Estimation of Distribution Algorithms with Solution Subset Selection for the Next Release Problem.

Víctor Pérez-Piqueras, Pablo Bermejo López, and José A. Gámez

This is the Accepted Manuscript version of the paper published online in the Logic Journal of the IGPL in October 2024 and DOI: https://doi.org/10.1093/jigpal/jzae052

Abstract The Next Release Problem (NRP) is a combinatorial optimization problem that aims to find a subset of software requirements to be delivered in the next software release, which maximize the satisfaction of a list of clients and minimize the effort required by developers to implement them. Previous studies have applied various metaheuristics, mostly genetic algorithms. Estimation of Distribution Algorithms (EDA), based on probabilistic modelling, have been proved to obtain good results in problems where genetic algorithms struggle. In this paper we propose to adapt three EDAs to tackle the multi-objective NRP in a fast and effective way. Results show that EDAs can be applicable to solve the NRP with rather good quality of solutions. Furthermore, we prove that their execution time can be significantly reduced using a per-iteration solution subset selection method while maintaining the overall quality of the solutions obtained, and they perform the best when limiting the search time as in an interactive tool that requires fast responsiveness. The experimental framework, code and datasets have been made public in a code repository.

1 Introduction

Successfully managing software releases is one of the major challenges in Software Engineering. As the product goal grows and project scope gets expanded, the difficulty of delivering valuable features to clients increases substantially. Client interests are usually defined in terms of software requirements, which are then prioritized and selected for the next software release. However, when there are many clients, it is frequent that their concerns are different or even opposed, leading to difficult choices to balance overall clients satisfaction. Furthermore, complexity of requirements has to be taken into account, in order to not surpass developers expected capacity for the next release. This problem, named Next Release Problem (NRP), pursues finding a set of requirements for a release that satisfy clients as much as possible and optimize development efforts. This is a strong NP-hard problem, as proved in [2], and is usually solved manually by experts judgement. Given

Víctor Pérez-Piqueras · Pablo Bermejo López · José A. Gámez
Department of Computing Systems, Intelligent Systems and Data Mining Laboratory (I3A), Universidad de Castilla-La Mancha, Albacete, 02071, Spain, e-mail: {victor.perezpiqueras,pablo.bermejo,jose.gamez}@uclm.es

that solving the NRP is critical for a software project success, and that it has to be solved every time a release is planned, it is an interesting candidate to be automated by means of optimization methods. Most previous works tackle the NRP by means of evolutionary techniques, being genetic algorithms (GAs) the preferred choices. However, Estimation of Distribution Algorithms (EDA), a subfamily of evolutionary algorithms, are known to guide the search in a more informed way than other population-based algorithms such as GAs, and they have previously been proved to perform better than local search in selection contexts such as feature selection [5].

In this work, we evaluate the applicability of EDAs, which have only been proposed to solve the NRP in a single previous study [29]. We have adapted three different EDAs to solve the multi-objective version of the NRP (see Section 2.2). We compared them against widely used GAs by following fair comparison state-of-art recommendations, and we optimize our proposed algorithms so that their search may finish in a short period of time while obtaining a high performance. Specifically, we pose the following research questions (RQ):

- **RQ1**: Do EDAs outperform state-of-the-art multi-objective NRP evolutionary algorithms?
- **RQ2**: Can we speed up the search by reducing the non-dominated solutions kept at each iteration by means of Solution Subset Selection without decreasing the quality indicators?
- **RQ3**: Which algorithm performs the best when setting a search time limit, as needed in a real-time application for the decision maker?

We aim to provide insights into the strengths and limitations of these algorithms, and their potential for real-world applications.

This article is structured as follows: Section 2 presents a summary of previous works, the problem at hand and state-of-art recommendations. In Section 3, EDAs and our three proposed algorithms are described. The experimental evaluation, along with the algorithms, datasets and methodology applied are listed in Section 4. Section 5 presents and discusses the results of the experimentation. Lastly, we conclude this work in Section 6.

2 Related work

2.1 Next Release Problem

The solution of the Next Release Problem (NRP) is one of the applications in the field of Search-Based Software Engineering (SBSE) [17], where Software Engineering related problems are tackled by means of search-based optimization algorithms [35].

The NRP was firstly formulated by Bagnall et al. [3]. In its definition, a subset of requirements has to be selected, having as goal meeting the clients needs, minimizing development effort and maximizing clients satisfaction. They applied a variety of metaheuristic techniques, such as simulated annealing, hill climbing and GRASP, but combining the objectives of the problem into a single-objective function. In a later study, Greer and Ruhe [15] examined the generation of feasible assignments of requirements to increments, considering various resource constraints and perspectives of stakeholders. They utilized GAs as the chosen optimization technique to solve the NRP. Subsequently, Baker et al. [4] showcased the application of metaheuristic techniques to real-world NRP, surpassing expert judgment. Their study employed simulated annealing and greedy algorithms.

In another research, del Sagrado et al. [11] applied ACO (Ant Colony Optimization) to solve the NRP. All of these approaches followed a single-objective formulation of the problem.

The current formulation of the NRP, proposed by Zhang et al. [36], consists in finding a candidate subset of requirements to implement in the next release of a software project. The set of n candidate requirements is denoted as $R = \{r_1, r_2, \ldots, r_n\}$. These requirements are proposed to a set $C = \{c_1, c_2, \ldots, c_m\}$ of m clients. Clients are not equally important, and a set of weights $W = \{w_1, w_2, \ldots, w_m\}$ defines the numeric importance of each client (higher means client more relevant). Each client provides an importance value v_{ij} to each requirement, that denotes how prioritary or valuable the requirement is. These importances are hold in a $m \times n$ matrix. Moreover, each requirement has an associated cost or effort of implementation defined by a set $E = \{e_1, e_2, \ldots, e_n\}$ of requirement efforts, in which each e_i belongs to a requirement r_i [18]. Finally, the total satisfaction of implementing each requirement is calculated and stored in a set $S = \{s_1, s_2, \ldots, s_n\}$, in which each requirement's satisfaction is measured as the weighted sum of all importance values from all clients for that requirement: $s_j = \sum_{i=1}^{m} w_i \times v_{ij}$.

