Evaluating the performance of NSGA-II and NSGA-III on Product Line Architecture Design

Lucas Wolschick, Paulo Cesar Gonçalves*, João Choma Neto, Willian Marques Freire, Aline Maria Malachini Miotto Amaral, Thelma Elita Colanzi
State University of Maringa
Maringa, Parana, Brazil
{ra123658,pg55461}@uem.br,{joao.choma,willianmarquesfreire}@gmail.com,{ammmamaral,teclopes}@uem.br

ABSTRACT

Product Line Architecture (PLA) design can be modeled as an optimization problem to be solved with search-based algorithms. PLA design optimization has successfully been done using the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) in scenarios involving up to three objectives, which involve software metrics for properties such as feature modularization, PLA extensibility, and cohesion. As many of these properties may be desired in a PLA at the same time, more than three objectives might need to be optimized simultaneously. The Non-Dominated Sorting Genetic Algorithm III (NSGA-III) was designed to solve problems impacted by more than three objectives, named many-objective problems, so it might suit this need. However, NSGA-III has not yet been applied in the context of PLA design. In this sense, this study aims to compare the performance of NSGA-II and NSGA-III for PLA design to uncover which algorithm best fits this problem. To accomplish this goal, we implemented a specialized version of NSGA-III and then ran experiments using both algorithms to optimize eight PLAs with three, four, and five objectives. We evaluate the algorithms' performance via quality indicators commonly used in search-based software engineering. The empirical results point out that: *(i)* NSGA-III had a slightly better performance than NSGA-II when optimizing four or five objectives in the context of our study; *(ii)* NSGA-II was the best or the algorithms tied when the PLA given as input is easier to optimize due to reduced solution space.

KEYWORDS

product-line architecture, search-based software engineering, multi-objective evolutionary algorithm

1 INTRODUCTION

A Software Product Line Engineering (SPL) consists of core assets with common and explicit variable features [23], which are used to derive products. The Product Line Architecture (PLA) encompasses the central design model of an SPL, and therefore, it needs to be generic and flexible enough to support the configuration of many distinct products. For an experienced architect, recognizing a good PLA design might be easy, but achieving one is not. Finding the PLA design that best combines different metrics demands a lot from the architect. Metrics may conflict, and in these cases architects need to choose which of them should be prioritized [1, 6, 25]. The conflict arises from the various possibilities for defining PLA modeling, which consider several different factors that can influence the PLA design each. This may lead into more than one possible solution existing for a given problem [23].

These trade-offs make PLA design suitable to be modeled as a multi-objective optimization problem [17]. Search-Based Software Engineering (SBSE) is a research field that aims to model software engineering problems as computational search problems, which can be solved with metaheuristics, to find near-optimal solutions automatically [18]. In the context of PLA design, a difficult and laborious process, SBSE techniques can be useful in decreasing human effort when exploring design alternatives.

The Multi-Objective Approach for Product Line Architecture Design (MOA4PLA) [9] applies SBSE techniques to optimize several architectural properties of a PLA design simultaneously. MOA4PLA has been implemented in an open-source tool named OPLA-Tool [16] using NSGA-II (Non-Dominated Sorting Genetic Algorithm II) [13], the most used multi-objective evolutionary algorithm (MOEA) in SBSE studies [7]. NSGA-II is a multi-objective algorithm that allows the simultaneous optimization of more than one objective (factor). For PLA optimization, the objectives consist of software metrics related to the architectural properties to be optimized.

NSGA-II shows satisfactory results when optimizing up to three objectives for PLA design [15, 28], which represent specific metrics in the PLA context that indicate the quality of the optimized solutions. However, when the architect wishes to optimize more than three objectives at once, NSGA-II does not maintain good results, in part due to a bigger search space and increased complexity of the problem [11]. In this situation, an algorithm specifically designed for this task (many-objective optimization) may provide better results instead, but to the best of our knowledge, no many-objective algorithms have been applied in the context of PLA design yet.

The Non-Dominated Sorting Genetic Algorithm III (NSGA-III) [12] was designed to solve problems containing more than three objectives, known as many-objective problems, and has already been applied in many contexts (see Section 7). Ishibuchi et al. [20] compared NSGA-II and NSGA-III using canonical problems usually studied in the optimization area. Their empirical results showed that NSGA-III does not consistently outperform NSGA-II even for ten-objective problems. They discovered that the results depend on the number of objectives and the characteristics of the problem being solved, which consists of PLA design for our context. In addition, their study found that the problem type had a larger effect on the results than the number of objectives. In the SPL context, NSGA-III was compared to NSGA-II to optimize SPL testing using three objective functions [21]. The results indicated that NSGA-III produced better results than NSGA-II for two SPL feature models.

*Also with Universidade Tecnológica Federal do Paraná.

Acknowledging that an SPL architect may be interested in optimizing more than three objectives simultaneously during PLA design optimization, and considering that NSGA-III was designed for many-objective problems, and that stochastic algorithms, such as NSGA-II and NSGA-III, are usually assessed by empirical studies following guidelines for SBSE studies [3, 7], an investigation on the performance of NSGA-III for PLA design becomes needed. To fulfill this gap, in this work we specialized an implementation of NSGA-III to OPLA-Tool and then conducted an empirical study comparing NSGA-II and NSGA-III in terms of many-objective optimization. Therefore, this study aims to analyze the performance of both algorithms in the context of PLA design optimization. Their performance is measured with the Hypervolume, and Euclidean distance to the ideal solution quality indicators, which are commonly used in the SBSE field [22].

A total of 48 experiments were conducted evaluating eight PLA designs, varying the number of objectives to be optimized (three, four, and five objectives). Our main findings are: *(i)* NSGA-III had a slightly better performance for many-objective optimization (four or five objectives); *(ii)* NSGA-II was the best or there was a tie between both algorithms in several cases, justified by the characteristics of the PLA design under optimization, which is less complex, leading to lower solution space. So, our results corroborate the literature [20] as they showed the results depend on the number of objectives and the problem characteristics to be solved.

In summary, the main contributions of our work are:

- we contribute an additional option of optimization algorithm to OPLA-Tool – NSGA-III – alongside existing ones, benefiting the research community that uses such a tool and allowing the efficient optimization of more than three objectives simultaneously;

- we enhance the state of the art with empirical results of the performance comparison between NSGA-II and NSGA-III in the PLA design context. This contribution is important to guide researchers and practitioners in choosing which MOEA to apply in their empirical studies and design problems.

2 PRODUCT LINE ARCHITECTURE DESIGN

Product Line Architectures are critical assets within the context of SPL Engineering. The development of a PLA demands significant effort to entail a design that involves all the commonalities and variabilities of the SPL's products. In addition, developing a PLA is an iterative, technical process that aims for maintainable, reusable architectures that meet desirable properties such as feature modularization, extensibility, and cohesion, among other factors. However, these factors can conflict, which leads to multiple modeling alternatives for a specific PLA design scenario.

SBSE techniques [17] can be applied to automate the optimization of PLA design, offering various near-optimal design alternatives based on predefined objectives. To assess these factors, software metrics are used as objectives that can be optimized through the search process. To achieve this goal, Colanzi et al. [9] proposed the approach named MOA4PLA (Multi-Objective Approach for Product Line Architecture Design), which employs Multi-Objective Evolutionary Algorithms (MOEAs) to evaluate and enhance the PLA design concerning architectural properties. MOA4PLA allows the application of various MOEAs in PLA optimization.

