences about the requirements allocation into the solution of release planning.

Keywords: Release planning, Multi-objective optimization, Human preferences, Search-based software engineering

*Correspondence: raphael.saraiva@aluno.uece.br
[1]State University of Ceará, Dr. Silas Munguba Avenue, 1700 Fortaleza, CE, Brazil
Full list of author information is available at the end of the article

Introduction

The incremental and iterative software life cycle is based on the idea of developing an initial system implementation and evolving it through several releases in a cyclic way [1]. Release planning (RP) addresses all decisions related to the requirements selection and assignment to a consecutive sequence of releases [2]. As stated by Ruhe and Saliu [3], good RP practices ensure that the software is built on providing the maximum business value by offering the best possible blend of features in the right sequence of releases. On the other hand, poor RP decisions can result in the following: (i) unsatisfied customers; (ii) release plans that are unlikely to be delivered within a given schedule, quality, and effort specifications; and (iii) release plans that do not offer the best business values.

Given the cognitive effort involved in dealing with RP, defining a "suitable" set of releases is inherently challenging. As "suitable," we may consider that it properly deals with variables that present complex relations such as stakeholders' preferences, technical constraints, limited resources, and subjective business aspects. In addition, this process can be time-consuming, requiring one to analyze an exhaustive list of possible combinations, which tend to be extremely large if the number of requirements is great. There are a number of existing approaches that are based on the belief that RP can be formalized as an optimization problem and widely explored by Search-based software engineering (SBSE) as well. In summary, SBSE proposes to apply search techniques to solve complex problems of software engineering [4].

However, RP is a wicked problem with a large focus on human intuition, communication, and human capabilities. For a wicked problem, it is not clear what the problem is and, therefore, what the solution is [3]. Additionally, turning the decision maker's (DM) feelings as a useful part of the resolution process may help avoid some resistance or little confidence in the final result [5]. Instead of just providing a simple weight factor for each requirement, for example, we emphasized the importance of providing a refined mechanism to efficiently capture the human preferences and, consequently, guide the search process. This mechanism must intuitively enable the DM to express his/her preferences in a broad scope, focusing the time in essential subjective aspects. As subjective aspects, we refer to the questions that are complex to define without human interaction, especially implicit information. For instance, the DM may want to allocate specific features in different releases, establish precedence relations or coupling relations between features according to his/her subjective knowledge.

In other words, we have to integrate the computational intelligence power with the human expertise to obtain more realistic and acceptable solutions for some wicked problems. In general, two main benefits arise from this perspective, which are to provide meaningful insights to the DM and increase the human engagement [6]. As discussed by Marculescu et al. [7], intuitive interaction with domain specialists is a key factor in industrial applicability, since it makes the system more usable and more easily accepted in an industrial setting. Despite the promising outlook, the definition regarding the type and how the preferences are exploited by optimization algorithms is a relevant challenge and has attracted attention in recent years.

Recently, the SBSE approaches based on this assumption have been discussed under the requirements engineering context. Araújo et al. [8] propose an architecture for the next release problem (NRP) based on the usage of an interactive genetic algorithm alongside a machine learning model. In that work, the preferences are gathered through a subjective mark provided by the DM to each solution, while a machine learning model learns his/her evaluation profile. After a certain number of subjective evaluations, the machine learning model replaces the DM and evaluates the remainder of the solutions. While considering the NRP, Ferreira et al. [9] propose an interactive ant colony optimization, where the DM is asked to specify which requirements he/she expects to be present or not in the next release. While these previous studies are focused on the requirements selection, Tonella, Susi, and Palma [10] present an interactive approach to the requirements prioritization. The information elicited from the user consists of a pairwise comparison between the requirements that are ordered differently in equally scored prioritization.

Regarding planning more than one release, an initial proposal of the present work was described by Dantas et al. [11]. It was designed as a single-objective model, which allows the DM to express different types of preferences considering the requirements allocation. Such preferences are stored in the preference base, in which the main purpose is to influence the search process. It was verified in the performed experiment that there was an unavoidable trade-off between the problem's metrics (score and risk) and the subjective preference. This conflict occurs because sometimes the solutions have to lose some value in score or risk to satisfy the DM's preferences.

To extend Dantas et al.'s [11] proposal, an earlier version of this work is investigated to deal with the DM preferences as another objective to be maximized in a multi-objective model [12]. Notwithstanding, the proposed approach included a strategy called the reference point method [13] to mitigate the usual cognitive effort of selecting a solution from the Pareto front. Empirical results were able to demonstrate the feasibility of the approach in an artificial environment.

Therefore, this paper significantly extends the previous work in two major aspects: (a) besides increasing the automatic experiment through two more search techniques, one large artificial dataset, and additional results, we also conduct a participant-based experiment to observe the behavior of the approach in a real-world context and (b) a prototype tool was developed and made available to enable a novel way to incorporate the human preferences during the release planning. The primary contributions of this paper can be summarized as follows:

- Experimental analyses considering both simulated and real human evaluations
- The presentation of a prototype tool for the release planning process

The remainder of this paper is organized as follows. The "Background" section presents the approach background, whereas the "Mathematical formulation" section details the mathematical model. The "Empirical study" section discusses the empirical study, and finally, the "Concluding remarks" section presents some conclusions and directions for future works.

Background

Search-based software engineering

Software engineers often face problems associated with balancing competing constraints, trade-off between concerns, and requirement imprecision. Software engineering is typically concerned with a near optimal solution or those that fall within a specified acceptable tolerance [14]. In these situations, automated optimization techniques are natural candidates. Thus, search-based software engineering (SBSE) seeks to reformulate software engineering problems as search problems. A search problem is one in which optimal or near optimal solutions are sought in a search space of candidate solutions [4].

As highlighted by Harman [15], there are only two key ingredients for the application of search-based optimization to software engineering problems:

- The choice of the problem representation amenable to symbolic manipulation
- The definition of the fitness function to characterize what is considered to be a good solution

SBSE has been applied to many fields within the general area of software engineering, such as requirements engineering [16], software design [17], and testing [18]. A wide range of different optimization and search techniques have been used by SBSE, with evolutionary algorithms (EAs) being the most popular ones [19]. EAs are generic and stochastic population-based algorithms that are inspired by biological and natural evolution concepts, such as reproduction, mutation, recombination, and selection [20]. In this work, we evaluated four different EAs, namely NSGA-II [21], MOCell [22], IBEA [23], and SPEA-II [24].

Modeling preferences for the search-based release planning

This work follows the concepts proposed by Dantas et al. [11] and includes the human's preferences in the release planning using search-based techniques. As shown in Fig. 1, such an approach is composed of three components: *interactions manager*, *preference base*, and *optimization process*.

The interactions manager is responsible for the user's interactions such as adding, modifying, or removing preferences; visualizing the best solutions; and initializing or finalizing the search process. The preference base stores every preference to facilitate the relevant information acquisition, for instance, the number of preferences that are in the base and satisfied by the solution. Finally, the optimization process is responsible for providing a solution through the search technique guided by the preference base.

An important aspect to be highlighted is how the DM can express his/her preferences about the requirements allocation throughout the releases. The authors have formalized eight types of preferences with different purposes, respectively named as *coupling joint*, *coupling disjoint*, *positioning precede*, *positioning follow*, *positioning before*, *positioning after*, *positioning in*, and *positioning no*.

The next section presents further details about the DM's preferences and mathematical formalization proposed by Dantas et al. [11].

Mathematical formulation

Consider that $R = \{r_i \mid i = 1, 2, 3, \cdots, N\}$ is the set of requirements available to be allocated to a set of releases $K = \{k_q \mid q = 0, 1, 2, \cdots, P\}$, where N and P are the

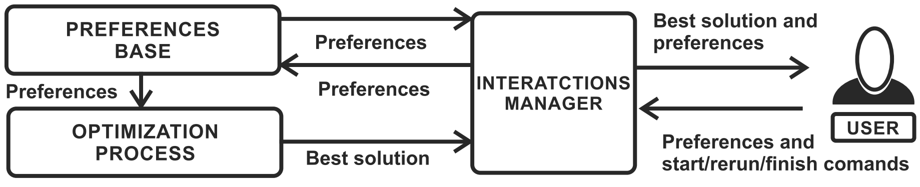


Fig. 1 Approach proposed by Dantas et al. [11]

number of requirements and releases, respectively. The vector $S = \{x_1, x_2, \cdots, x_N\}$ represents the solution, where $x_i \in \{0, 1, 2, \cdots, P\}$ stores the release k_q in which the requirement r_i is allocated, and $x_i = 0$ means that such a requirement was not allocated. In addition, consider $C = \{c_j \mid j = 1, 2, 3, \cdots, M\}$, where M is the number of clients and each client c_j has a weight w_j to estimate his/her importance to the company that develops the software. The function Value(i), which represents how valuable requirement r_i is, returns the weighted sum of the scores that each client c_j assigned to the requirement r_i as follows:

$$\text{Value}(i) = \sum_{j=1}^{M} w_j \times \text{Score}(c_j, r_i), \tag{1}$$

where Score(c_j, r_i) quantifies the perceived importance that a client c_j associates with a requirement r_i assigning a value ranging from 0 (no importance) to 10 (the highest importance). Thus, the value of the objective related to the overall client satisfaction is given by

$$\text{Satisfaction}(S) = \sum_{i=1}^{N} (P - x_i + 1) \times \text{Value}(i) \times y_i, \tag{2}$$

where $y_i \in \{0, 1\}$ is a decision variable that has a value of 1 if the requirement r_i is allocated to some release and 0 otherwise. This binary variable is necessary to avoid a requirement r_i being computed when it is not allocated. As suggested by Baker et al. [25], the clients are usually satisfied when the requirements they most prefer are implemented. Therefore, $(P - x_i + 1)$ is used for Satisfaction(S) to become higher when the requirements with a high Value(i) are allocated in the first releases, i.e, maximizing the overall clients' satisfaction.

In addition to maximizing the client's satisfaction, another relevant aspect to the project is to consider the minimization of the implementation risk of each requirement. We consider the "implementation risk" as the quantification of the danger to the project caused by the possible postponement of implementation for a defined requirement. The bigger the risk, the bigger is the probability of project failure. Identifying and dealing with risks early in development lessen long-term costs and help prevent software disasters [26]. According to Brasil et al. [27], the risk may be defined in terms of the risk impact analysis for the client's business and its probability of occurrence. As seen in Table 1, each value assume a range from 1 to 9 considering the level of impact (low, medium, and high) and of probability occurrence (low, medium, and high). For example, consider a requirement r_1 having a high negative impact on the business with a low chance of happening. This requirement will present a risk value of 7, whereas a requirement r_2 presents a risk value of 9 whether it presents a high negative impact with a maximum chance of happening.

Table 1 Impact analysis versus probability of occurrence

Impact analysis	Probability of occurrence		
	Low	Medium	High
Low	1	2	3
Medium	4	5	6
High	7	8	9

Thus, consider $D = \{d_1, d_2, ..., d_N\}$ where each d_i is the risk value associated with the requirement r_i. The value of the objective related to risk of a solution is defined as follows:

$$\text{Risk}(S) = \sum_{i=1}^{N} x_i \times d_i, \tag{3}$$

where the value of Risk(S) is smaller when the requirements with the highest risk are allocated to the first releases and, consequently, the overall project risk is minimized.

