Multi-Objective Software Effort Estimation: A Replication Study

Vali Tawosi, Federica Sarro, Alessio Petrozziello, Mark Harman

Abstract—Replication studies increase our confidence in previous results when the findings are similar each time, and help mature our knowledge by addressing both internal and external validity aspects. However, these studies are still rare in certain software engineering fields.
In this paper, we replicate and extend a previous study, which denotes the current state-of-the-art for multi-objective software effort estimation, namely CoGEE. We investigate the original research questions with an independent implementation and the inclusion of a more robust baseline (LP4EE), carried out by the first author, who was not involved in the original study. Through this replication, we strengthen both the internal and external validity of the original study.
We also answer two new research questions investigating the effectiveness of CoGEE by using four additional evolutionary algorithms (i.e., IBEA, MOCell, NSGA-III, SPEA2) and a well-known Java framework for evolutionary computation, namely JMetal (rather than the previously used R software), which allows us to strengthen the external validity of the original study.
The results of our replication confirm that: (1) CoGEE outperforms both baseline and state-of-the-art benchmarks statistically significantly ($p < 0.001$); (2) CoGEE's multi-objective nature makes it able to reach such a good performance; (3) CoGEE's estimation errors lie within claimed industrial human-expert-based thresholds. Moreover, our new results show that the effectiveness of CoGEE is generally not limited to nor dependent on the choice of the multi-objective algorithm. Using CoGEE with either NSGA-II, NSGA-III, or MOCell produces human competitive results in less than a minute. The Java version of CoGEE has decreased the running time by over 99.8% with respect to its R counterpart.
We have made publicly available the Java code of CoGEE to ease its adoption, as well as, the data used in this study in order to allow for future replication and extension of our work.

Index Terms—Software effort estimation; multi-objective evolutionary algorithm; confidence interval; estimates uncertainty.

1 Introduction

Many researchers have emphasised the idea of replication studies and its value to the Empirical Software Engineering (SE) field [15], [40], [100], [101].

Any body of knowledge mainly built upon experimental results, such as the Empirical SE one, need to be consolidated by performing an extensive verification of its experimental results [55].

This verification is possible by replicating an experiment to check whether its results are reproducible [55]: Replication increases our confidence if the results are similar each time. However, there are fields of software engineering where reproducibility and verification are still low [23].

Shepperd et al. [94] have recently performed a systematic review to identify replication experimental studies in the areas of software effort estimation up to August 2017. They found that there are only 22 unique articles replicating 30 original studies out of roughly 4600[1] published experiments on software project effort estimation [94].

We aim to further consolidate the body of knowledge in software effort estimation by carrying out an exact replication [101] of the work by Sarro et al. [89].[2] In their work, the authors propose a novel bi-objective software development effort estimation algorithm, named Confidence Guided Effort Estimation (CoGEE), which evolves multiple robust estimation models by simultaneously optimising for high accuracy and low uncertainty of the prediction [89]. CoGEE [89] has set the state-of-the-art for multi-objective effort estimation algorithms and has been the only one to achieve human-competitive results thus far [3].

In this paper, we replicate and extend their work [89] by further assessing CoGEE's effectiveness using the most recent best practice in search-based software engineering [6], [45], [67], [83] and predictive models for software engineering [88], [97], [109].

One important benefit of replications is that they help mature software engineering knowledge by addressing both internal and external validity aspects [101]. We have carried out a large empirical study involving 724 different software projects in order to replicate the same research questions by means of the same experimental design enriched by: (1) using a new robust state-of-the-art baseline benchmark

• *V. Tawosi, F. Sarro, A. Petrozziello, and M. Harman are with the Department of Computer Science, University College London, London, United Kingdom. M. Harman is also with Facebook London.*
E-mail: {vali.tawosi, f.sarro, a.petrozziello, mark.harman}@ucl.ac.uk

Manuscript received xxx; revised xxx.

1. Although it is hard to estimate the number of publications on effort estimation, Shepperd et al. attempt to corroborate this number with a general search on Scopus [94].

2. Shull et al. identify two types of replication: "*exact replications*, in which the procedures of an experiment are followed as closely as possible; and *conceptual replications*, in which the same research question is evaluated by using a different experimental procedure." Based on this categorization, our work is an exact replication.

3. Sarro et al. [89] won the ACM SIGEVO 13th Annual (2016) "Humies" Award for Human-Competitive Results Produced by Genetic and Evolutionary Computation http://www.human-competitive.org.

(i.e., Linear Programming for Effort Estimation) [88], which strength the conclusion and external validity of the previous study since the baseline used in the original study was found to be unstable [88]; (2) investigating four additional variants of Multi-Objective Evolutionary algorithms (MOEAs), each belonging to a different optimisation family, which enhances the external validity of the original study; (3) evaluating CoGEE's performance in terms of execution time; (4) a new and independent implementation of CoGEE in a different programming language (`Java`) based on a widely used evolutionary computation framework (namely `JMetal` [78]), which not only has decreased the running time with respect to the original `R` implementation, but has also strengthened the internal validity of the study.

The implementation of all approaches, the experiments and the analysis for the replicated research questions have been carried out independently by the first author of this paper who was not involved in the original study, though he had consulted with the authors through face to face interaction for explanations and clarifications. We used the same datasets as the original study to preserve the experimental context. As in the original study, we report our results in terms of Standard Accuracy (SA) and Mean Absolute Error (MAE) (see Section 2), and we test for statistical significance using the Wilcoxon Rank-Sum test (corrected with Bonferroni) with the Vargha-Delaney $\hat{A}_{12}$ non-parametric effect size.

The scientific findings of our replication are consistent with those reported in the original study. Specifically, they confirm that CoGEE outperforms both baseline and state-of-the-art techniques statistically significantly; and, also, that its error remains within the claimed thresholds for industrial best estimation practice [89]. We have also extended the previous findings by comparing the effectiveness of five different MOEAs (i.e., NSGA-II, NSGA-III, MOCell, IBEA, and SPEA2) as the underlying search algorithm for CoGEE. We found that three out of five algorithms produce similar high-quality solutions in similar running time; hence the effectiveness of CoGEE is generally not limited to nor depended on the choice of multi-objective algorithm. The two other algorithms (i.e., SPEA2 and IBEA) had a higher running time, and one of them (IBEA) produced lower quality solutions, although it could still outperform the baseline techniques.

We have made publicly available our `Java` implementation of CoGEE to facilitate its adoption in future studies [105]. The datasets used in the study are also available to facilitate the reproduction of our study [106].

The rest of the paper is organised as follows. Section 2 gives some background on software effort estimation, evaluation performance measures, search-based approaches for effort estimation, and CoGEE. Section 3 describes the design of our replication study in detail, including the research questions and the experimental method used to address them. Section 4 reports and discuss the results. Further insights on similarities and differences between the original study and this replication are discussed in Section 5. The validity of the study is discussed in Section 6. Section 7 reports on the related work, and Section 8 concludes the paper.

2 Software Effort Estimation

In software management and planning, producing an accurate estimation of the effort needed to complete or maintain a project is of great importance and concern. Meanwhile, estimating the most realistic amount of effort in the early stage of software development is difficult since the information available at that stage is usually incomplete and uncertain. Although construction of formal software effort estimation models started in the very early times of the industrialization of software production, expert judgement still remains the dominant strategy for effort prediction in practice where the accuracy of the estimate is sensitive to the practitioner's expertise and thus prone to bias [53], [87]. Early work to build an estimation technique tried to find a set of factors related to the software size and cost by using regression analysis [9]. With the advent of Artificial Intelligence-based techniques in software effort estimation, different approaches have been investigated, including analogy-based techniques (e.g., Case-Based Reasoning [98]), machine learning techniques (e.g., Classification and Regression Trees [59], Artificial Neural Networks [111], Support Vector Regression [80], Bayesian Networks [17]), Search-Based approaches [36] (e.g., Genetic Programming [29], [32], Tabu Search [31], [33]), and combinations of two or more of these methods (e.g., [21], [22], [60], [61]). These approaches usually exploit the relations between available information on a set of past projects (i.e., the training set) to build a model that can be used to predict the effort for a new project. The information presented in the training set (a.k.a. predictors or cost drivers) are the factors identified as related to the effort measured on the previous projects and stored in a database. Examples of such cost drivers are the functional size of the software, number of team members, team experience, etc. The prediction model takes as input the predictor values for a new project and returns a scalar value that represents the estimated effort to develop a software system, usually in terms of person-hour or person-month. Depending on the prediction approach, the predictors are used in a different way. For example, a linear regression technique combines the predictors through a linear equation, while in Case-Based reasoning the predictors are exploited to find the most similar projects from the past, then used to predict the effort for the new project.

2.1 Performance Measures

In the software effort estimation literature several measures have been introduced and used for evaluating the accuracy of a prediction model. These measures generally are built upon the prediction error (or absolute error) that is the distance between the predicted value and the actual value (i.e., $|Actual.value - Predicted.value|$). Among them, the Mean of Magnitude of Relative Error (MMRE), Mean of Magnitude of Relative Error Relative to Estimate (MEMRE) and Prediction at level l ($Pred(l)$) have been the most popular [68] until it was pointed out that they are biased towards underestimates [58], [62], [81], [95], [104], [107], and behave differently when comparing different prediction models [38]. Therefore, their use is discouraged and future studies should rely on standardised measures that are not biased towards under or overestimates, such as the *Sum* *of Absolute Errors* (SAE), *Mean Absolute Error* (MAE) and *Standard Accuracy* (SA) [64], [89], [97].

Given a set of n projects, each of which characterised by actual effort (a_i) and estimated effort (e_i), the SAE and MAE are computed as the sum of the absolute errors and the mean of the absolute errors across the n projects, respectively:

$$SAE = \sum_{i=1}^{n} |a_i - e_i| \tag{1}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |a_i - e_i| \tag{2}$$

SA was recommended by Shepperd and MacDonell [97] as a standard measure for comparing multiple prediction models against each other. It is based on MAE and defined as follows:

$$SA = \left(1 - \frac{MAE_{p_i}}{MAE_{p_0}}\right) \times 100 \tag{3}$$

where MAE_{p_i} is the MAE of the prediction model p_i being evaluated and MAE_{p_0} is the MAE of a large number (usually 1,000 runs) of *random guesses*. For a prediction model p_i whose accuracy outperforms that of random guessing, SA will produce a number in the range $[0, 1]$. An SA value closer to zero is discouraging since it means that the predictor p_i is performing just a little better than random guessing [89]. SA can also produce negative values, for those predictors which are performing even worse than random guessing.

A high-performance prediction model should have a lower SAE, lower MAE, and higher SA than its competitors.