According to Harman, Mansouri, and Zhang [19], two essential components are required to solve a problem using search algorithms: *(i)* a suitable problem representation that allows for symbolic manipulation; and *(ii)* an evaluation function, formulated based on the representation and specific metrics, to measure the proposed solutions' quality. In MOA4PLA, the first component is developed in the activity named Construction of the PLA Representation, while the second is established in the activity Definition of the Evaluation Model. Following these activities, the search algorithm is applied to generate a set of solutions (PLA representations). In the final step, these solutions are refined and returned, so that the software architect can choose the most appropriate one.

Within the scope of MOA4PLA, a PLA is represented by a class diagram that captures the static structural dimension of the architecture [9]. This representation is achieved through a metamodel, facilitating the manipulation of architectural elements by optimization algorithms. A PLA includes various architectural elements such as components, interfaces, and operations and their respective relationships. Each of these architectural elements is linked to one or more features through UML stereotypes, being either a common element across all products in the SPL or specific to some products, denoting its variability. Variable elements are connected to elements defining variation points and their possible variants [9].

OPLA-Tool is the open-source tool that fully automates the application of MOA4PLA [16]. The tool's optimization process begins with an XMI (XML Metadata Interchange) file containing the original PLA design. This file serves as the input to the approach, where, during the XMI conversion, each identified element in the XMI is transformed into an object within the metamodel, following its classification. After this point, the PLA representation will include objects that reflect the architectural components, the relationships among them, the variabilities, and the features associated with each architectural element [8].

MOA4PLA has an evaluation model, detailed in [29], which incorporates a set of objective functions derived from software metrics. These functions are essential for assessing key aspects of PLA design alternatives. Such functions measure properties such as feature modularization, PLA extensibility, variability, design elegance, coupling, cohesion, and size. The objective functions quantify the quality of the PLA design and its ability to meet these criteria during the optimization process. The SPL architect must select a subset of objective functions to be used during the process.

Among the objective functions encompassed in the model, five functions used in previous studies [8, 9, 15, 28] were also used in this study, namely COE (Relational Cohesion), ACLASS (Class Coupling), FM (Feature-based Metrics), CM (Conventional Metrics), and Eleg (Elegance). The COE function measures the cohesion of a PLA design through the internal relationship among its classes. ACLASS quantifies class coupling by counting the number of architectural elements that depend on other classes in the design and the number of elements on which each class depends. In turn, the FM function evaluates feature modularization in a PLA design, considering feature-based cohesion, feature tangling, and feature dispersion on architectural elements. CM is the sum of conventional cohesion, coupling, and size metrics. Finally, Eleg measures the elegance of an object-oriented design. The specific equations for calculating these functions and their respective metrics are detailed in [29].

MOA4PLA employs mutation and crossover operators to optimize the provided PLA design. Mutation operators also create the initial population from the original PLA design. OPLA-Tool allows different MOEAs to be chosen for optimizing PLA designs together with the mutation and crossover operators of MOA4PLA. Many MOEAs, such as NSGA-II and PAES, are readily available for use, having been implemented by the original authors of the tool. Users can also analyze the generated solutions after the optimization process with the tool.

3 SEARCH-BASED OPTIMIZATION

Many problems in Software Engineering (SE), including PLA design, are good candidates for being solved with search-based techniques due to their inherently complex and multi-faceted nature. Proposing solutions to a hard SE problem requires careful analysis and consideration of the problem's requirements, which can often be conflicting. This implies that often, no solution is single-handedly better than all the others. There are often many good solutions for a given problem, with different sets of desirable characteristics each, and finding and evaluating these alternatives is challenging and time-consuming. The software architect is responsible for exploring these alternatives in the design space and then picking whichever is the most adequate for the problem. By using search algorithms to generate alternatives, the architect can instead focus only on evaluating the generated solutions and their trade-offs.

In SBSE, SE problems are modeled as optimization problems, with their requirements numerically encoded as constraints and/or objective functions. In multi-objective or many-objective optimization problems, where there is more than one objective function being optimized at once, solutions can be described in terms of the values it assumes for each objective. These values can be considered components of a vector which positions the solution in a solution space. The solution space can contain feasible and unfeasible solutions, i.e., solutions that do not meet the problem constraints.

When comparing two solutions in a minimization problem, we say that a solution dominates another if it is numerically smaller in one dimension than the other and numerically equal to or smaller than the other in the remaining dimensions. If neither solution dominates the other, they are said to be non-dominated.

The set of all non-dominated points in the solution space forms the Pareto front, which corresponds to the set of solutions that are all optimal in one way or another but have different trade-offs. Therefore, we want to generate solutions that pertain to and are widely distributed in the Pareto front so that the software architect can explore and evaluate many design alternatives.

3.1 Multi-Objective Evolutionary Algorithms

Genetic algorithms are a class of metaheuristics composed of three types of search operators: selection, crossover, and mutation. All metaheuristics, including genetic algorithms, are stochastic algorithms, and a randomness factor is associated with the application of search operators [17, 18].

MOEAs are a generalization of genetic algorithms for multi-objective optimization problems, and are commonly used in many SBSE tasks. These algorithms maintain and iteratively improve a set of solutions, named population, in a manner inspired by evolution and natural selection theories.

Each iteration is called a generation; in every generation, each solution undergoes the steps of selection, crossover, and mutation. First, pairs of solutions in the population are selected with a selection operator and then crossed with a crossover operator. Usually, the selection favors the fittest solutions, so their traits, assumed to be more desirable, can be replicated and passed onwards. The crossover operator mixes traits of both solutions to generate a new solution, which can be subsequently randomly altered by a mutation operator. Lastly, after all descendant solutions are generated, a final population is constructed by mixing the ancestor and descendant solutions. This process is repeated until a desired stopping condition is reached.

The randomness factor associated with the selection, crossover, and mutation operators helps maintain a diverse population with members in many solution space regions. As the selection step favors fitter solutions, the overall effect is that after each generation, the resulting population converges closer to the Pareto front while maintaining diversity, which is our main goal with MOEAs.

Non-dominating Sorting Genetic Algorithm II (NSGA-II) [13] is an MOEA that aims to preserve diversity through elitism and a parameterless niching operator. Elitism in this context means that the best individuals from a given generation are preserved in the next generation without modification.

NSGA-II works by initially sorting the entire population into different "domination levels" and then inserting entire levels into the next population until there is no space for an entire level anymore. The remaining slots in the new population are filled by picking solutions from the least crowded regions in the solution space to maximize population diversity. Crowding in the solution space is approximated by a metric named "crowding distance", computed through the perimeter of the cuboid defined by a solution's two nearest neighbors [13].

NSGA-II is an effective algorithm widely used for multi-objective optimization problems where there are three or fewer objective functions [12], but it copes poorly when used with four or more objective functions (a scenario called many-objective optimization). Introducing new objective functions reduces the number of dominated solutions, as there are more ways in which one solution can tie with another, and makes estimating crowding of the solution space more expensive due to the increased number of dimensions. Furthermore, the higher dimensionality requires a greater quantity of solutions in the population so that the Pareto front can be effectively represented, which unacceptably harms the performance of NSGA-II and other popular MOEAs [12].