We assume that there are relations among releases and requirements which may be defined by the DM according to his/her subjective solution analysis. Consequently, he/she may obtain some insights about the problem and even adapt the decision criteria. As formalized by Dantas et al. [11], these relations may be expressed by the DM through defining a set of preferences. The process of defining these preferences follows the formalization below:

1. *Coupling joint*
   - Representation: coupling_joint(r_i, r_j).
   - Parameters: Requirements $r_i \in R$ and $r_j \in R$.
   - Interpretation: It is used to express that a DM wishes a requirement r_i to be placed together with a requirement r_j.
   - Formal interpretation: Is satisfied if, and only if, $x_i = x_j$.
2. *Coupling disjoint*
   - Representation: coupling_disjoint(r_i, r_j).
   - Parameters: Requirements $r_i \in R$ and $r_j \in R$.
   - Interpretation: It allows a DM to allocate r_i and r_j to different releases.
   - Formal interpretation: It is satisfied if, and only if, $x_i \neq x_j$.
3. *Positioning precede*
   - Representation: positioning_precede(r_i, r_j, [dist]).

- Parameters: Requirements $r_i \in R$, $r_j \in R$ and a minimum distance between the requirements, with a value always greater than zero.
- Interpretation: It enables a DM to specify that a requirement r_i must be positioned at least dist releases before a requirement r_j.
- Formal interpretation: It is satisfied if at least one of the following conditions is fulfilled:

  $(x_i, x_j \neq 0 \text{ and } x_j - x_i \geq \text{dist})$ OR
  $(x_i \neq 0 \text{ and } x_j = 0)$.

4. *Positioning follow*

- Representation: positioning_follow$(r_i, r_j, [\,\text{dist}])$.
- Parameters: Requirements $r_i \in R$, $r_j \in R$ and a minimum distance between the requirements, with a value always greater than zero.
- Interpretation: It expresses that a requirement r_i must be positioned at least dist releases after another requirement r_j.
- Formal interpretation: It is satisfied if at least one of the following conditions is met:

  $(x_i, x_j \neq 0 \text{ and } x_i - x_j \geq \text{dist})$ OR
  $(x_i = 0 \text{ and } x_j \neq 0)$.

5. *Positioning before*

- Representation: positioning_before(r_i, k_q).
- Parameters: Requirement $r_i \in R$ and a release k_q.
- Interpretation: It defines that a requirement r_i may be assigned to any release before a specific release k_q.
- Formal interpretation: It is satisfied when the following conditions are met:

  $(x_i \neq 0)$ AND $(k_q - x_i \geq 1)$.

6. *Positioning after*

- Representation: positioning_after(r_i, k_q).
- Parameters: Requirement $r_i \in R$ and a release k_q.
- Interpretation: It defines that a requirement r_i may be assigned to any release after a specific release k_q.
- Formal interpretation: It is satisfied when the following conditions are met:

  $(x_i \neq 0)$ AND $(x_i - k_q \geq 1)$.

7. *Positioning in*

- Representation: positioning_in(r_i, k_q).
- Parameters: Requirement $r_i \in R$ and a release k_q.
- Interpretation: It allows the DM to place a requirement r_i in a specific release k_q.
- Formal interpretation: It is satisfied if, and only if, $x_i = k_q$.

8. *Positioning no*

- Representation: positioning_no(r_i, k_q).
- Parameters: Requirement $r_i \in R$ and a release k_q.
- Interpretation: It defines that a requirement r_i should not be assigned in the release k_q.
- Formal interpretation: It is satisfied if, and only if, $x_i \neq k_q$.

Therefore, the main contribution of this model is the inclusion of the DM's preferences as one of the objectives to be optimized. Consider that $T = \{t_1, t_2, ..., t_Z\}$ is the set that represents the preference base, where Z is the number of preferences. Each preference t_k is a pair composed of the preference type and an importance level, which represents how valuable a preference is to the DM: the preference based on one of the types previously presented and the importance level ranges from 1 to 10 to distinguish each preference in terms of relevance. For instance, $t_1 =<$ positioning_before$(1, 2), 8 >$ denotes that the DM wishes that requirement 1 be positioned before release 2 with an importance level value of 8. Therefore, the value of the objective related to the subjective preferences is measured as follows:

$$\text{Pref}(S, T) = \begin{cases} \left(\frac{\sum_{i=1}^{Z} L_i \times \text{satisfy}(S, t_i)}{\sum_{i=1}^{Z} L_i} \right) & \textit{if } T \neq \emptyset \\ 0, & \text{otherwise,} \end{cases} \quad (4)$$

where L_i models the importance level defined by the DM for each respective preference t_i. The satisfy(S, t_i) returns 1 if the solution S satisfies the preference t_i and 0 otherwise. The objective Pref(S, T) is the unit percentage of the satisfied preferences' level in the solution compared with the total of the importance levels of all the preferences present in the preference base. Thus, this metric measures how satisfied, using the S solution, the user's preferences were.

It is important to highlight that the previous modeling does not define constraints capable of invalidating solutions that do not satisfy the DM's preferences, but it provides soft constraints that guide the search process through regions on space of solutions that are more preferred by him/her.

Regarding the hard constraints, that is, the ones that limit feasible solutions, we have considered three types that will be described from now. Firstly, we considered the technical interdependence relations between the requirements, wich are revealed a priori in the requirements specification document. The constraint Precedence(S) deals with precedence and coupling relations between the requirements as a binary matrix DEP$_{n \times n}$, as follows:

$$x_i \geq x_j, \forall i, j | DEP_{ij} = 1, \quad (5)$$

where DEP$_{ij}$ = 1, if the requirement r_i depends on the requirement r_j , 0 otherwise, and when DEP$_{ij}$ = DEP$_{ji}$ = 1, the requirements r_i and r_j must be implemented in the same release. The remainder of the matrix is filled with 0, indicating that there is no relation between the requirements and no technical limitation on their position assignment.

Furthermore, the Budget(S) constraint treats the resources available for each release. Thus, considering that each requirement r_i has a cost$_i$ and each release k_p has a budget value b_p, this constraint guarantees that the sum of the costs of all requirements allocated to each release does not exceed the correspondent budget:

$$\sum_{i=1}^{N} \text{inRelease}(x_i, k_j) \times \text{cost}_i < b_j, \quad \forall j \in \{1, 2, 3, \ldots, P\},$$

$$\text{inRelease}(x_i, j) = \begin{cases} 1, & \textit{if } x_i = j \\ 0, & \text{otherwise.} \end{cases}$$

Finally, the constraint ReqForRelease(S) guarantees that each release j has at least one allocated requirement. This constraint is described below:

$$\sum_{i=1}^{N} \text{inRelease}(x_i, j) > 0. \tag{6}$$

Therefore, our multi-objective formulation of the release planning consists of:

$$\begin{array}{ll} \text{maximize} & \text{Satisfaction}(S), \\ \text{maximize} & \text{Pref}(S, T), \\ \text{minimize} & \text{Risk}(S), \\ \text{subject to:} & 1)\text{Precedence}(S), \\ & 2)\text{ReqForRelease}(S), \\ & 3)\text{Budget}(S). \end{array} \tag{7}$$

Reference point method

An usual and challenging task associated with multi-objective problems is to decide which solution from the Pareto front will be chosen. The Pareto front is a set composed of the non-dominated solutions that represents the best trade-off among the objectives to be optimized [28]. Consequently, requesting the DM to analyze and choose a specific solution from this set can induce an excessive and additional cognitive effort. If there are more than two criteria in the problem, it may be difficult for the DM to analyze the large amount of information [29]. The first version of this work [12] proposed the use of the reference point method [13]. This method enables the DM to adjust different "aspiration levels" to each objective based on his/her preferences. These weights help the multi-objective technique to achieve solutions from the Pareto front suitable to the DM's needs.

Given G objectives, Eq. 8 is used to normalize the values reached by the solution S for each objective i in the range between 0 and 1.

$$\Delta f_i(S) = \frac{f_i(S) - o_i^*}{o_i^{\text{nad}} - o_i^*}, \tag{8}$$

where the vectors $O^{\text{nad}} = \{o_1^{\text{nad}}, \ldots, o_G^{\text{nad}}\}$ and $O^* = \{o_1^*, \ldots, o_G^*\}$ express, respectively, the highest and lowest values reached by the Pareto front and $f_i(S)$ represents the fitness function value for each objective i.

Regarding the normalization, it is done in Eq. 8 only within the reference point method, which is only used when selecting a solution from the Pareto front, but the solutions can be properly seen through the visualization interface (Fig. 4). We highlight the normalization follows a interval [0,1], where Δf_i close to 0 represents that a solution S is more close to the best achieved value for this objective i.

The DM defines the aspiration level a_i for each objective i. In our case, there are three objectives (Satisfaction, Pref, and Risk). The aspiration level is a weight that the DM specifies for each objective in order to subjectively differentiate each one. Supposing that the DM has 100 points available to distribute, he/she must decide how to allocate these points for each a_i according to their importance. The a_i values are used in the function MaxValue(S) that can be defined as

$$\text{MaxValue}(S) = \max_{i=1,\ldots,G} \Delta f_i(S) \times \Delta q_i, \tag{9}$$

where $\Delta q_i = a_i/100$. The MaxValue(S) generates a balance between the $\Delta f_i(S)$ and the DM's opinion. Considering that DM's opinion is represented by Δq_i and it is inversely proportional to $\Delta f_i(S)$, a solution that fulfills the aspiration levels generates a low MaxValue. On the other hand, when the solution S does not satisfy the DM's in one specific objective, it will have a Δf_i close to 1 that is multiplied by the Δq_i, generating a high MaxValue. Remembering that a high MaxValue implies a aspiration level to a objective which was not properly fulfilled.

As an hypothetical release planning scenario, consider that the DM has to distribute 100 points to the three objectives (Satisfaction(S), Pref(S), and Risk(S)). He/she assigned 34 points to the aspiration level a_1 and 33 points to a_2 and a_3. The multi-objective algorithm generates a Pareto front with E solutions, while a vector is created with E positions where each position represents a solution from the Pareto front, which is associated with a respective MaxValue defined by Eq. 9. Consequently, the solution that has the lowest MaxValue will be considered the one that meets the majority aspiration levels initially expressed by the DM. Finally, Eq. 10, also known as the scalarizing function, represents the solution search process from the Pareto front:

$$\begin{aligned} \text{minimize} \quad & \text{MaxValue}(S), \\ \text{subject to:} \quad & S \in E. \end{aligned} \tag{10}$$

Empirical study

The following sections present all of the details regarding the empirical study in which we followed some empirical software engineering guidelines, such as data collection procedure and quantitative results presentation [30, 31]. First, the experimental design specifications as well as research questions are presented. Then, the analysis and discussion of the achieved results are explained. Finally, the threats that may affect the validity of the experiments are emphasized.

