

FEDA-NRP: A fixed-structure multivariate estimation of distribution algorithm to solve the multi-objective Next Release Problem with requirements interactions

Víctor Pérez-Piqueras, Pablo Bermejo^{*}, José A. Gámez

Universidad de Castilla-La Mancha, Escuela Superior Ingeniería Informática s/n, Albacete, 02071, Spain

ARTICLE INFO

MSC:

68T20

Keywords:

Next Release problem

Estimation of distribution algorithms

Evolutionary multi-objective search

Search-based software engineering

Bayesian networks

Agile

ABSTRACT

In the development of a software product, the Next Release Problem is the selection of the most appropriate subset of requirements (tasks) to include in the next release of the product, such that the selected subset maximises the overall satisfaction of the stakeholders and minimises the total cost. Furthermore, in most cases, requirements or tasks cannot be developed independently, as there are dependencies between them, which must be respected in the selection for the next release. In this paper, we approach the Next Release Problem as a constrained bi-objective optimisation problem. The main contribution is the design of an Estimation of Distribution Algorithm that exploits domain knowledge, i.e. the dependencies between the requirements, to define the structure of a Bayesian network that models the relationships between the binary variables (requirements) to be optimised. The use of a Bayesian network with a fixed structure reduces the complexity of the search process, since it is unnecessary to learn the structure at each iteration of the algorithm. Moreover, it ensures that the sampled individuals are always valid with respect to the required dependencies. The second main contribution is the generation of a corpus of synthetic datasets with cost estimations derived from agile and classic management methodologies. Standard multi-objective metrics are computed in order to assess our proposal and compare it with other evolutionary multi-criterion optimisation algorithms, determining that it is the optimal choice when dealing with complex datasets.

1. Introduction

Software projects, both new development and enhancement projects, go through a series of phases during their life cycle. Among these phases are requirements elicitation, requirements selection, design, implementation, testing, deployment, etc. Many of these steps may be tedious, difficult to succeed in or very time-consuming; thus, a new set of techniques has been developed over the last two decades in order to alleviate these problems. Concretely, the Search-Based Software Engineering (SBSE) paradigm (Harman et al., 2012a) refers to the reformulation of software engineering problems into new or traditional search-based optimisation algorithms, which find optimal or near-optimal solutions in a search space. The main reason for this reformulation is that many problems in software engineering need to optimise two or more objectives, resulting in computationally intractable problems which do not allow for an exhaustive search. The definition of software engineering problems as optimisation ones opens the door to the use of a plethora of classic and novel methods; such as genetic algorithms (Katoch et al., 2021), particle swarm optimisation (Gad, 2022), ant colony optimisation (Dorigo et al., 2006) and estimation of distribution algorithms (Larrañaga and Lozano, 2001).

This work deals with the software development phase of requirements selection, known as the Next Release Problem (NRP) (Bagnall et al., 2001; Iqbal and Alam, 2021), one of the five most common Search-Based Software Engineering (SBSE) problems (Chen and Li, 2023). The NRP addresses the task of selecting a subset of requirements to be delivered in the next release or increment of the product, optimising certain objectives while fulfilling given restrictions. The NRP needs to be solved under any software project management methodology; that is, requirements selection is a problem to be solved in both classic (PMI, 2021; Imani, 2017) and agile (Beck, 1999; Schwaber and Sutherland, 2020) methodologies. In a classic or plan-driven approach, a large and complete set of requirements is previously elicited and then, for each release of the final product, a subset of these requirements needs to be selected under certain restrictions. When the development of a software product is being managed in an agile or value-driven manner, the NRP needs to be solved more frequently (at least once per increment) and, although the cardinality of the requirements set could be lower than in classic methods, this set is updated continuously and an NRP solution is thus not valid after a few days or weeks. Furthermore, some projects

^{*} Corresponding author.

E-mail address: Pablo.Bermejo@uclm.es (P. Bermejo).

scale agile teams and the set of requirements grows as large as in classic projects but still with the same update rate.

The basic formulation of the NRP mixes the objectives to be optimised, commonly minimising cost and maximising satisfaction, into a single objective in order to facilitate or simplify the search process. A budget B is then used to limit the search, establishing as many instances of the problem for the same dataset as values are set for B . Subsequently, NRP started to be addressed as a multi-objective problem (MONRP) (Zhang et al., 2007; Durillo et al., 2009; Geng et al., 2018; Rahimi et al., 2022), in such a way that objectives are not mixed. Solutions are then collected in Pareto fronts using the dominance criteria, and returned to the decision-maker in order to evaluate which fits best, given the current situation of the project. In the MONRP version of the problem, the search with a budget B is not typically limited, mainly because solution objectives are not merged, so the decision-maker can visually choose the desired satisfaction/cost ratio.

The method to deal with search objectives is not the only feature that distinguishes the formulation of NRP; the management of requirements dependencies is also particular (del Sagrado et al., 2015; Hamdy and Mohamed, 2019). Some planned requirements may imply the previous or posterior development of other requirements, or they may be mutually exclusive. Thus, the NRP does not only need to maximise satisfaction while minimising cost, but this must also be done while respecting certain constraints between requirements. This makes it a NP-hard problem, as proven in Almeida et al. (2018), and so it has been commonly solved with metaheuristic optimisation search algorithms, such as ACO (del Sagrado et al., 2015), PSO (Hamdy and Mohamed, 2019), GRASP (Pérez-Piqueras et al., 2022), Simulated Annealing (Baker et al., 2006) and NSGA-II (Durillo et al., 2011). Non metaheuristic approaches based on integer programming (Dong et al., 2022) and pursuing any-time behaviour (Domínguez-Ríos et al., 2019) have also been proposed. However, almost no attention has been focused on the use of Estimation of Distribution Algorithms (EDAs) (Larrañaga and Lozano, 2002). This is striking in view of the results obtained by this family of methods in a related problem such as feature selection (Guyon and Elisseeff, 2003; Abdollahzadeh and Gharehchopogh, 2022), both in its single-objective (Bermejo et al., 2011) and multi-objective versions (Maza and Touahria, 2019). In fact, it was only very recently that the first approach to the NRP using EDAs appeared (Pérez-Piqueras et al., 2023), but the proposal is limited to the use of univariate EDAs, which cannot explicitly deal with requirements dependencies.

In this work, we propose the FEDA-NRP (Fixed-structure EDA) algorithm, which takes advantage of the EDA search properties and of the fact that all dependencies among requirements in the Next Release Problem are known in advance. Furthermore, the preferred application niche for FEDA-NRP will be complex problems, with a large number of requirements and dependencies between them, since this is where exploiting domain knowledge directly in the algorithm will be a major advantage. The main contributions of this work are the following:

- (i) The development of a multivariate estimation of distribution algorithm (FEDA-NRP) to cope with the constrained bi-objective next release problem. The prior knowledge domain is used to construct a Bayesian network that models the requirement dependencies, thus avoiding the structural learning phase and accelerating the efficiency of the algorithm. Furthermore, all individuals sampled from this model are valid with respect to the constraints provided (dependencies).
- (ii) The generation of a public corpus of synthetic datasets with different complexities in the number of requirements and dependencies, and different project management context (classic and agile cost estimation). We hope that this corpus can be used as a reference in future studies, so it is available in a public repository.¹

- (iii) A comprehensive comparative experimental study to assess the strengths and weaknesses of FEDA-NRP with respect to state-of-the-art algorithms dealing with the bi-objective next release problem, which follows the guidelines and suggestions from recent literature for fair comparison of multi-objective search algorithms (Ishibuchi et al., 2022) and comparing the most suggested quality metrics (Li et al., 2020). It should be noted here that all algorithms are adapted in the sampling or selection phase to respect the constraints (dependencies) provided.
- (iv) The code of FEDA-NRP and the rest of the algorithms used in the comparison are publicly available, as well as the scripts needed to reproduce the experiments included in this work.²

In the following sections, we will explain a number of concepts in greater detail: We introduce the single and multi-objective NRP (Section 2.1) and then the management of requirements dependencies (Section 2.2). Next, in Section 3 we present an overview of the state of the art, citing classic and recent related works. Then, in Section 4 we present our proposed method FEDA-NRP, giving further detail with an example in Section 4.3. Our experimental framework is decomposed into several parts: First, Section 5.1 presents the process to sample a new corpus of public datasets; Sections 5.4, 5.2 and 5.5 provide the fine-grained details about the evaluation metrics, algorithms and experiment configuration. Reproducibility information is described in Section 5.6. Finally, Section 5.7 presents the quality metrics and visual indicators computed to assess the goodness of the compared algorithms, and, in Section 6, we summarise the main conclusions and propose future work.

2. The next release problem

In this section, we introduce some necessary concepts related to the NRP. First, we define the Multi-Objective NRP (MONRP) and then explain the concept of requirements dependencies or interactions. We introduce different types of dependencies, which are the most common, and also how to convert some types into others. We also cite relevant works, some of which do not deal with the interaction constraints and others that do.

2.1. The (multi-objective) next release problem

The Multi-Objective NRP (MONRP) can be defined by a set $R = \{r_1, r_2, \dots, r_n\}$ of n candidate software requirements, which are suggested by a set $C = \{c_1, c_2, \dots, c_m\}$ of m clients. In addition, a vector of costs or efforts is defined for the requirements in R , denoted $E = \{e_1, e_2, \dots, e_n\}$, in which each e_i is associated with a requirement r_i (Harman et al., 2012b). Each client has an associated weight, $W = \{w_1, w_2, \dots, w_m\}$, which measures its importance. Moreover, each client gives an importance value to each requirement, depending on their needs and goals with respect to the software product being developed. Thus, the importance that a requirement r_j has for a client c_i is given by a value v_{ij} , such that a zero value represents client c_i having no interest in implementation of the requirement r_j . An $m \times n$ matrix is used to hold all the importance values in v_{ij} . The overall satisfaction provided by a requirement r_j is denoted as $S = \{s_1, s_2, \dots, s_n\}$ and is measured as a weighted sum of all importance values for all clients, as expressed in Eq. (1):

$$s_j = \sum_{i=1}^m w_i \times v_{ij} \quad (1)$$

The MONRP involves finding a decision vector X , which includes the requirements to be implemented for the next software release, $X \subseteq R$, which contains the requirements that maximise clients satisfaction and minimise development efforts.

¹ <https://doi.org/10.5281/zenodo.7247877> (Pérez-Piqueras et al., 2022).

² https://github.com/UCLM-SIMD/MONRP/tree/eng_app_ai23.

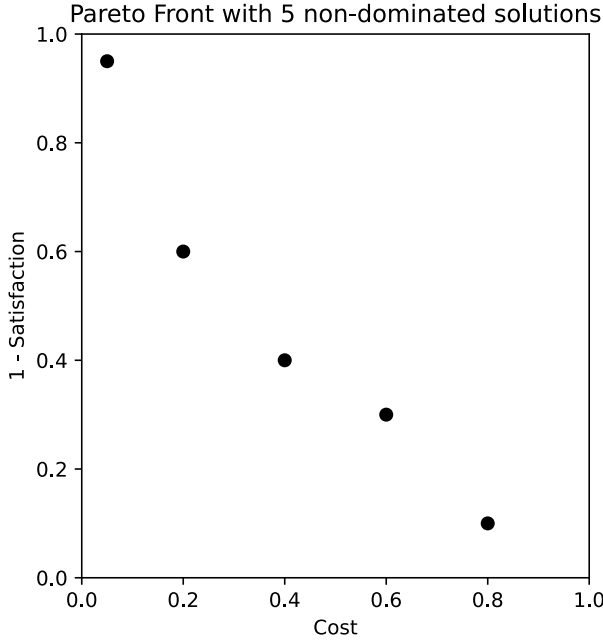


Fig. 1. Pareto Front with five non-dominated solutions in MONRP.

The MONRP objectives are expressed in Eqs. (2) and (3):

$$\text{Maximise } S(X) = \sum_{j \in X} s_j \quad (2)$$

$$\text{Minimise } E(X) = \sum_{j \in X} e_j \quad (3)$$

In addition, requirements in vector X might have to satisfy the constraints of the problem. These constraints are related to the interactions between requirements and to the total effort or budget of the development.

Defining the NRP as a multi-objective (cost-value) optimisation problem has the advantage that a single solution to the problem is not sought, but rather a set of solutions, the *Pareto front*. In this way, any solution from this set can be chosen by the decision-maker, according to the conditions, situation and restrictions of the software product development.

The Pareto front is a vector or set of configuration values for the decision variables that satisfies the problem constraints and optimises the objective functions. Thus, the Pareto front contains a set of solutions that are not dominated by any other, named *non-dominated solutions (NDS)*. Given a solution vector $x = [x_1, x_2, \dots, x_j]$ where j is the number of problem objectives, it dominates a solution vector $y = [y_1, y_2, \dots, y_j]$ if and only if y is not better than x for any objective $i = 1, 2, \dots, j$. In addition, there must exist at least one objective x_i that is better than the corresponding y_i of y . Conversely, two solutions are non-dominated as long as neither of them dominates the other.

A common practice consists on maintaining an archive of all the non-dominated solutions ($NDS_{archive}$) found during the search. Solutions in $NDS_{archive}$ are then used for the evaluation of quality metrics in order to assess the algorithm. Fig. 1 is an example of a Pareto front with an NDS set for MONRP, in which both objectives are normalised so that their value range is $[0,1]$. The value for Satisfaction is reversed so that it may be interpreted in the same sense as Cost; thus, point (0,0) is the ideal point and the nearer the solutions are to this point, the better they are.