Non-dominating Sorting Genetic Algorithm III (NSGA-III) is a proposed improvement over NSGA-II aimed at many-objective optimization problems [12]. This version of the algorithm replaces the previous technique for measuring crowding in the solution space with a reference-point-based niching procedure.

In this procedure, a set of reference points in the solution space is uniformly generated or supplied by the user. All reference points have an associated ray that crosses itself and the origin, and in each generation, new solutions are assigned to the reference point whose ray has the smallest perpendicular distance to the solution. When all solutions are assigned, each reference point will have a

number of solutions and their representatives associated with them. In order to construct the next generation, after all non-dominated fronts that fit in the new population have been wholly included, exactly as in NSGA-II, remaining slots in the population are filled by including representatives from the reference points that are the least represented in the new population. New solutions are included in the population until it is filled. This procedure preserves diversity in the solution space and is more efficient in that goal than the previous crowding measure, allowing for an arbitrarily high number of objectives to be picked for optimization.

In the original NSGA-III paper, the authors suggested a method based on Das and Dennis's approach [10] for systematically generating reference points on a normalized hyperplane, which intercepts each objective axis at coordinate one. The points are distributed on the hyperplane according to a specified number of subdivisions, p, and the number of objectives M so that they uniformly represent the entire Pareto front. The total number of generated points, H, is given by Equation 1. This number is taken as the population size to be used in the remainder of the algorithm [12].

$$H = \binom{M + p - 1}{p} \tag{1}$$

In their experiments, Deb and Jain used varying values of p depending on the problem and number of objectives. For three-objective problems, they set $p = 12$, which is the value we adopted for all NSGA-III experiments.

Many different quality metrics have been included as objective functions of MOA4PLA. Whether the use of multiple objectives has a positive impact on the generated solutions or not is an open question. As such, this work aims to investigate whether NSGA-III can be used as an effective search-based algorithm in MOA4PLA under a many-objective optimization context, especially when compared with NSGA-II.

3.2 Quality Indicators

As previously mentioned, NSGA-II and NSGA-III generate a set of non-dominated solutions. The performance assessment of such algorithms is usually done through quality indicators that evaluate different aspects of the set of resulting solutions [22]. In our study, we used the indicators Hypervolume and Euclidean Distance to the ideal solution.

To support the result analysis and computing the quality indicator, we composed three sets of solutions, namely, PF_{approx}, PF_{known}, and PF_{true}. PF_{approx} is the Pareto front of non-dominated solutions obtained in each run of an experiment. As we run each experiment 30 times, we have 30 PF_{approx} sets for each experiment. PF_{known} is the set of non-dominated solutions found by an experiment, considering the union of all solutions obtained in all its runs, eliminating the dominated ones. PF_{true} is conceptually known as the set with ideal solutions for a problem. As the PF_{true} of our problem is not known in advance, we adopted a common way to estimate this Pareto front: using the non-dominated solutions found by all algorithms in all runs [32].

The Hypervolume (HV) allows analyzing the convergence and diversity of solutions [22]. HV is a n-dimensional volume between PF_{true} and a specific reference point [32]. The higher the hypervolume, the greater the coverage area, reflecting a better front.

Euclidean Distance to the Ideal Solution (ED) is a distance measure that designates the closest solution to an "ideal solution". For minimization problems, the ideal solution is that one with the lowest value possible for the objective function being optimized [31]. The PF_{true} set is used to find the solution with the lowest ED of each experiment. The solution with the lowest ED tends to be preferred by the decision-makers as it represents the best trade-off among the objectives.

4 IMPLEMENTATION OF NSGA-III FOR PLA DESIGN

In this section, we briefly describe the current architecture of OPLA-Tool 4.1, and then we explain how we implemented NSGA-III to the OPLA-Tool code base 4.2.

4.1 Architecture of OPLA-Tool

OPLA-Tool v2[1] [16] is a software system organized in a set of modules, each of which handles a different portion of the application. In an overview, the modules can be roughly divided into two halves: those which implement the web-based interface with which the users can interact to set up experiments with desired parameters, and those in the back-end, which are responsible for taking the experiment requests, performing them, and returning their results so they can be shown to the user. The back-end modules, in conjunction, provide the actual implementation of the MOA4PLA process. Some of the responsibilities handled by the back-end modules are, for instance: providing common types for the rest of the application; implementing a REST HTTP API so the web-based interface can communicate with the back-end; and actually implementing and performing the optimization process using search algorithms, such as the ones specified in Section 3. We focused our implementation efforts on the second half of these modules.

OPLA-Tool v2 uses the free and open-source multi-objective optimization framework jMetal 4 [14], which is a Java library that provides a series of search algorithms and common interfaces for them, for use in the development of multi-objective optimization applications. OPLA-Tool v2's original code had a heavily modified fork of jMetal 4 included as a vendored dependency and relied on it for its implementation of MOA4PLA. This version of jMetal includes an implementation of NSGA-II, which was specialized by the tool's original authors for use with MOA4PLA.

Nonetheless, although OPLA-Tool v2 provides NSGA-II, it did not provide an implementation of NSGA-III, which is the focus of this comparative study. We wanted to assess whether NSGA-III could deliver better results than NSGA-II in the context of PLA optimization, and so incorporating an implementation of NSGA-III into OPLA-Tool became necessary.

4.2 Backporting methodology

Firstly, we compared different alternatives for bringing NSGA-III into OPLA-Tool. We could either implement the algorithm from scratch, port it from an existing implementation in another library or language, or attempt to backport a newer version of the jMetal framework, which contained an implementation of NSGA-III. Out

[1] https://github.com/otimizes/OPLA-Tool

of all three options, implementing the algorithm ourselves was judged the most unfeasible — there is no publicly available reference implementation from the original authors of the NSGA-III paper, which is light in detail in this regard, and our implementation would need to be thoroughly tested for correctness. We also considered the second option, which is porting an existing implementation of the algorithm into the code base, but ultimately, we chose to backport a newer version of the jMetal framework with NSGA-III into OPLA-Tool, due to similarities between different versions of jMetal and to avoid unfair comparisons between different implementation styles.

Due to the changes done to jMetal 4 described earlier in this section, lifting the entire project to jMetal 6 was deemed infeasible as a result of the amount of changes the existing codebase would need. Instead, we chose to bring over only the NSGA-III implementation with its minimal supporting code, changing interfaces and method signatures where possible so that code written for the new version could interface adequately with the old version of the framework.

The backport was conducted as follows: first, all transitive code dependencies for the NSGA-III implementation in jMetal 6 were identified and then copied over to the existing codebase. Second, dependencies were deduplicated where possible and merged with existing interfaces found in jMetal 4, adjusting any method signatures. Third, the new NSGA-III implementation was exposed in the API and in the web interface and then validated using tests present in OPLA-Tool. Lastly, any leftover unused code was culled from the application and the remaining code was tidied up.

During the backport, we aimed to adapt the implementation to the same guidelines and patterns used for the existing NSGA-II implementation, by existing interfaces components, in order to minimize major differences between both implementations. In effect, only those components strictly related to NSGA-III which were not found in the existing OPLA-Tool code base were backported alongside NSGA-III itself.