Experimental design

The empirical study was divided into two different experiments, (a) automatic experiment and (b) participant-based experiment. Essentially, the first one aims to analyze the approach using different search-based algorithms, over artificial and real-world instances, while the second one aims to evaluate the use of the proposal in a real scenario composed of human evaluations.

Automatic experiment

Three instances were used in this experiment, named dataset-1, dataset-2, and dataset-3. Both dataset-1 and dataset-2 are based on real-world projects extracted from [32]. The first one is based on a word processor software and is composed of 50 requirements and 4 clients. The second one is based on a decision support system and has 25 requirements and 9 clients. Due to the limited size of these instances, we artificially generated dataset-3 with 600 requirements and 5 clients. After the preliminary experiments, we defined for each dataset the budget as 60% of the maximum release cost.

In addition, we evaluated two scenarios to analyze a different number of preferences and their impact on the optimization process. In the first one, called LowPrefs scenario, we randomly generated 10, 5, and 120 preferences to dataset-1, dataset-2, and dataset-3, respectively. To the HighPrefs scenario, we generated 50, 25, and 600 preferences, which is equivalent to the same number of requirements for each corresponding dataset.

Regarding the optimization techniques, we evaluated four of the most used evolutionary algorithms in the literature (NSGA-II, MOCell, IBEA, and SPEA-II) and a random search for a sanity test. All parameters were empirically obtained through preliminary tests and configured for all evolutionary techniques: 256 individuals, 400 generations, a crossover rate of 90%, and a 1% mutation rate. The Pareto front returned by the random search was generated after 102,400 solution evaluations. As suggested by Arcuri and Briand [33], we also executed each technique 30 times to deal with the stochastic nature of the meta-heuristics, collecting both quality metrics and respective averages from the obtained results.

We used an off-line procedure presented by Zhang [34] to generate a reference Pareto front (PF_{ref}), since the true (optimal) Pareto front (PF_{true}) is unknown to the evaluated datasets. Consisting of the best solutions of each technique, the PF_{ref} denotes the best available approximation to the PF_{true}. For each instance and each scenario, we executed each evolutionary technique 30 times considering 256 individuals and 1200 generations. Thus, we reached almost 9,216,000 solutions evaluated by each evolutionary algorithm, as well as another 9,216,000 solutions evaluated by the random search, achieving more than 46,000,000 evaluations. Finally, we considered PF_{ref} the best non-dominated solutions generated by all search techniques for each instance and each scenario.

The quality metrics collected and analyzed in this experiment were the hypervolume, spread, and generational distance. The hypervolume (HV) calculates the search area dominated by the Pareto front and defined from a distant point W [35]. Such a point is the worst for all objectives when compared to the solutions of all Pareto fronts being evaluated.

$$\text{HV} = \text{volume}\left(\bigcup_{i=1}^{E} v_i\right), \tag{11}$$

where E is the set of solutions from the Pareto front to be evaluated and v_i is the hypercube area formed between each solution s_i and a far point W dominated by all solutions. Thus, the volume function calculates the occupied volume in the search space by the union of all the hypercubes v_i. In summary, HV reflects the convergence and dispersion of the solutions regarding the PF_{ref}. Thus, the higher the value of this metric, the closer the known Pareto front is to the PF_{ref}.

Spread (SP) denotes the diversity accounted for by the known Pareto front. The closer to 0 this value is, the more distributed and sparser are the set of non-dominated solutions from the known Pareto front.

$$\text{SP} = \frac{\sum_{g=1}^{G} h_g^e + \sum_{i=1}^{|E|} |h_i - \overline{h}|}{\sum_{g=1}^{G} h_g^e + |E|^{\overline{h}}}, \tag{12}$$

where G indicates the number of objectives of the problem, h_i can be any distance measured between the neighboring solutions, and $\overline{h}$ is the mean value between these distance measures. h_g^e is the distance between the extreme solutions of PF_{ref} and E, corresponding to the gth objective function.

Finally, the generational distance (GD) contributes to calculating the distance between the known Pareto front obtained by the optimization technique and the PF_{ref}.

$$GD = \frac{\sqrt{\sum_{i=1}^{E} euc_i^2}}{n}, \tag{13}$$

where the value of euc_i is the smallest Euclidean distance of a solution $i \in E$ to a solution from PF_{ref}.

Participant-based experiment

As previously mentioned, this experiment aims at observing the behavior and feasibility of the approach when it is used by software engineer practitioners. We chose NSGA-II from the automatic experiment due to its ability for generating solutions with good diversity, and it was employed with the same configurations used in the automatic experiment. To test our approach, we invited 10 participants to act as decision makers (DMs). First, a questionnaire with four simple questions was conducted to identify the general profile of each participant:

- Q_1: What is your current professional occupation?
- Q_2: How much experience do you have in the IT area?
- Q_3: On a scale of low, medium, and high, how would you rate your experience as a Software Developer?
- Q_4: On a scale of low, medium and high, how would you rate your experience with release planning?

From Table 2, all participants worked as a System Analyst or a Developer. The participants had between 1 and 21 years of experience in the software industry, resulting in a total of 71 years and an average of 7.1 years of experience. In relation to the IT experience, 50% of the participants selected "High" and no one selected "Low." Regarding the experience with the release planning process, 30% of them assigned "High," while 60% assigned "Medium," and only one assigned "Low." Consequently, we may assume that these results suggest a confidence level in the evaluations and feedbacks provided by the participants.

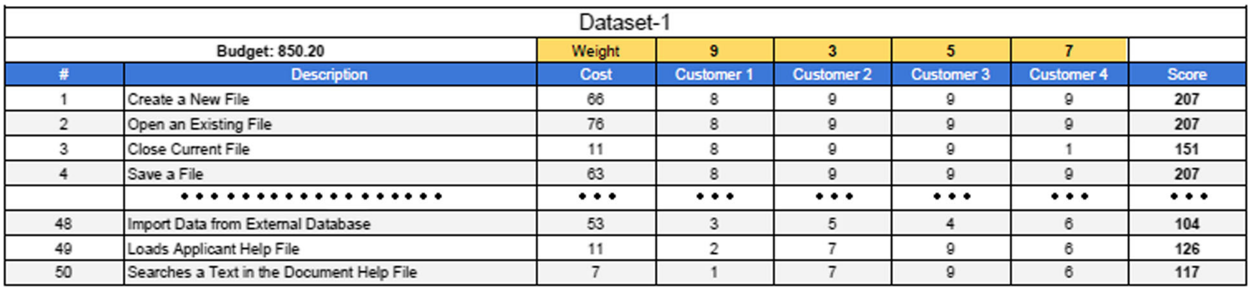
Table 2 Questionnaire answers from each participant

Participants	Q1	Q2	Q3	Q4
#1	System Analyst	12 years	High	High
#2	Developer	2 years	Medium	Medium
#3	Developer	6 years	High	Medium
#4	System Analyst	7 years	Medium	Medium
#5	System Analyst	4 years	High	Medium
#6	Developer	21 years	High	High
#7	Developer	1 year	Medium	Medium
#8	Developer	3 years	Medium	High
#9	Developer	10 years	High	Medium
#10	System Analyst	5 years	Medium	Low

The participant-based experiment consists of four major stages. In the first stage, named as "Context Guidelines," each participant was briefed about the task and scenario to be analyzed. Initially, we asked the participants (i) to perform the requirements engineer's role in a company where a software to be developed is a word processor (described in dataset-1) and (ii) all details regarding the use of our tool and how their preferences may be expressed. Subsequently, we presented a simple requirements specification document about dataset-1 (see Fig. 2), including all requirement descriptions, budget constraint, weight values given by the clients to each requirement, and, finally, the relevance of each client to the company.

After concluding general explanations, we carried out the second stage ("Non-preferences") attempting to present a solution without any preferences previously defined by the DM about the requirement allocation, i.e., considering just the Value and Risk objectives in the optimization process. Additionally, we asked the DM to weigh the objectives used in the reference point method to suggest a solution from the Pareto front. As illustrated in Fig. 3, a participant can adjust this weight configuration with a slider as he/she likes; see the requirements allocation throughout the releases, click "Optimize" to see other solutions or "Stop" when he/she is satisfied with the release plan. We also offer the opportunity to visualize the neighbor solutions of the one suggested by the reference point method. To obtain such a view, depicted in Fig. 4, the DM just has to click on "View" on the top menu.

After deciding which solution is suitable to his/her needs, the DM initiates the third stage called "Preferences Set." In this phase, we included the DM's preferences as an objective to be optimized. As Fig. 5 shows, the participants find information about the preference base on the right side of the window, including how to manage his/her preferences and to check which ones were able to be included in the suggested solution considering the weights configuration. Figure 6 exemplifies and shows the specifications required by the tool for the DM to express his/her preferences, as well as the importance level for each preference. Similar to the previous stage, the participant continuously interacts with the system until a solution is considered satisfactory. However, the main difference concerns the possibility of the DM to insert and manipulate his/her preferences.



Dataset-1							
Budget: 850.20		Weight	9	3	5	7	
#	Description	Cost	Customer 1	Customer 2	Customer 3	Customer 4	Score
1	Create a New File	66	8	9	9	9	207
2	Open an Existing File	76	8	9	9	9	207
3	Close Current File	11	8	9	9	1	151
4	Save a File	63	8	9	9	9	207
	• • • • • • • • • • • • • • • • • •	• • •	• • •	• • •	• • •	• • •	• • •
48	Import Data from External Database	53	3	5	4	6	104
49	Loads Applicant Help File	11	2	7	9	6	126
50	Searches a Text in the Document Help File	7	1	7	9	6	117

Fig. 2 A sample piece of the requirements specification document

Lastly, in the fourth stage ("Feedbacks"), we obtained feedback about how convinced the participants were about the subjective satisfaction provided by both solutions selected in stages 2 and 3, respectively called non-preference-based and preference-based solutions. As seen in Fig. 7, the participant simply estimates such evaluation to each solution on a scale of "Very ineffective," "Ineffective," "Indifferent," "Effective," and "Very effective."