2.2. Requirements dependencies in NRP

Software development commonly deals with requirements which are dependent on one another, mostly in an inclusive or sometimes in

an exclusive way. When solving the NRP (either in the basic or MONRP version), these dependencies also need to be taken into account. In Carlshamre et al. (2001), authors define several types of dependencies or interactions from the review of several software projects. In del Sagrado et al. (2015), they are paraphrased in the following set of 5 types of requirements interactions:

- Implication (*requires*; $r_i \Rightarrow r_j$): a given requirement must be implemented before implementing another.
- Combination (*AND*; $r_i \odot_j$): two requirements should be included in the same release.
- Exclusion (*OR*): implementing one requirement excludes another.
- Revenue: developing a given requirement changes the value or satisfaction of other requirements.
- Cost: developing a given requirement changes the cost of other requirements.

Carlshamre et al. (2001) suggests that *implication* and *combination* interactions are the most important types, and should thus be tackled with higher priority than *exclusion* and non-functional dependencies. A combination dependency in software development has more of a semantic need than a technological one; that is, sometimes it makes more sense to present two new functionalities together. It is the *requires* or implication which expresses a hard dependency for technological reasons. In terms of NRP, this means that, having an implication dependency $r_i \Rightarrow r_j$, if at some step the search algorithm selects r_i then r_j must also be selected. Since NRP solutions do not rank the selected requirements, making sure that both requirements are selected is enough to solve the implication. Moreover, in order to fulfil an AND dependency $r_i \odot_j$ (see Fig. 2(a)), this can be applied in practice as having a new requirement $r_{i,j}$ (Fig. 2(b)), which stands as the delivery of the two requirements together, as done in del Sagrado et al. (2015). Furthermore, in the case of having $r_i \odot_j$ and $r_j \odot_k$, we can associate the three requirements first as $r_{i,j} \odot_k$ and then join them as $r_{i,j,k}$. Note that as this group of variables is then managed as a single one, it has the side effect of reducing the problem dimensionality. Finally, after transforming this combination interaction, the implications from parents and to children of the resulting requirement is the union of the original ones (Fig. 2(c)). Thus, when designing the search algorithm, the most practical decision is to make it capable of dealing with implication dependencies.

3. Related work

The NRP was first formulated by Bagnall et al. (2001). In the current definition of NRP, a subset of requirements has to be selected, with the goal being to meet the clients' needs, minimising development effort and maximising clients' satisfaction. They applied a variety of metaheuristics techniques, such as simulated annealing, hill climbing and GRASP, but combining the objectives of the problem into a single-objective function. Later, Baker et al. (2006) also solved the NRP as a budget-constrained single-objective version of the problem, similarly to the feature subset selection problem.

Other studies started formulating the NRP as a multi-objective optimisation (MOO) problem, with the first being the proposal of Zhang et al. (2007). This new formulation, known as the Multi-Objective Next Release Problem (MONRP) or Cost-Value Model, is based on Pareto dominance (Coello Coello et al., 2007). In this approach, each objective is tackled separately, exploring the non-dominated solutions (NDS). Many works tackling the MONRP make use of the Pareto Archived Evolutionary Strategy (PAES) (Knowles and Corne, 1999), as in Chaves-Gonzalez et al. (2015), Chaves-González et al. (2015) and Marghny et al. (2022), consisting of maintaining a $NDS_{archive}$ found during the search. Finkelstein et al. (2009) also applied multi-objective optimisation considering different measures of fairness.

From the most recent reviews in SBSE, regarding Ramírez et al. (2020), only an EDA application to software testing (Sagarna and

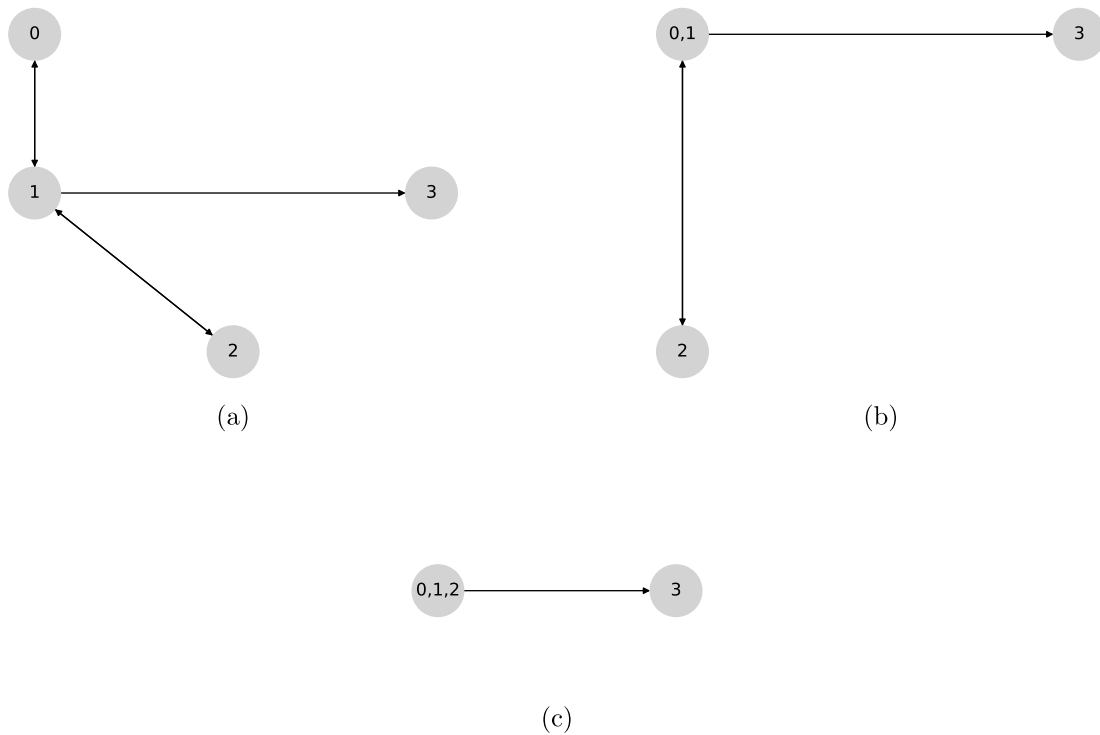


Fig. 2. Example of the transformation process for combination interactions.

Lozano, 2005) is referenced; and in Gupta et al. (2016) and Alba et al. (2021) EDA approaches are neither mentioned nor matched to any solution of the NRP. To the best of our knowledge, the first work using EDAs to solve the NRP is Pérez-Piqueras et al. (2023), where the authors solve the NRP with the UMDA and PBIL univariate EDA. In that study, both EDAs outperform genetic algorithms (NSGA-II), and this advantage increases as the number of requirements in the problem instances also gets larger.

Most NRP works, such as Durillo et al. (2011), Jiang et al. (2010) and Coello Coello et al. (2007), do not treat requirements dependencies. The first approaches that solve the single-objective NRP handling dependencies are authored by del Sagrado et al. (2011) and Souza et al. (2011), adapting global search algorithms based on Ant Colony Systems. Both proposals operate by representing dependencies in a graph that is also used to perform the search. In subsequent works, del Sagrado et al. (2015) also tackled requirements dependencies in the multi-objective version of the NRP, by formally defining graphs which represent all dependency types, reconstructing them, so that, in the end, only implication dependencies are present in the graph. Thus, during the search, they remove connections between a selected requirement and its OR-related requirements. Apart from ACO, other swarm-based approaches have been considered to deal with the MONRP by using implications dependencies, such as particle swarm optimisation (Hamdy and Mohamed, 2019) and artificial bee colony (Chaves-Gonzalez et al., 2015).

4. FEDA-NRP: a multi-objective EDA capable of using the a priori known structural relations between requirements dependencies for the NRP

In Pérez-Piqueras et al. (2023), the authors use univariate EDAs to cope with the MONRP. However, univariate EDAs are unable to explicitly manage requirement interactions, i.e. dependencies. This drawback can be overcome by using more complex EDA models, which explicitly manage dependencies between problem variables. In fact, n-variate EDAs (e.g. EBNA Larrañaga et al., 2000 and BOA Pelikan et al., 1999)

learn a probabilistic graphical model at each iteration in order to capture the underlying dependencies in the current population. However, this would be needless in the MONRP because the dependencies are one of the inputs (constraints) to the problem, and are thus known from the very beginning. In this section, we first briefly introduce some notions about EDAs, and then present our proposal, FEDA, a multivariate EDA with a fixed structure of dependencies to cope with the MONRP.

4.1. Estimation of distribution algorithms

Stochastic global search algorithms are expected to capture the underlying interdependencies among items under selection, in this case, requirements. That is why evolutionary algorithms such as GA and EDA typically obtain better results than local search ones.

In the EDA, the search is not guided by the use of genetic operators but by machine learning and inference processes. Thus, in EDA, the goal is to use the best individuals in the population as instances to learn the joint probability distribution (JPD) over the set of variables to be optimised, that is, the inclusion or not of requirements, in our case. This probability distribution explicitly collects the relationships between the variables in the best individuals, and so sampling from it should theoretically produce a better subset of individuals (population). However, the JPD is intractable even for a small number of variables, and thus Bayesian networks (BNs) (Koller and Friedman, 2009) emerge as the perfect candidate to model the JPD, given that their combination of graph and probability theory allows them to properly factorise the JPD. Furthermore, many machine learning and simulation algorithms are available in the literature to deal with BNs. Therefore, in practice, BNs constitute the core of EDAs, giving rise to different algorithms depending on the degree of probabilistic dependencies allowed to be modelled by the BN.

Formally, univariate EDAs assume that the n -dimensional JPD factorises as a product of n marginal and independent probability distributions, that is $p_I(x) = \prod_{i=1}^n p_I(x_i)$. The canonical representative of this idea is the Univariate Marginal Distribution Algorithm (UMDA) (Mühlenbein, 1997). UMDA^a, i.e. the adaptation of UMDA to the multi-objective case, starts by creating a random population, and at each

generation it selects the non-dominated individuals of the population, learns the probability model p_l from them, and samples a new population using p_l . In contrast to UMDA, in which populations are transformed into a probability model whose only purpose is to sample new populations, the Probability Based Incremental Learning (PBIL) (Baluja, 1994) algorithm attempts to create a probability model which can be considered a prototype for high evaluation vectors for the function space being explored. 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.

As mentioned, in order to explicitly manage the requirements interdependencies, more complex EDAs are needed, that is, EDAs able to cope with multivariate dependencies among the variables to be optimised. This is the case of the Estimation of Bayesian Networks Algorithm (EBNA) (Larrañaga et al., 2000) which uses an unconstrained Bayesian Network (BN) (Koller and Friedman, 2009) to model the dependencies between the variables. In a BN, the joint probability distribution is factorised as:

$$p(x_1, \dots, x_n) = \prod_{i=1}^n p(x_i | \text{Parents}(x_i)), \quad (4)$$

which means that a marginal probability distribution $p_j(x_j)$ is estimated for each configuration j of values for $\text{Parents}(x_i)$. Thus, the number of parameters to be estimated (and stored) grows exponentially on the number of parents. If x_i has no parents, a single marginal probability distribution is estimated.

In EBNA, at each iteration, the algorithm learns a BN from a dataset consisting of the best evaluated individuals in the current population. This BN is then sampled in order to obtain a new population. It is worth noting here that while the BN structure learning from data is an NP-hard problem, sampling from it can be performed with linear complexity on the number of variables.

4.2. FEDA-NRP

In both the basic formulations of NRP and the multi-objective MONRP, a common input available before solving the problem is the set of requirements dependencies or interactions. In Section 2.2 we saw that, of the different types of interactions, implication $r_i \Rightarrow r_j$ and combination $r_i \odot r_j$ are the most important, with it being possible to eliminate the latter by merging the two requirements together in the same delivery (see example in Fig. 2). Furthermore, implications of the type Finish-to-Start (PMI, 2021) are the most common type of interaction in software projects, since they are internal hard-logic dependencies with a technological sense and are usually provided by the development team.

Therefore, we can assume that the set of requirements and their dependencies, known in advance, can be described in the form of a directed acyclic graph (DAG), $G = (R, E)$, where R is the set of requirements and E is the set of edges which codifies the dependencies among them. For instance, $G = (R = \{r_0, r_1, r_2, r_3, r_4\}, E = \{r_0 \Rightarrow r_2, r_1 \Rightarrow r_2, r_2 \Rightarrow r_4\})$, represents a domain with 5 requirements $\{r_0, r_1, r_2, r_3, r_4\}$, such that requirements r_0, r_1 and r_3 do not depend on any other requirement, while r_2 is directly affected by the inclusion of r_0 or r_1 in the selection of requirements, and r_4 is directly affected by the inclusion of r_2 . Fig. 3 shows the resulting graph G .