After finishing the backport, we tested the resulting implementation to verify whether NSGA-III was working correctly in the existing OPLA-Tool v2. We also adapted a few of the existing automated tests in the repository so that the new implementation could be tested. Based on comparison runs with NSGA-II and NSGA-III, we attested that the implementation behaved as intended and concluded the backport. The resulting implementation is ready and available for use on the latest version of OPLA-Tool, which was used in the development of this study.

5 EMPIRICAL STUDY DEFINITION

5.1 Study Design

This section outlines the study's design conceived to analyze the performance of the existing NSGA-II algorithm and our implementation of NSGA-III in optimizing three, four, and five objectives. In our context, the performance of an algorithm refers to how effective it is in obtaining high-quality solutions in terms of the values of the objective functions and the quality indicators obtained from the objective function values. The objective functions express the quality of the achieved solutions in terms of the PLA design properties. The study includes two product lines in four different versions for each SPL. In this regard, the subjects of this empirical study are the PLA design alternatives resulting from the search process. The PLA designs and all objects of analysis, charts, spreadsheets, and measuring instruments are organized in the Experimental Package [2]. Details of the study design are presented below.

Planning. We used eight PLA designs, designed for two different SPLs. They are described in Section 5.2. As we are interested in comparing the performance of NSGA-III and NSGA-II for PLA design, we formulated the following hypotheses:

H_0: The NSGA-III algorithm, within the context of the OPLA-Tool, has equal or worse performance than NSGA-II for many-objective optimization (four or more objectives).

H_1: The NSGA-III algorithm, within the context of the OPLA-Tool, performs better than NSGA-II for many-objective optimization (four or more objectives).

Selection of Variables. The independent variables are the PLA, the Search-based Algorithm, and the Algorithm Configuration. The dependent variables are the objective functions, namely ACLASS, COE, FM, ELEG, and CM, and the quality indicators, namely HV and ED, obtained from the objective function values of the solutions generated in the experiments.

The PLA factor comprises eight treatments, consisting of four versions of AGM and four versions of MM PLAs (see Section 5.2). The Algorithm Configuration factor has one configuration, while the MOEA includes two treatments: the NSGA-II and NSGA-III algorithms. The configurations utilized in the Algorithm are detailed in the 'Operation' section. We employed five objective functions for the study: ACLASS, COE, FM, ELEG, and CM, from which the quality indicators HV and ED are assessed. These functions were selected because they are commonly used by previous studies involving MOA4PLA and OPLA-Tool. We categorized the objective functions into three distinct combinations: three objectives (ACLASS, COE, FM), four objectives (ACLASS, COE, FM, ELEG), and five objectives (ACLASS, COE, FM, ELEG, CM), as outlined in Table 1. In total, we worked with three evaluation configurations for each PLA design.

Operation. We used OPLA-Tool v2 to carry out the empirical study with the following base configuration: for NSGA-II, the Population Size was set to 200, the Number of Generations was 150, the Number of Fitness Evaluations was set to 30,000, and Mutation Rate of 0.9. For NSGA-III, the same parameters were used, but the population size varied according to the number of objectives: for three objectives, the Population Size was set to 91, for four objectives it was set to 455, and for five it was set to 1820 (these values are derived from Equation 1 using $p = 12$ and M set to three, four and five respectively). These base settings were then used with different combinations of objectives and PLAs. Eight PLA designs were used, four AGM versions and four MM versions (see Table 2). For each PLA, we executed three combinations of objectives: (i) ACLASS,

[2] https://doi.org/10.6084/m9.figshare.25583643

Table 1: Experiment Configurations

# Objectives	Experiment	MOEA	Objective Functions
3	II3OF	NSGA-II	ACLASS, COE, FM
	III3OF	NSGA-III	ACLASS, COE, FM
4	II4OF	NSGA-II	ACLASS, COE, FM, ELEG
	III4OF	NSGA-III	ACLASS, COE, FM, ELEG
5	II5OF	NSGA-II	ACLASS, COE, CM, FM, ELEG
	III5OF	NSGA-III	ACLASS, COE, CM, FM, ELEG

COE, FM, (ii) ACLASS, COE, FM, ELEG, and (iii) ACLASS, COE, FM, ELEG, CM. We carried out 30 runs of the search-based algorithm for each configuration, as recommended in [3]. The experiments executed in the study were named according to the second column of Table 1. The entire study involving the NSGA-II and NSGA-III algorithms resulted in 720 runs of each algorithm.

The experiments were conducted on machines with different CPU and RAM configurations. Due to this variation in the computational resources configuration, comparing the execution time of each experiment was not viable. This is not a problem as we are interested in assessing the performance of search-based algorithms in terms of quality of solutions for PLA design optimization.

Analysis and Interpretation: Data collected during the experiment execution was used to compute quality indicators for evaluating the algorithm's performance. To obtain the ideal solution, used to calculate the ED value for each generated solution, we used PF_{true}. In our work, the ideal solution has the minimum value obtained for each objective, since MOA4PLA addresses PLA design as a minimization problem. The HV was calculated using the PF_{known} generated per PLA configuration. Since the values used to calculate HV were normalized between zero and one, the reference point was set at 1.01, to represent the worst possible solution.

Furthermore, for HV values, we employed the statistical tests. Firstly, we analyzed if the sample sets have normal distribution using Shapiro-Wilk test with a confidence level of 95%. Then, to statistically compare the results of HV, we applied the Kruskal-Wallis Pairwise test for datasets with non-normal distribution. For datasets with a normal distribution, we utilized ANOVA. In both tests, we maintained a confidence level of 95% (p-value $\leq$0.05) to ascertain statistical differences among the sample sets. Additionally, we computed the effect size using the Vargha-Delaney's $\hat{A}_{12}$ measure for further analysis. The gathered data was structured into multiple files to facilitate statistical analysis, utilizing R Studio and Google Sheets software.

5.2 Used PLA designs

Versions of two SPLs were used in our study, namely Arcade Game Maker (AGM) and Mobile Media (MM). AGM [27] was created by the Software Engineering Institute (SEI). It is composed of three arcade games: Brickles, Bowling and Pong. MM [30] is a mobile application composed of features that handle music, video, and photos for mobile devices.

The eight PLA designs were obtained from the OPLA-Tool repository [3]. All versions of each SPL capture different element structures and feature modularization patterns to evaluate PLA optimization strategies at distinct points. In the AGM family, AGM-2 was created from AGM-1 by adding two new features: ranking and logging. AGM-3 and AGM-4 contain the same architectural elements as AGM-1 and AGM-2, respectively; however, the features are mapped to different elements than the other two versions. For the MM family, MM-1 and MM-2 have different numbers of components, and the MediaMgr component is less coupled and more cohesive in the MM-2 version. MM-3 and MM-4 were obtained from MM-1 and MM-2, respectively, but have different feature mappings. Such differences were proposed to exercise the optimization process in different situations and in order to avoid deriving results too tied to the specificities of any specific design(s). Each version has a different number of components, interfaces, classes, and variabilities, as shown in Table 2.