Research questions

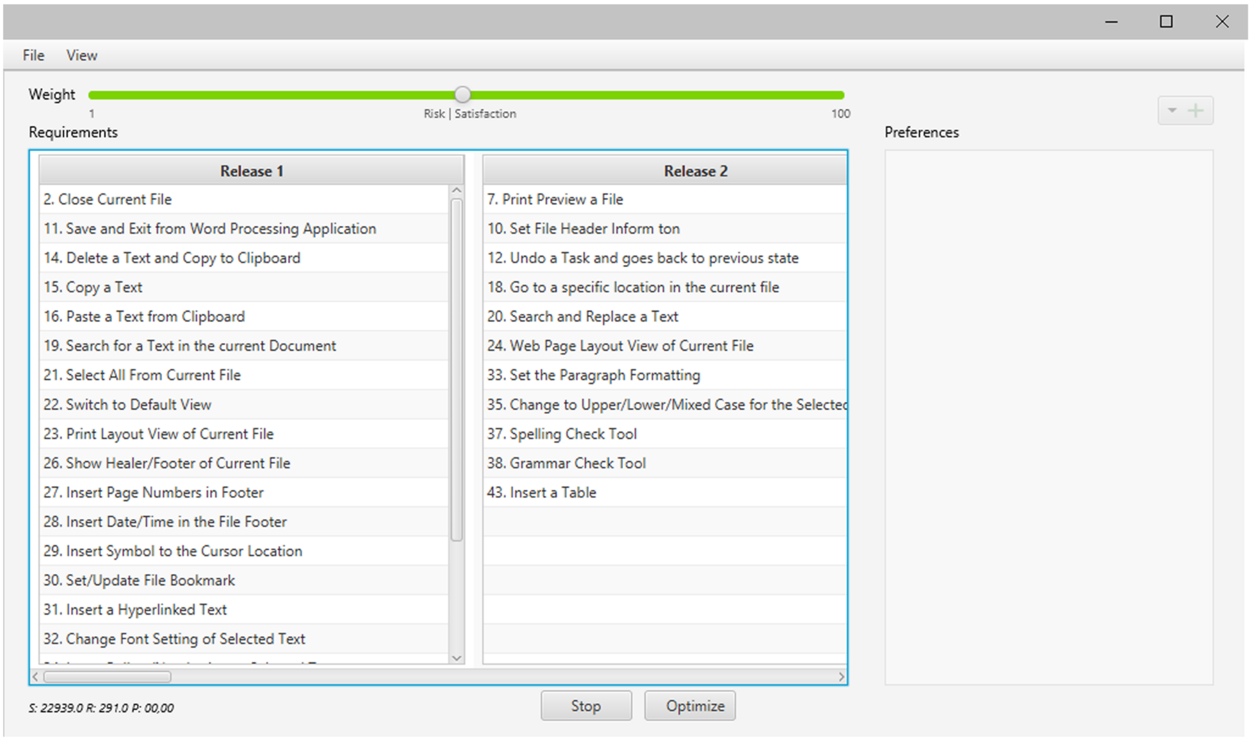
Three research questions were asked to assess and analyze our approach behavior. They are presented as follows:

- RQ_1: Which search-based techniques, among the evaluated ones, produce better solutions?
- RQ_2: Which is the subjective benefit when considering the DM's preferences as an objective to be optimized?
- RQ_3: Which is the relation between the inclusion of the preferences and the DM's subjective evaluation in the final solution?

Results and analysis

The results of the empirical study are presented in this section by analyzing the previous three research questions.

- RQ_1: Which search-based techniques, among the evaluated ones, produce better solutions?



Release 1	Release 2
2. Close Current File	7. Print Preview a File
11. Save and Exit from Word Processing Application	10. Set File Header Inform ton
14. Delete a Text and Copy to Clipboard	12. Undo a Task and goes back to previous state
15. Copy a Text	18. Go to a specific location in the current file
16. Paste a Text from Clipboard	20. Search and Replace a Text
19. Search for a Text in the current Document	24. Web Page Layout View of Current File
21. Select All From Current File	33. Set the Paragraph Formatting
22. Switch to Default View	35. Change to Upper/Lower/Mixed Case for the Selectec
23. Print Layout View of Current File	37. Spelling Check Tool
26. Show Healer/Footer of Current File	38. Grammar Check Tool
27. Insert Page Numbers in Footer	43. Insert a Table
28. Insert Date/Time in the File Footer	
29. Insert Symbol to the Cursor Location	
30. Set/Update File Bookmark	
31. Insert a Hyperlinked Text	
32. Change Font Setting of Selected Text	

S: 22939.0 R: 291.0 P: 00,00

Fig. 3 GUI used in the "Non-preferences" stage of participant-based experiment

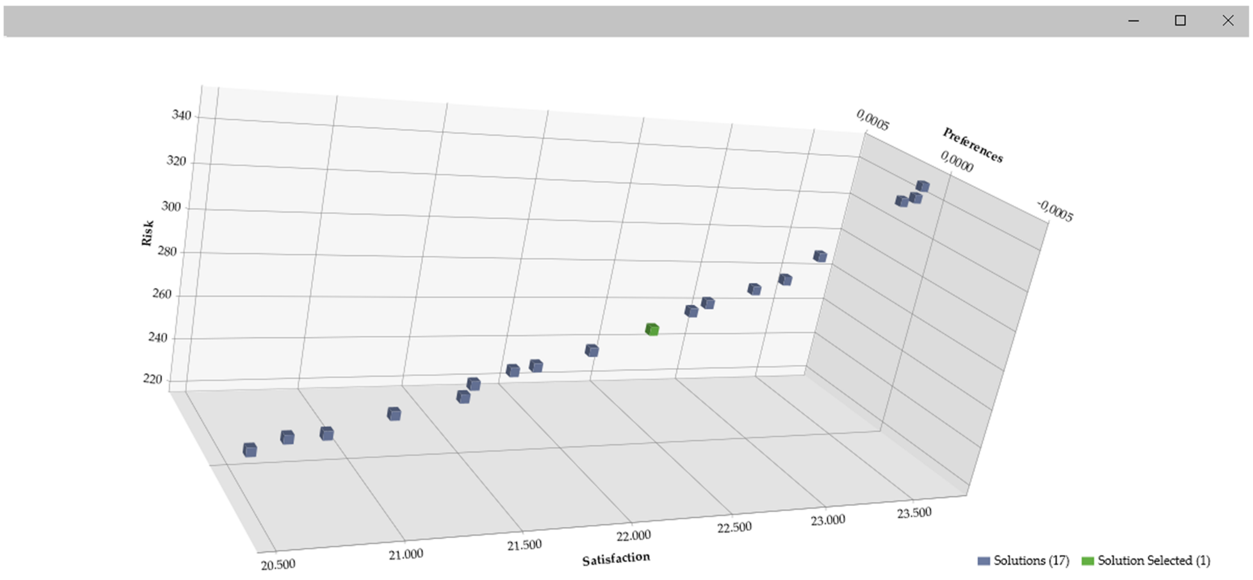
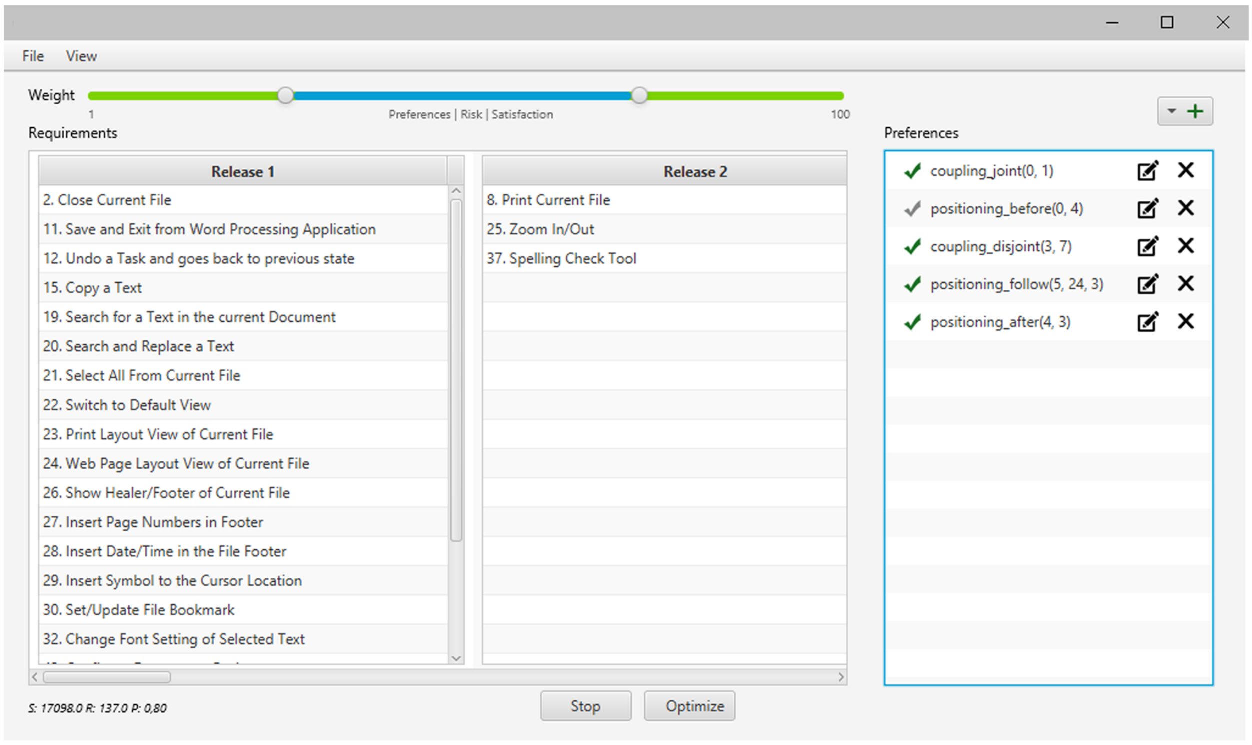


Fig. 4 Interface for visualizing solutions from the Pareto front

Aiming to answer this question, we analyzed the results produced by all algorithms for each instance and the scenario previously presented in the automatic experiment design.

Table 3 presents the values of statistical tests considering the metrics hypervolume (HV), spread (SP), and generational distance (GD) for each dataset. The Wilcoxon test (WC) was applied, using the Bonferroni correction, to identify the occurrence of statistical differences among the samples considering a confidence level of 95%. The Vargha-Delaney $\hat{A}_{12}$ test was used to measure the effect size, in other words, the relative number of