The availability of the requirements graph as domain knowledge is of great importance because it actually represents the structure of a BN modelling the interdependencies between all the requirements. Therefore, we can avoid the most computationally demanding process in a multivariate EDA, i.e. structural learning, because parameter learning can be done in a very efficient manner once the structure is known. However, we can go one step further by simplifying the complexity of the BN model considered. To do so, we must be aware of the semantics induced by the set of dependencies between the requirements. Thus, we can observe that the inclusion of requirement $r_i \in \text{Parents}(r_j)$ in

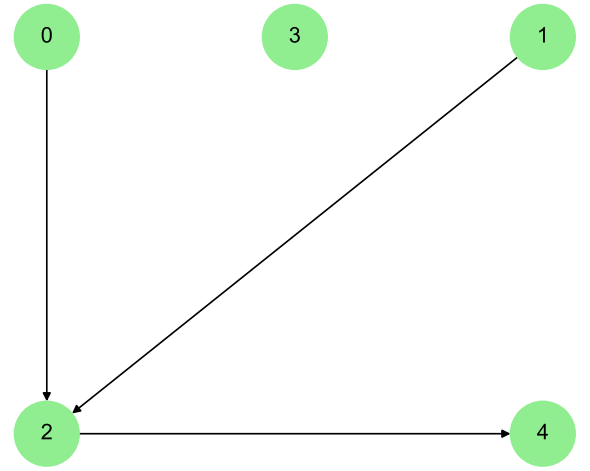


Fig. 3. Graph example with a prefixed structure for requirements dependencies.

the selected set of requirements is enough to force also the inclusion of r_j in the selection, regardless of whether the other requirements in $\text{Parents}(r_j)$ have been included. On the other hand, we must consider that r_j can be selected even if no element of $\text{Parents}(r_j)$ has been selected. This behaviour is well known in the BN literature under the name of *leaky binary noisy-OR gate* (see e.g. Rohmer (2020) and Onisko et al. (2001)):

“Consider a set of n possible causes $\{x_1, \dots, x_n\}$ for an effect y , such that, each cause x_i can produce effect y with probability p_i independently of the presence of any other subset of causes. Furthermore, there is a *leak* probability, p_0 , of the effect being true even if all the causes are false”.

The advantage of using this type of gate to model a relation $\langle x_j | \text{Parents}(x_j) \rangle$ in a BN is that only $|\text{Parents}(x_j)| + 1$ parameters are needed instead of $2^{|\text{Parents}(x_j)|}$. In our problem, we can go even further, because as a dependency $r_i \Rightarrow r_j$ means a deterministic relation, that is, $p(r_j | r_i) = 1$, then $p_1 = p_2 = \dots = p_n = 1$, and we only have to store the *leak* probability p_0 of r_j being true (selected) when all its parents are false (non selected). As a result, our proposal FEDA-NRP has the same learning and space complexity as a univariate EDA algorithm, e.g. UMDA, as we only have to estimate and store vector of length n , $\text{probs}[i]$, containing a single parameter for each requirement: $\text{probs}[i] = p(r_i = 1)$.

Having described the probabilistic graphical model and its advantages, which can be used as basis for our proposal FEDA-NRP (Fixed dependency model Estimation of Distribution Algorithm for the NRP) to cope with the MONRP, we now describe the main FEDA-NRP parts/steps, showing its pseudocode in Algorithm 2.

A high-level overview of FEDA-NRP, as depicted in Fig. 4, performs as follows. The population of individuals, defined by requirements and dependencies (see Section 4.2.1), is firstly initialised following the ancestral order of the BN structure. The algorithm manages two different archives of non-dominated solutions: a local one containing the NDS in the current population and a global one that contains the NDS visited (and currently being non-dominated) from the first iteration up to the current one. The main loop updates a global set of NDS with the current population (see Section 4.2.2), which will be returned at the end of the execution, as well as a local set of NDS, while the termination condition is not met. The probability model previously defined (see Section 4.2.3) is updated using the local set of NDS (see Section 4.2.5). Then, a new population is sampled using the probability model (see Section 4.2.4). Sampled individuals will respect the BN dependencies structure and will thus not require repairing.

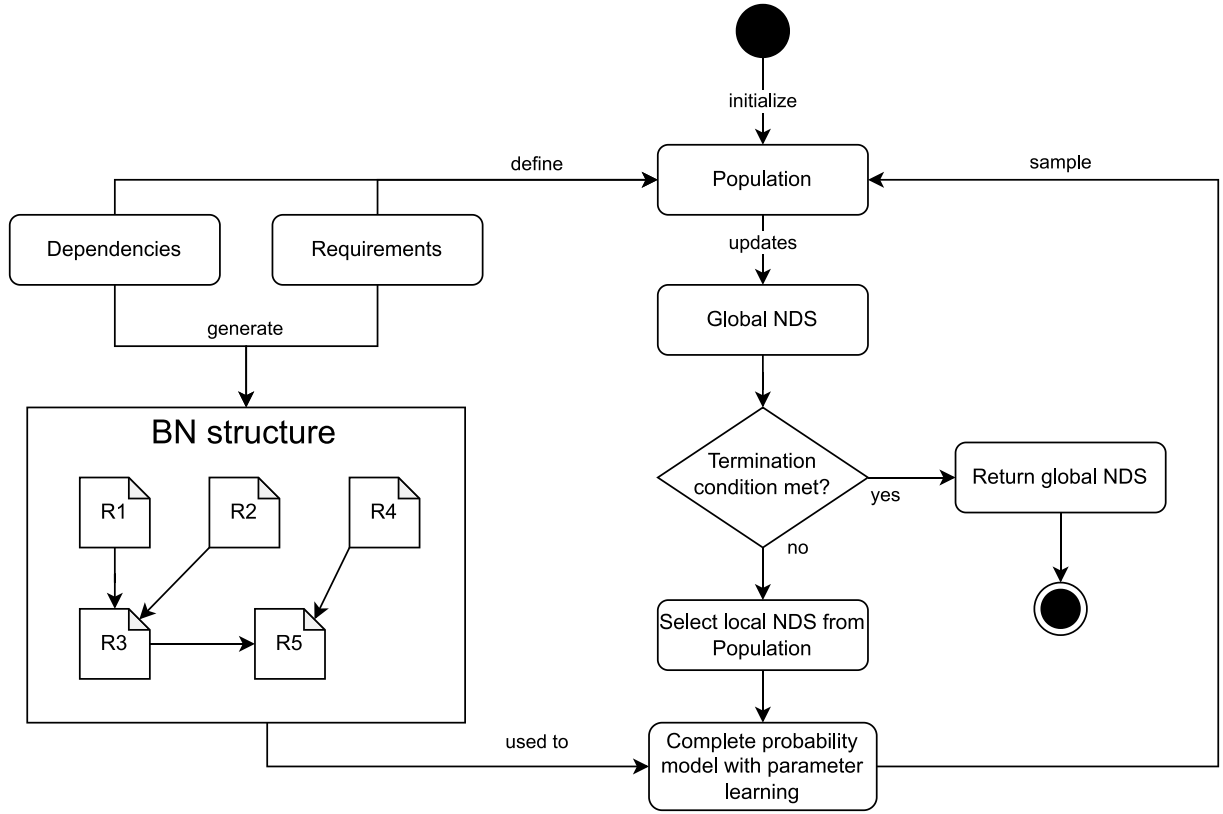


Fig. 4. High-level overview diagram of FEDA-NRP.

4.2.1. Individual representation

As in the rest of population-based algorithms used to solve the MONRP, each solution (individual) is represented by a vector of booleans of length $n = |R|$, where the i th value indicates the inclusion or not of a requirement r_i in the next release.

4.2.2. NDS

FEDA-NRP applies a PAES strategy, and so non-dominated solutions found during the search are stored. In particular, two archives are maintained: a global one, $nds_{archive}$, containing all the non-dominated solutions visited; and a local one, nds_{local} , which contains the non-dominated solutions in the current population. nds_{local} will be used as the dataset to learn the parameters of the probabilistic model at each FEDA-NRP iteration.

4.2.3. Model initialisation

Our probabilistic graphical model $probModel$ is a BN formed by the graphical structure G and the probabilities $probs[]$. The graphical structure G to be used is obtained from the problem domain, as dependencies between the requirements are provided as input (G). The leaky binary noisy-OR is used to model all the nodes having a non-empty set of parents.

With respect to the parameters needed, a uniform distribution is used to initialise the probability of each requirement, i.e. $probs[i] = p(r_i = 1) = 0.5$, for $i = 1, \dots, n$.

4.2.4. Sampling

As is common in multivariate EDAs, probabilistic logic sampling (PLS) (Henrion, 1988) is used, adapted, in our case, to the use of leaky noisy-OR gates.

First, an ancestral order σ with respect to G is needed. That is, an ordering of the requirements such that r_i cannot precede any of its parents. The existence of at least one ancestral order for the variables in G is guaranteed because G is a DAG. In our notation, $\sigma[i] = r_j$ means

that requirement r_j is placed in the i th position of the ordering. For instance, in the graph depicted in Fig. 3, a possible topological ordering is $\sigma = [r_3, r_0, r_1, r_2, r_4]$.

To obtain a new sample s , we apply Algorithm 1, which iterates over the requirements following the ordering stored in σ . If the requirement r_i has no parents in G , then it is selected (i.e. $s[i] = 1$) if a random number generated in $[0, 1]$ is smaller than or equal to $p(r_i = 1)$ ($probs[i]$). On the other hand, if r_i has parents in G , then we are sure they have been previously visited by the sampling process because of the ancestral ordering, and thus its sampled values are already stored in s . If so, we select the values in s corresponding to $Parents(r_i)_G$ ($s^{Parents(r_i)}$) and if this (sub)vector contains at least one value equal to 1, then r_i is selected because of the dependencies. Otherwise, we proceed as in the no-parents case, sampling from the leak probability stored in $probs[i]$.

4.2.5. Learning

As described in previous sections, there is no need for structural learning in FEDA-NRP, because the graph structure is predefined (or fixed) by the dependencies between requirements. Therefore, only parameter learning is needed.

Because of the use of the leaky noisy-OR gate to model the nodes corresponding to requirements with parents, we only have to estimate the probability vector $probs[]$ of length n , with:

$$probs[i] = \begin{cases} p(r_i = 1) & \text{if } Parents(r_i) = \emptyset \\ p(r_i = 1 | \text{all parents} = 0) & \text{if } Parents(r_i) \neq \emptyset \end{cases}$$

Therefore, all the probabilities required can be computed using e.g. maximum likelihood estimation (MLE) from the frequencies obtained in a single pass over the data (subset of individuals). However, there is a special case that should be treated with caution, which is that of a requirement r_i whose value is 0 in all the individuals that constitute the dataset, that is, r_i is never selected. If we use MLE in

Algorithm 1 Sampling pseudocode (modified PLS)

```

procedure GETSAMPLE( $probModel = (G = (R, E), probs); \sigma$ )
   $s \leftarrow [0, 0, \dots, 0]$  ▷ vector of zeroes of length  $|R|$ 
   $P \leftarrow \text{initPop}(G)$  ▷ initialisation
  evaluate( $P$ ) ▷ compute both MONRP objectives
  for  $j = 0$  to  $|R|$  do
     $r_i \leftarrow \sigma[j]$ 
    if  $\text{Parents}_G(r_i) \neq \emptyset$  then ▷ leaky noisy-OR gate
      if  $s^{\downarrow \text{Parents}_G(r_i)}$  contains at least a 1 then
         $s[i] \leftarrow 1$ 
      else if ( $\text{random}([0, 1]) \leq probs[i]$ ) then
         $s[i] \leftarrow 1$ 
      end if
    else ▷ node with no parents
      if ( $\text{random}([0, 1]) \leq probs[i]$ ) then
         $s[i] \leftarrow 1$ 
      end if
    end if
  end for
  return  $s$ 
end procedure

```

Algorithm 2 FEDA-NRP pseudocode

```

procedure FEDA( $G, popSize, iterations$ )
   $nds_{archive} \leftarrow \emptyset$ 
   $\sigma \leftarrow \text{ancestralOrdering}(G)$ 
   $probModel \leftarrow \text{initProbabilisticModel}(G)$  ▷ Section 4.2.3
  for  $i = 0$  to  $iterations$  do
     $P \leftarrow \text{samplePop}(probModel, \sigma, popSize)$  ▷ call  $popSize$  times Algorithm 1
    evaluate( $P$ ) ▷ compute both MONRP objectives
     $nds_{local} \leftarrow \text{getLocalNDS}(P)$ 
     $nds_{archive} \leftarrow \text{updateNDS}(nds_{local})$ 
     $probModel \leftarrow \text{learnProbModel}(probModel, nds_{local})$  ▷ Section 4.2.5
  end for
  return  $nds_{archive}$ 
end procedure

```

this case, $P(r_i = 1)$ will be zero, and thus solutions selecting r_i will no longer be obtained. Therefore, to avoid this undesired behaviour, we keep the value of $probs[i]$ unchanged, that is, the value is not modified in the learning step. Evidently, some other possibilities are available, such as smoothing (e.g. Laplace) for parameter estimation, but this one has worked properly for the problem at hand.

4.2.6. FEDA-NRP scheme

Finally, using the previously described processes as building blocks, the pseudocode of FEDA-NRP is shown in Algorithm 2. As can be observed, FEDA-NRP follows a PAES strategy, that is, as explained in Section 2.1, an $NDS_{archive}$ is maintained along the whole process, being updated with the set of non-dominated solutions found at each iteration.

4.3. Example

This section provides an example based on the dependencies codified by the graph G shown in Fig. 3, considering $[r_3, r_0, r_1, r_2, r_4]$ as the ancestral ordering for sampling.

4.3.1. Model initialisation

Given G in Fig. 3, prior probabilities for selecting requirements without parents and leak probabilities for selecting requirements with parents are set to 0.5. Thus, $probs = [0.5, 0.5, 0.5, 0.5, 0.5]$.