Although EDA approaches have been applied to SBSE problems, only one work [29] has used EDAs to solve the NRP without modelling dependencies, to the authors' knowledge. From the most recent reviews, in Ramírez et al. [31] only an EDA application to software testing [32] is referenced; and in Gupta et al. [16] and Alba et al. [1] EDA approaches are not mentioned or matched to any solution of the NRP.

2.2 Multi-Objective Next Release Problem

A multi-objective optimization (MOO) version of the NRP is a suitable approach. In MOO, the problem objectives are not combined but tackled separately. This implies that instead of obtaining a single solution as output of the search, a Pareto front of non-dominated solutions [8] is returned. This Pareto front is formed by a set of solutions that are non-dominated by any other of the set. This new formulation applied to NRP is called Multi-Objective Next Release Problem (MONRP) [36]. In this case, a solution $x = [e_x, s_x]$ that holds the two objectives of the problem, is said to dominate other solution $y = [e_y, s_y]$ only if e_y and s_y are worse than e_x and s_x, respectively. Conversely, the solutions are non-dominated among them as long as neither of them dominates the other. Thus, the MONRP consists of finding a subset X of R, containing the requirements to be implemented for the next software release that maximize clients satisfaction and minimize development efforts. The MONRP objectives are the following:

$$\begin{aligned} Maximize\ S(X) &= \sum_{j \in X} s_j \\ Minimize\ E(X) &= \sum_{j \in X} e_j \end{aligned} \tag{1}$$

Several works have solved the MONRP, being the first one the proposal of Zhang et al. [36]. In their work, they tackled each objective separately, exploring the non-dominated solutions (NDS). Finkelstein et al. [14] also used a multi-objective approach considering different measures of fairness,

and applying evolutionary algorithms, such as ParetoGA and NSGA-II [10] to solve the MONRP. Other works that continued to explore evolutionary algorithms to solve the MONRP are those of Durillo et al. [13, 12]. They proposed two GAs, NSGA-II and MOCell (Multi-Objective Cellular genetic algorithm), as well as an evolutionary procedure, PAES (Pareto Archived Evolution Strategy). Jiang et al. [23] presented a different approach using an ACO (Ant Colony Optimization) algorithm. Charan Kumari et al. [6] proposed the use of a hybrid differential evolution strategy [30]. Notably, none of these works considered the interactions between requirements. Subsequently, studies started considering and designing requirement interactions. Sagrado et al. [33] and Souza et al. [34] addressed the MONRP with interactions by applying ACO. Finally, Chaves-González et al. [7] proposed the use of a Multi-Objective Ant Bee Colony (MOABC), a swarm intelligence evolutionary-based algorithm. Both Sagrado et al. [33] and Chaves-González et al. [7] compared their proposals against genetic (NSGA-II) and greedy (GRASP) algorithms.

In the literature, there are studies that tackled the MONRP without modelling problem constraints [14, 23, 13, 12, 6] and studies that considered them [34, 33, 7]. These constraints include not only requirement interactions, but also a total effort limitation or Budget (B). However, this Budget limit is not commonly used in the MOO version of the problem, because it just filters the most expensive solutions, which can be easily done in a visual manner by the decision maker. In our proposal, we did not model requirement dependencies, neither the constraint of total effort.

2.3 Fair comparison

In the studies previously referenced, there is a wide variety of metrics used to measure the quality of the Pareto front obtained by an algorithm: Hypervolume (HV), Δ-Spread, Spacing, etc. However, not all quality metrics are appropriate to evaluate MOO Pareto fronts. Li et al. [26] provide a methodological guidance to choose the appropriate quality indicators to evaluate the quality of a Pareto front. Pareto compliant [38] metrics (those that for a solution X that dominates other solution Y, always provide a metric evaluation for X better than the metric evaluation obtained by Y) are recommended to tackle MOO problems such as the MONRP. A subset of recommended Pareto compliant metrics that we have chosen is the following:

- Hypervolume (HV) [37]. Is the most widely used metric to assess Pareto fronts in multi-objective problems in SBSE. It denotes the space covered by the set of non-dominated solutions. In order to compute it, a reference point is needed, and it should be the same for all algorithms under comparison.
- Δ-Spread [9]. It measures the dispersion of the solutions in the Pareto front. Thus, the smaller the Δ-Spread value is for a set of non-dominated solutions, the better (more uniform). Interpretation of Δ-Spread is only fair on bi-objective problems, which is the case for the canonical MONRP with Effort and Satisfaction objectives. Thus, although it cannot be said it always behaves as Pareto compliant, it is reliable when computed on bi-objective problems.

- Unique Non Dominated Front Ratio (UNFR) [26]. It measures the ratio of solution points in the P_{ref}[1] which belong to the solution set of the evaluated algorithm. That is, it measures the contribution (from 0 to 1) of an algorithm to the P_{ref}.