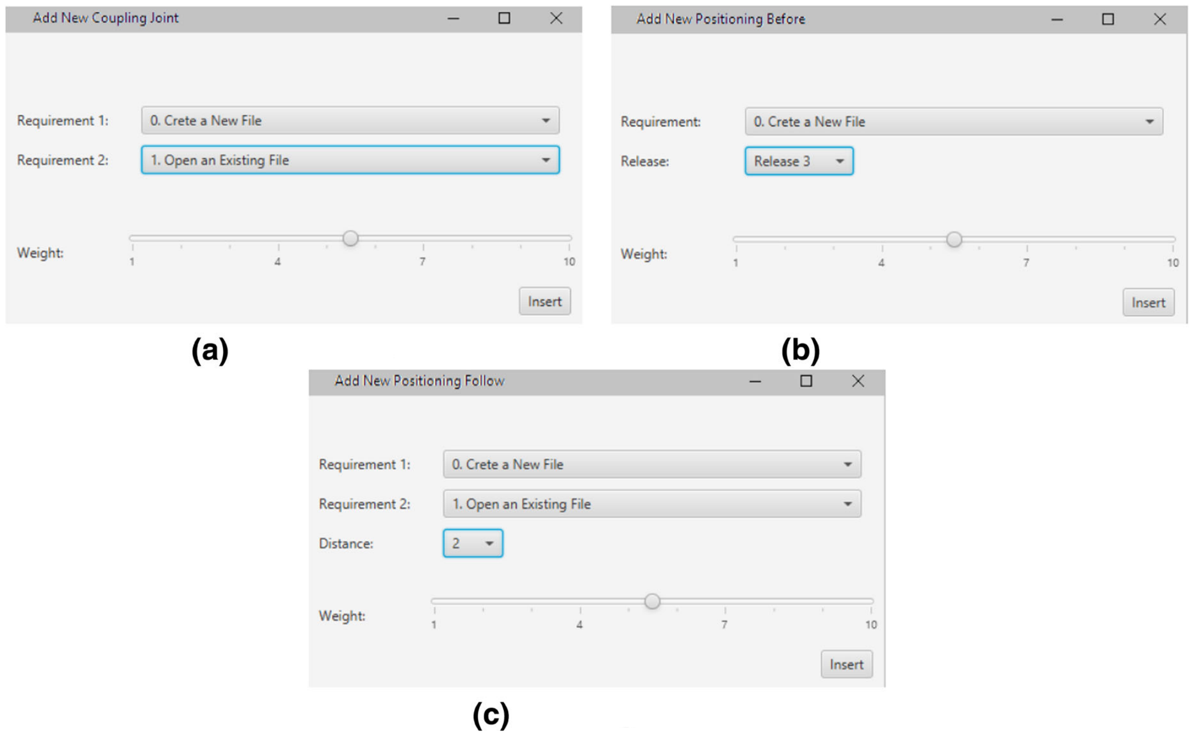


Release 1	Release 2
2. Close Current File	8. Print Current File
11. Save and Exit from Word Processing Application	25. Zoom In/Out
12. Undo a Task and goes back to previous state	37. Spelling Check Tool
15. Copy a Text	
19. Search for a Text in the current Document	
20. Search and Replace a Text	
21. Select All From Current File	
22. Switch to Default View	
23. Print Layout View of Current File	
24. Web Page Layout View of Current File	
26. Show Healer/Footer of Current File	
27. Insert Page Numbers in Footer	
28. Insert Date/Time in the File Footer	
29. Insert Symbol to the Cursor Location	
30. Set/Update File Bookmark	
32. Change Font Setting of Selected Text	

- ✓ coupling_joint(0, 1)
- ✓ positioning_before(0, 4)
- ✓ coupling_disjoint(3, 7)
- ✓ positioning_follow(5, 24, 3)
- ✓ positioning_after(4, 3)

S: 17098.0 R: 137.0 P: 0,80

Fig. 5 GUI used in the "Preferences Set" stage of participant-based experiment



(a) Add New Coupling Joint

Requirement 1: 0. Crete a New File

Requirement 2: 1. Open an Existing File

Weight: 1 4 7 10

Insert

(b) Add New Positioning Before

Requirement: 0. Crete a New File

Release: Release 3

Weight: 1 4 7 10

Insert

(c) Add New Positioning Follow

Requirement 1: 0. Crete a New File

Requirement 2: 1. Open an Existing File

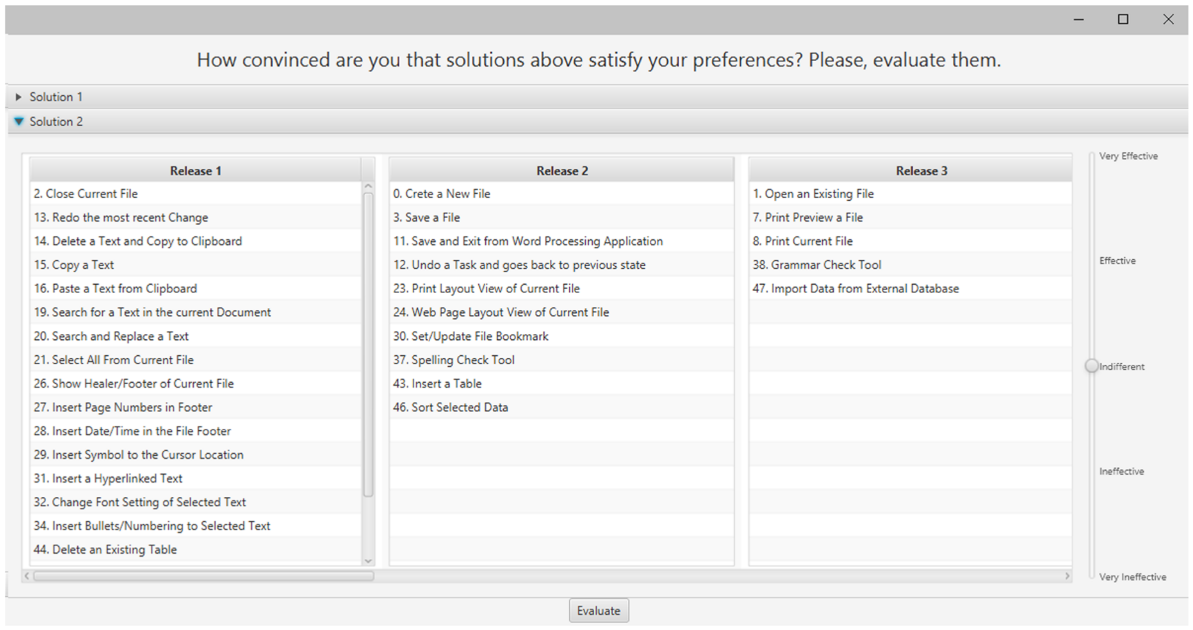
Distance: 2

Weight: 1 4 7 10

Insert

Fig. 6 Example of preferences defined by the DM. **a** Coupling Joint. **b** Positioning Before. **c** Positioning Follow

times that an algorithm produced superior values over another. Further details about each statistical test may be seen in the work of Arcuri and Briand [33]. $\hat{A}_{12}$ measures the probability that a technique (table line) with a particular parameter setting yields a higher result than another technique (table column). For instance, considering MOCell (1) and IBEA (2), the probability of MOCell returning a front better than IBEA's one according to the GD metric for dataset-1 in HighPrefs is 82.7% given that the corresponding $\hat{A}_{12}$ is 0.827.



How convinced are you that solutions above satisfy your preferences? Please, evaluate them.

▸ Solution 1

▾ Solution 2

Release 1	Release 2	Release 3
2. Close Current File	0. Crete a New File	1. Open an Existing File
13. Redo the most recent Change	3. Save a File	7. Print Preview a File
14. Delete a Text and Copy to Clipboard	11. Save and Exit from Word Processing Application	8. Print Current File
15. Copy a Text	12. Undo a Task and goes back to previous state	38. Grammar Check Tool
16. Paste a Text from Clipboard	23. Print Layout View of Current File	47. Import Data from External Database
19. Search for a Text in the current Document	24. Web Page Layout View of Current File	
20. Search and Replace a Text	30. Set/Update File Bookmark	
21. Select All From Current File	37. Spelling Check Tool	
26. Show Healer/Footer of Current File	43. Insert a Table	
27. Insert Page Numbers in Footer	46. Sort Selected Data	
28. Insert Date/Time in the File Footer		
29. Insert Symbol to the Cursor Location		
31. Insert a Hyperlinked Text		
32. Change Font Setting of Selected Text		
34. Insert Bullets/Numbering to Selected Text		
44. Delete an Existing Table		

Fig. 7 GUI presented to the DM in the fourth stage, i.e., "Feedbacks"

Table 3 Wilcoxon and Vargha-Delaney statistical values from all search algorithms, considering all metrics on each scenario

Algorithm	Datasets	LowPrefs scenario								HighPrefs scenario							
		IBEA		MOCell		NSGA-II		Random		IBEA		MOCell		NSGA-II		Random	
		WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$
Generational distance (GD)																	
MOCell	Dataset-1	3.0E−10	1.000	–	–	–	–	–	–	1.3E−04	0.827	–	–	–	–	–	–
	Dataset-2	3.0E−10	1.000	–	–	–	–	–	–	3.0E−04	0.814	–	–	–	–	–	–
	Dataset-3	3.0E−10	1.000	–	–	–	–	–	–	3.0E−10	1.000	–	–	–	–	–	–
NSGA-II	Dataset-1	3.0E−10	1.000	0.001	0.211	–	–	–	–	6.0E−04	0.197	3.4E−07	0.084	–	–	–	–
	Dataset-2	4.1E−10	0.996	0.042	0.284	–	–	–	–	*0.435*	*0.652*	0.083	0.301	–	–	–	–
	Dataset-3	3.0E−10	1.000	5.0E−10	0.000	–	–	–	–	3.0E−10	1.000	1.2E−08	0.042	–	–	–	–
Random	Dataset-1	3.0E−10	1.000	3.0E−10	0.000	3.0E−10	1.000	–	–	3.0E−10	1.000	3.0E−10	0.172	3.0E−10	0.000	–	–
	Dataset-2	3.0E−10	1.000	3.0E−10	0.000	3.0E−10	1.000	–	–	3.0E−10	1.000	3.0E−10	0.185	3.0E−10	1.000	–	–
	Dataset-3	3.0E−10	1.000	3.0E−10	0.000	3.0E−10	1.000	–	–	3.0E−10	1.000	3.0E−10	0.000	3.0E−10	1.000	–	–
SPEA-II	Dataset-1	3.0E−10	1.000	4.3E−07	0.087	*0.191*	*0.323*	3.0E−10	0.000	2.2E−07	0.078	5.1E−09	0.032	0.011	0.254	3.0E−10	0.000
	Dataset-2	6.1E−10	0.992	1.1E−04	0.168	*1.000*	*0.406*	3.0E−10	0.000	*0.112*	*0.308*	3.8E−06	0.117	6.4E−04	0.198	3.0E−10	0.000
	Dataset-3	3.0E−10	1.000	5.5E−10	0.006	*1.000*	*0.592*	3.0E−10	0.000	3.7E−10	0.997	2.6E−09	0.024	*1.000*	*0.400*	3.0E−10	0.000
Spread (SP)																	
MOCell	Dataset-1	3.0E−10	0.000	–	–	–	–	–	–	3.3E−10	0.000	–	–	–	–	–	–
	Dataset-2	3.0E−10	0.000	–	–	–	–	–	–	3.0E−10	0.000	–	–	–	–	–	–
	Dataset-3	*1.000*	*0.555*	–	–	–	–	–	–	0.117	0.690	–	–	–	–	–	–
NSGA-II	Dataset-1	3.0E−10	0.000	2.0E−08	0.951	–	–	–	–	3.3E−10	0.000	*1.000*	*0.614*	–	–	–	–
	Dataset-2	3.0E−10	0.000	6.7E−10	0.991	–	–	–	–	3.3E−10	0.000	3.0E−10	1.000	–	–	–	–
	Dataset-3	*1.000*	*0.515*	*1.000*	*0.462*	–	–	–	–	*0.063*	*0.705*	*1.000*	*0.497*	–	–	–	–
Random	Dataset-1	3.0E−10	0.000	*1.000*	*1.000*	9.1E−07	0.097	–	–	3.3E−10	0.000	*0.610*	*1.000*	0.010	0.252	–	–
	Dataset-2	3.0E−10	0.000	9.8E−07	1.000	3.0E−10	0.000	–	–	3.0E−10	0.000	5.9E−03	1.000	6.1E−10	0.007	–	–
	Dataset-3	1.3E−06	0.897	2.8E−04	0.444	2.9E−05	0.852	–	–	6.0E−06	0.875	2.5E−03	0.310	6.0E−04	0.802	–	–
SPEA-II	Dataset-1	3.0E−10	0.000	*1.000*	*0.393*	8.9E−09	0.038	*1.000*	*0.426*	3.3E−10	0.000	c 3.3E−10	0.000	3.3E−10	0.000	3.3E−10	0.000
	Dataset-2	3.0E−10	0.000	*0.900*	*0.627*	6.5E−08	0.063	8.5E−08	0.933	3.3E−10	0.000	3.3E−10	0.000	3.3E−10	0.000	2.0E−09	0.021
	Dataset-3	1.3E−06	0.102	4.3E−07	0.087	8.4E−07	0.096	9.0E−10	0.012	1.5E−08	0.045	4.2E−09	0.030	5.6E−09	0.033	8.2E−10	0.011
Hypervolume (HV)																	
MOCell	Dataset-1	3.3E−10	0.998	–	–	–	–	–	–	3.3E−10	0.998	–	–	–	–	–	–
	Dataset-2	3.0E−10	1.000	–	–	–	–	–	–	3.0E−10	1.000	–	–	–	–	–	–
	Dataset-3	3.0E−10	0.000	–	–	–	–	–	–	3.0E−10	0.000	–	–	–	–	–	–
NSGA-II	Dataset-1	3.0E−10	1.000	*1.000*	*0.581*	–	–	–	–	3.3E−10	1.000	7.0E−06	0.873	–	–	–	–
	Dataset-2	3.0E−10	1.000	0.012	0.743	–	–	–	–	3.0E−10	1.000	*1.000*	*0.607*	–	–	–	–
	Dataset-3	6.5E−08	0.063	3.0E−10	1.000	–	–	–	–	8.2E−10	0.011	8.2E−10	0.988	–	–	–	–
Random	Dataset-1	3.0E−10	0.000	3.0E−10	0.001	3.0E−10	0.000	–	–	3.3E−10	0.000	3.3E−10	0.001	3.3E−10	0.000	–	–
	Dataset-2	3.0E−10	0.000	3.0E−10	0.000	3.0E−10	0.000	–	–	3.0E−10	0.000	3.0E−10	0.000	3.0E−10	0.000	–	–
	Dataset-3	3.0E−10	0.000	3.0E−10	1.000	3.0E−10	0.000	–	–	3.0E−10	0.000	3.0E−10	1.000	3.0E−10	0.000	–	–
SPEA-II	Dataset-1	3.2E−09	0.973	4.7E−05	0.155	5.2E−06	0.122	3.0E−10	1.000	3.3E−10	1.000	0.013	0.742	*0.484*	*0.351*	3.3E−10	1.000
	Dataset-2	3.7E−10	0.997	*0.076*	*0.701*	*1.000*	*0.436*	3.0E−10	1.000	3.0E−10	1.000	*1.000*	*0.520*	*1.000*	*0.430*	3.0E−10	1.000
	Dataset-3	2.4E−07	0.080	4.5E−10	0.995	*0.820*	*0.368*	3.0E−10	1.000	3.7E−10	0.002	8.2E−10	0.980	*1.000*	*0.430*	3.0E−10	1.000