2.2 Search-Based Effort Estimation

Search-Based Software Engineering (SBSE) reformulates software engineering problems as search problems. The SBSE solution space is explored using a search technique equipped with some software metrics able to discriminate between good and bad solutions, which will tend to guide the search towards the optimal solution(s) [44].

In Search-Based Effort Estimation (SBEE), the optimisation method builds many candidate models and tries to identify the optimal model, i.e., the one providing the most accurate estimates [36], [86]. Such a model can be described by the following equation [89]:

$$EstimatedEffort = w_1 op_1 f_1 + ... + w_n op_n f_n + C \tag{4}$$

where f_i represents the value of the i^{th} feature (i.e., cost driver), w_i its weight, and C represents a constant, while op_i represents the i^{th} mathematical operator (e.g., $+$, $-$, $\times$, $\div$, exp) of the model. Any value except for the features in Equation (4) can be optimised in order to maximise the accuracy of the model. Consequently, the search space consists of all the models that could possibly be built by varying the weights and operators in Equation (4). Considering the number of project features and the range of weights and operators, the search space can become dramatically large, that makes this problem suitable to be solved by search-based approaches.

The fitness of a model is evaluated by its prediction accuracy. In a single objective SBEE, a single measure of accuracy is used to compare different models and consequently derive the best one [13], [32]. However, in the context of effort estimation, there are several measures of accuracy, each one focusing on a different aspect. Since there is no a defined way of aggregating different accuracy measures for SEE, a multi-objective solution is appropriate [34], where several competing measures are optimised simultaneously [45], [89].

SBEE has been widely explored for effort estimation over the last 20 years [13], [28], [31], [32], [33], [34], [66], [89], [92]. Unlike the previous work on multi-objective software effort estimation, Sarro et al. [89] considered the confidence interval of the estimated efforts by a model as an objective to be minimised alongside the maximisation of the accuracy of the point estimates. Since this paper is a replication of this work, we briefly describe the approach in the following.

2.3 Confidence Guided Effort Estimation (CoGEE)

CoGEE is a bi-objective estimation method introduced in the original study by Sarro et al. [89].

Representation: The effort estimation model in Equation (4) is encoded as a Genetic Algorithm individual by representing it as an expression syntax tree with operators in the internal nodes and weights in the leaf nodes (see Figure 1). As in the original study [89], the values for the weights and the constant C are randomly drawn from a range of real numbers bounded by $[min, max] = [-100, 100]$ and the operators are drawn from $op \in \{+, -, \times, \div\}$.[4] The feature values in Equation (4) come from the training data and do not change during the evolution process, so they are not present in the expression syntax tree. An equation (i.e., model) that produces negative values for $EstimatedEffort$ is considered infeasible and penalised by receiving the worst possible fitness value. The initial population is generated by building 100 random trees of fixed depth. However, it should be noted that the fixed depth of the tree does not force a model to use all of the features in the training set: a feature might be discarded from a given model if its weight is set to zero and its operator is set to multiplication during the optimisation process.

Fitness: The fitness of a solution is evaluated using two objective functions; one minimising the prediction error (i.e., maximise accuracy), and the other minimising the uncertainty of the estimate distribution. In the original study, the Sum of Absolute Errors (SAE) is used as the accuracy objective function (see Equation (1)), where the confidence interval associated with the estimation model is used to assess the uncertainty of the mean value of the distribution of absolute errors produced by the model [89]. The confidence interval is defined as follows:

4. Equation (4) used herein is the one used in the original study [89], which was actually inspired by previous work using genetic programming to build prediction models such as the one by Lefley and Shepperd [66], which also used the operators $+$, $-$, $\times$, and $\div$.

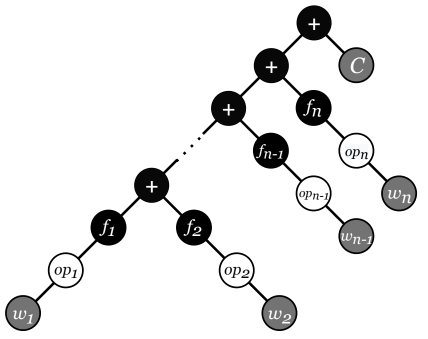


Fig. 1: Syntax Tree of the Estimation Model of Equation (4).

$$CI = \phi(p, df) \times \frac{std(AbsoluteErrors)}{\sqrt{n}} \tag{5}$$

where the second term is the sample standard deviation of the distribution of absolute errors with n being the size of the sample and $\phi(p, df)$ the quantile function (Equation (6)) which returns a threshold value x below which random draws from the given cumulative distribution function would fall p percent of the time [46]:

$$\phi(p, df) = inf\{x \in \mathbb{R} : p \leq F(x, df)\} \tag{6}$$

for a probability $0 < p < 1$, $F(x, df)$ is the probability density function of *t-distribution* function, which is a function of x and df [39]. Confidence intervals are calculated so that this percentage is 95%; the degree of freedom, df, depends on the number of parameters we are estimating: in regression models, an n-sized sample usually leads to $n - k$ degrees of freedom, where k is the number of parameters to be estimated (here $k = 1$).

Handling Multiple Objectives: Pareto optimality is used to rank the solutions based on their fitness and to shortlist the final solution(s). Pareto optimality states: "A solution x_1 is said to dominate another solution x_2 if x_1 is no worse than x_2 in all objectives and strictly better in at least one objective" [16].

Computational Search: The original study used NSGA-II [25], a widely used multi-objective evolutionary algorithm, as a ranking method for CoGEE. NSGA-II is a fast non-dominated sorting-based multi-objective genetic algorithm with elitism, in which the diversity of the population is preserved by a parameterless crowded-comparison approach [25]. NSGA-II is used with roulette wheel and tournament operators [63] to select individuals for reproduction and determine the individuals that are included in the next generation (i.e., survivals), respectively [89]. The roulette wheel operator (a.k.a. fitness proportionate selection) assigns a roulette slice to each chromosome in such a way that the size of the slices is proportionate to the fitness value of the chromosomes. A random point is then selected on the wheel, and the chromosome under such point is chosen. In this way, even if candidate solutions with higher fitness have more chances to be selected, there is still a chance that they may remain unselected; likewise, even the weakest candidate solutions have got a chance, though small, to be selected for reproduction. This selection happens multiple time in each generation to fill the mating pool [65]. The tournament selector, on the other hand, is used to select the best n solutions (usually $n \in [1, 10]$) to be copied straight into the next generation [89]. Each tournament randomly picks a number of solutions from the population and selects the fittest one.

Single point crossover and uniform mutation operators are used and defined such as to preserve well-formed equations, in the form of Equation (4), in all offspring [89]. More specifically, the crossover operator randomly selects the same point in every two mating individuals and swaps the sub-parts corresponding to the selected point. The length of the two offspring chromosomes (i.e., predictive model expressions) is guaranteed to be of the same length as the parent individuals since both parent chromosomes are cut at the same point. The crossover operator fires at a rate determined by the algorithm parameters, specifically, the crossover probability rate. It means that not all of the selected individuals will end up participating in reproduction. The algorithm uses a uniform mutation operator that selects a random node in the tree and changes its value to a randomly selected possible value. The mutation operator can change internal nodes (i.e., operators) as well as leaf nodes (i.e., weights) of the tree. In particular, when the mutation operator affects an internal node, a new operator $op'_i \in \{\{+, -, \times, \div\} \backslash op_i\}$ is randomly selected and assigned to the node. Whilst, if the mutation operator affects a leaf node (i.e., weights and the constant C), a new weight $w'_i \in \mathbb{R}$ is assigned to the node, which in both the original and replication study is selected randomly to be in $[-100, 100]$. The crossover and mutation rates are set to 0.5 and 0.1, respectively, and the evolutionary process uses a population of 100 individuals and terminates after 250 generations. These parameters are the same as in the original study, where no tuning was performed to search for optimal parameter values.

3 Empirical Study Design

Since this study is an exact replication of Sarro et al.'s work [89], we follow the original study as closely as possible to assess whether the same results will be obtained.

In this section, we present the research questions, the datasets and techniques used for the experiments, and the validation and evaluation criteria used to assess the results.

3.1 Research Questions

The first four research questions for this replication study are identical to RQs 1–4 in the original study. Additionally, we investigate two new research questions (RQs 5–6).

In order to be accepted as a software effort estimation model, CoGEE needs to outperform baseline techniques. Therefore, the first research question is:

RQ1. Sanity Check: *Is the proposed approach CoGEE suitable for effort estimation?*

To perform the sanity check, the original study compared CoGEE with three common baselines used in the context of effort estimation [89]. Specifically, the three baselines are Mean and Median Effort and Random Guessing (described in Section 3.3). Performing better than baseline techniques is necessary but not sufficient for an effort estimation model to be adopted in practice. Therefore, as in the original study, we investigate whether CoGEE outperforms state-of-the-art effort estimation techniques to guarantee that it can advance the state-of-the-art:

RQ2. State-of-the-Art Benchmark: *Does CoGEE provide more accurate and robust estimates than currently used effort estimation methods?*

In the original study, three effort estimation techniques were used to benchmark CoGEE: Automatically Transformed Linear Model (ATLM) [109], Classification and Regression Tree (CART) [12], Case-Based Reasoning (CBR) [98]. Each of these techniques represents a different approach to solve the software effort estimation problem. Specifically, they are gradient, regression, and analogy based techniques. We use these algorithms as-is, i.e., we kept their original formulation both in terms of equation and error measure used as detailed in Section 3.3. In our replication, we replace ATLM with a recent and more robust state-of-the-art benchmark (i.e., Linear Programming for Effort Estimation (LP) [88]) because ATLM has been shown to be unstable [88].