4.3.2. Population sampling

Let us detail the sampling of an individual by using Algorithm 1 and $\sigma = [r_3, r_0, r_1, r_2, r_4]$. We start with $s = [0, 0, 0, 0, 0]$.

- $\sigma[0] = r_3$. As r_3 has no parents in G , we sample a random uniform number in $[0, 1]$, say $u = 0.72$. As $u \leq p(r_3 = 1)$? ($0.72 \leq 0.5$?) is false, $s[3]$ remains 0.
- $\sigma[1] = r_0$. As r_0 has no parents in G , we sample a random uniform number in $[0, 1]$, say $u = 0.91$. As $u \leq p(r_0 = 1)$? ($0.91 \leq 0.5$?) is false, $s[0]$ remains 0.
- $\sigma[2] = r_1$. As r_1 has no parents in G , we sample a random uniform number in $[0, 1]$, say $u = 0.51$. As $u \leq p(r_1 = 1)$? ($0.51 \leq 0.5$?) is false, $s[1]$ remains 0.
- $\sigma[3] = r_2$. As r_2 has $\{r_0, r_1\}$ as parents, we first check whether $s[0]$ or $s[1]$ are 1. As this is not the case, we sample a random uniform number in $[0, 1]$, say $u = 0.17$. As $u \leq p(r_2 = 1 | s_0 = s_1 = 0)$? ($0.17 \leq 0.5$?) is true, we set $s[2] = 1$.
- $\sigma[4] = r_4$. As r_4 has $\{r_2\}$ as parent, we first check whether $s[2]$ is 1. As this is the case we directly set $s[4] = 1$.

Therefore, our sample is $s = [0, 0, 1, 0, 1]$. Let us consider $popSize = 6$, and that the sampling process produces:

$P = \{[0, 0, 1, 0, 1]; [0, 0, 0, 0, 1]; [0, 0, 1, 1, 1]; [0, 0, 0, 0, 0]; [0, 1, 1, 0, 1]; [1, 1, 0, 1, 1]\}$

4.3.3. Selecting non-dominated solutions

Let us assume that, after evaluating the population, the following set of non-dominated solutions are identified:

$nds_{local} = \{[0, 0, 1, 0, 1]; [0, 0, 0, 0, 1]; [0, 0, 0, 0, 0]; [1, 1, 0, 1, 1]\}$

They will constitute our local archive of NDS as well as the global one, because this is the first iteration of FEDA-NRP.

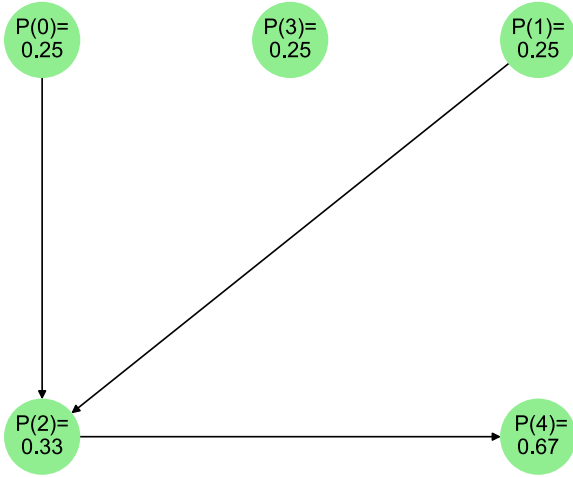


Fig. 5. Marginal/conditional probabilities for each requirement.

4.3.4. Probabilistic model learning

By taking $nds_{local} = \{[0, 0, 1, 0, 1]; [0, 0, 0, 0, 1]; [0, 0, 0, 0, 0]; [1, 1, 0, 1, 1]\}$ as our dataset, we simply compute the proper frequencies to estimate the new *probs* vector.

- $probs[0] = p(r_0 = 1) = \frac{1}{4} = 0.25$
- $probs[1] = p(r_1 = 1) = \frac{1}{4} = 0.25$
- $probs[2] = p(r_2 = 1 | r_0 = r_1 = 0) = \frac{1}{3} = 0.33$
- $probs[3] = p(r_3 = 1) = \frac{1}{4} = 0.25$
- $probs[4] = p(r_4 = 1 | r_2 = 0) = \frac{2}{3} = 0.67$

Please note that for requirements having a non-empty set of parents (r_2 and r_4), only the individuals for which all the parents have a value equal to zero are considered to compute the estimated probability.

Therefore, the new probabilistic model to be used in the next iteration has the *fixed* graphical structure and parameters $probs = [0.25, 0.25, 0.33, 0.25, 0.67]$. Fig. 5 shows the result.

5. Experimental framework

We performed an exhaustive experimentation not only with our proposed method FEDA-NRP but also, for comparison reasons, with univariate and bivariate EDAs (UMDA, PBIL, MIMIC), and two advanced multi-objective genetic algorithms: AGE-MOEA-II and C-TAEA, all adapted to deal with dependencies. Due to the lack of public NRP datasets, we sampled a corpus with instances of software products managed under different contexts, which also constitutes an important contribution of this work. In the following sections, we introduce the sampling process and the resulting datasets. We then provide some detail about the quality metrics used for the evaluation. Finally, we explain the methodology followed in the execution of the different algorithms and the results obtained.

5.1. Datasets

In 2015, Chaves-Gonzalez et al. (2015) explained that due to the privacy policies followed by software development companies, the only two publicly available datasets were those included in their work, which we denote as *p1* (Greer and Ruhe, 2004) and *p2* (del Sagrado et al., 2015) in Table 1 (*p* stands for *public*). These public datasets have a low number of requirements, and present only a few implication and combination dependencies.

Later, Almeida et al. (2018) dealt with several corpuses of different sizes and interactions complexity: Classic and Realistic datasets (Ren

et al., 2012), and a new corpus of their own sampling. Souza et al. deal with 72 synthetic datasets in Souza et al. (2011), with a number of requirements from 20–50 and implication interactions with a density of implications of up to 20% of requirements; however, they are not public yet. Apart from Greer and Ruhe (2004) and del Sagrado et al. (2015), we also found two public datasets in Karim and Ruhe (2014); however their complexity is similar to *p1* and *p2*, and thus we do not consider them, as our goal is to deal with more complex instances.

In the absence of public datasets of higher dimensionality, we decided to sample synthetic datasets with a wide range in the number of requirements, stakeholders, number of implication interactions, density of interactions and average length of implied requirements by the same requirement. The reason we are interested in a large range in the number of requirements is that software projects may contain not only tens of requirements, but in the order of hundreds. This is even more common in projects managed in a plan-driven manner, where requirement elicitation results in a very large output of features to be developed.

Furthermore, we decided to sample datasets related to both projects managed under classic and agile methodologies. Thus, we generated four agile and four classic datasets (named *aX* and *cX*). Lastly, we sampled the *dX* datasets with a classic management context, similar to in *cX*, but with a greater complexity in requirements and dependencies. Table 1 summarises the different properties of the public and synthetic datasets sampled and used in our experiments. Our corpus can be divided into four kinds of datasets:

- *pX datasets*: two small datasets used in several works, such as in del Sagrado et al. (2015). *p1* comes from a real dataset, and it has very few requirements (20) and also very few dependencies (7). *p2*, a public synthetic dataset, provides a more realistic dataset in the sense of its complexity, although the number of dependencies and requirements is still quite small.
- *aX datasets*: commonly, agile projects have a low number of actively managed stakeholders, although the received buy-in in the development of the product is more constant. Thus, *aX* datasets present a lower number of stakeholders. Since requirements are not decided a priori, with a long elicitation process, requirements and dependencies between them are not typically large for a given minor or functional release. Thus, we produced datasets with few dependencies and requirements. On the other hand, two *aX* datasets have many requirements. Effort estimations are computed using a Fibonacci scale, similar to common agile estimation techniques.
- *cX datasets*: classic or plan-driven datasets tend to have many requirements and, due to long and expensive planning, also numerous dependencies. Furthermore, due to usual processes of managing stakeholders interests, it is also common to identify more stakeholders than in agile datasets. Effort values were simulated by using the Function Points (FP) size metric extracted from the 2015 version of the International Software Benchmarking Standards Group³ (ISBSG) dataset, using the two highest categorical values from the *Unadjusted Function Points rating* column, ‘New development’ from *Development type* and ‘IFPUG 4+’ from the *Count approach* column. This procedure is used to generate percentiles 25, 50, 75 of total FP of a classic project, in order to generate a realistic sample of a classic estimation of requirements. This size generation is done by selecting randomly, for a given number of requirements, a list of costs that sums up to the percentile value.
- *dX datasets*: following the same procedure as in *cX*, we simulate the most complex classic projects, with the largest number of requirements and dependencies. In fact, this is the case in which the MONRP might be of greater help for the decision-maker.

³ <https://www.isbsg.org/>.

Table 1

Datasets used in our experiments: 2 public dataset, 4 agile, 4 low scale classic, and 4 large scale classic management.

Dataset	#Stakeholders	R	O	density	\bar{D}
p1	5	20	7	0.350	1.857
p2	5	100	29	0.290	2.690
a1	5	50	18	0.360	2.222
a2	15	50	18	0.360	2.722
a3	5	200	74	0.370	1.946
a4	15	200	75	0.375	2.253
c1	15	50	20	0.400	2.400
c2	100	50	17	0.340	3.529
c3	15	200	69	0.345	1.942
c4	100	200	75	0.375	2.093
d1	15	200	88	0.440	3.352
d2	50	200	88	0.440	4.852
d3	15	300	131	0.437	3.771
d4	50	300	145	0.483	3.697

Table 1 summarises all the datasets used in this work. $|R|$ is the number of requirements, $out_degree(r_i)$ is the number of requirements implied by requirement r_i , and O is the number of requirements with an implication dependency towards one or more requirements, as expressed in Eq. (5). $\%R$ is $O/|R|$, that is, the percentage of requirements in the dataset which imply other(s) requirement(s), and is referred to as *density*, ranging $[0,1]$. Finally, \bar{D} is the average number of implications in requirements with implication interaction with one or more requirements (Eq. (6)).

$$O = \sum_{i=1}^{|R|} I(out_degree(r_i) > 0) \quad (5)$$

$$\bar{D} = \frac{\sum_{i=1}^{|R|} out_degree(r_i)}{O} \quad (6)$$

The benchmark of datasets created is publicly available for download in Pérez-Piqueras et al. (2022).

5.2. Algorithms

Apart from FEDA, three well-known EDA algorithms form part of our experimentation framework (UMDA, PBIL and MIMIC) and two recent multi-objective genetic algorithms (AGE-MOEA-II and C-TAEA) that are capable of dealing with dependencies. These were found to perform better than the NSGA-II (Deb et al., 2002) and NSGA-III (Deb and Jain, 2014), two advanced versions of NSGA (Srinivas and Deb, 1994), which is commonly used for comparison in MONRP studies.

- AGE-MOEA-II (Adaptive Geometry Estimation based MOEA II) (Panichella, 2022) is presented and evaluated, being found it outperforms (in the hypervolume indicator) four multi- and many-objective evolutionary algorithms, using a novel method to model the non-dominated front.
- C-TAEA (Li et al., 2019), an Evolutionary Algorithm that maintains two archives (one for convergence and the other for diversity) during a constrained search, outperforming in a benchmark of multi- and many-objective problems with a relatively small number of variables (*requirements* if applied to MONRP).

Since datasets *p1* and *p2* contain combination interactions ($r_i \odot_j$), those dependencies were eliminated by joining both requirements in a new one $r_{i,j}$, as in del Sagrado et al. (2015). The two genetic algorithms are explicitly designed to deal with dependencies.

The five competing algorithms, UMDA, PBIL, MIMIC, AGE-MOEA-II and C-TAEA, were adapted to deal with implication dependencies in a straightforward manner. Thus, after generating a set of individuals, if an individual contains r_i with $r_i \Rightarrow r_j$, then r_j is also set in the individual vector of selected requirements.

FEDA-NRP and all the competing algorithms maintain the $NDS_{archive}$ through their execution. Since the competing algorithms were adapted, we refer to them as UMDA^a, PBIL^a, MIMIC^a, AGE-MOEA-II^a and C-TAEA^a.

5.3. Evaluation indicators

The following are some of the most commonly applied and reliable quality metrics used to assess both convergence and diversity of solutions:

- Hypervolume (HV) (Zitzler and Thiele, 1998). This 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, which should be the same for all algorithms under comparison.
- Unique Nondominated Front Ratio (UNFR) (Li et al., 2020). This metric requires computing a Pareto reference (PRef), which is a set of non-dominated solutions obtained by merging all Pareto fronts returned by the algorithms being evaluated after their execution. Thus, UNFR measures the ratio of solution points in the PRef that belong to the solution set of the evaluated algorithm. That is, it measures the contribution (from 0 to 1) of an algorithm to the PRef.
- Generational Distance+ (GD+) (Veldhuizen and Lamont, 1998). This covers the convergence aspect of the quality of a solution set, measuring the Euclidean distance of the solution set to the ideal PRef. Consequently, this metric is to be minimised. For each solution, its GD is the minimum of the distances to each point in the Pareto front. In order to make GD compliant with Pareto dominance, GD+ enhances GD by measuring distances between points, using only the objective coordinates that are superior in the PRef to those from the solutions set being measured.
- Δ -Spread (Deb, 2001). This measures the dispersion of the solutions in the Pareto front. Thus, the smaller the Δ value is for a set of non-dominated solutions, the better (more uniform).