Observing the data from Table 3, it is noticeable that there was a statistical difference in most of the comparisons, except for 14.4% of the times in which there was no statistical difference (values in italic).

Before analyzing the results obtained from the GD metric, it is important to highlight that a value close to 0 is desirable because it indicates that the Pareto front is closer to the PF_{ref}. Thus, looking at dataset-1 and LowPref scenario, we can note that IBEA achieved better results of GD, due to $\hat{A}_{12}$ being 1 when the other algorithms are compared with IBEA. This means that 100% of the times, other techniques produced results higher than IBEA, for the GD metric. Such a behavior is verified in almost all of the scenarios. However, in dataset-1, when the number of preferences is high, IBEA lost to NSGA-II. Therefore, in general, IBEA outperforms all the other algorithms in terms of GD, and NSGA-II is the second best search technique among the evaluated ones.

As noted in the "Automatic experiment" section, the metric SP measures the dispersion between the Pareto front solutions. In practice, lower values for this metric bring about more uniformly distributed solutions from the front. Thus, observing the statistical test results for SP, we can see that IBEA achieves the worst results 100% of the times in comparison with all the other techniques for each scenario of dataset-1 and dataset-2. This observation indicates that even solutions returned by IBEA are next to the reference Pareto front. However, these solutions are not well distributed in the search space. On the other hand, SPEA-II obtained the best results in SP for all instances, although no statistical difference was verified in dataset-1 and dataset-2 with low preferences when compared with MOCell.

Observing both approximations to the PF_{ref} (convergence) and diversity, hypervolume (HV) is essential for evaluating the multi-objective algorithms. Because the HV value is near 1, the result is better. Thus, analyzing the results from this metric for dataset-1 and dataset-2, which are based on real data, NSGA-II shows a better performance in both datasets. For instance, in almost all of the scenarios, NSGA-II and SPEA-II achieved the highest HV values. Only in dataset-1 and LowPrefs scenario did MOCell reach the higher HV values than SPEA-II and with no statistical difference from NSGA-II, while for dataset-3 and Low and High scenarios, IBEA outperformed all algorithms in more than 90% of runs.

Figure 8 shows a comparison between the best Pareto fronts obtained by each search technique for dataset-3 on the HighPrefs scenario taking into account the hypervolume metric considering 30 runs.

Notice that in Fig. 8, all of the algorithms present different distributions of solutions. Regarding the HV metric, IBEA returns solutions that are more concentrated and closer to PF_{ref}. Among the evaluated algorithms, MOCell returns solutions that are nearer to the extreme points from the PF_{ref} and thus has good diversity. In addition, in that case, although SPEA-II provides solutions with better convergence than NSGA-II and MOCell, its front does not have good dispersion. Consequently, it represents a bad diversity. Finally, it is self-evident that random search was inferior to all of the evolutionary techniques investigated in this work.

Figure 9 shows the comparison of all the investigated search techniques considering the average calculated from each evaluated metric for all the scenarios and datasets. Because the metrics have different ranges of values, their results were normalized to the [0,1] interval.

We noticed that, on average, NSGA-II obtained the best results in HV. However, the difference from other evolutionary techniques was not high. Regarding the SP, SPEA-II considerably outperformed all of the other algorithms, followed by random search and MOCell. Finally,

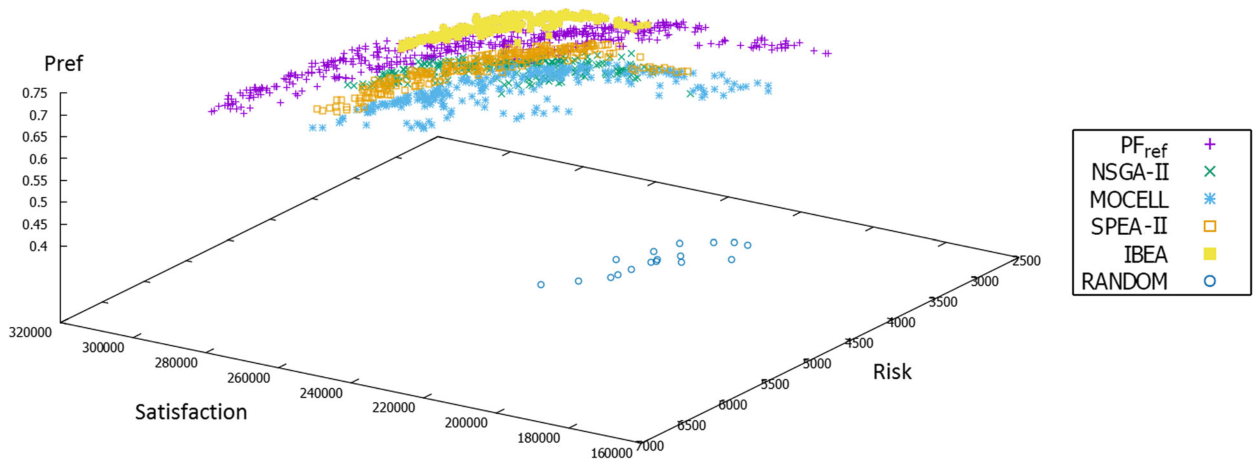


Fig. 8 The best Pareto fronts produced by each algorithm for datatset-3 and HighPrefs scenario on 30 runs, considering only HV metric

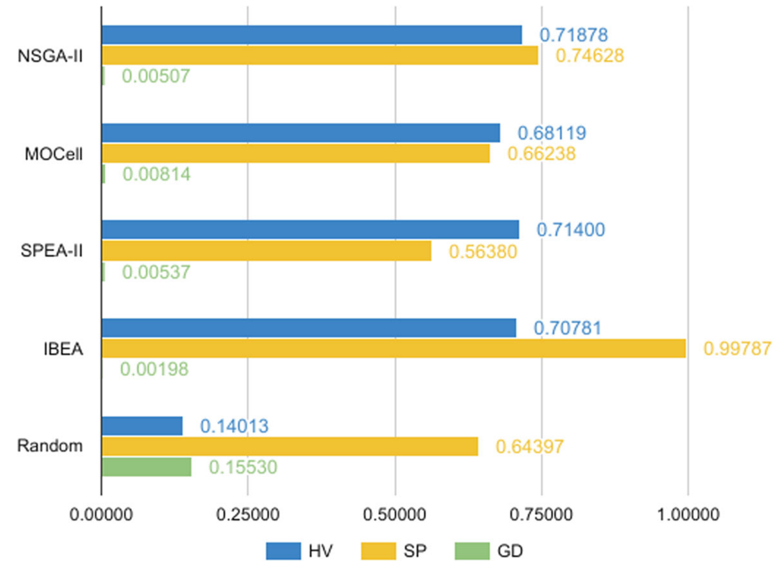


Fig. 9 Comparison between all search techniques considering metrics HV, SP, and GD

observing the GD results, the best algorithm was IBEA, followed by NSGA-II and SPEA-II.

Regarding the execution time of the meta-heuristics, NSGA-II and MOCell obtained the smallest execution times with a little difference between them. To determine which algorithm presents a better time performance, a statistical test was performed for these two algorithms shown in Table 4. Thus, NSGA-II achieved better time results for dataset-3 considering both scenarios and dataset-1 considering only the HighPrefs scenario, while MOCell was shown to be superior for dataset-2 considering both scenarios and dataset-1 in the LowPrefs scenario. However, even though NSGA-II is better in some scenarios and MOCell in others, the magnitude of the differences in time between these algorithms was small. The greatest difference among all the meta-heuristics for all instances is 151,344 ms. Thus, due to this magnitude of time, we considered the execution time irrelevant in comparison with the Pareto front quality metrics.

In summary, IBEA has proven to be a good choice because it presents qualified solutions for all the datasets, as demonstrated by GD. However, its solutions do not present a wide coverage of the search space for instances with a low number of requirements. If the diversity in Pareto front is a required aspect, NSGA-II is more recommended for scenarios composed of a small number of requirements.