Outperforming these benchmarks provides evidence that CoGEE does, indeed, advance the state-of-the-art. However, to get an insight into whether it is the multi-objective nature of CoGEE that makes it able to reach an accuracy higher than the state-of-the-art techniques, we investigate the benefits from multi-objective formulation, as done in the original study:

RQ3. Benefits from Multi-objective Formulation: *Does CoGEE provide more accurate and robust estimates than alternative single and multi-objective approaches?*

This research question has been investigated on two fronts by Sarro et al. [89]. First, it examines whether the two objectives that CoGEE considers together outperform the single objective variants of each considered individually. Therefore, the first sub-goal of this research question compares CoGEE with two single objective variants of evolutionary effort estimation algorithm (GA-SAE and GA-CI), each optimising one of the two objectives considered by CoGEE. The goal of GA-SAE and GA-CI will be minimising the Sum of Absolute Errors (SAE) and the Confidence Interval (CI), respectively:

RQ3.1. *Does CoGEE provide more accurate and robust estimates than GA-SAE and GA-CI?*

Secondly, Sarro et al. [89] examine the effectiveness of the objective chosen. Minimising the SAE implicitly minimises the sum of underestimate and the sum of overestimates at the same time. These two objectives are conflicting when considered separately. Because minimising overestimates, for instance, could lead to high underestimates and vice versa. The SAE combines these two separate objectives in one formulation. RQ3.2 investigates whether underestimates and overestimates should be optimised separately or whether it is sufficient to combine them as a single objective (SAE). Therefore, another multi-objective algorithm (called NSGAII-UO), which optimises the two objectives of underestimate and overestimate separately, is compared with CoGEE:

RQ3.2. *Does CoGEE provide better results than NSGAII-UO?*

Positively answering RQ1, RQ2, and RQ3 shows that CoGEE can improve the state-of-the-art in automated software effort estimation with strong scientific evidence. However, for an industrial uptake, CoGEE must outperform current industrial practice as well. Since reports about the actual estimation accuracy of current industrial practice are not reliably consistent [89], the original study compared CoGEE with the current beliefs about the industrial practice:

RQ4. Comparison to Industrial Practices: *Does CoGEE provide more accurate and robust estimates than the ones claimed for current industrial best practice?*

To answer RQ4, we compare the performance of CoGEE (and other state-of-the-art estimators) against claims made for best human-expert-based results achievable in the industry [53], [75], as done in the original study. In particular, we investigate the magnitude of relative error (compared to claimed industrial best practice). Because industrialists tend to be more concerned with underestimated results (rather than overestimated results) [70], following the original work [89], we also evaluate the budget overrun that would accrue from using our technique, compared to these claimed for industrial best practice and the state-of-the-art.

Sarro et al. [89] used NSGA-II with CoGEE since it is a widely used Multi-objective Evolutionary Algorithm (MOEA) [26], [89]. However, there are many variants of MOEAs used in search-based software engineering [84], each designed to improve a different aspect of the Pareto Front quality. Therefore, we are interested to know if CoGEE achieves different results with other variants of MOEAs:

RQ5. Using other MOEAs:. *Dose any other MOEA provide CoGEE with better results than NSGA-II and in less time?*

To answer RQ5, we implemented CoGEE using four additional MOEAs, namely NSGA-III, SPEA2, MOCell, and IBEA, to investigate whether these algorithms can improve the accuracy and quality of CoGEE. Each of these four variants is the most prominent and widely used variation of their family [84]. Because of the differences in the algorithm design, each of these multi-objective evolutionary algorithms may achieve different results as well as have a different running time. Thus, we are also interested in the running time produced by each of the MOEAs.

In order to strengthen the external validity of the original study the first author has independently implemented CoGEE. To this purpose, we decided to use `JMetal`, the `Java` framework for Evolutionary Computation [78], rather than the `R` software used in the original paper. This choice was motivated by two main reasons: (1) `JMetal` is a widely known and publicly available framework; (2) previous work [76], [110] pointed out that using `R` may become inefficient when performing computationally intensive tasks for a number of reasons, including lazy evaluation of the function arguments, name lookup with mutable environments, and being dynamically typed. We also observed this issue with the original CoGEE implementation in `R`, thus we devote our last research question to investigate whether using a `Java`-based implementation is more efficient than its `R` counterpart. If the new implementation of CoGEE runs faster, without reducing its accuracy, its adoption would be more appealing in practice, and it would make it easier for other researchers to extend CoGEE or to use it to benchmark their newly proposed techniques.

RQ6. CoGEE's Running Time: Java vs. R: *Does the Java implementation of CoGEE run faster than its R counterpart?*

To answer this question, we compare the running time of the `Java` implementation of NSGA-II we built upon the `JMetal` framework (version 5.4) [78] against the `R` version proposed in the original study, which was built using the package `nsga2R` (version 1.0) [1].

In the following, we will refer to the CoGEE algorithm implemented in the original study as CoGEE$_{NSGAII-R}$, to our new `Java` implementation as CoGEE$_{NSGAII}$, and the other four alternative CoGEEs based on other MOEAs as CoGEE$_{IBEA}$, CoGEE$_{MOCell}$, CoGEE$_{NSGAIII}$, and CoGEE$_{SPEA2}$.

3.2 Datasets

We used the same five publicly available datasets used in the original study to empirically investigate our research questions. These five datasets, namely China, Desharnais, Finnish, Maxwell, and Miyazaki, have been extensively used in previous software effort estimation studies [10].

Table 1 summarises the descriptive statistics of the features of the datasets we considered. We observe that these datasets differ in terms of: observation number (from 38 to 499 projects); number and type of features (from 4 to 17 features, including a variety of features describing the software projects); companies involved, i.e., within-company (WC), or cross-company (CC); and geographical locations [10]. The software projects included in these datasets also differ in technical characteristics. For instance, software projects developed in different programming languages and for different application domains, ranging from telecommunications to commercial information systems.

TABLE 1: Descriptive Statistics of the Datasets.

Dataset	Type	Variable	Min	Max	Mean	Std. Dev.
China	CC	Input	0	9404	167.10	486.34
(499 projects)		Output	0	2455	113.60	221.27
		Enquiry	0	952	61.60	105.42
		File	0	2955	91.23	210.27
		Interface	0	1572	24.23	85.04
		Effort	26	54620	3921.05	6480.86
Desharnais	WC	TeamExp	0	4	2.30	1.33
(77 projects)		ManagerExp	0	7	2.65	1.52
		Entities	7	386	121.54	86.11
		Transactions	9	661	162.94	146.09
		AdjustedFPs	73	1127	284.48	182.26
		Effort	546	23940	4903.94	4188.19
Finnish	CC	HW	1	3	1.26	0.64
(38 projects)		AR	1	5	2.24	1.50
		FP	65	1814	763.58	510.83
		CO	2	10	6.26	2.73
		Effort	460	26670	7678.29	7135.28
Maxwell	CC	SizeFP	48	3643	673.31	784.04
(62 projects)		Nlan	1	4	2.55	1.02
		T01	1	5	3.05	1.00
		T02	1	5	3.05	0.71
		T03	2	5	3.03	0.89
		T04	2	5	3.19	0.70
		T05	1	5	3.05	0.71
		T06	1	4	2.90	0.69
		T07	1	5	3.24	0.90
		T08	2	5	3.81	0.96
		T09	2	5	4.06	0.74
		T10	2	5	3.61	0.89
		T11	2	5	3.42	0.98
		T12	2	5	3.82	0.69
		T13	1	5	3.06	0.96
		T14	1	5	3.26	1.01
		T15	1	5	3.34	0.75
		Effort	583	63694	8223.21	10499.90
Miyazaki	CC	SCRN	0	281	33.69	47.27
(48 projects)		FORM	0	91	22.38	20.55
		FILE	2	370	34.81	53.36
		Effort	896	253760	13996	36601.56

The **China** dataset [112] comprises 499 projects from multiple companies across China measured and recorded in 19 attributes. We used the basic elements used to calculate Function Points (i.e., Input, Output, Inquiry, File, Interface) as independent variables and the Effort variable as the dependent variable. This dataset was made available in 2010.

The **Desharnais** dataset, including 81 software projects, was collected from 10 organizations in Canada between 1983 and 1988. We considered the total effort as a dependent variable but not the length of the code. Following other studies [88], [89], [96], [98], in this study 77 of the 81 records of this dataset are used since there are missing data in four records. We also excluded from our analysis the categorical variables (i.e., Language and YearEnd) and used the following independent variables: TeamExp (i.e., the team experience measured in years), ManagerExp (i.e., the manager experience measured in years), Entities (i.e., the number of the entities in the system data model), Transactions (i.e., the number of basic logical transactions in the system), AdjustedFPs (i.e., the Adjusted Function Points).

The **Finnish** dataset [99] was collected from nine different firms in Finland, and made available in 1997. It consists of nine attributes and 38 records, with size measured in Function Points and the Effort expressed in person-hour. Following previous research [88], [89], we excluded the PROD variable since it represents the productivity expressed in terms of Effort and size, and only used HW (i.e., the type of hardware), FP (i.e., Function Points), AR and CO as the independent variables to build effort estimation models.

The **Maxwell** dataset was collected from the biggest Finnish commercial bank between 1985 and 1993. It comprises 22 categorical attributes that were asserted to influence software productivity [69], and the size attribute is measured in Function Points. As Sarro et al. [89], we exploited 17 features: Function Points (SizeFP) and 16 ordinal variables, i.e., number of different development languages used (Nlan), customer participation (T01), development environment adequacy (T02), staff availability (T03), standards used (T04), methods used (T05), tools used (T06), software's logical complexity (T07), requirements volatility (T08), quality requirements (T09), efficiency requirements (T10), installation requirements (T11), staff analysis skills (T12), staff application knowledge (T13), staff tool skills (T14), and staff team skills (T15). As for the Desharnais dataset, we did not use categorical variables.

The **Miyazaki** dataset was collected by Fujitsu's Large Systems Users Group [74] and made available in 1994. The data were obtained from 48 COBOL systems developed in 20 different organizations and across multiple departments within those organizations. For this dataset, we considered the following independent variables: SCRN (i.e., the number of different input or output screens), FORM (i.e., the number of different report forms), and FILE (i.e., the number of different record format). The dependent variable is Effort, defined as the number of person-hours needed from system design to system test, including indirect effort such as project management.

3.3 Techniques

This section describes the techniques used and how they have been implemented and configured.

Evolutionary Algorithms. The original study used *NSGA-II* [25], a widely used multi-objective evolutionary algorithm, as a ranking method under the hood of CoGEE (see Section 2.3). In this study, we use NSGA-II with the same parameters as used in the original study (see Section 2.3 for details). Furthermore, we have considered four additional MOEAs, namely NSGA-III [24], [51], SPEA2 [116], MOCell [77], and IBEA [114], each representative of a family of multi-objective algorithms widely used in SBSE [84]. We explain each of these algorithms below.

Non-dominated Sorting Genetic Algorithm III (NSGA-III) [24], [51] is a reference-point based NSGA-II in which the diversity among solutions is maintained by providing and adaptively updating a number of reference points. We set the number of the reference points equal to the number of the individuals in the population, with the initial reference-points generated by the framework. All of the operators and other parameters used for NSGA-III are the same as for NSGA-II.

Strength Pareto Evolutionary Algorithm 2 (SPEA2) [116] exploits elitism by preserving an external archive of the best solutions found so far and using a Strength value for each chromosome, in addition to the fitness value, to determine how many solutions are dominated by the chromosome. SPEA2 uses a fine-grained fitness assignment strategy, a density estimation technique, and an enhanced archive truncation method, all of which help the algorithm in preserving a usually better spread of the solutions on the Pareto Front in comparison with NSGA-II. The archive size for SPEA2 is set to the population size and all other parameters and operators are equal to the original parameters set for NSGA-II.