[3]https://github.com/otimizes/OPLA-Tool/tree/master/plas/PLASMarty

Table 2: Numbers of Architectural Elements of the PLAs

PLA	Components	Interfaces	Classes	Variabilities	Features
AGM1	9	14	20	4	9
AGM2	9	14	21	5	11
AGM3	9	14	20	4	9
AGM4	9	14	21	5	11
MM1	11	16	13	7	13
MM2	8	13	10	7	13
MM3	11	16	13	7	13
MM4	8	13	10	7	12

6 EMPIRICAL RESULTS

In this section, we present the results obtained from our empirical study. We extracted the quantitative results through the objective function values of the solutions resulting from the optimization process. The results were analyzed using the HV and ED quality indicators, and statistical tests were applied to examine the behavior of the solutions. All results, including independent and dependent variables, are organized in the experimental package [4].

Table 3 presents the number of non-dominated solutions each algorithm achieves in each experiment (PF_{known}). Clearly, for each SPL, the higher the number of objectives, the higher the number of solutions.

Hypervolume. As mentioned, HV was measured within the range of zero to one. The higher the Hypervolume (HV) value, the better the results, indicating better solution space exploration. First, we used the HV values in the normality test of Shapiro-Wilk before evaluating the statistical difference in the performance between NSGA-II and NSGA-III. Then, we employed the Kruskal-Wallis test for the data sets with non-normal distribution and the ANOVA test for data with normal distribution, as explained in Section 5.1. The data were verified with a confidence level of 95%.

Table 4 presents the medians of the Hypervolume indicator by number of objectives. Values highlighted in bold represent the best significant HV value. This table also presents which test was applied to assess the statistical difference between the algorithms, the p-values, and the effect size according to the Vargha-Delaney test. The hatched cells refer to results with statistical significance.

[4]https://doi.org/10.6084/m9.figshare.25583643

Table 3: Number of Non-dominated Solutions

PLA	3OF		4OF		5OF	
	NSGA-II	NSGA-III	NSGA-II	NSGA-III	NSGA-II	NSGA-III
AGM1	23	19	79	92	92	129
AGM2	21	6	55	103	144	146
AGM3	29	29	182	197	376	198
AGM4	19	16	169	62	527	238
MM1	16	12	63	62	127	111
MM2	11	13	47	37	160	85
MM3	19	9	69	48	163	73
MM4	7	9	51	35	133	64

Table 4: Median of Hypervolume and Results of Kruskal-Wallis (K)/ANOVA (A) Statistical Tests

PLA	3OF					4OF					5OF				
	NSGA-II	NSGA-III	Test	p-value	Effect size[a]	NSGA-II	NSGA-III	Test	p-value	Effect size[a]	NSGA-II	NSGA-III	Test	p-value	Effect size[a]
AGM1	**0.41**	0.32	K	< 0.001	0.8589▲	0.34	**0.46**	A	< 0.001	0.1544▲	**0.39**	0.25	A	< 0.001	0.9533▲
AGM2	0.42	0.47	K	0.668	0.4678≃	0.45	**0.51**	A	0.019	0.2944△	**0.39**	0.29	A	< 0.001	0.833▲
AGM3	**0.60**	0.48	A	< 0.001	0.9167▲	**0.51**	0.47	A	< 0.001	0.8711▲	**0.45**	0.39	A	< 0.001	0.9389▲
AGM4	**0.49**	0.26	A	< 0.001	0.8411▲	0.36	**0.58**	A	< 0.001	0.0000▲	**0.37**	0.34	A	0.004	0.6967△
MM1	**0.56**	0.47	K	< 0.001	0.8267▲	0.39	0.40	K	0.525	0.4522≃	0.32	**0.40**	K	< 0.001	0.1500▲
MM2	**0.67**	0.56	K	< 0.001	0.8500▲	0.46	0.45	A	0.190	0.5833∇	0.37	**0.51**	K	< 0.001	0.0289▲
MM3	0.29	**0.34**	K	0.012	0.3111△	0.39	0.38	K	0.894	0.5100≃	0.24	**0.29**	K	< 0.001	0.2089▲
MM4	0.72	0.70	K	0.383	0.5656≃	0.59	0.55	A	0.029	0.6600∇	0.41	**0.59**	A	< 0.001	0.0378▲

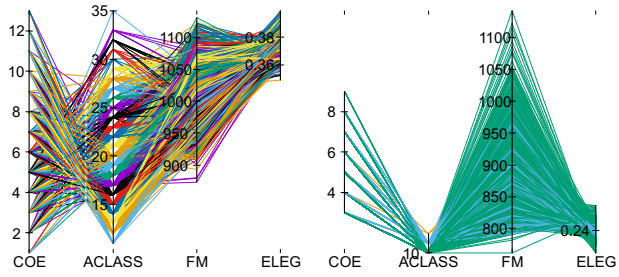
[a] Values in gray cells indicate statistical difference. The symbols for the magnitude of the effect size are: "≃" negligible, "∇" small magnitude, "△" a medium magnitude, and "▲" a large magnitude

Table 5: Lowest Euclidean Distance to the Ideal Solution

PLA	3OF		4OF		5OF	
	NSGA-II	NSGA-III	NSGA-II	NSGA-III	NSGA-II	NSGA-III
AGM1	7.62	**6.40**	**6.71**	11.66	**6.62**	24.38
AGM2	6.71	**5.39**	11.58	**10.30**	**4.58**	19.70
AGM3	10.05	**9.00**	**6.00**	9.54	**8.97**	15.04
AGM4	6.40	**5.39**	10.82	**5.00**	**15.22**	23.35
MM1	6.16	**6.10**	6.40	**5.10**	**8.69**	12.58
MM2	**8.00**	10.00	5.83	5.83	**10.14**	11.46
MM3	10.00	**5.00**	9.00	**6.08**	10.94	**9.34**
MM4	**7.00**	8.00	**4.00**	5.00	**18.63**	31.34

Table 6: Median of ED of the Non-dominated Solutions

PLA	3OF		4OF		5OF	
	NSGA-II	NSGA-III	NSGA-II	NSGA-III	NSGA-II	NSGA-III
AGM1	**62.30**	76.78	69.21	**63.53**	**76.00**	109.48
AGM2	71.37	68.11	**82.88**	90.77	**83.13**	108.98
AGM3	**34.80**	63.51	79.54	**65.63**	69.85	67.98
AGM4	**66.48**	117.67	139.18	138.03	141.92	**113.90**
MM1	68.62	**49.25**	105.15	101.12	**116.62**	120.32
MM2	60.13	59.03	**98.41**	122.07	144.12	138.23
MM3	252.07	**210.06**	199.13	**181.07**	226.34	**129.71**
MM4	**72.06**	96.13	**121.02**	147.00	159.26	**117.15**



(a) $NSGA - II - AGM4 - 4OF$ **(b)** $NSGA - III - AGM4 - 4OF$

Figure 1: Parallel coordinates for AGM4 - Experiments II4OF and III4OF

For AGM versions, NSGA-II achieved better hypervolume than NSGA-III for three cases with three objectives, three cases with four objectives, and four cases with five objectives, almost all with large magnitude. NSGA-III was the best in only one case with four objectives. There was also one tie for AGM2 with three objectives.

For MM versions, considering three objectives, we observed that NSGA-II was better than NSGA-III for MM1 and MM2 with large magnitude. For MM3, NSGA-III was the better. Considering four objectives, NSGA-III overcame NSGA-II for MM4, and both algorithms tied for the other three cases. For five objectives, NSGA-III achieved the best results in hypervolume with high magnitude.