2.4 Solution Subset Selection

Multi-objective algorithms update a NDS (commonly named $NDS_{archive}$) set at each iteration of their execution and return it when finished. However, the size of the $NDS_{archive}$ set directly affects quality indicators of the resulting Pareto front, such as HV. Furthermore, algorithms under comparison can be run under different population sizes or number of iterations, which can lead to $NDS_{archive}$ sets of unequal size that, ultimately, will provoke biased comparison of quality indicators. To solve this issue, [22] recommends that each algorithm returns a subset of the final $NDS_{archive}$ set found. Usually, this is done by running a solution subset selection (SSS) by means of a greedy forward search based on HV at the end of the execution. Besides, in [20] it is stated that the size of the subset has only minor effects in the results of a comparison, so a fixed SSS size can be set for all algorithms under comparison. Lastly, reducing the returned Pareto fronts, that usually contain hundreds of solutions, can help the decision maker by simplifying the process of navigating them [21].

3 Estimation of Distribution Algorithms

EDAs are evolutionary algorithms based on probabilistic modelling and were designed as an alternative to GAs [25]. As GAs, EDAs are population-based algorithms, however instead of relying upon the goodness of genetic operators, EDAs apply a more normative approach. This approach consists of three steps: (i) select a subset of promising individuals from the current population; (ii) from this subset, learn a probability distribution of variables to optimize; (iii) sample a new population using the estimated probability distribution. As no crossover nor mutation operators are needed, the number of hyperparameters decreases, thus simplifying algorithm configuration. The pseudocode of a multi-objective version of a generic EDA adapted to solve the MONRP is detailed in Algorithm 1.

The complexity of an EDA is related to the degree of explicit interrelations (dependencies) it allows. Thus, we can find univariate EDAs, such as UMDA and PBIL, with no explicit dependencies modelling, that implicitly catch the interrelations between variables by means of an evaluation function (as in GAs); and multivariate EDAs such as the bivariate MIMIC, where the dependencies among the variables are explicitly tackled by constructing a graphical structure that codifies the dependencies in the probabilistic model (e.g. a Bayesian network [24]).

UMDA_{NRP}

In Univariate Marginal Distribution Algorithm (UMDA) [25, Ch. 4] the JPD is factorized as the product of marginal distributions:

[1] A Pareto reference (P_{ref}) is the set of non-dominated solutions obtained by merging all Pareto fronts returned by the algorithms being evaluated after their execution

Algorithm 1 EDA_{NRP}

procedure $EDA_{NRP}(maxGenerations, SS_{size})$
  $nds \leftarrow \emptyset$ ▷ initialize the $NDS_{archive}$
  $P \leftarrow$ generateRandomPopulation() ▷ generate first population randomly
  for $i = 0$ **to** $maxGenerations$ **do**
    $individuals \leftarrow$ selectIndividuals(P) ▷ select non-dominated individuals
    $probModel \leftarrow$ learnProbModel($individuals$) ▷ learn the probability model
    $P \leftarrow$ sampleNewPopulation($probModel$) ▷ sample individuals based on distribution
    $nds \leftarrow$ updateNDS(P, nds) ▷ update set of non-dominated solutions
  end for
  $nds \leftarrow$ solutionSubsetSelection(nds, SS_{size}) ▷ find subset that maximizes HV
  return nds
end procedure

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i)$$

The $UMDA_{NRP}$ follows a structure similar to that of Algorithm 1. It starts creating a random population and, at each generation, it selects the non-dominated individuals of the population, learns the probability model p from them (using maximum likelihood estimation with or without Laplace smoothing), and samples a new population using p (notice that as no dependency is considered, each variable/requirement is sampled from its marginal probability distribution). Finally, new individuals are evaluated, and the $NDS_{archive}$ is updated with the new non-dominated individuals found. After execution, it returns a subset of the final $NDS_{archive}$. This subset is the result of applying an SSS process with a pre-fixed SS_{size} of solutions.

PBIL$_{NRP}$

Population-Based Incremental Learning (PBIL) [25, Ch. 4] combines the mechanisms of a generational GA with simple competitive learning. Differently to UMDA, in which populations are transformed into a probability model whose only purpose is to sample new populations, PBIL algorithm attempts to create a probability model which can be considered a prototype for high evaluation vectors for the function space being explored. Furthermore, UMDA learns a new model at each iteration without keeping nor evolving any knowledge from the past iterations, while PBIL refines one model through all the generations. In a manner similar to the training of a competitive learning network, the values in the probability model are gradually shifted towards representing those in high evaluation vectors. Thus, the behaviour of PBIL diverges from the default EDA (Algorithm 1) by initializing a probability vector instead of a population. Then, rather than selecting individuals, learning a probability model and sampling a new population, PBIL first samples individuals from its probability model and then selects a random individual from the $NDS_{archive}$ and use it to update the probability vector in two steps:
(1) $p(x_i) = p(x_i) \cdot (1.0 - LR) + bestIndividual_i \cdot LR$, being i the i^{th} variable position, and LR the learning rate hyperparameter, ranging from 0 to 1.
(2) If $(Prob_{rand} < Prob_{mut})$: $p(x_i) = p(x_i) \cdot (1.0 - MS) + r \cdot MS$, r being a random number

$\in \{0, 1\}$ and MS the mutation shift hyperparameter, ranged from 0 to 1.

$\text{MIMIC}_{\text{NRP}}$

Mutual Information Maximization for Input Clustering (MIMIC) [25, Ch. 4] is an EDA capable of explicitly modelling some of the interdependencies among variables. In particular, in MIMIC a chain is used as graphical model, so all the variables (but the root) directly depend on exactly one variable. Given a permutation (or chain) $\pi = (i_1, i_2, ..., i_n)$, the model is factorized as:

$$P_\pi(x) = P(x_{\pi_1} \cdot \prod_{i=2}^{n} P(x_{\pi_i} | x_{\pi_{i-1}}),$$

where $p(x_{i_n})$ and $p(x_{i_j} | x_{i_{j+1}})$, $j = 1, ..., n-1$ are, respectively, estimated by the marginal and conditional relative frequencies of the corresponding variables within the subset of selected individuals at each generation. MLE with or without Laplace smoothing is used for parameter estimation from the subset of selected individuals at each generation. The objective of MIMIC is to find a permutation π such that its $p_\pi(x)$ minimizes the Kullback-Leibler information divergence (expressed using the Shannon entropy) between the probability function $p(x)$ and the probability functions $p_\pi(x)$ of the class $P_\pi(x)$. Its learning phase behaves as a greedy algorithm that firstly selects x_{i_n} as the variable with the smallest estimated entropy. Then, it consecutively selects a variable from the set of not chosen variables whose average conditional entropy (with respect to the previous variable) is the smallest.