To support replication, all the datasets, results, and source code are available online for public access[1]. In addition, an interactive 3D chart version of Fig. 8 is available to provide a better visualization of Pareto front for all the scenarios.

- *RQ*$_2$: Which is the subjective benefit when considering the DM's preferences as an objective to be optimized?

To evaluate such a subjective benefit, we analyzed the subjective evaluation provided by each participant to the non-preference-based and preference-based solutions from the participant-based experiment. Figure 10 depicts that 7 out of 10 participants evaluated the preference-based as solution more satisfactory than the non-preference-based solution. In addition, more than half of the participants considered the preference-based solution as "Effective" or "Very Effective," while 5 participants

Table 4 Wilcoxon and Vargha-Delaney statistical values from NSGA-II and MOCell, considering the execution times

Algorithm	Datasets	MOCell			
		LowPrefs scenario		HighPrefs scenario	
		WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$
NSGA-II	Dataset-1	1.1E−09	0.958	0.00013	0.212
	Dataset-2	4.6E−10	0.968	2.2E−09	0.950
	Dataset-3	5.1E−10	0.032	5.1E−10	0.032

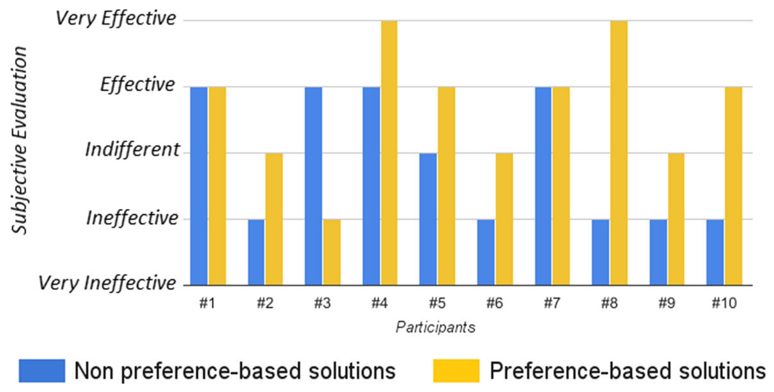


Fig. 10 Subjective evaluations defined by the participants to the non preference-based and preference-based solutions

judged the non-preference-based solution as "Ineffective" regarding their subjectivity interests. Only participant #2 negatively evaluated the preference-based solution as "Ineffective." The relation between the inclusion of the preferences and the subjective evaluation will be investigated in the next research question.

Therefore, answering RQ$_2$, such analysis suggests a considerable benefit when considering the DM's preferences as an objective of the optimization process. In addition, in Table 5, we provide more information, i.e., the number of preferences added and time spent to perform the experiment for each participant.

On average, there were 5 preferences per participant. As can be seen, participant #3 added the smallest number of preferences, only one, while participant #9 added 13 preferences, the greatest quantity. In relation to the time taken to perform the experiment, each participant took on average 23 min. Participant #7 took the greatest amount of time, 35 min, while #8 was the one who took the smallest amount of time, 5 min.

Table 5 Number of preferences and time for each participant on the experiment

Participants	Number of preferences	Time (min)
#1	5	30
#2	6	13
#3	1	12
#4	4	25
#5	3	33
#6	6	24
#7	3	35
#8	2	5
#9	13	26
#10	11	29

- RQ_3: Which is the relation between the inclusion of the preferences and the DM's subjective evaluation regarding the final solution?

The subjective evaluation is provided by each participant to the preference-based solutions, while the value of the Pref(T, S) objective is obtained by Eq. (4). As seen in Fig. 11, we trace an adjusting line through the solutions, indicating that there is a correlation between such variables. However, a metric was necessary to evaluate the intensity of this relation because some solutions were visually far apart.

To evaluate the correlation between the subjective evaluation and Pref, we used the Spearman's rank coefficient [36]. This metric is a uniform value between −1 and 1 and indicates no correlation if this value is equal to 0. In addition, the farther from 0 this value is, the more correlated the series are. The value r_s calculated for the data series is 0.74, indicating that these values are directly proportional. The two-tailed p value of 0.0144 also suggests that this correlation is statistically significant with a significance level $\alpha = 0.05$.

In Fig. 12, the subjective evaluation values are often close to the corresponding Pref ones. The cases in which they are not may suggest that there are other aspects that influence the DM's satisfaction and are not covered by the available types of preferences in this work.

Regarding the analysis presented above, we can conclude that there is a directly proportional relation between the subjective evaluation and the objective Pref. However, some samples suggest that the preference types adopted may not cover all of the DM's wishes.

Participants' feedback

At the end of the participant-based experiment, each participant was invited to answer a feedback questionnaire about the experience using the tool. We asked four

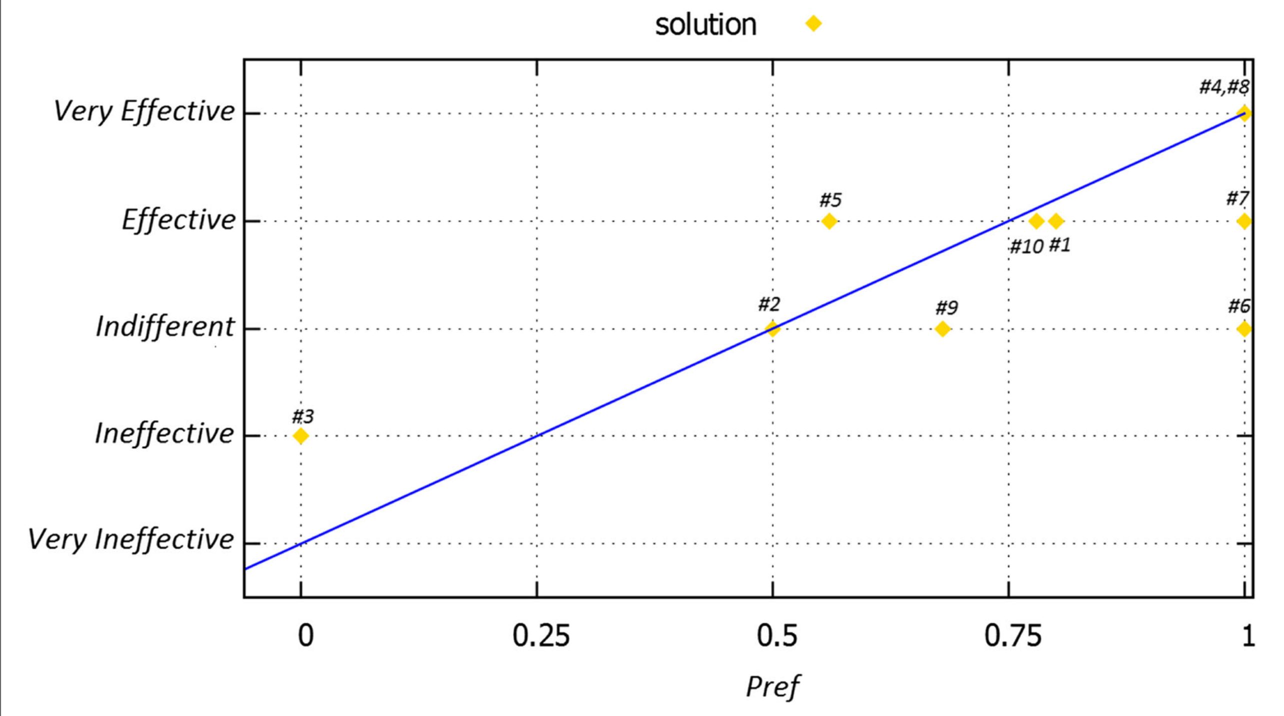


Fig. 11 Distribution of preference-based solutions based on subjective evaluation and Pref

questions covering different aspects of usability (three objective questions and one subjective question):

- Q_1: How efficient do you judge the experience at interactively assisting the tool to plan the releases to be?
- Q_2: How much easier was it to express your opinions considering the available preferences?
- Q_3: Would you use this tool in your workplace?

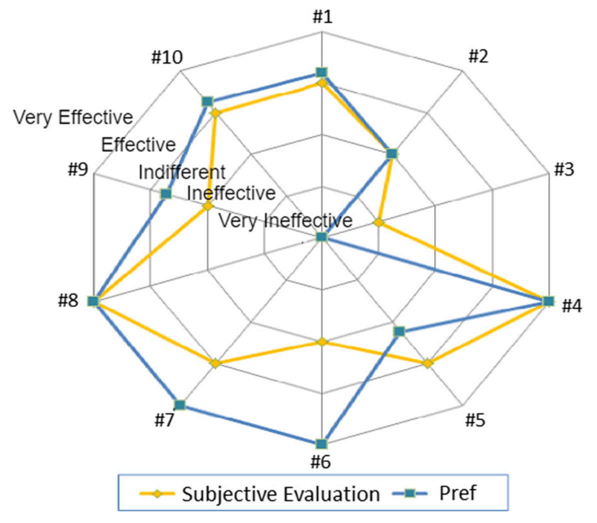


Fig. 12 Relation between subjective evaluation and the Pref value

- Q_4: What changes would you suggest regarding the tool interface?

First, for Q_1, 80% of the participants selected "Effective" or "Very effective" on a scale of "Very ineffective," "Ineffective," "Indifferent," "Effective," or "Very effective." Complementing such a result, for Q_2, 50% considered "Easy" to express the preferences on a scale of "Very hard," "Hard," "Indifferent," "Easy," and "Very easy." These answers reinforce the conclusions achieved in RQ_3 about the subjective benefit from considering the DM's preferences in the optimization process.

We used a scale from 1 ("No way") to 5 ("Certainly") for Q_3. Four participants rated with 5, another four with 3, and only one with 2. This feedback encourages the investigation of the presented tool in a real-world scenario of release planning.

Regarding the subjective question (Q_4), the answers were generally divided between improving the requirements allocation visualization and providing a better way to adjust the weight configuration for each one of the objectives.

Threats to validity

Below, we discuss the threats to the validity of our empirical evaluation, classifying them into Internal, External, Construction and Conclusion [37].

Taking into account the internal characteristics of the experiments, we have to notice that preliminary tests were carried out for defining the search technique parametrization. However, some specific settings on a given algorithm can obtain better results for some instances. Despite the fact that two datasets were based on real data, some information was necessary to be randomly generated (risk values and number of releases), that is, they do not represent a fully real-world scenario. The risk of implementing each requirement, which did not originally exist, was manually defined by a Developer and appended to the instances. The number of releases was changed from 3 to 5 and 8 for dataset-1 and dataset-2, respectively. This choice was made to increase the variation of the DM's preferences.

Regarding the participant-based experiment, the participants may have changed their behavior since they knew that they were under evaluation, corroborating the Hawthorne's effect [38]. To mitigate this problem, the participants received an explanation about the approach but not about the assumptions that were under investigation.

We believe that our empirical study has a weakness regarding the generalization of the achieved results. For instance, the datasets based on real information have few requirements, which makes it hard to conclude that the results would be similar for large-scale instances. Such a circumstance was the motivation to generate and use the artificial dataset with a large number of requirements. A similar problem is encountered in the participant-based experiment because the number of participants was not high enough to represent expressive scenarios.