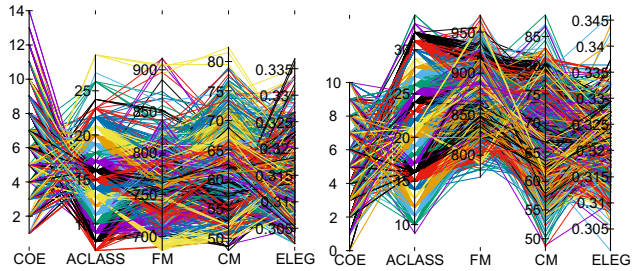
Euclidean Distance to the Ideal Solution. For this quality indicator, we are interested in the solution with the lowest ED because it represents the solution with the best compromise among all objectives optimized during the search. After analyzing which is the solution with the lowest ED by experiment, we present in Table 5 the value of the lowest ED achieved in each experiment. Table 6 shows the median value of the solutions' ED for each experiment. As PLA design was modeled as a minimization problem, the lower the ED value, the better the result. The best values for each pair of experiments are highlighted in bold. There is no predominance of better results, as NSGA-III found the solution with the lowest ED for 11 cases whereas NSGA-II achieved the lowest ED for 12 cases.

For AGM, each algorithm generated the solution with the lowest ED for six cases. For MM, NSGA-II achieved the lowest ED in six cases, whereas NSGA-III was the best in this indicator in five cases. Both algorithms were tied in one case (MM2 - 4OF).

Given such results, we decided to analyze the median of ED, looking for a trend. These numbers are presented in Table 6, where we can also note both algorithms achieved good results for this indicator without a common behavior. The statistical test pointed out some cases where there are significant differences, but there are also several ties. The best significant results are highlighted in bold. Considering only the significant cases, NSGA-II had the best ED median in six cases for AGM versions, while NSGA-III achieved the best ED medians in three cases. For MM versions, NSGA-III had the best ED median in five cases, whereas NSGA-II was the best in four cases for this SPL.

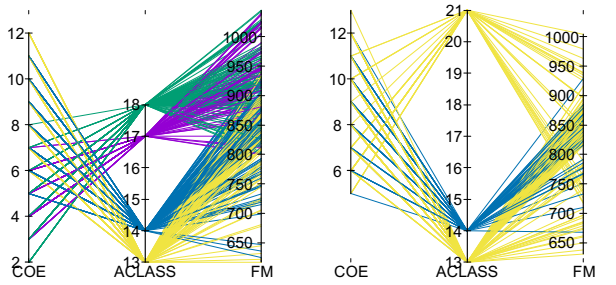
6.1 Discussion

As we observed no predominant behavior from the quality indicator results, we discuss the results in this section by analyzing the fitness of the non-dominated solutions, shedding light on some particularities of the PLA designs used in our study. For doing so, we rely on parallel coordinates graphics, such as the ones depicted in Figure 1b. Such kind of graphic presents the fitness of each solution for the set of non-dominated solutions (PF_{known}) obtained by an algorithm in an experiment considering the 30 independent runs. Each line in the picture represents a solution, demonstrating the trade-off among the objective functions in the solution sets.



(a) $NSGA - II - AGM1 - 5OF$ **(b)** $NSGA - III - AGM1 - 5OF$

Figure 2: Parallel coordinates for AGM1 - Experiments II5OF and III5OF



(a) $NSGA - II - MM3 - 3OF$ **(b)** $NSGA - III - MM3 - 3OF$

Figure 3: Parallel coordinates for MM3 - Experiments II3OF and III3OF

Figure 1 presents the set of non-dominated solutions (PF_{known}) obtained by NSGA-II and NSGA-III for AGM4 using four objectives (Experiments II4OF and III4OF) as parallel coordinate graphs. Subfigures (a) and (b)'s objective bars are aligned and on the same scales. In this experiment, NSGA-III had a better median HV and a similar median ED to NSGA-II. We can note in Figure 1 that, although NSGA-II was able to maintain a wider spread of values for COE and ACLASS, NSGA-III managed to attain lower ACLASS and ELEG values which are concentrated around narrower, smaller ranges. Since our optimization problem is minimization, in this example, NSGA-III's ability to find good solutions in the presence of many objectives becomes evident, even if it converged heavily to a small set of possible values for ACLASS and ELEG. This may indicate that, at some point in the search process, the search space collapsed to a smaller, more optimized set of values for NSGA-III.

On the other hand, Figure 2 depicts a parallel coordinates graph for the experiments involving the optimization of five objectives for AGM1, where surprisingly NSGA-II achieved the best results for both quality indicators. Both algorithms generated a similar number of solutions (Table 3). Here, NSGA-III's quicker convergence to narrower ranges worked against it: NSGA-II maintained a good spread across all objectives while NSGA-III lost lower-valued portions of its ACLASS and FM axes. Its range of FM values is approximately 200 units smaller and higher than NSGA-II. This harmed NSGA-III's HV and ED values and was responsible for its reduced performance compared to NSGA-II. This may again indicate that the solution search space was reduced irreversibly at some point during NSGA-III's search process, losing access to a valuable class of solutions (those with lower FM values).

Figure 3 presents the parallel coordinates for the experiments that optimize three objectives for MM3. While NSGA-II maintained a wider diversity of solutions for the ACLASS axis, maintaining representatives in four different values, and generated solutions with lower COE values, NSGA-III obtained a richer solution trade-off distribution by generating and keeping a portion of its population with high ACLASS and low COE and FM values, represented in the parallel coordinate graphs by the lines crossing the ACLASS axis at value 21. Furthermore, most of its solutions have ACLASS values near the lower end of its range, which helped bring down the median ED value for this solution set. This allowed it to beat NSGA-II in that scenario for both ED and HV, despite having less unique values for ACLASS and a higher-valued COE distribution as well. This goes in line with the previous graphs, which suggest that NSGA-III converged to a smaller region of the search space.

Overall, NSGA-II seemed to maintain a more even distribution of values across the objective axes than NSGA-III, which converged on a smaller set of values more quickly, probably due to its reference-point-based niching procedure (see Section 3.1), — when these values were lower than NSGA-II's, NSGA-III beat NSGA-II. In the opposite scenario, NSGA-II beat NSGA-III.

The solution space is directly impacted by the original PLA design given as input to the search-based algorithm. The possible values for feature modularization (FM) are impacted by the number of features and their mapping in the architectural elements. For instance, AGM2 and AGM4 have one additional variability and two additional features, enabling more alternative solutions in the solution space than AGM1 and AGM3. Analyzing Figures 1b and 2, ACLASS seems more conflicting with FM for AGM4 than for AGM1, impacting the solution space. We can also note the impact on the range of FM values, whose highest value is greater for AGM4. As seen in Table 2, MM3 is the version of MM PLA with the highest number of architectural elements. It is also the version of MM with the worst feature modularization. These two characteristics provide more opportunities for PLA design optimization, enabling NSGA-III to achieve better results as discussed above.

Given such empirical results, we cannot prove the hypothesis H_1 because it is clear that for the context of our empirical study, the performance of NSGA-III and NSGA-II depends on both the number of objectives and the characteristics of the PLA design, corroborating the findings of Ishibuchi et al. [20].

6.2 Threats to Validity

In this section, we present the threats to the validity of the results achieved in this study.