$MIMIC_{NRP}$'s method also fits into the general EDA-based scheme described in Algorithm 1, using the explained approach for learning. In the sampling phase, probabilistic logic sampling (PLS) [19] is used to respect the chain order during the simulation.

Solution Subset Selection

At the end of each iteration in Algorithm 1, the NDS found is used to update the general $NDS_{archive}$. This update process tends to be very slow as the population size increases, and it is a problem which does not exist in other optimized algorithms such as NSGA-II, which contains an embedded process to perform a fast filtering of non-dominated solutions.
When the search loop finishes in Algorithm 1, we apply a SSS process (*solutionSubsetSelection*), which consists in selecting a fixed-size subset from the $NDS_{archive}$ which maximizes its HV. This is done to perform fair comparison among algorithms (Section 2.3). However, it may be used also to reduce the current $NDS_{archive}$ at each iteration, hopefully not decreasing the performance of quality indicators, what would lead to faster search time. Thus, this addition of SSS at the end of each iteration is also proposed and evaluated, and consequently used to answer RQ2.

4 Experimental evaluation

In this section we present the algorithms, datasets and experimental approach. To ensure reproducibility, the source code for the algorithms, implemented in Python 3.8.8, along with the experi-

mentation setup and datasets used are available at the following repository: https://github.com/uclm-simd/monrp/tree/igpl23.

4.1 Algorithms under comparison

Our experimentation framework includes 6 algorithms, for which four metrics have been computed: the three quality indicators (HV, Δ-Spread and UNFR) introduced in Section 2.3 and execution time (seconds).

- **EDA$_{\mathbf{NRP}}$**. Instanced with the three multi-objective EDAs proposed in Section 3: $UMDA_{NRP}$, $PBIL_{NRP}$ and $MIMIC_{NRP}$.

- **Single-Objective GA**. It combines the two objective functions of the MONRP into a single objective function by using a weighted aggregation. Then, it updates its $NDS_{archive}$ with new individuals after each generation.

- **NSGA-II.** The Non-dominated Sorting Genetic Algorithm-II [10] is a state-of-the-art multi-objective GA. It uses elitism and ranks each individual based on the level of non-dominance.

- **AGE-MOEA-II.** The Adaptive Geometry Estimation based MOEA II [28] is a recent multi-objective GA that uses a novel method to model the non-dominated front. It has been adapted to keep the global $NDS_{archive}$ and perform the SSS stage, making it possible to use it in our comparison framework.

The ranges of parameters used in the experimentation for each algorithm are described in Section 4.2, for further detail.

Algorithm performance has been measured using two widely used public datasets (P1 and P2), taken from previous NRP studies [15, 33], and another four created synthetically (S1, S2, S3 and S4) due to the lack of datasets with high number of requirements, so that algorithms can be tested in significantly larger instances. The whole evaluation corpus is shown in Table 1. Each dataset contains a set of proposed requirements, defined by a vector of efforts, one effort value for each requirement. Clients are also included, defined by a vector of weights. The importance that each client gives to each requirement is also contained in the dataset, by means of a matrix of values, in which each value represents the importance of a requirement for a client. Dataset P1 [15] includes 20 requirements and 5 clients. Dataset P2 [33] includes 100 requirements and 5 clients. Datasets S1-S4 range from 40 to 200 requirements, and reach up to 150 clients; thus, S3 and S4 are the most complex datasets in our evaluation corpus.

Table 1: Two publicly available and four synthetic datasets.

Dataset	P1	P2	S1	S2	S3	S4
#Clients	5	5	15	50	100	150
#Requirements	20	100	40	80	140	200

4.2 Methodology

Each algorithm is run 30 times. Based on guidance by [20] and [22], after each execution a set **S** of size $SS_{size} = 10$ solutions is selected by means of a greedy HV-based forward SSS run over the $NDS_{archive}$ returned by the algorithm. Then, the mean HV (reference point = (1.1,1.1)) and Δ-Spread metrics are computed from **S**. After all algorithms have finished their executions, a reference Pareto (P_{ref}) is constructed selecting the NDS from a pool made of all the **S** sets returned by all executions from all algorithms.

In order to answer RQ1 and RQ3, and also following guidance by [22], we run all algorithms configured through a grid search of hyperparameter values. Then, the best configuration found for each algorithm (in terms of HV) is the one used when comparing their performance. All algorithms have two hyperparameters in common, and these are the ranges of values set for them: $Population\ Size = \{100, 200, 500, 700, 1000\}$ and $\#Iterations = \{50, 100, 200, 300, 400\}$. GA and NSGA-II were also tested with $MutationProbability = \{0.1, 0.3\}$.
All the algorithms find their best results when configured with the maximum $Population\ Size$, except for AGE-MOEA-II which converges with $Population\ Size = 700$. Regarding the number of iterations, both GA and $UMDA_{NRP}$ are very fast to converge, only needing 50 and 100 iterations, respectively. On the other hand, $PBIL_{NRP}$, $MIMIC_{NRP}$, NSGA-II and AGE-MOEA-II use the largest number of iterations (400). GA and NSGA-II obtain better results with $MutationProbability = 0.3$.