Concerning the experiment construction threats, the metrics that we used to estimate client's satisfaction and overall risk are based on values that are defined a priori by the development team. These estimated values may vary as the project goes on, which requires rerunning the approach to adapt these changes. Despite this limitation, this strategy is widely used in the literature, such as [3, 39, 40]. In addition, it is known that meta-heuristics can vary their final solutions and execution time according to the instance. Unfortunately, we did not investigate time concerns. In addition, there is no longer an explanation about the evaluated metrics. Nevertheless, all of them have been widely used in the related works as well as the multi-objective optimization literature. Still, considering the participant-based experiment, the major metric used to measure the satisfaction of each participant was a subjective evaluation provided for the final solution following a scale of "Very ineffective," "Ineffective," "Indifferent," "Effective," and "Very effective." This feedback may not properly represent the DM's feeling.

Finally, the threats to the experiment conclusions' validity are mainly related to the characteristics of the algorithms that were investigated. Meta-heuristics present a stochastic behavior, and thus, distinct runs may produce different results for the same problem. Aiming at minimizing such a weakness, for each combination between datasets, scenarios and the DM's preferences for the search algorithms were executed 30 times in the automatic experiment. Given all obtained results, we conducted statistical analyses as recommended by Arcuri and Briand [41]. Analyzing the conclusions of the participant-based experiment, some of them may be affected by each participant's understanding level after receiving an explanation about the study as well as their experience on release planning using automatic tools.

Conclusions

Release planning is one of the most complex and relevant activities performed in the iterative and incremental software development process. Recently, the SBSE approaches have been discussed based on the strength of the computational intelligence with the human expertise. In other words, it allows the search process to be guided by the human's knowledge and, consequently, provide valuable solutions to the decision maker (DM). Thus, we claim the importance of providing a mechanism to capture the DM preferences in a broader scope, instead of just requiring a weight factor, for instance. Besides increasing the human's engagement, he/she will progressively gain more consciousness of how feasible the preferences are.

The evaluated multi-objective approach consists of treating the human's preferences as another objective to be maximized, as well as maximizing the overall client satisfaction and minimizing the project risk. In sum, the DM defines a set of preferences about the requirements allocation, which are stored in a preference base responsible for influencing the search process.

Therefore, we have significantly extended our previous work through the accomplishment of new experimental analysis considering both simulated and real human evaluations. The automatic experiment points out that NSGA-II obtained overall superiority in two of the three datasets investigated, positioning itself as a good search technique for smaller scenarios, while IBEA showed a better performance for large datasets, since the loss in the initial diversity of the algorithm decreases as the number of requirements increases. Regarding the participant-based experiment, it was found that two thirds of the participants evaluated the preference-based solution better than the non-preference-based one, encouraging the investigation of the presented tool in a real-world scenario of release planning. In addition, we made a novel tool for the release planning process to be able to incorporate the human preferences during the optimization process[1].

As future work, we intend to evolve our GUI to provide a more intuitive interaction, solutions visualization and preferences specification by the DM. We also intend to compare our approach with other search-based proposals, which explore the human's preferences in the optimization process.

Endnote

[1] Webpage: http://goes.uece.br/raphaelsaraiva/multi4rp/en/.

Abbreviations

DM: Decision maker; EA: Evolutionary algorithm; GD: Generational distance; HV: Hypervolume; IBEA: Indicator-based evolutionary algorithm; MOCell: Multi-objective cellular genetic algorithm; NRP: Next release problem; NSGA-II: Nondominated sorting genetic algorithm II; PF_{true}: Pareto front; PF_{ref}: Reference Pareto front; RP: Release planning; SBSE: Search-based software engineering; SP: Spread; SPEA-II: Strength Pareto evolutionary algorithm II

Acknowledgements

The authors acknowledged all the participants for their availability to the experiment.

Funding

We would like to thank the Foundation of Scientific and Technological Development Supporting from Ceará (FUNCAP) for the financial support.

Availability of data and materials

The datasets and results supporting the conclusions of this article are available at webpage: http://goes.uece.br/raphaelsaraiva/multi4rp/en/.

Authors' contributions

All authors have contributed to the methodological and experimental aspects of the research. The authors have also read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Author details

[1]State University of Ceará, Dr. Silas Munguba Avenue, 1700 Fortaleza, CE, Brazil. [2]Federal University of Ceará, Cratéus, CE, Brazil. [3]Federal University of Goiás, Goiânia, GO, Brazil.

Received: 9 January 2017 Accepted: 10 July 2017
Published online: 28 July 2017

References

1. Sommerville I (2011) Software engineering. Addison Wesley, Boston
2. Ngo-The A, Ruhe G (2008) A systematic approach for solving the wicked problem of software release planning. Soft Comput Fusion Found Methodologies Appl 12(1):95–108
3. Ruhe G, Saliu MO (2005) The art and science of software release planning. Softw IEEE 22(6):47–53
4. Harman M, McMinn P, de Souza JT, Yoo S (2012) Search based software engineering: techniques, taxonomy, tutorial. Empir Softw Eng Verification 7007:1–59
5. Miettinen K (2012) Nonlinear multiobjective optimization, Vol. 12. Springer, New York
6. Zhang Y, Finkelstein A, Harman M (2008) Search based requirements optimisation: existing work & challenges. In: Proceedings of the 14th International Working Conference, Requirements Engineering: Foundation for Software Quality (RefsQ '08). Springer, Montpellier Vol. 5025. pp 88–94
7. Marculescu B, Poulding S, Feldt R, Petersen K, Torkar R (2016) Tester interactivity makes a difference in search-based software testing: A controlled experiment. Inf Softw Technol 78:66–82
8. Araújo AA, Paixao M, Yeltsin I, Dantas A, Souza J (2016) An Architecture based on interactive optimization and machine learning applied to the next release problem. Autom Softw Eng 3(24):623–671
9. Ferreira TdN, Arajo AA, Baslio Neto AD, de Souza JT (2016) Incorporating user preferences in ant colony optimization for the next release problem. Appl Soft Comput 49(C):1283–1296
10. Tonella P, Susi A, Palma F (2013) Interactive requirements prioritization using a genetic algorithm. Inf Softw Technol 55(1):173–187
11. Dantas A, Yeltsin I, Araújo AA, Souza J (2015) Interactive software release planning with preferences base. In: International Symposium on Search Based Software Engineering. Springer, Cham. pp 341–346
12. Saraiva R, Araújo AA, Dantas A, Souza J (2016) Uma Abordagem Multiobjetivo baseada em Otimização Interativa para o Planejamento de Releases. In: VII Workshop em Engenharia de Software Baseada em Busca. CBSoft, Maringá
13. Wierzbicki AP (1980) The use of reference objectives in multiobjective optimization. In: Multiple Criteria Decision Making Theory and Application. Springer, West Germany Vol. 117. pp 468–486
14. Harman M, Jones BF (2001) Search-based software engineering. Inf Softw Technol 43(14):833–839
15. Harman Mark (2007) The current state and future of search based software engineering. In: 2007 Future of Software Engineering. IEEE Computer Society, Washington. pp 342–357
16. Pitangueira AM, Maciel RSP, de Oliveira Barros M (2015) Software requirements selection and prioritization using sbse approaches: a systematic review and mapping of the literature. J Syst Softw 103:267–280
17. Räihä O (2010) A survey on search-based software design. Comput Sci Rev 4(4):203–249
18. McMinn P (2004) Search-based software test data generation: a survey. Softw Testing Verification Reliab 14(2):105–156
19. Harman M, Mansouri SA, 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
20. Back T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press, New York
21. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans Evol Comput 6(2):182–197
22. Nebro AJ, Durillo JJ, Luna F, Dorronsoro B, Alba E (2009) Mocell: a cellular genetic algorithm for multiobjective optimization. Int J Intell Syst 24(7):726–746
23. Zitzler E, Künzli S (2004) Indicator-based selection in multiobjective search. In: International Conference on Parallel Problem Solving from Nature. Springer, Berlin. pp 832–842
24. Zitzler E, Laumanns M, Thiele L, et al (2001) SPEA2: Improving the strength Pareto evolutionary algorithm. In: Eurogen. ETH-TIK, Zürich Vol. 3242(103). pp 95–100
25. Baker P, Harman M, Steinhofel K, Skaliotis A (2006) Search based approaches to component selection and prioritization for the next release problem. In: 2006 22nd IEEE International Conference on Software Maintenance. IEEE, Philadelphia. pp 176–185
26. Boehm BW (1991) Software risk management: principles and practices. IEEE Softw 8(1):32–41
27. Brasil MMA, da Silva TGN, de Freitas FG, de Souza JT, Cortés MI (2012) A Multiobjective Optimization Approach to the Software Release Planning with Undefined Number of Releases and Interdependent Requirements. In: Enterprise Information Systems: 13th International Conference, ICEIS 2011, Revised Selected Papers. Springer, Beijing Vol. 102. p 300
28. Deb K (2001) Multi-objective optimization using evolutionary algorithms, Vol. 16. John Wiley & Sons, United Kingdom
29. Thiele L, Miettinen K, Korhonen PJ, Molina J (2009) A preference-based evolutionary algorithm for multi-objective optimization. Evol Comput 17(3):411–436
30. Kitchenham BA, Pfleeger SL, Pickard LM, Jones PW, Hoaglin DC, Emam KE, Rosenberg J (2002) Preliminary guidelines for empirical research in software engineering. IEEE Trans Softw Eng 28(8):721–734
31. Yin RK (2003) Case study research: Design and methods. In: Applied Social Research Methods Series. Sage Publications, California Vol. 5
32. Karim MR, Ruhe G (2014) Bi-objective genetic search for release planning in support of themes. In: SSBSE'14. Springer, Cham. pp 123–137
33. Arcuri A, Briand L (2014) A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. Softw Testing Verification Reliab 24(3):219–250
34. Zhang Y (2010) Multi-objective search-based requirements selection and optimisation. University of London, London

35. ZITZLER E (1999) Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Swiss Fed Inst Technol (ETH) Zurich. TIK-Schriftenr 30
36. Siegel S (1956) Nonparametric statistics for the behavioral sciences. 1st edn. McGraw-Hill, Columbus
37. de Oliveira Barros M, Dias-Neto AC (2011) 0006/2011-Threats to Validity in Search-based Software Engineering Empirical Studies. RelaTe-DIA 5(1):1–12
38. McCambridge J, Witton J, Elbourne DR (2014) Systematic review of the hawthorne effect: new concepts are needed to study research participation effects. J Clin Epidemiol 67(3):267–277
39. Colares F, Souza J, Carmo R, Pádua C, Mateus GR (2009) A new approach to the software release planning. In: Software Engineering, 2009. SBES'09. XXIII Brazilian Symposium on. IEEE, New York. pp 207–215
40. Greer D, Ruhe G (2004) Software release planning: an evolutionary and iterative approach. Inf Softw Technol 46(4):243–253
41. Arcuri A, Briand L (2011) A practical guide for using statistical tests to assess randomized algorithms in software engineering. In: 2011 33rd International Conference on Software Engineering (ICSE). IEEE, New York. pp 1–10