Internal Validity. The sample used in the study poses a threat to the internal validity of the results due to the use of academic SPL. However, this threat was mitigated using PLAs from different domains, one related to media management for mobile devices (MM) and another to arcade games (AGM). To diversify, we explored four PLA design versions for each of the domains used, as presented in Table 2, which had differences in the arrangement of architectural elements. The parameter values adopted to configure NSGA-II and NSGA-III impact on the algorithms´ performance. To mitigate this

threat we used the values adopted in previous studies on PLA design optimization for both algorithms. The only difference is in the population size which varies for NSGA-III according to the number of objectives under optimization, as proposed by its authors. Another internal threat is related to the randomness of the algorithms used, NSGA-III and NSGA-II. To mitigate this threat, we followed a recommendation by Arcuri and Briand [2], executing the experimental studies 30 times and analyzing the set of results obtained in the 30 independent rounds. Another threat to the study's internal validity is related to the infrastructure configurations used on the servers to execute the experiments. However, this did not affect the study's results, as we did not measure performance in terms of execution time but rather the quality of the solutions.

External Validity. The sample size is identified as a threat to external validity, as a sample with eight PLA designs was used. The sample consists of four versions of the AGM and four versions of the MM. Thus, it is not possible to generalize the results identified in the study to different samples. However, with the samples used in the study, it was possible to identify some differences between the algorithm's performance and corroborate the literature's findings.

Conclusion Validity. We minimized the main threats to conclusion validity identified during the study using quality indicators and statistical tests commonly used in SBSE studies [4, 16–18].

7 RELATED WORK

NSGA-III has already been used to solve many different SE problems and has been compared with NSGA-II in those contexts.

In Carvalho et al. [5], the authors adopted NSGA-III as the base algorithm to automatically extract microservices from legacy systems using SBSE techniques. The problem was modeled as a five-objective optimization problem, and their choice of using NSGA-III was taken based on the results of an experimental comparison between NSGA-II and NSGA-III. The comparison was done on a restricted version of the original problem with only two objective functions, but nonetheless NSGA-III attained better hypervolume and inverted generational distance (IGD) values than NSGA-II (these quality indicators were used to assess their performance).

Mkaouer et al. [24] proposed an SBSE approach for finding good sequences of code refactorings using NSGA-III. They used a set of fifteen software quality metrics related to aspects such as complexity, coupling, and cohesion of the resulting code as objectives. The authors performed a comparative analysis using subsets of three to ten objectives from the original 15 with the MOEAs NSGA-III, IBEA, MOEA/D, and NSGA-II. They found that NSGA-III presented the best convergence among all test cases and attained the best IGD metric when using all objectives, which suggests that the use of more objectives can lead to better results in SBSE problems.

Ishibuchi et al. [20] also compared NSGA-II and NSGA-III in different many-objective problems with three to ten objectives. The authors found that both the number of objectives and the nature of the problem being solved impacted the performance of MOEAs, and noted that, for their problem and parameter sets, the type of the problem had a larger impact on the performance than the number of objectives. They also verified that NSGA-III did not always outperform NSGA-II, even for ten-objective problems. Finally, they also remarked that results may vary a lot with different implementations of each algorithm, especially regarding NSGA-III, which did not have a "canonical" implementation as the original papers did not provide source code for the algorithm [12]. In our study, we compared NSGA-II and NSGA-III implementations originating from the same framework (jMetal) aiming to minimize this risk.

Ramirez et al. [26] realized a comparative study of eight different MOEAs, including NSGA-II and NSGA-III, with the problem of software architecture discovery, or extracting an existing architecture from a code base. The authors ran the eight MOEAs with subsets of two to nine objective functions, totaling 256 configurations. They reported that, when comparing NSGA-II against NSGA-III, NSGA-II attained better HV values on average than NSGA-III. Still, NSGA-III generated better-spaced solutions as the number of objectives grew from two to four. In configurations from six to nine objectives, NSGA-III performed better than NSGA-II, although it ranked worse than other many-objective algorithms, such as ϵ-MOEA. These results corroborate the findings in [20] which suggest that the algorithms' performance may depend on the problem and on the number of objectives.

Jamil et al. [21] do a comparative study between NSGA-II and NSGA-III to solve the problem of testing SPLs. They ran both algorithms with only one case using three objective functions. Their results indicated that NSGA-III produced better results than NSGA-II.

In general, NSGA-III was found to be a competitive algorithm, outperforming or rivaling NSGA-II in most multi-objective problems. It may sometimes fall short of NSGA-II depending on the problem and the number of objectives used. There is a reasonable expectation that it may work well for some SBSE problems, and we found a perceived gap in the literature regarding its application in PLA optimization which this study aims to fill.

8 CONCLUDING REMARKS

We conducted an empirical study to assess the performance of the NSGA-III algorithm for PLA design optimization using OPLA-Tool. To this end, we performed 48 experiments evaluating eight PLAs in two domains, varying the number of objectives to be optimized (three, four, and five objectives). The performance of NSGA-II and NSGA-III were compared in terms of hypervolume and Euclidean distance to the ideal solution regarding the solutions found.

The empirical results, considering the quality indicators, pointed out a slightly better performance for NSGA-III when optimizing four or five objectives. However, this did not happen in all cases. More importantly, we observed that according to the PLA design characteristics, such as the feature modularization, the number of features, and the number of architectural elements, the performance of the algorithms varies. This demonstrates that NSGA-III does not always surpass NSGA-II, and that the results depend on both the number of objectives and the characteristics of the problem being solved, corroborating the literature's findings.

This suggests that NSGA-III may provide a more diverse set of solutions with more spread trade-offs in the solution space than NSGA-II for larger designs and many objectives. On the other hand, NSGA-II still remains competitive for less complex PLAs. With this in mind, we suggest that, when using many objectives for optimizing complex PLAs, practitioners use NSGA-III. On the other

hand, for simpler PLAs, the optimization algorithm must be chosen after careful analysis of the PLA characteristics, the number of objectives to be optimized, and previous literature results. Further research remains necessary in order to derive more actionable, hard practical guidelines for software architects and developers regarding algorithm choice for PLA design.

The study results enhance the state of the art with empirical findings for a performance comparison between NSGA-III and NSGA-II for the PLA design context. Additionally, it demonstrates that the NSGA-III algorithm could be an additional optimization algorithm option for OPLA-Tool for many-objective purposes.

In future work, we intend to conduct further studies involving larger, real PLAs to map how the characteristics of PLA design influence the results. Also, it is necessary to carry out other empirical studies with other objective functions and with a higher number of objectives aiming at discovering how the objective functions impacted the NSGA-III results for PLA design optimization. In such further studies, including additional quality indicators to provide other points of view of the search-based algorithms' performance would be interesting.

ACKNOWLEDGMENTS

This work is supported by CNPq grant 404027/2023-7 and CAPES - Finance Code 001.

REFERENCES

[1] Sven Apel and Dirk Beyer. 2011. Feature cohesion in software product lines: an exploratory study. In *Proc. of the ICSE'11* (Waikiki, Honolulu, HI, USA). New York, USA, 421–430.

[2] Andrea Arcuri and Lionel Briand. 2011. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proceedings of the 33rd international conference on software engineering*. 1–10.