RQ2 aims to find if search time can be decreased by reducing the size of the $NDS_{archive}$ maintained by the algorithm, without decreasing the algorithm performance. Thus, we add the same SSS process (previously only run at the end of each execution) at the end of each iteration. Then, we measure the change caused in the four metrics. The SSS cannot be applied to NSGA-II, which already runs its own embedded fast non-dominated sorting strategy, not compatible with applying SSS at the end of each iteration.

Lastly, in order to answer RQ3, we set a time threshold for the search of each algorithm. Then, we identify the best configuration and compare algorithms again using only the results from executions that do not last over that time. Following this procedure, we aim to find which algorithm would best fit in a real scenario where a decision maker uses an interactive computed-aided project management tool, with a maximum search time of 150 seconds.

5 Results and analysis

In this section, results are shown and ordered in the appropriate flow to answer the three RQs stated in Section 1.

5.1 RQ1: Do EDAs outperform state-of-the-art MONRP evolutionary algorithms?

Given the best configuration found for each algorithm, as explained in Section 4.2, we compared the three EDA proposals and the three GAs, using quality indicators that are Pareto compliant (HV, UNFR and Δ-Spread), and execution time. We also logged the number of non-dominated solutions found in the final $NDS_{archive}$ returned by each algorithm. We show the results for the largest dataset, S4, in Table 2, and we summarize the conclusions obtained, which generalize for the rest of datasets. All tables and plots for all datasets are available in the same public repository as the source code.

Table 2: Results for dataset S4 using the best configurations.

Method	**HV**	**UNFR**	**Δ−Spread**	**Time(s)**	**#NDS**
GA	0.6475	0.0011	0.6928	3531.5	181.3
$UMDA_{NRP}$	0.7635	0.0012	0.5717	693.9	359.8
$PBIL_{NRP}$	0.4819	0.0008	0.5985	635.9	70.1
$MIMIC_{NRP}$	0.7827	0.0019	0.6050	9681.6	614.9
$NSGA-II$	0.8231	0.0079	0.6179	75704.9	1000.0
$AGE-MOEA-II$	0.7597	0.0038	0.6065	5110.9	922.7

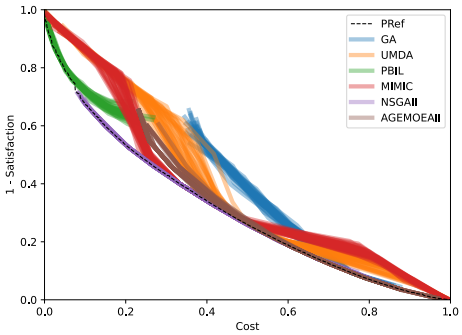

Fig. 1: Pareto Front of each algorithm and Pareto Reference, dataset S4.

In Table 2 we can see results that are representative of the general case: NSGA-II outperforms the other algorithms in terms of HV and UNFR. However, its execution time is extremely long. The $MIMIC_{NRP}$ proposal is the one which is closer to it. Conversely, $UMDA_{NRP}$ and $PBIL_{NRP}$ outperform in terms of Δ-Spread. Figure 1 plots, for each algorithm, the 10 solutions selected in

the SSS process, for each of the 30 executions. That is, 300 solutions are plotted and, thus, the general area covered by each one can be easily noticed. Although $MIMIC_{NRP}$ and $UMDA_{NRP}$ are capable of finding solutions as balanced as those found by the NSGA-II, the former can also find good extreme solutions.
Regarding the results considering all datasets: in terms of HV and UNFR, NSGA-II outperforms in 5/6 datasets. On the contrary, $UMDA_{NRP}$, $PBIL_{NRP}$ and $MIMIC_{NRP}$ obtain the best Δ-Spread results in 5/6 datasets. In terms of execution time, NSGA-II is, by far, the most time-consuming algorithm.
As a conclusion for RQ1, NSGA-II performs the best in two out of the three metrics used to evaluate algorithms' performance, given that algorithms are allowed to run during all the time they may need to converge.

5.2 RQ2: Can we speed up the search by reducing the non-dominated solutions kept at each iteration by means of SSS without decreasing the quality indicators?

There are two main tasks at each iteration of all the algorithms under evaluation: learning + sampling and updating the $NDS_{archive}$ with the new NDS set found in the iteration. We computed the mean time used in each of these tasks. Results are similar for all datasets, and we provide the corresponding to dataset S4 in Figure 2, showing the mean times obtained by all possible hyperparameters configurations of the algorithms (see Section 4.2). Clearly, NGSA-II has a reduced percentage of time used in its embedded fast filtering and ordering process at each iteration; however, the other algorithms spend a great portion of time to update the $NDS_{archive}$: while NSGA-II spends 20% of its time, $UMDA_{NRP}$ and $PBIL_{NRP}$ use more than 80% of the total time, $MIMIC_{NRP}$ takes 30%, GA up to 96%, and AGE-MOEA-II uses near 80% of its search time to filter non-dominated solutions. Thus, although GA, AGE-MOEA-II and the EDA proposals are already faster to converge than NSGA-II, it is clear that they can be even faster by reducing the time to update the $NDS_{archive}$ at each iteration. In order to do so, we decided to apply the SSS process not only at the end of the execution, but also as a last step at the end of each execution. Hence, the size of $NDS_{archive}$ would be kept constant ($SS_{size} = 10$ in our case) and thus its update should be significantly faster.