HV may serve as a general representation of the four quality aspects, but the other three metrics give a more granular detail of spread and uniformity (Δ -Spread), convergence (GD+), and cardinality related to the PRef (UNFR).

5.4. Evaluation

Since datasets are sampled using different scales for agile and classic types, evaluation of Satisfaction and Cost of each solution vector is computed with a prior scaling of requirement value and cost, using a min-max normalisation. Consequently, all non-dominated solutions returned by the executed algorithms are in the range $[0,1]$.

Based on guidance by Ishibuchi et al. (2022, 2020) for fair comparison, we compute all metrics from a subset selection of solutions from the resulting $NDS_{archive}$. Concretely, we run a hypervolume-based greedy forward subset selection to choose 10 solutions.

With regard to HV, it is always computed using the same reference point. In order to decide an appropriate reference point, we drew on the guidance in Li et al. (2020) to set the HV reference point in a bi-objective problem:

$$\begin{aligned} ref_x &= nadir_x + range_x/10 \\ ref_y &= nadir_y + range_y/10 \end{aligned}$$

The nadir point is the worst point found by algorithms during a search. Since we normalise both Satisfaction and Cost, our worst value is 1 for both metrics (Satisfaction is plotted as $1 - \text{Satisfaction}$). Range is the difference between the best and worst point found. The best possible value for each metric is 0, so the x and y values of the reference point for both goals are set as:

$$ref_x = ref_y = 1 + (1 - 0)/10 = 1.1$$

Regarding the GD+ metric, the necessary PRef is a proxy set for the ideal Pareto front. Thus, we build the PRef filtering the non-dominated solutions set from a pool made of all the $NDS_{archive}$ sets found by all the algorithms executed under all the hyperparameter configurations detailed in Section 5.5.

Finally, it is worth mentioning an important aspect of the UNFR metric. In the case of this work, since the PRef is obtained from such a great number of algorithms and configuration combinations (see Section 5.5), it presents a high number of solutions. Furthermore, each algorithm is not evaluated using all solutions from its corresponding $NDS_{archive}$, but using a selected subset of 10 points from its corresponding $NDS_{archive}$. Thus, the UNFR value for each algorithm tends to be quite low, and the maximum possible is never 1, since $|PRef| \gg 10$. In any event, greater UNFR values are desirable.

5.5. Experiment configuration

Each algorithm is executed 30 times. After each execution, a set S of 10 solutions is selected from $NDS_{archive}$ to compute HV and Δ -Spread metrics. In the case of GD+ and UNFR, they cannot be computed until the PRef is constructed; that is, until the 30 executions for all algorithms have finished. Then, GD+ and UNFR are computed from the set S obtained in each execution. Finally, all the metrics are averaged over the 30 runs.

Lastly, with respect to the hyperparameter configuration of algorithms (population size, maximum number of generations, ...), we follow the recommendations in Ishibuchi et al. (2022), which suggest that algorithms should be run under their best possible configuration. Thus, the experimental results for algorithms shown in Section 5.7 come from the 30 executions of each algorithm under its best configuration found. Appendix contains the range of values for each hyperparameter, and we identify the best configuration found for each algorithm after evaluating a grid search over all the hyperparameters. Two hyperparameters are common in all algorithms (Population Size and Number of Iterations), with values: Population Size = {100, 200, 500, 700, 1000} and #Iterations = {50, 100, 200, 300, 400}. All algorithms yield their best results (measured by HV) when using a Population Size = 1000, the maximum value among the 5 given for this hyperparameter. With respect to the number of iterations, all algorithms but UMDA^a need many iterations to converge, usually the maximum (400) or almost the maximum (300 in FEDA) value among the possible ones. UMDA^a seems to converge very quickly, since in 7 out of 14 datasets it yields its best results with just 100 or fewer generations, in both agile and classic projects.

5.6. Reproducibility

All experiments were run under the same runtime environment. The machines used had 32 Gb of RAM, of which only 8 Gb were used, and two 3.00 GHz 4-core Intel Xeon E5450 processors. The operating system used was a CentOS Linux 7 with a 64-bit architecture. All the algorithms and the experimentation setup were implemented by the authors of this work using Python 3.8.8, with the exception of the AGE-MOEA-II and C-TAEA algorithms, which were run using the Python-based Pymoo package (Blank and Deb, 2020). All our code is available at the following repository: https://github.com/UCLM-SIMD/MONRP/tree/eng_app_ai23; the sampled datasets are also available at the following repository: <https://doi.org/10.5281/zenodo.7247877> (Pérez-Piqueras et al., 2022). Additional packages and libraries used to support numeric operations, metric calculations and data visualisation can be found in the list of package requirements inside the GitHub repository, with the following being the most important packages: matplotlib, numpy, pandas, scipy, pymoo.

Table 2

Quality metrics in datasets: public (pX) and agile (aX).

Dataset	Method	HV	UNFR	GD+	Δ -Spread	NDS
p1	UMDA ^a	0.898	0.070	0.036	0.642	35.8
	PBIL ^a	0.908	0.079	0.028	0.532	38.7
	MIMIC ^a	0.835	0.105	0.063	0.650	39.0
	AGE-MOEA-II ^a	0.908	0.079	0.028	0.532	40.0
	C-TAEA ^a	0.888	0.059	0.037	0.621	20.4
	FEDA-NRP	0.895	0.053	0.039	0.685	35.3
p2	UMDA ^a	0.780	0.002	0.046	0.559	145.0
	PBIL ^a	0.732	0.002	0.017	0.572	101.5
	MIMIC ^a	0.817	0.001	0.019	0.597	350.8
	AGE-MOEA-II ^a	0.794	0.019	0.007	0.641	326.6
	C-TAEA ^a	0.680	0.013	0.026	0.639	173.6
	FEDA-NRP	0.793	0.001	0.033	0.602	209.7
a1	UMDA ^a	0.864	0.022	0.030	0.576	67.0
	PBIL ^a	0.897	0.046	0.003	0.608	110.8
	MIMIC ^a	0.850	0.029	0.028	0.561	131.7
	AGE-MOEA-II ^a	0.898	0.056	0.006	0.531	133.5
	C-TAEA ^a	0.838	0.055	0.003	0.551	90.7
	FEDA-NRP	0.873	0.011	0.018	0.581	94.9
a2	UMDA ^a	0.929	0.047	0.028	0.592	47.2
	PBIL ^a	0.974	0.077	0.003	0.610	73.6
	MIMIC ^a	0.913	0.043	0.034	0.596	79.1
	AGE-MOEA-II ^a	0.989	0.105	0.000	0.626	81.5
	C-TAEA ^a	0.890	0.095	0.007	0.652	48.1
	FEDA-NRP	0.943	0.013	0.018	0.631	51.0
a3	UMDA ^a	0.803	0.002	0.067	0.581	126.5
	PBIL ^a	0.656	0.002	0.028	0.570	62.9
	MIMIC ^a	0.814	0.000	0.052	0.606	85.4
	AGE-MOEA-II ^a	0.772	0.016	0.013	0.656	307.6
	C-TAEA ^a	0.665	0.013	0.034	0.686	189.5
	FEDA-NRP	0.824	0.003	0.048	0.592	153.9
a4	UMDA ^a	0.792	0.001	0.064	0.587	143.4
	PBIL ^a	0.658	0.001	0.024	0.568	64.6
	MIMIC ^a	0.805	0.000	0.049	0.585	80.5
	AGE-MOEA-II ^a	0.746	0.014	0.013	0.633	324.3
	C-TAEA ^a	0.647	0.010	0.034	0.659	182.2
	FEDA-NRP	0.802	0.003	0.052	0.617	170.3

5.7. Results

We ran the six algorithms under comparison (FEDA, UMDA^a, PBIL^a, MIMIC^a, AGE-MOEA-II^a and C-TAEA^a) using their best hyperparameter configuration found (see Appendix), and run over our corpus of 14 datasets (Section 5.1).

In this section, we provide the values obtained for each dataset in the four quality metrics (HV, UNFR, GD+, Spread) and the mean cardinality of the final $NDS_{archive}$. We also show a visual indicator (scatter plot) for qualitative assessment.

Quality metrics

In Tables 2, 3 and 4, for each dataset, we highlight in bold the best metric value among the six algorithms.

The most important and widely used quality metric in the literature is HV, since it is Pareto compliant and summarises the four dimensions to be assessed on solution vectors.

With respect to the datasets with the lowest complexity, that is, those with fewer than 200 requirements ($p1$ - $a2$, $c1$, $c2$), it can be observed that there is no clear winner for the HV metric; with AGE-MOEA-II^a being that with more wins (three out of six datasets). On the other hand, FEDA-NRP is the winner in the case of the datasets with the highest complexity in terms of number of requirements ($|R|$) and O (see Table 1). Our proposed FEDA-NRP reaches the highest hyper-volume in all cases but one ($a4$ with 200 requirements, .802 vs .805 in MIMIC). That is, as the number of requirements and requirements interactions increases, FEDA-NRP is able to find better solution subsets than the other algorithms. Furthermore, the greatest outperformance of FEDA-NRP with respect to the other algorithms is in dX datasets that represent classic-managed software products, with this scenario

Table 3
Quality metrics in classic (*cX*) datasets.

Dataset	Method	HV	UNFR	GD+	Δ -Spread	[NDS]
c1	UMDA ^a	0.882	0.027	0.030	0.607	79.3
	PBIL ^a	0.903	0.043	0.004	0.615	105.7
	MIMIC ^a	0.888	0.039	0.016	0.602	143.5
	AGE-MOEA-II ^a	0.924	0.052	0.005	0.661	135.0
	C-TAEA ^a	0.846	0.056	0.002	0.633	93.5
	FEDA-NRP	0.897	0.014	0.017	0.605	90.2
c2	UMDA ^a	0.858	0.020	0.051	0.563	69.6
	PBIL ^a	0.942	0.046	0.003	0.603	103.5
	MIMIC ^a	0.848	0.025	0.030	0.618	159.0
	AGE-MOEA-II ^a	0.945	0.055	0.003	0.607	142.3
	C-TAEA ^a	0.802	0.045	0.011	0.601	97.2
	FEDA-NRP	0.932	0.038	0.007	0.556	98.3
c3	UMDA ^a	0.878	0.003	0.063	0.570	65.7
	PBIL ^a	0.741	0.003	0.039	0.652	36.9
	MIMIC ^a	0.884	0.002	0.050	0.605	50.1
	AGE-MOEA-II ^a	0.869	0.033	0.014	0.648	131.9
	C-TAEA ^a	0.747	0.029	0.024	0.672	76.8
	FEDA-NRP	0.906	0.003	0.044	0.638	57.4
c4	UMDA ^a	0.820	0.001	0.077	0.583	61.0
	PBIL ^a	0.734	0.004	0.041	0.602	48.4
	MIMIC ^a	0.817	0.004	0.046	0.633	56.3
	AGE-MOEA-II ^a	0.795	0.042	0.011	0.663	95.2
	C-TAEA ^a	0.670	0.032	0.032	0.684	67.8
	FEDA-NRP	0.850	0.004	0.051	0.615	59.5

Table 4
Quality metrics in complex classic (*dX*) datasets.

Dataset	Method	HV	UNFR	GD+	Δ -Spread	[NDS]
d1	UMDA ^a	0.794	0.003	0.072	0.563	123.8
	PBIL ^a	0.720	0.002	0.039	0.582	71.2
	MIMIC ^a	0.803	0.001	0.055	0.592	77.2
	AGE-MOEA-II ^a	0.711	0.016	0.016	0.674	278.1
	C-TAEA ^a	0.610	0.011	0.036	0.704	161.7
	FEDA-NRP	0.810	0.003	0.052	0.594	141.5
d2	UMDA ^a	0.801	0.003	0.056	0.571	147.9
	PBIL ^a	0.693	0.001	0.026	0.575	79.6
	MIMIC ^a	0.795	0.000	0.047	0.567	81.9
	AGE-MOEA-II ^a	0.721	0.014	0.016	0.652	322.4
	C-TAEA ^a	0.619	0.012	0.033	0.671	211.5
	FEDA-NRP	0.808	0.002	0.047	0.596	200.3
d3	UMDA ^a	0.763	0.001	0.07	0.570	124.1
	PBIL ^a	0.644	0.002	0.033	0.563	71.0
	MIMIC ^a	0.757	0.000	0.061	0.612	74.6
	AGE-MOEA-II ^a	0.649	0.014	0.023	0.663	236.6
	C-TAEA ^a	0.569	0.011	0.041	0.700	147.5
	FEDA-NRP	0.798	0.002	0.045	0.600	166.1
d4	UMDA ^a	0.750	0.001	0.066	0.569	148.0
	PBIL ^a	0.645	0.002	0.032	0.572	80.3
	MIMIC ^a	0.750	0.000	0.056	0.583	79.6
	AGE-MOEA-II ^a	0.630	0.012	0.022	0.673	295.8
	C-TAEA ^a	0.554	0.007	0.040	0.709	190.5
	FEDA-NRP	0.779	0.001	0.045	0.606	192.4

being where the MONRP is most commonly applied and more useful due to the need for long-term planning. Fig. 6 shows the mean HV for each algorithm as the number of requirements increases. Together with requirements, the number and density of dependencies also increases. Thus, the search space is more difficult to explore since it is much more restricted. Consequently, the diversity of solutions tends to decrease as well as the HV. In the case of a larger number of requirements (200 and 300), FEDA-NRP presents a better average HV than the other algorithms. This difference in HV (distance from the FEDA-NRP brown line to the other lines) increases with the number of requirements. This can be better appreciated in Fig. 7, which shows the difference in the mean HV, between FEDA-NRP and each of the other algorithms, ordered by number of requirements. As can be seen, this difference increases in all cases when the number of requirements exceeds 100, particularly in the case of PBIL^a and the two recent multi-objective

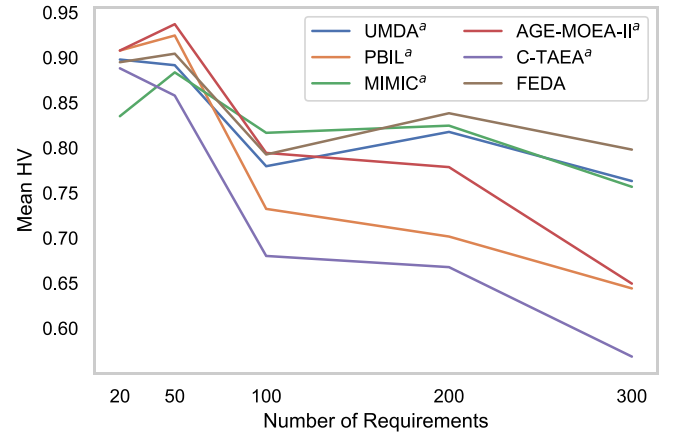


Fig. 6. Mean hypervolume in datasets with the same number of requirements.