MultiObjective Cellular Genetic Algorithm (MOCell) [77] is a cellular genetic algorithm that, similar to SPEA2, uses an external archive to store non-dominated solutions and randomly replaces existing individuals in the population with selected individuals from the archive after each iteration. MOCell tries to preserve a grid-like spread over the Pareto Front to provide a better sampling of the search space. The archive size of MOCell is set to the square root of the population size, and all other parameters and operators are equal to the original parameters set for NSGA-II.

Indicator-Based Evolutionary Algorithm (IBEA) [115] is a representative of a family of MOEAs that consider Pareto Front quality indicators, such as the Hypervolume, in the evolution process, to preserve a good quality Pareto Front. The archive size for IBEA is set equal to the population size, and all other parameters and operators are equal to the original parameters set for NSGA-II.

We used `JMetal` 5.4 [78] to implement all the GA based algorithms, including NSGA-II, NSGAII-UO, NSGA-III, SPEA2, IBEA, and MOCell. Generational Genetic Algorithm implementation of the `JMetal` framework, which is the standard implementation of GA, is used for two single-objective variants that we call GA-SAE and GA-CI. These two single objective algorithms differ only in the fitness functions where GA-SAE minimises the Sum of Absolute Errors and GA-CI minimises the Confidence Interval associated with the mean of the absolute errors. We used GA and NSGA-II with the same settings as the `R` implementation used in the original study [89] (see Section 2.3) to be able to compare the results in an equitable setting.

Random Guessing (RG) is a naïve and general method that simply assigns the effort of a randomly selected project to the target project [97]. More formally, random guessing predicts an effort value y for the target case $project_t$ by randomly sampling (with equal probability) over all the remaining $n-1$ cases and taking $y = r$; where r is the effort value for the randomly drawn $project_r$ from $1...n \mid project_r \neq project_t$ [97]. This method does not need any parameter estimation and any prediction system is expected to outperform it over time; otherwise, the prediction system is not using any target case information. We implemented the Random Guessing effort estimator in `R` programming language version 3.5.3 [4].

Mean and Median Effort are two baseline benchmarks commonly used for effort estimation techniques [73], [89], [109]. Specifically, the mean or median of the past projects is used as the predicted effort for a new project.

Linear Programming for Effort Estimation (LP) has been recently proposed by Sarro and Petrozziello [88] as a more robust and accurate benchmark than ATLM for effort estimation techniques. LP forms a mathematical model with a linear objective function (i.e., Sum of Absolute Errors) subject to linear equality and inequality constraints draw out of the observation data. The feasible region is given by the intersection of the constraints and LP is able to find a point in the polyhedron where the function has the smallest value in polynomial time. It has been shown that LP is robust to different data splits under different cross-validation methods, easy and fast to apply, and have comparable prediction performance to standard methods [88]. Moreover, this approach does not have any hyper-parameter to tune [88]. In this study, we used the LP `R` package [3] made publicly available by Sarro and Petrozziello [88].

Case-Based Reasoning (CBR) is an artificial intelligence technique based on the premise that "similar problems are best solved with similar solutions". It has been successfully used in Software Engineering for prediction [93], [98]. For a software effort estimation problem in which the software projects are described by their features, the effort value of the k most similar projects in the feature space are retrieved and used as the final prediction for the new project. The choice of k is a tuning decision, and its effect in software prediction has been studied in previous work [56]. We report the results of each of the choices of k, between $k = 1$ and $k = 3$, as done in the original study. As in the original work [89] we used ANGEL (ANaloGy Estimation tool) [98] to obtain CBR predictions. ANGEL is a tool introduced by Shepperd and Schofield [98] to estimate the development effort of a software project using analogy. It supports the Euclidean distance measure between vectors and uses this metric to compute project similarity. The final estimation is computed as the mean effort of the k nearest cases.

Classification and Regression Tree (CART) is a machine learning technique that constructs prediction models by recurrently partitioning the data and fitting a simple model in the form of a decision tree within each partition. For software effort estimation problem in which the target variable (i.e., effort) takes continuous and ordered values, the resulting decision tree is called a regression tree. We used the `R` (version 3.5.3) package `rpart` (version 4.1-13) to generate regression trees by using `anova` as a splitting method for `rpart` as done in the original study. This method has no other parameters.

3.4 Validation and Evaluation Criteria

In this study we have adopted the validation and evaluation criteria used in the original study.

In order to validate the prediction models, we used a 3-fold cross-validation and the same folds as in the original study. This allows us to avoid possible data bias produced by different sampling, and thus to assess the consistency of the results between the two studies. explain the 3-fold validation?

To evaluate the estimation performance of the prediction models, we used the Mean Absolute Error (MAE) and the Standard Accuracy (SA) (see Section 2.1 for their definition). To measure the accuracy performance of MOEAs, which produce more than one final solution, we compute and use the mean MAE over all the solutions in their Pareto Front. Strictly speaking, we use all the solutions (i.e., models) in a given Pareto Front obtained by CoGEE on the training set to estimates the effort for the (unseen) projects contained in the test set. Then we compute the CI and SAE of these estimates per model and compute the mean value of these measures over all the models in a given Pareto Front. We use these mean values to compare CoGEE against the other techniques that produce only one solution.

The following statistical hypothesis (null hypothesis) is tested to verify the difference between the prediction performance of the models (in RQs 1–5):

H_0: *There is no difference between the absolute errors provided by the prediction models* P_i *and* P_j.

If the test rejects the null hypothesis, the following alternative hypothesis is accepted:

H_1: *The absolute errors provided by the prediction model* P_i *are significantly different from those provided by the prediction model* P_j.

Similarly, we tested the following null hypothesis to verify the difference between the running time of the different multi-objective evolutionary algorithms (RQ5):

H_0: *There is no difference between the running time achieved by the algorithms* A_i *and* A_j.

Where the alternative hypothesis is as follows:

H_1: *The running time of the algorithm* A_i *is significantly different than the running time of the algorithm* A_j.

Finally, we tested the following null hypothesis to verify the difference between the running time of the NSGA-II `Java` implementation (CoGEE$_{NSGAII}$) and the `R` implementation (CoGEE$_{NSGAII-R}$) (RQ6):

H_0: *The running time achieved by CoGEE*$_{NSGAII}$ *is not lower than the one achieved by CoGEE*$_{NSGAII-R}$.

The alternative hypothesis is as follows:

H_1: *The running time of CoGEE*$_{NSGAII}$ *is significantly lower than the running time of CoGEE*$_{NSGAII-R}$.

Since many of the samples comes from non-normally distributed populations [10], [89] a non-parametric method ought to be used as the statistical significance test. To this end, we use the Wilcoxon Rank-Sum test [20], which is a non-parametric test that makes no assumption about underlying data distribution, hence, raises the bar for significance for both normally and non-normally distributed data. We set the confidence limit, α, at 0.05 and applied the standard Bonferroni correction (α/K, where K is the number of hypotheses) when multiple hypotheses were tested.

As noted by Arcuri and Briand [6], when two randomised algorithms are compared against each other, given a large enough number of runs n, it is not unlikely to obtain statistically significant differences even if the difference is so small as to be of no practical value. Consequently, it is inadequate to merely show statistical significance alone; we also need to know whether the effect size is worthy of interest. To this end, a standardised non-parametric effect size measure, namely the Vargha Delaney's $\hat{A}_{12}$ statistic, is used to assess the effect size of the difference between two methods with regard to their accuracy performance [6], [89]. Running two algorithms A and B, $\hat{A}_{12}$ measures the probability of A performing better than B with reference to a performance measure. $\hat{A}_{12}$ is computed using Equation (7), where R_1 is the rank sum of the first data group we are comparing, and m and n are the number of observations in the first and second data sample, respectively.

$$\hat{A}_{12} = \frac{\left(\frac{R_1}{m} - \frac{m+1}{2}\right)}{n} \tag{7}$$

Based on Equation (7), if two algorithms are equally good, $\hat{A}_{12} = 0.5$. Respectively, $\hat{A}_{12}$ higher than 0.5 means that the first algorithm is more likely to produce better results. The effect size is considered small for $0.6 \leq \hat{A}_{12} < 0.7$, medium for $0.7 \leq \hat{A}_{12} \leq 0.8$, and large for $\hat{A}_{12} \geq 0.8$, although these thresholds are not definitive [89]. To perform statistical tests, we used the implementation of the Wilcoxon Rank-Sum test and Vargha Delaney's $\hat{A}_{12}$ effect size available from `stats` library in `R` [4] version 3.5.3. Since we are interested in *any* improvement, no transformation is performed on the $\hat{A}_{12}$ effect size [79], [89].

Analysing the solutions produced by MOEAs by looking only at their accuracy or confidence interval independently does not give us information about the trade-off between the two competing objectives. Thus, we need to quantify the overall quality of prediction models with respect to both objectives at the same time. For completeness, we report the percentage of runs in which a solution found by CoGEE$_{NSGAII}$ dominates, or is equal to, the solution found by the single objective GA (i.e., GA-SAE, or GA-CI) on the test sets, based on both objectives simultaneously, and also based on each of them separately. We also report the average SAE and CI values achieved by all the evolutionary approaches in our online supplementary appendix [106]. However, we highlight that looking at one objective at a time is discouraged as it may result in misleading conclusions about the trade-off between two equally important objectives [42], [83], [85].

On the other hand, the Pareto Front's quality indicators allow us to quantify the overall quality of prediction models by measuring the trade-off between multiple competing objective values (in our case, SAE and CI). These indicators are well-known in the multi-objective optimisation literature [14], [19], [113] and have been extensively used in previous software engineering work to evaluate and compare multi- and single-objective algorithms performance as an alternative to using the average values (see e.g., [35], [41], [42], [43], [67], [85]).

As done in the original work, we use three Pareto Front quality indicators, namely Contributions (I_C), Hypervolume (I_{HV}), and Generational Distance (I_{GD}) [34], [67], as these three indicators complement each other in quantifying the overall quality of the prediction models in the objective space in terms of convergence, spread, uniformity, and cardinality [67].

The Contribution indicator (I_C) measures the share of the contribution of each approach to a Reference Set (RS, which usually is the true optimal Pareto Front, if known) [37]. In other words, I_C for approach A is the proportion of the solutions in the reference set that are produced by A. A high I_C indicates that more of the solutions in the reference set are produced by approach A which means either the solutions from the other approaches are dominated by the solutions produced by A or A produced more high quality solutions than the other approaches. It means that even if approach B produces a few but very high-quality solutions, it will get a lower I_C score. This is why two other quality indicators are used as well.

The Generational Distance indicator (I_{GD}) [108] computes the average distance between the Pareto Front of approach A and the RS. In an n-objective space, I_{GD} calculates the n-dimensional Euclidean distance between each point in the solution set of approach A and its nearest neighbouring point in the reference set. The average distance is then considered as the I_{GD} indicator for approach A.