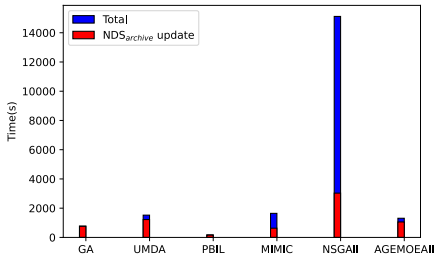

Fig. 2: Mean time in algorithms execution used to update the $NDS_{archive}$.

Time results, shown in Table 3(a), for GA, AGE-MOE-II and the three EDA proposals prove that a great reduction of execution time when solving S4 makes it worth to insert the SSS process at the end of each iteration. All time reductions are statistically significant (Wilcoxon paired test), in all datasets.
Thus, RQ2 has been partially answered. Evolutionary algorithms, such as GA, AGE-MOEA-II and our EDA_{NRP} proposals, can be sped up by reducing the size of their $NDS_{archive}$ once per iteration. With regards to the second part of RQ2, Table 3(b, c, d) shows the change experimented by the algorithms in the three quality indicators. It can be observed that HV increases in the case of $PBIL_{NRP}$. In the other 4 algorithms, the decrease of HV is so small that it can be said to remain constant, while having a great reduction in execution time. UNFR increases in 4 out of the 5 algorithms. The uniformity in the Δ-Spread of solutions found is the only quality indicator which is slightly worse. In fact, in other datasets and algorithms, this change is not significant. Figure 3 shows the box plot for the four mentioned metrics, where it can be seen that not only mean values are improved, but also variability in results is commonly reduced. Thus, we conclude that applying the SSS procedure at each iteration is a worth decision, since HV (the most representative and used quality indicator of NDS) remains constant or even increases in some cases, while drastically reducing the execution time, improving UNFR and slightly worsening Δ-Spread.
Figures and tables for the rest of our experimentation corpus provide the same conclusions, and are available in the same public repository as our source code.

Table 3: Change in mean metrics for S4 when applying SSS at each iteration.

Time(s)					**HV**			
Algorithm	Before	After	%Change	p-value	Before	After	%Change	p-value
GA	775.5	109.5	-78.19%	5.96E-08	.631	.630	-0.07%	2.98E-07
$UMDA_{NRP}$	1523.2	695.9	-52.54%	5.96E-08	.753	.751	-0.26%	5.96E-08
$PBIL_{NRP}$	173.6	61.7	-57.08%	5.96E-08	.421	.472	12.14%	5.96E-08
$MIMIC_{NRP}$	1645.5	1158.6	-25.99%	5.96E-08	.762	.759	-0.29%	5.96E-08
$AGE-MOEA-II$	1310.6	510.4	-49.63%	5.96E-08	.715	.715	-0.05%	6.56E-06
(a)					**(b)**			
UNFR					**Δ-Spread**			
GA	.0007	.0008	16.94%	2.91E-04	0.6922	0.6953	0.46%	5.58E-03
$UMDA_{NRP}$	.0007	.0009	40.61%	2.06E-05	0.5709	0.6027	5.59%	5.96E-08
$PBIL_{NRP}$	.0002	.0001	-46.31%	1.81E-03	0.5797	0.5922	2.20%	6.31E-04
$MIMIC_{NRP}$	.0004	.0005	3.03%	9.69E-02	0.5997	0.6251	4.22%	8.34E-07
$AGE-MOEA-II$	.0019	.0019	3.3%	3.94E-01	0.6236	0.6141	-1.45%	3.67E-02
(c)					**(d)**			

5.3 RQ3: Which algorithm performs the best when setting a search time limit, as needed in a real time application for the decision maker?

The final aim in solving the MONRP is to help the decision maker to create a plan for a given set of requirements. Commonly, this is performed in a support or CARE (Computer Aided Requirement

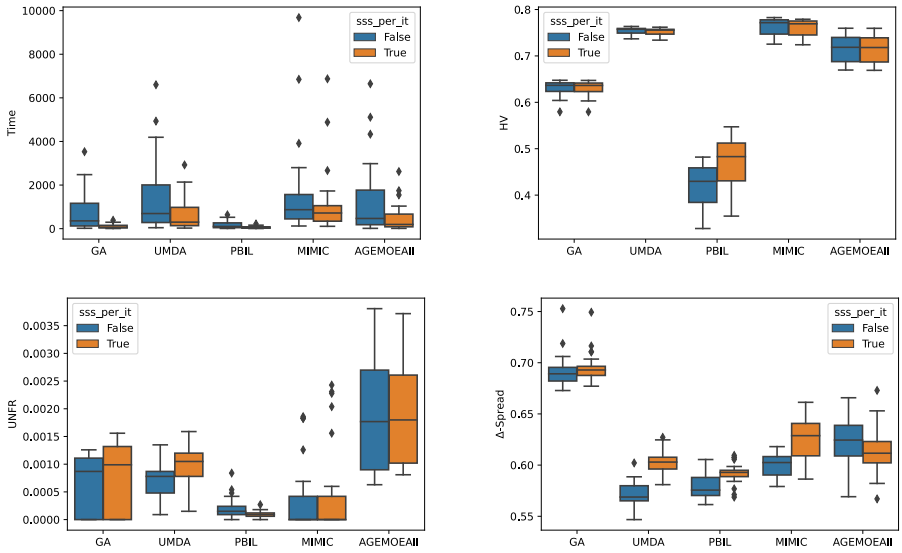

Fig. 3: Box plot illustrating the change in metrics for S4 when applying SSS at each iteration.

Engineering) [27] tool. Thus, the output returned by the search is expected not to take a very long time. In this case, we have set a time threshold of up to 150 seconds. Any result obtained by the algorithms longer than the threshold is ignored when identifying the best result for each algorithm among all its configurations. Furthermore, we make the comparisons using the SSS process per iteration, since in RQ2 we learned that this does not affect performance while reducing execution time.