[3] Andrea Arcuri and Lionel Briand. 2014. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Software Testing, Verification and Reliability* 24, 3 (2014), 219–250.

[4] Carlos Vinicius Bindewald, Willian M Freire, Aline M M Miotto Amaral, and Thelma Elita Colanzi. 2019. Towards the support of user preferences in search-based product line architecture design: an exploratory study. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering (SBES)*. 387–396. https://doi.org/10.1145/3350768.3351993

[5] Luiz Carvalho, Alessandro Garcia, Thelma Elita Colanzi, Wesley K. G. Assunção, Juliana Alves Pereira, Baldoino Fonseca, Márcio Ribeiro, Maria Julia de Lima, and Carlos Lucena. 2020. On the Performance and Adoption of Search-Based Microservice Identification with toMicroservices. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 569–580. https://doi.org/10.1109/ICSME46990.2020.00060

[6] Soo Ho Chang, Hyun Jung La, and Soo Dong Kim. 2006. Key Issues and Metrics for Evaluating Product Line Architectures. In *Proc. of the 18th Int. Conf. on Softw. Engineering & Knowledge Engineering (SEKE)* (San Francisco, CA, USA). 212–219.

[7] Thelma Elita Colanzi, Wesley K.G. Assunção, Silvia R. Vergilio, Paulo Roberto Farah, and Giovani Guizzo. 2020. The Symposium on Search-Based Software Engineering: Past, Present and Future. *Information and Software Technology* 127 (2020), 106372. https://doi.org/10.1016/j.infsof.2020.106372

[8] Thelma Elita Colanzi and Silvia Regina Vergilio. 2016. A feature-driven crossover operator for multi-objective and evolutionary optimization of product line architectures. *Journal of Systems and Software* 121 (2016), 126–143. https://doi.org/10.1016/j.jss.2016.02.026

[9] Thelma Elita Colanzi, Silvia Regina Vergilio, Itana Gimenes, and Willian Nalepa Oizumi. 2014. A search-based approach for software product line design. In *Proceedings of the 18th International Software Product Line Conference (SPLC)*, Vol. 1. 237–241. https://doi.org/10.1145/2648511.2648537

[10] Indraneel Das and J. E. Dennis. 1998. Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. *SIAM J. on Optimization* 8, 3 (mar 1998), 631–657. https://doi.org/10.1137/S1052623496307510

[11] Kalyanmoy Deb and Himanshu Jain. 2012. Handling many-objective problems using an improved NSGA-II procedure. In *2012 IEEE Congress on Evolutionary Computation*. 1–8. https://doi.org/10.1109/CEC.2012.6256519

[12] Kalyanmoy Deb and Himanshu Jain. 2014. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation* 18, 4 (2014), 577–601.

[13] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6 (2002), 182–197. https://doi.org/10.1109/4235.996017

[14] Juan J. Durillo and Antonio J. Nebro. 2011. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software* 42, 10 (2011), 760–771. https://doi.org/10.1016/j.advengsoft.2011.05.014

[15] Willian Freire, Cláudia Rosa, Aline Amaral, and Thelma Colanzi. 2022. Validating an Interactive Ranking Operator for NSGA-II to Support the Optimization of Software Engineering Problems. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering (SBES)*. 337–346.

[16] Willian Marques Freire, Mamoru Massago, Arthur Cattaneo Zavadski, Aline M M Miotto Amaral, and Thelma Elita Colanzi. 2020. OPLA-Tool v2.0: a Tool for Product Line Architecture Design Optimization. In *34th Brazilian Symposium on Software Engineering (SBES)*. https://doi.org/10.1145/3422392.3422498

[17] Mark Harman, Yue Jia, Jens Krinke, William B Langdon, Justyna Petke, and Yuanyuan Zhang. 2014. Search based software engineering for software product line engineering: a survey and directions for future work. In *Proceedings of the 18th International Software Product Line Conference-Volume 1*. 5–18.

[18] Mark Harman and Bryan F Jones. 2001. Search-based software engineering. *Information and software Technology* 43, 14 (2001), 833–839.

[19] Mark Harman, S Afshin Mansouri, and Yuanyuan Zhang. 2009. Search based software engineering: A comprehensive analysis and review of trends techniques and applications. *Department of Computer Science, King's College London, Tech. Rep. TR-09-03* (2009), 23.

[20] Hisao Ishibuchi, Ryo Imada, Yu Setoguchi, and Yusuke Nojima. 2016. Performance comparison of NSGA-II and NSGA-III on various many-objective test problems. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. 3045–3052. https://doi.org/10.1109/CEC.2016.7744174

[21] Muhammad Abid Jamil, Ahmad Alhindi, Muhammad Arif, Mohamed K Nour, Normi Sham Awang Abubakar, and Tareq Fahad Aljabri. 2019. Multiobjective Evolutionary Algorithms NSGA-II and NSGA-III for Software Product Lines Testing Optimization. In *2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*. 1–5. https://doi.org/10.1109/ICETAS48360.2019.9117500

[22] Miqing Li and Xin Yao. 2019. Quality Evaluation of Solution Sets in Multiobjective Optimisation: A Survey. *Comput. Surveys* 52 (03 2019), 1–38. https://doi.org/10.1145/3300148

[23] F. v. d. Linden, F. Schmid, and E. Rommes. 2007. *Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering*. Springer.

[24] Mohamed Wiem Mkaouer, Marouane Kessentini, Slim Bechikh, Kalyanmoy Deb, and Mel Ó Cinnéide. 2014. High dimensional search-based software engineering: finding tradeoffs among 15 objectives for automating software refactoring using NSGA-III. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation* (Vancouver, BC, Canada) *(GECCO '14)*. Association for Computing Machinery, New York, NY, USA, 1263–1270. https://doi.org/10.1145/2576768.2598366

[25] Klaus Pohl, Günter Böckle, and van Der Linden Frank J. 2005. *Software product line engineering: foundations, principles and techniques* (1 ed.). Springer Science & Business Media.

[26] Aurora Ramírez, José Raúl Romero, and Sebastián Ventura. 2016. A comparative study of many-objective evolutionary algorithms for the discovery of software architectures. *Empirical Software Engineering* 21 (2016), 2546–2600.

[27] SEI. 2009. *Software Engineering Institute - The Arcade Game Maker Pedagogical Product Line*. https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=485941. Accessed in 2023 April.

[28] Diego Fernandes Silva, Luiz Fernando Okada, Wesley K. G. Assunção, and Thelma Elita Colanzi. 2022. Intensifying the search-based optimization of product line architectures with crossover operators. *Empirical Software Engineering* 27 (2022), 166. https://doi.org/10.1007/s10664-022-10198-3

[29] Yenisei Delgado Verdecia, Thelma Elita Colanzi, Silvia Regina Vergilio, and Marcelo C Benitez Santos. 2017. An Enhanced Evaluation Model for Search-based Product Line Architecture Design.. In *CIbSE*. 155–168.

[30] T. Young. 2005. *Using AspectJ to Build a Software Product Line for Mobile Devices*. Master's thesis. University of British Columbia.

[31] Milan Zeleny and James L Cochrane. 1973. *Multiple criteria decision making*. University of South Carolina Press.

[32] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert Da Fonseca. 2003. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation* 7 (2003), 117–132.