The Hypervolume indicator (I_{HV}) [117] measures the volume of the dominated portion of the objective space by members of the non-dominated set of solutions from each approach. A large I_{HV} indicates that the solutions produced by the approach covered more of the objective space. Hypervolume is one of the few quality indicators that can evaluate a solution set's quality in terms of all the four quality aspects including convergence, spread, uniformity, and cardinality [67]. Zitzler [118] demonstrated that Hypervolume measure is also strictly 'Pareto compliant' which means that Hypervolume of *A* is higher than *B*, if the Pareto set of *A* dominates that of *B*. To compute the I_{HV}, we used $[1.01,\ 1.01]$ as the reference point, which is the boundary of the optimisation problem after normalization with an added 1% offset, so that the boundary solutions could contribute to the I_{HV} value [67].

If the optimal Pareto Front (PF_{true}) is known, this is usually used as a reference set to compute the above three quality indicators. Otherwise, these indicators can be calculated with respect to a set of reference points that is the closest known front to the PF_{true} (i.e., PF_{known}). In the original and our study, the PF_{known}, is obtained by combining the Pareto Fronts produced by the approaches compared, and selecting the non-dominated solutions as done in previous work [37], [41], [43], [67], [89].[5] Specifically, when we compare CoGEE$_{NSGAII}$ with GA-SAE and GA-CI (RQ3.1), we construct the reference set PF_{known} by combining all the solutions provided by the three algorithms and then select the non-dominated ones from this union set (i.e., we select all those solutions that are better than all the other solutions in at least one objective and not worse in the other objective). Whereas when we compare CoGEE$_{NSGAII}$ with NSGAII-UO (RQ3.2) we construct the reference set PF_{known} by combining all the solutions provided by these two algorithms and then selecting the non-dominated ones. Once we have obtained the PF_{known}, we normalized the fitness values for all the fronts before computing the quality indicators. The normalization helps avoid unwanted scaling effects, since SAE and CI vary in different ranges. In particular, we used Min-Max feature scaling to normalise these values to the range of $[0,\ 1]$. Minimum and maximum values are acquired from the union of the Pareto Fronts produced by the approaches being compared.

Due to the stochastic nature of the evolutionary algorithms, best practice is to run the experiments as many times as possible to show with high confidence that the results are statistically significant [6]. Therefore, and following the original study, we performed 30 independent runs per algorithm on each of the cross-validation folds (for a total of 90 runs per dataset) to mitigate the variance in the results due to the stochastic nature of the approaches considered and allow for such confident statistical testing.

To answer RQs 5-6, we compute the running time of each of the multi-objective algorithms under a same configuration (i.e., 250 generations for 100 individuals), and disregard any time needed for I/O operations. To obtain a fair measurement, we run all the experiments on a single node of the UCL Computer Science HP Clusters, which is equipped with the Intel® Xeon® CPU E3-1240 v3@3.40GHz and RAM 12 GB. To measure the change in the running time between the `Java` version and the `R` version (RQ6), we compute the Percentage Change[6], which is defined as follows:

$$PercentageChange = \frac{(T_{Java} - T_R)}{T_R} \times 100 \quad (8)$$

where T_{Java} is the mean running time for CoGEE$_{NSGAII}$ and T_R is the mean running time for CoGEE$_{NSGAII-R}$ over 90 independent runs (i.e., 30 runs on each of the three folds). A negative value of the Percentage Change indicates a decrease, while a positive value shows an increase in the running time. We use the distribution of the running time

5. Note that to obtain the Pareto Fronts of the single-objective variants GA-SAE and GA-CI to answer RQ 3.1, we compute both the SAE and CI values of the estimation models built by the single-objective algorithms in order to assess their effectiveness in the two-objective space. Since each solution provided by these algorithms is an estimation model, we can compute the SAE and CI values associated with the predictions made by these solutions (models) regardless of the objective that they have been optimised for. Similarly, we compute both SAE and CI of the solutions built by NSGAII-UO to answer RQ 3.2.

6. Comparing the exact running time could be misleading since it inherently depends on several aspects, including but not limited to the size of the dataset (number of instances as well as the number of features) and the computational power of the machine. Therefore, we chose to analyse the change in the running time as a rate (e.g., percentage change) rather than the scalar values.

measured over 90 independent runs of each algorithm to test for the statistical difference in RQ5 and RQ6.

4 Results

In this section, we present the results obtained answering the research questions stated in Section 3.1, and compare them with the results of the original study [89].

4.1 RQ1. Sanity Check

Table 2 presents the Standard Accuracy (SA) of the estimations obtained by CoGEE$_{NSGAII}$[7] and the other techniques we experimented with.[8] The SA results show that CoGEE$_{NSGAII}$ outperforms the baseline methods (i.e., Random Guessing, Mean, and Median effort) with a large difference: CoGEE$_{NSGAII}$ is almost twice as accurate as of the best baseline estimator on each of the datasets. The results of the Wilcoxon test comparing CoGEE$_{NSGAII}$ with the baseline methods are shown in Table 3. The p-values are very small ($p < 0.001$) for all the datasets, indicating that the difference is statistically significant. The results remained statistically significant after correcting the p-values for multiple statistical tests. The $\hat{A}_{12}$ effect size (which is presented in brackets) is large for all the datasets. This evidence, which is consistent with the original study, confirms that CoGEE$_{NSGAII}$ outperforms the baseline methods and positively answers RQ1.

4.2 RQ2. Comparison to State-of-the-Art

We analysed the SA of the state-of-the-art techniques alongside that of CoGEE$_{NSGAII}$, shown in Table 2. We can observe that the SA of CoGEE$_{NSGAII}$ is better than CBR for all datasets, better than LP for all but the Desharnais dataset, and better than CART for all but the Finnish dataset.

The results of the Wilcoxon test performed to check the significance of the difference between the accuracy performance of CoGEE$_{NSGAII}$ and the other state-of-the-art techniques are shown in Table 4. We can observe that the differences are statistically significant ($p < 0.001$) for 23 out of 25 comparisons with a large effect size ($\hat{A}_{12} \geq 0.87$) in 22 cases and medium effect size ($\hat{A}_{12} = 0.63$) in one. These results are consistent with the original study and provide evidence to answer RQ2 positively.[9]

4.3 RQ3. Does Multi-Objectivity Help?

Table 6a reports on the quality of the solutions found using CoGEE$_{NSGAII}$, GA-SAE, and GA-CI, with respect to the three Pareto Front quality indicators: I_{HV}, I_{GD}, and I_C.

The analysis shows that for all datasets, CoGEE$_{NSGAII}$ produces comfortably better results than GA-SAE and GA-CI. As disclosed by the I_C indicator, the majority of the solutions on the reference front are produced by CoGEE$_{NSGAII}$, which means that GA-SAE and GA-CI have comparatively fewer good quality solutions. Nevertheless, we should note that unlike CoGEE$_{NSGAII}$, which provides a Pareto set of solutions at the end of each run, GA-SAE and GA-CI produce only one ultimately good solution. Thus, it is reasonable that CoGEE$_{NSGAII}$ contributes more to the reference front than single-objective benchmarks.

The I_{HV} indicator, on the other hand, is not affected by number of the solutions on the front. The relative difference in the volume of the objective space dominated by the three algorithms varies from dataset to dataset, yet CoGEE$_{NSGAII}$ has the largest value of I_{HV} for all datasets. As for the I_{GD} indicator, sometimes solutions produced by the single objective algorithms are close to the reference Pareto Front. However, CoGEE$_{NSGAII}$ has almost all of its solutions on the reference front (i.e., the mean distance of its solutions from the reference front is zero) for four out of five datasets, and there is a very small distance for China.

The results of the Wilcoxon test comparing CoGEE$_{NSGAII}$ with GA-SAE and GA-CI over I_{HV} and I_C indicators confirm that CoGEE$_{NSGAII}$'s excellence in the production of more good-quality solutions is statistically significant with a large effect size for 18 out of 20 cases (see Table 7a). For the I_{GD} indicator on four out of five datasets, CoGEE$_{NSGAII}$ has significantly better results than at least one of the single objective algorithms, while on China, CoGEE$_{NSGAII}$ is significantly better than both. However, the medium effect size measured for the significance of the difference on I_{GD} indicator suggests that almost for all of the cases, all three algorithms were able to discover solutions that are near-optimal for the problem instances, with respect to the objective they were evolved upon, whereas CoGEE$_{NSGAII}$ produces a Pareto Frontier of those solutions which gives a project manager the option to pick the solution that best fits their concerns.

For completeness, we also report in Table 5 the percentage of runs in which a solution found by the multi-objective algorithm dominates, or is equal to, the solution found by the single-objective algorithm. We can observe that this percentage is large for all datasets considered in this study. Thus, confirming that CoGEE provides more accurate and robust estimates than GA-SAE and GA-CI, as found in the original study [89].

These results corroborate previous findings in the literature on multi-objective optimisation [8], [50], [82], [89] that multi-objective genetic algorithms can outperform single-objective genetic algorithms, even when compared against the specific single objective targeted by the single-objective genetic algorithm. This arises because search spaces are non-monotonic, and therefore disimproving moves may be required in order to arrive at overall results that lie closer to global optima [89]. In other words, multi-objective genetic algorithms are more likely to escape from local optima and it is not uncommon that they outperform single-objective ones even when they are evaluated with respect to the same single fitness function used in the single-objective optimisation [50].

7. The SA values are obtained by using an ensemble composed of the models on the Pareto Front. These results can be attained in practice if a practitioner uses as the final estimation the average of the estimates provided by all the models in the Pareto Front, as we did herein for evaluation purposes. On the other end, a practitioner can also choose to use a single model based on their trade-off preference (i.e., small prediction error vs. narrow confidence interval).

8. For completeness we provide the MAE values in our on-line appendix [106] to facilitate comparison against other works that may not have reported SA.

9. Based on the results published in the original study, and correction for Miyazaki, CoGEE$_{NSGAII}$ outperforms ATLM in four out of five datasets.

TABLE 2: RQs1–2: Standard Accuracy (SA) values achieved by CoGEE$_{NSGAII-R}$ (original study), CoGEE$_{NSGAII}$ (this replication), the baseline (Mean and Median Effort), and state-of-the-art techniques (CBR1–3, LP, and CART) for each of the five datasets. For completeness, SA results are also included for the other three alternative evolutionary algorithms considered later in answer to RQ3 (i.e., GA-CI, GA-SAE, and NSGAII-UO) and four variants of CoGEE in answer to RQ5 (i.e., CoGEE$_{NSGAIII}$, CoGEE$_{SPEA2}$, CoGEE$_{MOCell}$, and CoGEE$_{IBEA}$).