In this case, as it can be seen in Table 4, in dataset S4 our EDA proposals $UMDA_{NRP}$ and $MIMIC_{NRP}$ outperform NSGA-II in the three quality indicators. In fact, by looking at UNFR and Figure 4 we can see that NSGA-II does not have time to find any solution which belongs to the P_{ref}. In the rest of our corpus, the only different result is that $MIMIC_{NRP}$ obtains higher HV than $UMDA_{NRP}$. With respect to AGE-MOEA-II, it is a very competitive algorithm. It obtains a very good HV value and one of the lowest execution times. However, $UMDA_{NRP}$ and $MIMIC_{NRP}$ outperform it in HV, although only $UMDA_{NRP}$ outperforms it regarding UNFR and $\Delta-$Spread.

Method	**HV**	**UNFR**	**$\Delta-$Spread**	**Time(s)**	**#NDS**
GA	.6468	.0010	.6877	117.9	10
$UMDA$	.7563	.0010	.5809	133.8	10
$PBIL_{NRP}$	.5356	.0001	.5866	113.3	10
$MIMIC_{NRP}$	.7692	.0001	.6422	254.7	10
$NSGA-II$	.6282	.0000	.7455	112.9	100
$AGE-MOEA-II$	.7321	.0009	.6292	114.5	10

Table 4: Results for dataset S4 using the best configuration with search time < 150s.

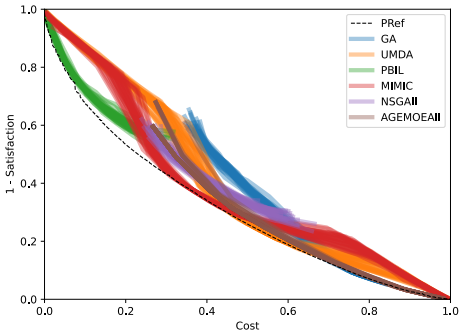

Fig. 4: Pareto Front of each algorithm and Pareto Reference, with search time<150s, dataset S4.

6 Conclusions

We proposed an adaptation of three EDAs to solve the MONRP, and compared them against three well-known GAs. We found that if algorithms are given all the time they need to converge, NSGA-II performs the best. Regarding execution time, NSGA-II is the slowest due to its search process, but not due to the time used to sort non-dominated solutions, since it implements a fast sorting algorithm. We inserted an SSS step at the end of each iteration, in order to alleviate the cardinality of $NDS_{archive}$ in GA, $AGE-MOEA-II$ and our EDA_{NRP} proposals, and we observed that it drastically reduced the mean execution time while significantly maintaining overall quality indicators' performance. Additionally, variability in all metrics was reduced in most cases.
Lastly, we conclude that NSGA-II is not an appropriate algorithm to solve the MONRP in interactive tools due to its execution time. After thresholding the search time to an applicable value in order to make the decision maker not wait too long, our $MIMIC_{NRP}$ and $UMDA_{NRP}$ proposals arise as the best choices to be applied to maximize the HV obtained, as well as UNFR and $\Delta-$Spread in the case of $UMDA_{NRP}$. In future works, we plan to focus on exploiting SSS in the constrained version of the MONRP, and to explore the inclusion of more objectives.

Acknowledgements

This work has been partially funded by the Spanish Government through the project PID2019-106758GB-C33 MCIN/AEI/10.13039/501100011033, by the Regional Government (JCCM) through the project SBPLY/21/180225/000062, and by Universidad de Castilla-La Mancha and "ERDF A way of making Europe" under the project 2023-GRIN-34437.