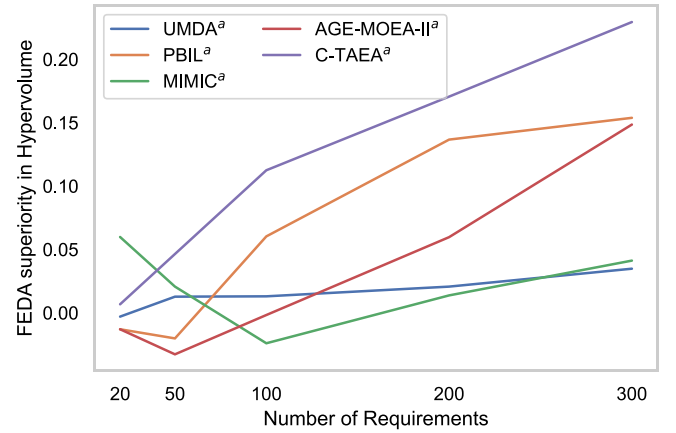


Fig. 7. FEDA-NRP diverges from the rest of algorithms, in terms of hypervolume, as the number of requirements increases.

algorithms AGE-MOEA-II^a and C-TAEA^a. It is also of interest to mention that only when datasets have more than 100 requirements does FEDA-NRP start to diverge from MIMIC^a. With 300 requirements, FEDA-NRP is not just superior in HV to all algorithms, but is also much faster than its main competitors (AGE-MOEA-II^a, C-TAEA^a, and MIMIC^a), as explained later in the *Time* discussion.

Looking at Table 4, it might seem that FEDA-NRP obtains the best HV values because of the high cardinality of the final $NDS_{archive}$ obtained. However, as previously stated, all the metrics are computed from a subset of the same size (10). Thus, it is worth recalling here that in pursuit of a fair comparison (see Section 5.4) all quality metrics are computed from solution sets of the same size.

Nevertheless, in complex datasets and when the focus is not set on HV but on quality metrics that measure just one of the four measurement aspects of non-dominated sets, AGE-MOEA-II^a performs the best in terms of UNFR and GD+, and UMDA^a performs the best in terms of Δ -Spread. With respect to these metrics, it is again worth mentioning that, as explained in Section 5.4, since PRef is computed from all the solutions found by all the algorithms, the UNFR computed for a given algorithm is always very low and far from 1. For example, in dataset *p1*, the PRef is made up of 38 points. Since each execution of an algorithm returns a 10-points subset from the resulting $NDS_{archive}$, its maximum UNFR possible is $10/38 = 0.2632$.

Statistical analysis of quality metrics

For each metric, we run the Friedman test, a non-parametric group test for dependent samples (the same 14 datasets are used as input in the 6 algorithms). When the Friedman test detects a significant

Table 5

Mean values for each algorithm in less and more complex datasets.

Method	HV	UNFR	GD+	Δ-Spread
Simple datasets ($ R \leq 100$)				
UMDA ^a	0.868	0.031	0.037	0.590
PBIL ^a	0.893	0.049	0.009	0.590
MIMIC ^a	0.858	0.040	0.032	0.604
AGE-MOEA-II ^a	0.910	0.061	0.008	0.600
C-TAEA ^a	0.824	0.054	0.014	0.616
FEDA-NRP	0.889	0.022	0.022	0.610
Complex datasets ($ R \geq 200$)				
UMDA ^a	0.800*	0.002*	0.067*	0.574•
PBIL ^a	0.686*	0.002*	0.033	0.586*
MIMIC ^a	0.803*	0.001*	0.052*	0.598
AGE-MOEA-II ^a	0.737*	0.020•	0.016•	0.658*
C-TAEA ^a	0.635*	0.016*	0.034*	0.686*
FEDA-NRP	0.822•	0.003*	0.048*	0.607*

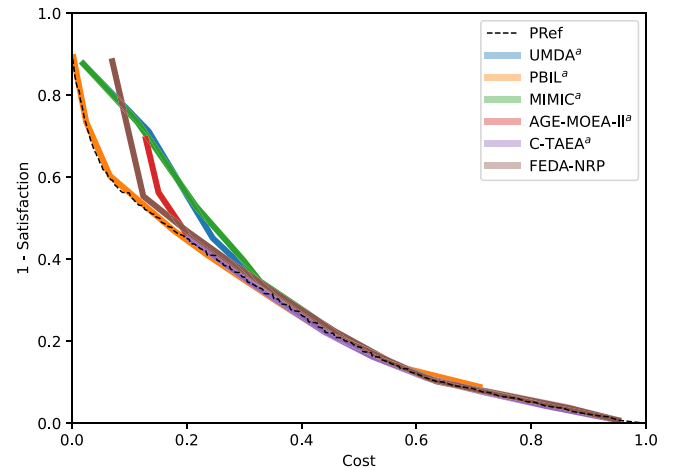
difference between the compared algorithms, a post hoc test is run: pairwise comparisons with Wilcoxon signed-rank test, with correction of the p -value for multiple comparisons reducing the false discovery rate. A significance level of 5% was used. Table 5 shows the mean value obtained for each metric, distinguishing between simple datasets (≤ 100 requirements) and the most complex datasets (≥ 200 requirements). If the post-hoc test shall be run (because the Friedman test is significant), then we mark with • the algorithm with the best mean value, and then with * the algorithms found to be statistically different from the control algorithm in the post-hoc tests. Thus, if an algorithm is not marked with *, it is not significantly different from the control. Conclusions vary from simple to complex datasets; that is, the best performing algorithms depend on the number of requirements and dependencies. When dealing with simple datasets ($p1$, $p2$, $a1$, $a2$, $c1$ and $c2$), the post-hoc tests find no algorithm to be statistically worse than the control algorithm, in any of the quality indicators.

In the case of algorithms with more dependencies and requirements ($a3$, $a4$, $c3$, $c4$, $d1$ - $d4$), which is the target of our proposal, FEDA-NRP obtains the highest HV, being significantly better than UMDA, PBIL, MIMIC, AGE-MOEA-II and C-TAEA (all the algorithms). In the case of our interest lying in metrics which solely reflect one quality dimension with respect to closeness to PRef, then AGE-MOEA-II is the one which performs the best. That is, it finds most of its solutions close to, or on, the PRef, but as can be seen in the visual indicators (plots), this is at the cost of finding medium or high Cost solutions. However, although not shown here for the sake of clarity, the execution time of AGE-MOEA-II, together with C-TAEA, is much greater than that of any of the other algorithms. This, supports us in recommending FEDA-NRP rather than any of the other algorithms in our experimental framework (see Time discussion below) for the case of problems (software projects) with a high number of requirements to be planned.

As a final comment on these results, let us analyse the role played by the type of BN used as graphical model in the different EDAs considered. If we focus on the degree of dependencies allowed, we can order the models as FEDA-NRP (multivariate) > MIMIC (bivariate) > UMDA and PBIL (univariate). Thus, comparing them can be viewed as a sort of ablation study where, at each step, we reduce the number of dependencies allowed, thus simplifying the graphical model. From the results in Table 5, in particular for the complex cases, which are the main target of this proposal, we can observe that the rank is exactly FEDA-NRP > MIMIC > UMDA > PBIL. Therefore, as expected, the expressiveness of the Bayesian networks considered plays a key role in the behaviour of the corresponding EDA.

Visual indicator with plotting of solution subsets

Apart from quality metrics, visual indicators are also useful not only to qualitatively assess the goodness of the solutions subsets, but also to compare the regions in which each algorithm behaves the best. If the decision-maker prefers middle or knee solutions, algorithms with

**Fig. 8.** Visual indicator of algorithm searches for dataset $a1$.

high HV would again be the best choice. In this work, we present three plots as visual indicators for three datasets with different number of requirements: $a1$ ($|R| = 50$), $c3$ ($|R| = 200$) and $d3$ ($|R| = 300$), although the complete set of images is available in the public repository.⁴ In Figs. 8–10, for each algorithm, we show the S found in one of the 30 executions. Since $|S|$ is set to 10, we plot the Pareto Front with a line across the 10 points. Evidently, statistical tests and values in tables are computed from the 30 executions. The PRef constructed is plotted in order to qualitatively assess the goodness of each algorithm with respect to the optimal Pareto front, and is also constructed using all the points from all the execution for all algorithms, thus seeking to define an ideal Pareto Front.

Fig. 8 shows that in the case of a small dataset, all the algorithms are capable of searching through the balanced (knee) solutions area. PBIL^a, MIMIC^a and FEDA-NRP are the only ones that find close solutions in the areas that minimise the Cost of development, while FEDA-NRP is the only one capable of finding extreme solutions in the area of Satisfaction maximisation. Similar behaviour is observed in the rest of the datasets with a low number of attributes. With this case being more common in agile project management, because requirements are created dynamically, any of the mentioned algorithms would be a good choice to solve the MONRP in such contexts. As shown in Table 5, in less complex datasets, there is no significant difference for HV, and so a good choice would be to use the fastest of the four mentioned algorithms, which are FEDA-NRP and PBIL^a (see Time discussion below).

With regards to classic management datasets with greater complexity in terms of requirements and dependencies, we can observe the behaviour of the algorithms in Figs. 9 and 10. The scatter plot in Fig. 9 shows that PBIL^a only remains close to PRef in the Cost minimisation area, and is unable to find solutions in the balanced (knee) or maximisation (extreme right) zones for Satisfaction. Thus, its performance in terms of HV decreases drastically with the increase in the complexity of the project, which is exactly the opposite in the case of FEDA. As can be seen, AGE-MOEA-II^a and C-TAEA^a solutions both fail at minimising Cost and can only maximise Satisfaction; furthermore, not only does FEDA-NRP find balanced solutions close to PRef, but is also able to find solutions along the entire search space, returning solutions which minimise Cost, maximise Satisfaction or balances both objectives.

A similar situation can be interpreted from Fig. 10: PBIL^a really contributes to the PRef in solutions that minimise Cost until the limit in which balanced (knee) solutions start. Exactly the opposite happens for AGE-MOEA-II^a and C-TAEA^a, which contribute to the PRef in solutions that maximise Satisfaction until the balanced solutions area starts,

⁴ https://github.com/UCLM-SIMD/MONRP/tree/eng_app_ai23.

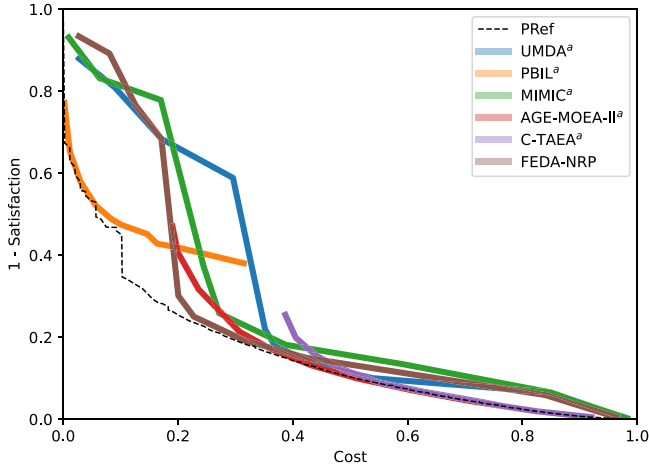


Fig. 9. Visual indicator of algorithm searches for dataset c3.