China	SA	Desharnais	SA	Finnish	SA	Maxwell	SA	Miyazaki	SA
CoGEE$_{MOCell}$	0.51	CoGEE$_{NSGAII-R}$	0.47	CART	0.52	CoGEE$_{NSGAII-R}$	0.56	CoGEE$_{NSGAII}$	0.66
CoGEE$_{NSGAII}$	0.48	GA-SAE	0.45	CoGEE$_{SPEA2}$	0.46	CoGEE$_{MOCell}$	0.56	CoGEE$_{NSGAIII}$	0.66
CoGEE$_{SPEA2}$	0.48	LP	0.44	CoGEE$_{NSGAII}$	0.45	CoGEE$_{NSGAIII}$	0.56	CoGEE$_{SPEA2}$	0.66
CoGEE$_{NSGAII-R}$	0.48	CoGEE$_{MOCell}$	0.43	CoGEE$_{MOCell}$	0.45	CoGEE$_{NSGAII}$	0.55	CoGEE$_{NSGAII-R}$	0.66[a]
GA-SAE	0.48	CoGEE$_{NSGAIII}$	0.42	CoGEE$_{NSGAII-R}$	0.45	CoGEE$_{SPEA2}$	0.55	GA-SAE	0.66
LP	0.48	CoGEE$_{SPEA2}$	0.42	CoGEE$_{NSGAIII}$	0.44	GA-SAE	0.55	GA-CI	0.66
CoGEE$_{NSGAIII}$	0.47	CoGEE$_{NSGAII}$	0.41	GA-CI	0.44	LP	0.52	CoGEE$_{MOCell}$	0.65
GA-CI	0.45	CART	0.38	GA-SAE	0.42	CART	0.51	LP	0.63
CART	0.40	CoGEE$_{IBEA}$	0.38	CBR3	0.41	CBR3	0.51	NSGAII-UO	0.57
CBR3	0.40	GA-CI	0.36	LP	0.39	CoGEE$_{IBEA}$	0.49	CBR3	0.56
CoGEE$_{IBEA}$	0.39	CBR3	0.34	CoGEE$_{IBEA}$	0.39	GA-CI	0.48	CBR2	0.56
Median	0.38	Median	0.33	CBR2	0.38	CBR2	0.47	CBR1	0.55
CBR2	0.35	CBR2	0.32	CBR1	0.31	Median	0.33	CoGEE$_{IBEA}$	0.52
CBR1	0.29	CBR1	0.27	Mean	0.17	NSGAII-UO	0.29	Median	0.49
Mean	0.25	Mean	0.26	Median	0.14	Mean	0.27	CART	0.46
NSGAII-UO	-0.18	NSGAII-UO	-0.05	NSGAII-UO	0.09	CBR1	0.26	Mean	0.30

a. This value differs from the original study, which incorrectly reported a value of 0.90. The correct value of 0.66 is shown in this table.

TABLE 3: RQ1. Results of the Wilcoxon test ($\hat{A}_{12}$ effect size in brackets) performed on the Mean of Absolute Errors provided by CoGEE$_{NSGAII}$, compared to the baselines: Mean, Median, and Random Guessing. For completeness, we also include the other three alternative evolutionary algorithms considered later in answer to RQ3 (i.e., GA-CI, GA-SAE, and NSGAII-UO), and four variants of CoGEE in answer to RQ5 (i.e., CoGEE$_{IBEA}$, CoGEE$_{MOCell}$, CoGEE$_{NSGAIII}$, and CoGEE$_{SPEA2}$).

Dataset	Technique	Mean	Median	Random
China	CoGEE$_{NSGAII}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{IBEA}$	<0.001 (0.80)	0.320 (0.53)	<0.001 (0.91)
	CoGEE$_{MOCell}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{NSGAIII}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{SPEA2}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	GA-CI	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	GA-SAE	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	NSGAII-UO	0.999 (0.10)	0.999 (0.00)	0.678 (0.46)
Desharnais	CoGEE$_{NSGAII}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{IBEA}$	<0.001 (0.93)	<0.001 (0.77)	<0.001 (0.99)
	CoGEE$_{MOCell}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{NSGAIII}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{SPEA2}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	GA-CI	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	GA-SAE	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	NSGAII-UO	0.999 (0.00)	0.999 (0.00)	0.601 (0.48)
Finnish	CoGEE$_{NSGAII}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{IBEA}$	<0.001 (0.97)	<0.001 (0.97)	<0.001 (0.99)
	CoGEE$_{MOCell}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{NSGAIII}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{SPEA2}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	GA-CI	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	GA-SAE	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	NSGAII-UO	0.991 (0.33)	0.831 (0.43)	<0.001 (0.75)
Maxwell	CoGEE$_{NSGAII}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{IBEA}$	<0.001 (0.97)	<0.001 (0.97)	<0.001 (1.00)
	CoGEE$_{MOCell}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{NSGAIII}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{SPEA2}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	GA-CI	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	GA-SAE	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	NSGAII-UO	<0.001 (0.83)	0.999 (0.03)	<0.001 (1.00)
Miyazaki	CoGEE$_{NSGAII}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{IBEA}$	<0.001 (1.00)	<0.001 (0.80)	<0.001 (1.00)
	CoGEE$_{MOCell}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{NSGAIII}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE$_{SPEA2}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	GA-CI	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	GA-SAE	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	NSGAII-UO	<0.001 (1.00)	<0.001 (0.97)	<0.001 (1.00)

To answer RQ3.2, we compared CoGEE$_{NSGAII}$ with a bi-objective formulation of the effort estimation problem that minimises underestimates and overestimates simultaneously (NSGAII-UO). This algorithm uses the same implementation of the CoGEE$_{NSGAII}$, while only the objectives are different. As shown in Table 2, NSGAII-UO has poor SA values compared to CoGEE$_{NSGAII}$ for all the datasets. NSGAII-UO is performing even worse than Random Guessing on two out of five datasets (i.e., it achieves a negative SA value on China and Desharnais). This finding is in line with the previous study, indicating that the selection of the objective function plays a crucial role in guiding the algorithm to find high accuracy models for prediction. Table 6b shows the results of the comparison between CoGEE$_{NSGAII}$ and NSGAII-UO with respect to their resulting Pareto Front quality indicators. As we can see, CoGEE$_{NSGAII}$ has achieved considerably better values, especially, considering I_{GD}; the Pareto Front of NSGAII-UO is always dominated by that of CoGEE$_{NSGAII}$. As for the I_C indicator, CoGEE$_{NSGAII}$ dominates the Pareto Front of NSGAII-UO to a large extent for all the datasets. However, the I_{HV} indicator suggests that the difference in the volume of the objective space dominated by the two algorithms is not as much different. This is explainable by the fact that both the algorithms shape a high degree of spread in the solutions in the objective space, which is one of the strong aspects of bi-objective algorithms in general.

The results of the Wilcoxon test presented in Table 7b shows that the Pareto Front quality achieved by CoGEE$_{NSGAII}$ is statistically significantly different from that of NSGAII-UO with a large effect size for all the datasets. Therefore, CoGEE$_{NSGAII}$ passes RQ3.2 by providing higher accuracy and better quality of solutions than NSGAII-UO.

The positive answers for both RQ3.1 and RQ3.2 provide scientific evidence that it is CoGEE's multi-objective nature (i.e the simultaneous optimisation for CI and SAE) that makes it able to outperform all the other approaches.

TABLE 4: RQ2. Results of the Wilcoxon test ($\hat{A}_{12}$ effect sizes in brackets) comparing the Mean of the Absolute Errors for CoGEE$_{NSGAII}$, with those for the state-of-the-art techniques, CBR1–3, CART, and LP.

CoGEE$_{NSGAII}$ vs	CBR1	CBR2	CBR3	CART	LP
China	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (0.63)
Desharnais	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (0.97)	1.000 (0.00)
Finnish	<0.001 (1.00)	<0.001 (1.00)	<0.001 (0.87)	1.000 (0.00)	<0.001 (1.00)
Maxwell	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
Miyazaki	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)

TABLE 5: RQ3.1. Percentage of runs in which a solution found by CoGEE$_{NSGAII}$ dominates (D), or is equal (E) to, the solution found by the single objective GA (i.e., GA-SAE, or GA-CI) on the test sets, based on both objectives simultaneously (column one), and based on each of them separately (columns two and three for SAE and CI, respectively).

Dataset	SAE & CI				SAE				CI			
	CoGEE$_{NSGAII}$ vs.				CoGEE$_{NSGAII}$ vs.				CoGEE$_{NSGAII}$ vs.			
	GA-SAE		GA-CI		GA-SAE		GA-CI		GA-SAE		GA-CI	
	D	E	D	E	D	E	D	E	D	E	D	E
China	71%	0%	84%	0%	72%	0%	88%	5%	66%	2%	86%	2%
Desharnais	59%	0%	64%	0%	61%	0%	83%	0%	66%	0%	77%	0%
Finnish	66%	6%	76%	0%	81%	7%	76%	0%	66%	14%	72%	4%
Maxwell	83%	0%	66%	0%	87%	0%	100%	0%	100%	0%	66%	1%
Miyazaki	8%	92%	80%	20%	8%	92%	80%	20%	6%	94%	26%	74%

TABLE 6: RQ3.1 and RQ3.2. Mean of the quality indicators (I_{GD}, I_{HV}, I_C) computed on the Pareto Fronts of CoGEE$_{NSGAII}$ and the considered single-objective evolutionary approaches (a), and NSGAII-UO (b) over 30 runs. The best value per each indicator is printed in bold face.

(a)

Dataset	Technique	I_{GD}	I_{HV}	I_C
China	CoGEE$_{NSGAII}$	**0.01**	**0.70**	**0.91**
	GA-SAE	0.14	0.28	0.05
	GA-CI	0.18	0.30	0.04
Desharnais	CoGEE$_{NSGAII}$	**0.00**	**0.72**	**0.99**
	GA-SAE	0.06	0.17	0.01
	GA-CI	0.04	0.12	0.01
Finnish	CoGEE$_{NSGAII}$	**0.00**	**0.58**	**0.98**
	GA-SAE	0.01	0.07	0.01
	GA-CI	0.01	0.01	0.01
Maxwell	CoGEE$_{NSGAII}$	**0.00**	**0.82**	**0.99**
	GA-SAE	0.03	0.22	0.01
	GA-CI	0.03	0.19	0.01
Miyazaki	CoGEE$_{NSGAII}$	**0.00**	**0.67**	**0.98**
	GA-SAE	0.07	0.61	0.01
	GA-CI	0.22	0.58	0.00

(b)

Dataset	Technique	I_{GD}	I_{HV}	I_C
China	CoGEE$_{NSGAII}$	**0.00**	**1.00**	**0.97**
	NSGAII-UO	0.83	0.24	0.03
Desharnais	CoGEE$_{NSGAII}$	**0.00**	**0.99**	**0.99**
	NSGAII-UO	0.09	0.74	0.01
Finnish	CoGEE$_{NSGAII}$	**0.00**	**1.00**	**0.99**
	NSGAII-UO	0.13	0.77	0.01
Maxwell	CoGEE$_{NSGAII}$	**0.00**	**0.99**	**1.00**
	NSGAII-UO	0.06	0.96	0.00
Miyazaki	CoGEE$_{NSGAII}$	**0.00**	**1.00**	**0.92**
	NSGAII-UO	0.05	1.00	0.08

TABLE 7: RQ3.1 and RQ3.2. Results of the Wilcoxon test (with $\hat{A}_{12}$ effect sizes in brackets) which compare the quality indicators (I_{GD}, I_{HV}, I_C) of CoGEE$_{NSGAII}$, to the ones obtained by single-objective evolutionary approaches (a), and NSGAII-UO (b) over 30 runs.