References

1. Alba, E., Ferrer, J., Villalobos, I.: Metaheuristics and Software Engineering: Past, Present, and Future. International Journal of Software Engineering and Knowledge Engineering **31**(09), 1349–1375 (2021)
2. Almeida, J., Pereira, F., Reis, M., Piva, B.: The Next Release Problem: Complexity, Exact Algorithms and Computations: 5th International Symposium, ISCO 2018, Marrakesh, Morocco, 2018, Revised Selected Papers, pp. 26–38 (2018)
3. Bagnall, A.J., Rayward-Smith, V.J., Whittley, I.M.: The next release problem. Information and Software Technology **43**(14), 883–890 (2001)
4. Baker, P., Harman, M., Steinhöfel, K., Skaliotis, A.: Search based approaches to component selection and prioritization for the next release problem. In: 2006 22nd IEEE International Conference on Software Maintenance, pp. 176–185 (2006)
5. Bermejo, P., de la Ossa, L., Puerta, J.M.: Global feature subset selection on high-dimensional datasets using re-ranking-based edas. In: CAEPIA, *Lecture Notes in Computer Science*, vol. 7023, pp. 54–63. Springer (2011)
6. Charan Kumari, A., Srinivas, K., Gupta, M.P.: Software requirements optimization using multi-objective quantum-inspired hybrid differential evolution. In: O. Schütze, C. Coello (eds.) EVOLVE, pp. 107–120. Springer Berlin Heidelberg (2013)
7. Chaves-González, J.M., Pérez-Toledano, M.A., Navasa, A.: Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm. Knowledge-Based Systems **83**(1), 105–115 (2015)
8. Coello Coello, C.A., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-Objective Problems. Springer US (2007)
9. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Jhon Wiley and Sons Ltd (2001)
10. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
11. del Sagrado, J., del Águila, I.M., Orellana, F.J.: Ant colony optimization for the next release problem: a comparative study. Proceedings - 2nd International Symposium on Search Based Software Engineering, SSBSE 2010 pp. 67–76 (2010)
12. Durillo, J.J., Zhang, Y., Alba, E., Harman, M., Nebro, A.J.: A study of the bi-objective next release problem. Empirical Software Engineering **16**(1), 29–60 (2011)
13. Durillo, J.J., Zhang, Y., Alba, E., Nebro, A.J.: A study of the multi-objective next release problem. Proceedings - 1st International Symposium on Search Based Software Engineering, SSBSE 2009 (2009)
14. Finkelstein, A., Harman, M., Mansouri, S.A., Ren, J., Zhang, Y.: A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. Requirements Engineering **14**(4), 231–245 (2009)
15. Greer, D., Ruhe, G.: Software release planning: An evolutionary and iterative approach. Information and Software Technology **46**, 243–253 (2004)
16. Gupta, P., Arora, I., Saha, A.: A review of applications of search based software engineering techniques in last decade. 5th Int. Conference on Reliability, Infocom Technologies and Optimization, ICRITO 2016: Trends and Future Directions (978), 584–589 (2016)
17. Harman, M., Jones, B.: Search-based software engineering. Information and Software Technology **43** (2001)
18. Harman, M., McMinn, P., de Souza, J.T., Yoo, S.: Search based software engineering: Techniques, taxonomy, tutorial. In: B. Meyer, M. Nordio (eds.) Empirical Software Engineering and Verification: International Summer Schools, pp. 1–59. Springer Berlin Heidelberg (2012)
19. Henrion, M.: Propagating uncertainty in bayesian networks by probabilistic logic sampling. In: J.F. Lemmer, L.N. Kanal (eds.) Uncertainty in Artificial Intelligence, *Machine Intelligence and Pattern Recognition*, vol. 5, pp. 149–163. North-Holland (1988)
20. Ishibuchi, H., Pang, L.M., Shang, K.: Population size specification for fair comparison of multi-objective evolutionary algorithms. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 1095–1102 (2020)
21. Ishibuchi, H., Pang, L.M., Shang, K.: Solution subset selection for final decision making in evolutionary multi-objective optimization. CoRR **abs/2006.08156** (2020)
22. Ishibuchi, H., Pang, L.M., Shang, K.: Difficulties in fair performance comparison of multi-objective evolutionary algorithm. IEEE Comput. Intell. Mag. **17**(1), 86–101 (2022)

23. Jiang, H., Zhang, J., Xuan, J., Ren, Z., Hu, Y.: A Hybrid ACO Algorithm for the Next Release Problem. In: The 2nd International Conference on Software Engineering and Data Mining, pp. 166–171 (2010)
24. Larrañaga, P., Etxeberria, R., Lozano, J.A., Peña, J.M.: Combinatonal optimization by learning and simulation of bayesian networks. In: UAI '00: Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence, Stanford University, Stanford, California, USA, June 30 - July 3, 2000, pp. 343–352. Morgan Kaufmann (2000)
25. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, USA (2001)
26. Li, M., Chen, T., Yao, X.: How to evaluate solutions in pareto-based search-based software engineering: A critical review and methodological guidance. IEEE Transactions on Software Engineering **48**(5), 1771–1799 (2020)
27. Orellana, F.J., Cañadas, J., del Águila, I.M., Túnez, S.: INSCO Requisite - A Web-Based RM-Tool to support Hybrid Software Development. In: J. Cordeiro, J. Filipe (eds.) ICEIS 2008 - Proceedings of the Tenth International Conference on Enterprise Information Systems, Volume ISAS-1, Barcelona, Spain, June 12-16, 2008, pp. 326–329 (2008)
28. Panichella, A.: An improved Pareto front modeling algorithm for large-scale many-objective optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '22, p. 565–573. Association for Computing Machinery (2022). DOI 10.1145/3512290.3528732. URL https://doi.org/10.1145/3512290.3528732
29. Pérez-Piqueras, V., López, P.B., Gámez, J.A.: Estimation of distribution algorithms applied to the next release problem. In: 17th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2022), Lecture Notes in Networks and Systems, pp. 98–108. Springer (2023)
30. Price, K., Storn, R.: Differential Evolution-A simple evolution strategy for fast optimization (1997)
31. Ramírez, A., Delgado-Pérez, P., Ferrer, J., Romero, J.R., Medina-Bulo, I., Chicano, F.: A systematic literature review of the sbse research community in spain. Progress in Artificial Intelligence **9**(2), 113–128 (2020)
32. Sagarna, R., Lozano, J.A.: On the performance of estimation of distribution algorithms applied to software testing. Applied Artificial Intelligence **19**(5), 457–489 (2005)
33. del Sagrado, J., del Águila, I., Orellana, F.J.: Multi-objective ant colony optimization for requirements selection. Empirical Software Engineering **20**, 577–610 (2015)
34. de Souza, J.T., Maia, C.L.B., Ferreira, T.d.N., Carmo, d.R.A.F., Brasil, M.M.A.: An ant colony optimization approach to the software release planning with dependent requirements. In: M.B. Cohen, M. Ó Cinnéide (eds.) Search Based Software Engineering, pp. 142–157. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
35. Vescan, A., Pintea, C.M., Pop, P.C.: Test Case Prioritization—ANT Algorithm With Faults Severity. Logic Journal of the IGPL **30**(2), 277–288 (2020)
36. Zhang, Y., Harman, M., Mansouri, A.: The multi-objective next release problem. In: GECCO 2007: Genetic and Evolutionary Computation Conference, pp. 1129–1137 (2007)
37. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms — a comparative case study. In: A.E. Eiben, T. Bäck, M. Schoenauer, H.P. Schwefel (eds.) Parallel Problem Solving from Nature — PPSN V, pp. 292–301. Springer Berlin Heidelberg (1998)
38. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation **7**(2), 117–132 (2003)