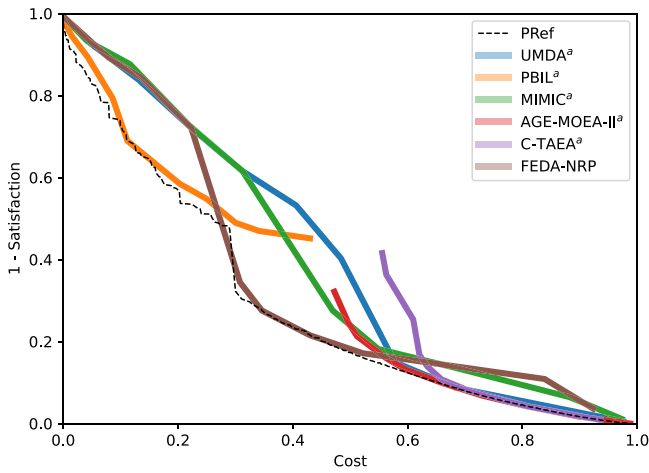


Fig. 10. Visual indicator of algorithm searches for dataset d3.

which is where FEDa-NRP outperforms in the search process, in terms of closeness to PRef and, in general, in HV. Thus, it is visually clear (corroborated from results in Table 4) that FEDa-NRP would be the best option to be applied in large and complex datasets, specially in those commonly found in classic project management.

We thus prove that our proposed method is able to successfully use the knowledge about dependencies in the learning and sampling phases of EDA, which truly helps to find good solution sets in large datasets while maintaining a linear memory complexity similar to univariate algorithms such as UMDA.

With the help of the visual indicators, we have found that MIMIC^a, UMDA^a and FEDa-NRP algorithms are those that explore the complete width of the search space, with FEDa-NRP being the one that gets closest to PRef, thus obtaining greater HV in the most complex datasets (especially for the classic-management datasets dX). Also in complex datasets, PBIL and genetic algorithms can only obtain good solutions for one of the objectives (left or right side of the PRef). Regarding UMDA^a, the solutions it finds are not excessively close to each other, which is why this algorithm obtained the best Δ -Spread values, so its strongest point would be the ability to provide more different solutions to the decision-maker.

Time

C-TAEA^a is the algorithm that takes longest to finish its search, taking from 1.5 (best case) to 2.5 h (worst case). This time does not just depend on the complexity of the dataset, but also on other two issues: (1) the complexity of the algorithm itself and (2) the cardinality

Table A.6

Number of wins for each set of hyperparameter configurations in the FEDa-NRP algorithm.

PopSize	#Iters.	Datasets	HV	Wins
1000	300	[a1, a2, c2, c3, d2, p1]	[0.873, 0.944, 0.932, 0.906, 0.808, 0.895]	6
1000	50	[a3, a4, d4]	[0.8244, 0.8047, 0.781]	3
1000	400	[c1, c4]	[0.8999, 0.8521]	2
1000	200	[d1, p2]	[0.8099, 0.7955]	2
1000	100	[d3]	[0.8011]	1

Table A.7

Number of wins for each set of hyperparameter configurations in the AGE-MOEa-II^a algorithm.

PopSize	#Iters.	Datasets	HV	Wins
1000	400	[a1,a2,a3,a4, c1,c2,c3,c4, d2,d3,d4,p2]	[0.8978,0.9891,0.7723,0.7456, 0.9242,0.9454,0.8692,0.795, 0.7211,0.6495,0.6302,0.7943]	12
700	400	[d1]	[0.7156]	1
100	200	[p1]	[0.9078]	1

of the $NDS_{archive}$, which depends on the capability of the algorithm to find non-dominated solutions in the search space. As the number of non-dominated solutions found increases, the filtering of $NDS_{archive}$ takes longer after each iteration.

The next most time-consuming algorithms are AGE-MOEa-II^a, which may take up to 50 min to run its search, and MIMIC^a with up to 30 min. Then, FEDa-NRP runs its search with a maximum duration of 13 min. Hence, we conclude that, in a real-world context in which plan-driven projects (to which dX datasets belong) schedule a very large set of requirements to be delivered in the following months, our results support that FEDa-NRP is the best algorithm to solve the MONRP. It is capable of finding the statistically best solutions in the global quality metric of HV, and is also about twelve times faster than C-TAEA^a, five times faster than AGE-MOEa-II^a. FEDa-NRP is also twice as fast as MIMIC^a, and statistically better in terms of HV, UNFR and GD+. Lastly, FEDa-NRP is the only algorithm that can provide the decision-maker with a variety of candidate solutions which range from both extreme areas (minimise Cost or maximise Satisfaction) to the balanced zones adjacent to the PRef in the search space from which to choose (as shown in the Visual indicators).

6. Conclusions and future work

The main contribution of this work is FEDa-NRP, a multivariate EDA to solve the MONRP which explicitly models dependencies between requirements. Embedding the knowledge about requirements interactions in the learning and sampling phases of an EDA helps to obtain good solution sets for the MONRP problem in complex datasets, in terms of hypervolume, balance and cardinality of the number of solutions. The use of the leaky binary noisy-OR gate in the model allows us to maintain a linear complexity while dealing with multivariate information.

The second contribution of the study is the creation of a benchmark of synthetic datasets covering different dimensions of requirements management in agile and classic software projects developments. The variety of this benchmark facilitated the evaluation of the tested algorithms in different regimes. We made this corpus publicly available to promote future research on MONRP, as well as our software to ensure reproducibility and fair comparison. In particular, in this paper, a rigorous experimental evaluation was carried out considering this corpus and involving six algorithms, the main conclusion of which is the superiority of FEDa-NRP over the rest of the tested algorithms when considering the most complex cases and the balance obtained between time required and accuracy of the solutions. It is in the case of complex problems (projects with a large number of requirements)

Table A.8
Number of wins for each set of hyperparameter configurations in the C-TAEA^a algorithm.

PopSize	#Iters.	Datasets	HV	Wins
1000	400	[a1, a2, a3, a4, c1, c2, c3, c4, d4, d1, d2, d3, p2]	[0.8375, 0.8902, 0.6647, 0.6471, 0.8459, 0.8024, 0.7467, 0.6696, 0.6104, 0.619, 0.5686, 0.5541, 0.6801]	13
100	400	[p1]	[0.8881]	1

Table A.9
Number of wins for each set of hyperparameter configurations in the UMDA^a algorithm.

PopSize	#Iters.	Datasets	HV	Wins
1000	50	[a1, a3, c3, c4, d1, p1]	[0.864, 0.803, 0.878, 0.820, 0.795, 0.878]	6
1000	400	[a2, d4]	[0.930, 0.752]	2
1000	300	[a4, c1]	[0.794, 0.882]	2
1000	200	[c2, d2]	[0.863, 0.802]	2
1000	100	[d3, p2]	[0.765, 0.771]	2

Table A.10
Number of wins for each set of hyperparameter configurations in the PBIL^a algorithm.

PopSize	#Iters.	Datasets	HV	Wins
1000	400	[a1, c1, c2, c3, c4, d1, d2, d3, d4, p2]	[0.897, 0.903, 0.942, 0.741, 0.734, 0.718, 0.693, 0.644, 0.645, 0.739]	10
700	400	[a2, a3, a4]	[0.974, 0.657, 0.658]	3
200	300	[p1]	[0.895]	1

Table A.11
Number of wins for each set of hyperparameter configurations in the MIMIC^a algorithm.

PopSize	#Iters.	Datasets	HV	Wins
1000	400	[a4, c1, c3, c4, d1, d4, p2]	[0.8051, 0.8885, 0.8841, 0.8165, 0.8026, 0.75, 0.8166]	7
1000	50	[a1]	[0.854]	1
1000	100	[a2]	[0.9141]	1
1000	300	[a3, c2, d3]	[0.8158, 0.8634, 0.7574]	3
1000	200	[d2]	[0.7962]	1
200	200	[p1]	[0.835]	1

that FEDA-NRP obtained the best HV results of all the algorithms. It is worth noting that its main competitors (in terms of HV) are also EDAs: UMDA and MIMIC, the latter taking twice as long as FEDA-NRP, and UMDA performing worse in terms of UNFR and GD+, besides HV. Another substantial advantage of FEDA-NRP with respect to these EDAs is that it finds the best balanced solutions close to the Pareto Reference, facilitating the decision-maker's choice for a candidate solution.

On the other hand, the main limitation of our proposal is that it requires as input the definition of all dependencies between requirements, although this is also the main motivation of our work. Without explicitly defined dependencies, the Bayesian network would be an empty graph, i.e., all variables would be independent of each other, so FEDA-NRP would be reduced to the UMDA algorithm.

As future work, it would be interesting to measure the complexity added by keeping the $NDS_{archive}$ updated during the search, and to optimise this process by updating the solution subset selection per iteration, instead of only once after the search has finished.

CRedit authorship contribution statement

Víctor Pérez-Piqueras: Software, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Pablo Bermejo:** Conceptualization, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration. **José A. Gámez:** Conceptualization, Investigation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared public links inside the paper to download the code and datasets.

Acknowledgements

This work has been funded by the project SBPLY/21/180225/000062 funded by the Government of Castilla-La Mancha and “ERDF A way of making Europe”. This work has also been supported by Universidad de Castilla-La Mancha, Spain and “ERDF A way of making Europe” under project 2023-GRIN-34437. It is also partially funded by MCIN/AEI/10.13039/501100011033 and “ESF Investing your future” through project PID2019-106758GB-C33. We also wish to thank the international Software Benchmarking Standards Group (ISBSG) for allowing us to use the 2015 version of their dataset for research purposes.

Appendix. Algorithm hyperparameter selection

In this section, we show the range of values for hyperparameters tuned in each algorithm included in our experiments, as well as the number of wins over datasets.

All the algorithms were configured with hyperparameters PopSize = {100, 200, 500, 700, 1000} and #Iterations = {50, 100, 200, 300, 400}. In total, 25 configurations per dataset.

With respect to concrete PBIL^a hyperparameters, we fixed learning rate, mutation probability and mutation shift to 0.1. UMDA and MIMIC used a selection scheme based on non-dominance and selection of a fixed number of 50 individuals.

The goodness of a configuration is measured in terms of the HV of the returned $NDS_{archive}$.

Since our global corpus contains 14 datasets (Table 1), the total number of wins for a given algorithm is 14. The configuration (row) with more wins is then selected as the configuration of the corresponding algorithm for the experiment results shown in Section 5.7. Please note that there are cases in which a configuration with fewer #Iterations or #PopSize than others obtain the same HV; that is, the algorithm converges quickly. In that case, the configuration with less #Iterations or #PopSize is considered as the winner.

Tables A.6–A.11 show the results for FEDA, AGE-MOEA-II^a, C-TAEA^a UMDA^a, PBIL^a and MIMIC^a, respectively. In each table, the configuration with a greater number of wins can be interpreted as the recommended best configuration, and is that applied in our experiments.