(a)

Dataset	CoGEE$_{NSGAII}$ vs	I_{GD}	I_{HV}	I_C
China	GA-SAE	<0.025 (0.69)	<0.025 (0.90)	<0.025 (0.95)
	GA-CI	<0.025 (0.77)	<0.025 (0.86)	<0.025 (0.91)
Desharnais	GA-SAE	<0.025 (0.68)	<0.025 (1.00)	<0.025 (1.00)
	GA-CI	<0.025 (0.61)	<0.025 (1.00)	<0.025 (1.00)
Finnish	GA-SAE	0.802 (0.47)	<0.025 (0.96)	<0.025 (1.00)
	GA-CI	<0.025 (0.69)	<0.025 (1.00)	<0.025 (1.00)
Maxwell	GA-SAE	<0.025 (0.64)	<0.025 (1.00)	<0.025 (1.00)
	GA-CI	<0.025 (0.61)	<0.025 (0.94)	<0.025 (0.94)
Miyazaki	GA-SAE	<0.025 (0.54)	0.166 (0.54)	<0.025 (1.00)
	GA-CI	<0.025 (0.90)	<0.025 (0.69)	<0.025 (1.00)

(b)

Dataset	CoGEE$_{NSGAII}$ vs	I_{GD}	I_{HV}	I_C
China	NSGAII-UO	<0.05 (1.00)	<0.05 (0.99)	<0.05 (0.98)
Desharnais	NSGAII-UO	<0.05 (1.00)	<0.05 (1.00)	<0.05 (1.00)
Finnish	NSGAII-UO	<0.05 (1.00)	<0.05 (0.99)	<0.05 (1.00)
Maxwell	NSGAII-UO	<0.05 (1.00)	<0.05 (1.00)	<0.05 (1.00)
Miyazaki	NSGAII-UO	<0.05 (0.99)	<0.05 (0.81)	<0.05 (1.00)

4.4 RQ4. Comparison to Industrial Practice

Figure 2a shows the box plots of the Magnitude of Relative Error (MRE) for CoGEE$_{NSGAII}$ and four state-of-the-art techniques. As in the original study, we compared the accuracy of our model with two thresholds of error (i.e., 30% and 40% of the true value of the effort, plotted on Figure 2a by two dotted-lines set at 1.3 and 1.89). These two thresholds are derived from evidence that industrial practice based on expert judgement hopes/claims to produce predictions within [75], [89]. As the box plots show, for three of the datasets, the distribution of MRE for CoGEE$_{NSGAII}$ lies below these thresholds.[10]

It is arguable that overestimate and underestimate of the effort of a software project can have very different consequences. The former may cause a company to miss a contract, or if engaged in the project, may lead to misallocation of resources where they are not needed, while the latter can harm a company financially and can also affect its reputation. It is also revealed that managers of software companies are more concerned about underestimated project efforts [70] and might be interested to see the distribution of the magnitude of underestimated projects for each technique. The industrial thresholds used in Figure 2a is derived from the current industrial claims concerning project overrun. That being, the distribution of underestimates of an estimation technique are expected to fall within this threshold, in order to make the technique competitive and actionable for industrial uptake.

Figure 2b reports the distribution of overrun project budgets obtained by each of the effort prediction techniques, including CoGEE$_{NSGAII}$. We can observe that the median project overrun for CoGEE$_{NSGAII}$ lies within these thresholds for all the datasets. More specifically, for Desharnais and Miyazaky the entire distribution of overrun values from CoGEE$_{NSGAII}$ lie within the upper bound, and for Maxwell and Finish, the vast majority of the distribution of overruns lies within this bound, while this can not be said for the other state-of-the-art techniques.

This evidence supports the claim that CoGEE$_{NSGAII}$ advances the state-of-the-art that can be expected from automated estimators within the bounds of current claims for industrial best practice. These findings are in line with the original study in that CoGEE$_{NSGAII}$ can advance the claimed state-of-best-practice as well as the known scientific state-of-the-art.

4.5 RQ5. Using other MOEAs

The comparison between multi-objective evolutionary algorithms used as the search algorithm under the hood of CoGEE is carried out by analysing the SA achieved, investigating their ability in providing high-quality Pareto Fronts, and their running time.

Table 2 shows the SA values achieved by each of the MOEAs for each dataset. All variants of CoGEE, except for CoGEE$_{IBEA}$ achieved very similar SA values. We can see that CoGEE$_{MOCell}$ resulted in the highest SA value on three out of five datasets.

Table 8 shows the analysis of three quality indicators for MOEAs. Except for Miyazaki, for all datasets, SPEA2 is the MOEA with the highest I_C indicator. The I_C is so high that for three out of five dataset SPEA2 appears to provide more than 80% of the solutions in the Reference Front. However, very similar results for I_{GD} and I_{HV} indicators by all the MOEAs, except for IBEA, suggests that the Pareto Fronts provided by these algorithms are very close and equally spread, which is in line with the fact that they achieved very similar SA values.

10. In Figure 2a we show the MRE distributions achieved by the different models solely to depict the relationship of each technique to the two industrial thresholds. This measure is not recommended for comparing prediction models (i.e., selecting the best technique) as, similarly to using MMRE, its use could be misleading (see Section 2.1).

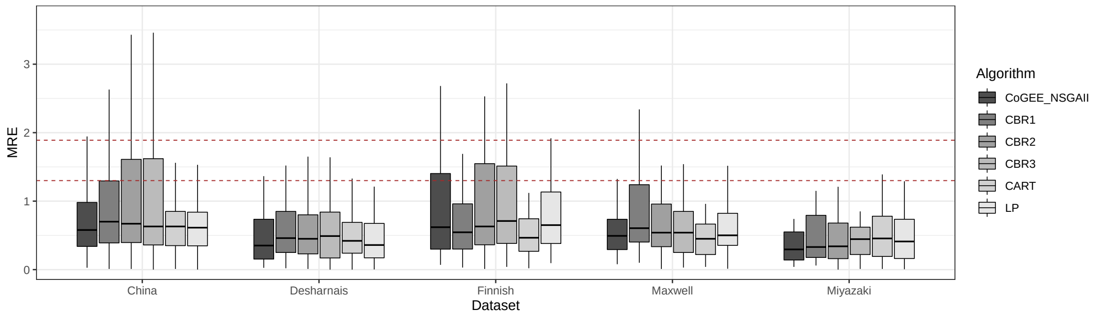
TABLE 8: RQ5. Mean of the quality indicators (I_{GD}, I_{HV}, I_C) computed on the Pareto Fronts of CoGEE$_{NSGAII}$ and the four alternative MOEAs (i.e., CoGEE$_{IBEA}$, CoGEE$_{MOCell}$, CoGEE$_{NSGAIII}$, and CoGEE$_{SPEA2}$) over 30 runs. The best value per each indicator is printed in bold face.

Dataset	Technique	I_{GD}	I_{HV}	I_C
China	CoGEE$_{IBEA}$	0.21	0.75	0.00
	CoGEE$_{MOCell}$	**0.00**	**1.00**	0.01
	CoGEE$_{NSGAII}$	**0.00**	**1.00**	0.06
	CoGEE$_{NSGAIII}$	**0.00**	**1.00**	0.08
	CoGEE$_{SPEA2}$	**0.00**	**1.00**	**0.84**
Desharnais	CoGEE$_{IBEA}$	0.23	0.56	0.00
	CoGEE$_{MOCell}$	**0.00**	0.97	0.03
	CoGEE$_{NSGAII}$	**0.00**	0.96	0.04
	CoGEE$_{NSGAIII}$	**0.00**	0.97	0.05
	CoGEE$_{SPEA2}$	**0.00**	**0.98**	**0.89**
Finnish	CoGEE$_{IBEA}$	0.23	0.59	0.00
	CoGEE$_{MOCell}$	**0.00**	**0.99**	0.18
	CoGEE$_{NSGAII}$	**0.00**	0.98	0.11
	CoGEE$_{NSGAIII}$	**0.00**	**0.99**	0.32
	CoGEE$_{SPEA2}$	**0.00**	**0.99**	**0.39**
Maxwell	CoGEE$_{IBEA}$	0.24	0.60	0.00
	CoGEE$_{MOCell}$	**0.00**	0.97	0.01
	CoGEE$_{NSGAII}$	**0.00**	0.97	0.03
	CoGEE$_{NSGAIII}$	**0.00**	0.97	0.05
	CoGEE$_{SPEA2}$	**0.00**	**0.99**	**0.91**
Miyazaki	CoGEE$_{IBEA}$	0.78	0.19	0.00
	CoGEE$_{MOCell}$	**0.00**	**1.00**	0.00
	CoGEE$_{NSGAII}$	**0.00**	**1.00**	0.31
	CoGEE$_{NSGAIII}$	**0.00**	**1.00**	**0.37**
	CoGEE$_{SPEA2}$	**0.00**	**1.00**	0.32

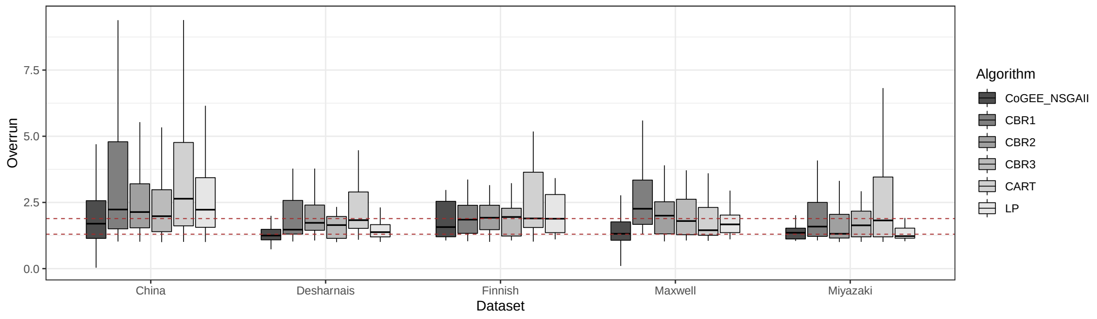
TABLE 9: RQ5. Results of the Wilcoxon test (with $\hat{A}_{12}$ effect sizes in brackets) which compare the quality indicators (I_{GD}, I_{HV}, I_C) of CoGEE$_{NSGAII}$, to the ones obtained by four alternative MOEAs over 30 runs.