References

- Abdollahzadeh, B., Gharehchopogh, F.S., 2022. A multi-objective optimization algorithm for feature selection problems. *Eng. Comput.* 38 (Suppl 3), 1845–1863. <http://dx.doi.org/10.1007/s00366-021-01369-9>.
- Alba, E., Ferrer, J., Villalobos, I., 2021. Metaheuristics and software engineering: Past, present, and future. *Int. J. Softw. Eng. Knowl. Eng.* 31 (09), 1349–1375. <http://dx.doi.org/10.1142/S0218194021500443>.
- Almeida, J., Pereira, F., Reis, M., Piva, B., 2018. The next release problem: Complexity, exact algorithms and computations. In: 5th International Symposium, ISCO 2018, Marrakesh, Morocco, 2018, Revised Selected Papers. pp. 26–38. http://dx.doi.org/10.1007/978-3-319-96151-4_3.
- Bagnall, A.J., Raymond-Smith, V.J., Whitley, I.M., 2001. The next release problem. *Inf. Softw. Technol.* 43 (14), 883–890. [http://dx.doi.org/10.1016/S0950-5849\(01\)00194-X](http://dx.doi.org/10.1016/S0950-5849(01)00194-X).
- Baker, P., Harman, M., Steinhöfel, 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. pp. 176–185. <http://dx.doi.org/10.1109/ICSM.2006.56>.
- Baluja, S., 1994. Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Tech. rep, USA.
- Beck, K., 1999. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Publishing Company.
- Bermejo, P., de la Ossa, L., Puerta, J.M., 2011. Global feature subset selection on high-dimensional datasets using re-ranking-based EDAs. In: Advances in Artificial Intelligence - 14th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2011, la Laguna, Spain, November 7–11, 2011. Proceedings. In: Lecture Notes in Computer Science, vol. 7023, Springer, pp. 54–63. http://dx.doi.org/10.1007/978-3-642-25274-7_6.
- Blank, J., Deb, K., 2020. Pymoo: Multi-objective optimization in python. *IEEE Access* 8, 89497–89509. <http://dx.doi.org/10.1109/ACCESS.2020.2990567>.
- Carlshamre, P., Sandahl, K., Regnell, B., Dag, J., 2001. An industrial survey of requirements interdependencies in software product release planning. pp. 84–91. <http://dx.doi.org/10.1109/ISRE.2001.948547>.
- Chaves-Gonzalez, J.M., Perez-Toledano, M., Navasa, A., 2015. Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm. *Knowl.-Based Syst.* 83, <http://dx.doi.org/10.1016/j.knosys.2015.03.012>.
- Chaves-González, J.M., Pérez-Toledano, M.A., Navasa, A., 2015. Teaching learning based optimization with Pareto tournament for the multiobjective software requirements selection. *Eng. Appl. Artif. Intell.* 43, 89–101. <http://dx.doi.org/10.1016/j.engappai.2015.04.002>.
- Chen, T., Li, M., 2023. The weights can be harmful: Pareto search versus weighted search in multi-objective search-based software engineering. *ACM Trans. Softw. Eng. Methodol.* 32 (1), 5:1–5:40. <http://dx.doi.org/10.48550/arXiv.2202.03728>.
- Coello Coello, C.A., Lamont, G.B., Veldhuizen, D.A.V., 2007. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer US, <http://dx.doi.org/10.1007/978-0-387-36797-2>.
- Deb, K., 2001. *Multi-Objective Optimization using Evolutionary Algorithms*. Jhon Wiley and Sons Ltd, New York.
- Deb, K., Jain, H., 2014. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Trans. Evol. Comput.* 18 (4), 577–601. <http://dx.doi.org/10.1109/TEVC.2013.2281535>.
- 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. <http://dx.doi.org/10.1109/4235.996017>.
- del Sagrado, J., del Águila, I.M., Orellana, F.J., 2011. Requirements interaction in the next release problem. In: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation. Association for Computing Machinery, pp. 241–242. <http://dx.doi.org/10.1145/2001858.2001994>.
- del Sagrado, J., del Águila, I., Orellana, F., 2015. Multi-objective ant colony optimization for requirements selection. *Empir. Softw. Eng.* 20, 577–610. <http://dx.doi.org/10.1007/s10664-013-9287-3>.
- Domínguez-Ríos, M.Á., Chicano, F., Alba, E., del Águila, I., del Sagrado, J., 2019. Efficient anytime algorithms to solve the bi-objective Next Release Problem. *J. Syst. Softw.* 156, 217–231. <http://dx.doi.org/10.1016/j.jss.2019.06.097>.
- Dong, S., Xue, Y., Brinkkemper, S., Li, Y.-F., 2022. Multi-objective integer programming approaches to next release problem — Enhancing exact methods for finding whole Pareto front. *Inf. Softw. Technol.* 147 (C), <http://dx.doi.org/10.1016/j.infsof.2022.106825>.
- Dorigo, M., Birattari, M., Stutzle, T., 2006. Ant colony optimization. *IEEE Comput. Intell. Mag.* 1 (4), 28–39. <http://dx.doi.org/10.1109/MCI.2006.329691>.
- Durillo, J., Zhang, Y., Alba, E., Harman, M., Nebro, A., 2011. A study of the bi-objective next release problem. *Empir. Softw. Eng.* 16, 29–60. <http://dx.doi.org/10.1007/s10664-010-9147-3>.
- Durillo, J., Zhang, Y., Alba, E., Nebro, A., 2009. A study of the multi-objective next release problem. In: Proceedings - 1st International Symposium on Search Based Software Engineering, SSBSE 2009. <http://dx.doi.org/10.1109/SSBSE.2009.21>.
- Finkelstein, A., Harman, M., Mansouri, S.A., Ren, J., Zhang, Y., 2009. A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. *Requir. Eng.* 14 (4), 231–245. <http://dx.doi.org/10.1007/s00766-009-0075-y>.
- Gad, A.G., 2022. Particle swarm optimization algorithm and its applications: A systematic review. *Arch. Comput. Methods Eng.* 29, 2531–2561. <http://dx.doi.org/10.1007/s11831-021-09694-4>.
- Geng, J., Ying, S., Jia, X., Zhang, T., Liu, X., Guo, L., Xuan, J., 2018. Supporting many-objective software requirements decision: An exploratory study on the next release problem. *IEEE Access* 6, 60547–60558. <http://dx.doi.org/10.1109/ACCESS.2018.2875122>.
- Greer, D., Ruhe, G., 2004. Software release planning: An evolutionary and iterative approach. *Inf. Softw. Technol.* 46, 243–253. <http://dx.doi.org/10.1016/j.infsof.2003.07.002>.
- Gupta, P., Arora, I., Saha, A., 2016. A review of applications of search based software engineering techniques in last decade. In: 2016 5th International Conference on Reliability, Infocom Technologies and Optimization, ICRITO 2016: Trends and Future Directions (978). IEEE, pp. 584–589. <http://dx.doi.org/10.1109/ICRITO.2016.7785022>.
- Guyon, I., Elisseeff, A., 2003. An introduction to variable and feature selection. *J. Mach. Learn. Res.* 3, 1157–1182.
- Hamdy, A., Mohamed, A.A., 2019. Greedy binary particle swarm optimization for multi-objective constrained next release problem. *Int. J. Mach. Learn. Comput.* 9 (5), 561–568. <http://dx.doi.org/10.18178/jimlc.2019.9.5.840>.
- Harman, M., Mansouri, A., Zhang, Y., 2012a. Search-based software engineering: Trends, techniques and applications. *ACM Comput. Surv.* 45, 1–61. <http://dx.doi.org/10.1145/2379776.2379787>.
- Harman, M., McMinn, P., de Souza, J.T., Yoo, S., 2012b. Search based software engineering: Techniques, taxonomy, tutorial. In: Meyer, B., Nordio, M. (Eds.), *Empirical Software Engineering and Verification: International Summer Schools*. Springer Berlin Heidelberg, pp. 1–59. http://dx.doi.org/10.1007/978-3-642-25231-0_1.
- Henrion, M., 1988. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In: Lemmer, J.F., Kanal, L.N. (Eds.), *Uncertainty in Artificial Intelligence*. In: Machine Intelligence and Pattern Recognition, vol. 5, North-Holland, pp. 149–163. <http://dx.doi.org/10.1016/B978-0-444-70396-5.50019-4>.
- Imani, T., 2017. Does a hybrid approach of agile and plan-driven methods work better for IT system development projects? *Int. J. Eng. Res. Appl.* 07, 39–46. <http://dx.doi.org/10.9790/9622-0703043946>.
- Iqbal, U., Alam, K.A., 2021. Next release problem: A systematic literature review. *KIET J. Comput. Inf. Sci.* 3 (1), 65–78.
- Ishibuchi, H., Pang, L.M., Shang, K., 2020. 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. <http://dx.doi.org/10.1109/SMC42975.2020.9282850>.
- Ishibuchi, H., Pang, L.M., Shang, K., 2022. Difficulties in fair performance comparison of multi-objective evolutionary algorithms [research frontier]. *IEEE Comput. Intell. Mag.* 17 (1), 86–101.
- Jiang, H., Zhang, J., Xuan, J., Ren, Z., Hu, Y., 2010. A hybrid ACO algorithm for the next release problem. In: The 2nd International Conference on Software Engineering and Data Mining. pp. 166–171.
- Karim, M.R., Ruhe, G., 2014. Bi-objective genetic search for release planning in support of themes. In: Le Goues, C., Yoo, S. (Eds.), *Search-Based Software Engineering*. Springer International Publishing, Cham, pp. 123–137. http://dx.doi.org/10.1007/978-3-319-09940-8_9.
- Katoch, S., Chauhan, S.S., Kumar, V., 2021. A review on genetic algorithm: past, present, and future. *Multimedia Tools Appl.* 80 (5), 8091–8126. <http://dx.doi.org/10.1007/s11042-020-10139-6>.
- Knowles, J., Cornes, D., 1999. The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimization. In: Proceedings of the Congress on Evolutionary Computation. pp. 98–105. <http://dx.doi.org/10.1109/CEC.1999.781913>.
- Koller, D., Friedman, N., 2009. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press.
- Larrañaga, P., Etxeberria, R., Lozano, J.A., Peña, J.M., 2000. Combinatorial 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. Morgan Kaufmann, pp. 343–352. <http://dx.doi.org/10.48550/arXiv.1301.3871>.
- Larrañaga, P., Lozano, J.A., 2001. Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, USA, <http://dx.doi.org/10.1007/978-1-4615-1539-5>.
- Larrañaga, P., Lozano, J., 2002. Estimation of distribution algorithms: a new tool for evolutionary computation. <http://dx.doi.org/10.1007/978-1-4615-1539-5>.
- Li, K., Chen, R., Fu, G., Yao, X., 2019. Two-archive evolutionary algorithm for constrained multiobjective optimization. *IEEE Trans. Evol. Comput.* 23 (2), 303–315. <http://dx.doi.org/10.1109/TEVC.2018.2855411>.
- Li, M., Chen, T., Yao, X., 2020. How to evaluate solutions in Pareto-based search-based software engineering: A critical review and methodological guidance. *IEEE Trans. Softw. Eng.* 48 (5), 1771–1799. <http://dx.doi.org/10.1109/tse.2020.3036108>.

- Marghny, M., Zanaty, E., Dukhan, W., Reyad, O., 2022. A hybrid multi-objective optimization algorithm for software requirement problem. *Alex. Eng. J.* 61 (9), 6991–7005. <http://dx.doi.org/10.1016/j.aej.2021.12.043>.
- Maza, S., Touahria, M., 2019. Feature selection for intrusion detection using new multi-objective estimation of distribution algorithms. *Appl. Intell.* 49 (12), 4237–4257. <http://dx.doi.org/10.1007/s10489-019-01503-7>.
- Mühlenbein, H., 1997. The equation for response to selection and its use for prediction. *Evol. Comput.* 5 (3), 303–346. <http://dx.doi.org/10.1162/evco.1997.5.3.303>.
- Onisko, A., Druzdzal, M.J., Wasylyuk, H., 2001. Learning Bayesian network parameters from small data sets: application of Noisy-OR gates. *Internat. J. Approx. Reason.* 27 (2), 165–182. [http://dx.doi.org/10.1016/S0888-613X\(01\)00039-1](http://dx.doi.org/10.1016/S0888-613X(01)00039-1).
- Panichella, A., 2022. An improved Pareto front modeling algorithm for large-scale many-objective optimization. In: *Proceedings of the Genetic and Evolutionary Computation Conference. GECCO '22*, Association for Computing Machinery, pp. 565–573. <http://dx.doi.org/10.1145/3512290.3528732>.
- Pelikan, M., Goldberg, D.E., Cantú-Paz, E., 1999. BOA: The Bayesian optimization algorithm. In: *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1. GECCO '99*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 525–532. <http://dx.doi.org/10.5555/2933923.2933973>.
- Pérez-Piqueras, V., López, P.B., Gámez, J.A., 2022. Datasets for the next release problem (agile and classic costs). <http://dx.doi.org/10.5281/zenodo.7247877>.
- Pérez-Piqueras, V., López, P.B., Gámez, J.A., 2022. GRASP-based hybrid search to solve the multi-objective requirements selection problem. In: *Optimization and Learning - 5th International Conference, OLA 2022, Syracuse, Sicilia, Italy, July 18-20, 2022*, Proceedings. In: *Communications in Computer and Information Science*, vol. 1684, Springer, pp. 189–200. http://dx.doi.org/10.1007/978-3-031-22039-5_15.
- Pérez-Piqueras, V., López, P.B., Gámez, J.A., 2023. 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)*. In: *Lecture Notes in Networks and Systems*, Springer, pp. 98–108. http://dx.doi.org/10.1007/978-3-031-18050-7_10.
- PMI (Ed.), 2021. *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, seventh ed. Project Management Institute.
- Rahimi, I., Gandomi, A.H., Nikoo, M.R., et al., 2022. A comparative study on evolutionary multi-objective algorithms for next release problem. <http://dx.doi.org/10.21203/rs.3.rs-1929133/v1>, Preprint.
- Ramírez, A., Delgado-Pérez, P., Ferrer, J., Romero, J.R., Medina-Bulo, I., Chicano, F., 2020. A systematic literature review of the SBSE research community in Spain. *Prog. Artif. Intell.* 9 (2), 113–128. <http://dx.doi.org/10.1007/s13748-020-00205-3>.
- Ren, Z., Luo, Z., Xuan, J., Jiang, H., 2012. Solving the large scale next release problem with a backbone-based multilevel algorithm. *IEEE Trans. Softw. Eng.* 38 (05), 1195–1212. <http://dx.doi.org/10.1109/TSE.2011.92>.
- Rohmer, J., 2020. Uncertainties in conditional probability tables of discrete Bayesian Belief Networks: A comprehensive review. *Eng. Appl. Artif. Intell.* 88, 103384. <http://dx.doi.org/10.1016/j.engappai.2019.103384>.
- Sagarna, R., Lozano, J.A., 2005. On the performance of estimation of distribution algorithms applied to software testing. *Appl. Artif. Intell.* 19 (5), 457–489. <http://dx.doi.org/10.1080/08839510590917861>.
- Schwaber, K., Sutherland, J., 2020. *The Scrum Guide*. Scrum.org.
- Souza, J., Maia, C., Ferreira, T., Carmo, R., Brasil, M., 2011. An ant colony optimization approach to the software release planning with dependent requirements. pp. 142–157. http://dx.doi.org/10.1007/978-3-642-23716-4_15.
- Srinivas, N., Deb, K., 1994. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.* 2 (3), 221–248. <http://dx.doi.org/10.1162/evco.1994.2.3.221>.
- Veldhuizen, D.A.V., Lamont, G.B., 1998. *Evolutionary Computation and Convergence to a Pareto Front*. Stanford University, Morgan Kaufmann, California, pp. 221–228.
- Zhang, Y., Harman, M., Afshin, S., 2007. The multi-objective next release problem. In: *Proceeding of the 9th Annual Conference on Genetic and Evolutionary Computation*. pp. 1129–1137. <http://dx.doi.org/10.1109/SSBSE.2010.16>.
- Zitzler, E., Thiele, L., 1998. Multiobjective optimization using evolutionary algorithms — A comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (Eds.), *Parallel Problem Solving from Nature — PPSN V*. Springer Berlin Heidelberg, pp. 292–301. <http://dx.doi.org/10.1007/BFb0056872>.