Dataset	CoGEE$_{NSGAII}$ vs	I_{GD}	I_{HV}	I_C
China	CoGEE$_{IBEA}$	<0.0125 (1.00)	<0.0125 (1.00)	<0.0125 (0.63)
	CoGEE$_{MOCell}$	<0.0125 (0.67)	0.7479 (0.47)	0.9777 (0.43)
	CoGEE$_{NSGAIII}$	0.9325 (0.44)	0.9522 (0.43)	0.9981 (0.39)
	CoGEE$_{SPEA2}$	1.0000 (0.02)	1.0000 (0.13)	1.0000 (0.00)
Desharnais	CoGEE$_{IBEA}$	<0.0125 (1.00)	<0.0125 (1.00)	<0.0125 (0.71)
	CoGEE$_{MOCell}$	0.4240 (0.57)	0.9563 (0.43)	0.8969 (0.45)
	CoGEE$_{NSGAIII}$	0.9977 (0.38)	0.7130 (0.48)	0.9607 (0.43)
	CoGEE$_{SPEA2}$	1.0000 (0.67)	1.0000 (0.21)	1.0000 (0.00)
Finnish	CoGEE$_{IBEA}$	<0.0125 (1.00)	<0.0125 (1.00)	<0.0125 (0.89)
	CoGEE$_{MOCell}$	1.0000 (0.23)	1.0000 (0.33)	1.0000 (0.26)
	CoGEE$_{NSGAIII}$	1.0000 (0.15)	0.9903 (0.40)	1.0000 (0.12)
	CoGEE$_{SPEA2}$	1.0000 (0.30)	0.9965 (0.38)	1.0000 (0.06)
Maxwell	CoGEE$_{IBEA}$	<0.0125 (1.00)	<0.0125 (1.00)	<0.0125 (0.75)
	CoGEE$_{MOCell}$	0.1525 (0.55)	0.2521 (0.53)	<0.0125 (0.59)
	CoGEE$_{NSGAIII}$	0.9999 (0.26)	0.3428 (0.52)	0.9994 (0.36)
	CoGEE$_{SPEA2}$	1.0000 (0.00)	1.0000 (0.06)	1.0000 (0.00)
Miyazaki	CoGEE$_{IBEA}$	<0.0125 (1.00)	<0.0125 (1.00)	<0.0125 (1.00)
	CoGEE$_{MOCell}$	1.0000 (0.50)	1.0000 (0.50)	<0.0125 (1.00)
	CoGEE$_{NSGAIII}$	1.0000 (0.50)	1.0000 (0.50)	1.0000 (0.00)
	CoGEE$_{SPEA2}$	1.0000 (0.50)	1.0000 (0.50)	0.9219 (0.49)

Table 9 shows the results of the Wilcoxon test between the distribution of the quality indicator values achieved by CoGEE$_{NSGAII}$ against each of the other MOEAs. These results are corrected for multiple tests using Bonferroni correction (making α level at $\alpha < 0.0125$). Therefore, we can reject our null hypothesis (see Section 3.4) only for CoGEE$_{IBEA}$, with a medium to high effect size depending on the datasets. On the other hand, all other MOEAs produce solutions with equal quality in comparison with CoGEE$_{NSGAII}$.



(a) Magnitude of Relative Error



(b) Overrun

Fig. 2: RQ4. Comparison with claimed optimal industrial practice when the (a) the Magnitude of Relative Error (MRE) and (b) overrun of each project in a given dataset is considered.

On the other end, when comparing the running time of CoGEE with each of the five MOEAs (see Table 10), we can observe that IBEA and SPEA2 are the slowest algorithms (their running time ranges from 10,000 to more than 100,000 milliseconds), while all the other MOEAs execute in less than 1,500 milliseconds. Table 11 shows the results of the Wilcoxon test comparing the running time of CoGEE$_{NSGAII}$ against other MOEAs. The results suggest that the null hypothesis (see Section 3.4) is rejected for CoGEE$_{NSGAIII}$, CoGEE$_{IBEA}$, and CoGEE$_{SPEA2}$, which means that the median running time for these three algorithms is higher than that of CoGEE$_{NSGAII}$.

Based on both the SA and quality indicator analysis, and considering the running time of the MOEAs, we conclude that CoGEE$_{MOCell}$ and CoGEE$_{NSGAII}$ are able to achieve the best quality with a faster running time.

4.6 RQ6. CoGEE's Running Time: Java vs. R

To assess whether the running time of CoGEE decreases due to the new `Java` implementation, we analyse and compare it with the running time of the original `R` implementation. Table 10 shows the average running time and standard deviation of CoGEE$_{NSGAII}$ implemented in `Java`, and its `R` counterpart CoGEE$_{NSGAII-R}$.

Overall, CoGEE$_{NSGAII}$ runs 718 and 5,358 times faster than CoGEE$_{NSGAII-R}$ for Finnish (the smallest dataset) and China (the biggest one), respectively. This translates in a decrease in running time between 99.86% and 99.98% when comparing the `R` version to its `Java` counterpart. CoGEE$_{NSGAII-R}$ is extremely slow and it took days to execute all experiments. On the other hand, the same experiments run by using CoGEE$_{NSGAII}$ terminated in few minutes. Given the magnitude of this difference, unsurprisingly, the results of the Wilcoxon test (see Table 11) confirm that, for all datasets, the running time of CoGEE$_{NSGAII}$ is statistically significantly lower than the running time of CoGEE$_{NSGAII-R}$ with a large effect size in all cases considered.

These results are in line with previous studies suggesting that `R` is not well-suited for computationally intensive algorithms [76], [110]. Since the results in terms of accuracy are similar (see Table 2), we recommend using the `Java` version and we have made it publicly available [105], [106].

5 Discussion

Overall, we can observe that the results of our replication (RQ1, RQ2, RQ3) are consistent with the original study, thus reinforcing the conclusions previously made. There are some minor discrepancies, which, however, do not change the conclusions, as discussed in the following.

The results for RQ1 and RQ2 provide scientific evidence that, overall, CoGEE$_{NSGAII}$ outperforms both baseline and state-of-the-art approaches, which confirm previous finding. CoGEE$_{NSGAII}$ significantly outperformed all the baseline benchmarks for all datasets, and also the state-of-the-art techniques in the three largest cross-company datasets (i.e., China, Maxwell, and Miyazaki) with large effect sizes for 22 out of 23 cases. For the other two datasets, i.e., Desharnais and Finnish, CoGEE$_{NSGAII}$ is ranked second only to LP and CART, respectively, where the results of the statistical analysis shows a small effect size. We observe that the SA

TABLE 10: RQs5–6. Mean, Median and Standard Deviation (SD) of the running time over 30 runs of all MOEA variants, including the `R` version used in the original study (CoGEE$_{NSGAII-R}$).

		Running Time (milliseconds)		
Dataset	Technique	Mean	Median	Std. Dev.
China	CoGEE$_{IBEA}$	11,550	11,537	66
	CoGEE$_{MOCell}$	1,035	1,035	47
	CoGEE$_{NSGAII}$	1,100	1,101	35
	CoGEE$_{NSGAIII}$	1,381	1,381	51
	CoGEE$_{SPEA2}$	212,582	212,251	3,292
	CoGEE$_{NSGAII-R}$	5,894,548	5,746,701	272,092
Desharnais	CoGEE$_{IBEA}$	11,699	11,688	60
	CoGEE$_{MOCell}$	473	476	18
	CoGEE$_{NSGAII}$	530	526	19
	CoGEE$_{NSGAIII}$	725	725	18
	CoGEE$_{SPEA2}$	148,815	150,599	6,504
	CoGEE$_{NSGAII-R}$	1,257,656	1,248,304	13,946
Finnish	CoGEE$_{IBEA}$	11,698	11,701	39
	CoGEE$_{MOCell}$	570	592	54
	CoGEE$_{NSGAII}$	545	534	55
	CoGEE$_{NSGAIII}$	742	741	9
	CoGEE$_{SPEA2}$	171,462	172,003	5,958
	CoGEE$_{NSGAII-R}$	391,719	392,129	20,187
Maxwell	CoGEE$_{IBEA}$	11,682	11,678	37
	CoGEE$_{MOCell}$	539	538	17
	CoGEE$_{NSGAII}$	562	556	16
	CoGEE$_{NSGAIII}$	743	740	14
	CoGEE$_{SPEA2}$	128,248	128,514	504
	CoGEE$_{NSGAII-R}$	2,606,284	2,598,778	77,340
Miyazaki	CoGEE$_{IBEA}$	11,616	11,612	35
	CoGEE$_{MOCell}$	451	452	6
	CoGEE$_{NSGAII}$	553	552	5
	CoGEE$_{NSGAIII}$	770	770	5
	CoGEE$_{SPEA2}$	143,218	143,216	119
	CoGEE$_{NSGAII-R}$	514,834	503,486	96,071

values obtained by our study are slightly different from those reported in the original study. This is mainly due to the stochastic nature of these algorithms involved, and are small enough to not change the ranking of the techniques considered, so can be neglected and the previous findings can be confirmed.

The results for RQ3 show that CoGEE$_{NSGAII}$ is able to produce higher quality solutions than the two single-objective variants as well as than NSGAII-UO. The SA results alongside the Pareto Front analysis, reveal that the single-objective variants were not able to produce satisfactory solutions for neither of the two objectives (i.e., they often produced solutions which had worse SAE or CI with respect to CoGEE$_{NSGAII}$ as can be observed for example for the Finnish dataset in Figure 3)[11], whereas NSGAII-UO was able to produce a few good quality solutions but the majority were very poor (see, for example, Figure 4)[11]. These results suggest that simultaneously optimising for CI and SAE (which basically combines underestimates and overestimates into one objective) makes CoGEE$_{NSGAII}$ able to reach such a good performance. The results of the quality indicators for RQ3.1 and RQ3.2 are slightly different from those obtained in the original study but consistent. The replication study showed more significant differences in the comparison of CoGEE$_{NSGAII}$ vs. GA-SAE and GA-CI, which reinforces the conclusions drawn for RQ3.1 in the original study. The reason is mainly due to the stochastic nature of the approaches considered and the different frameworks used[12], which has been mitigated by executing 30 independent executions. The differences observed, though, did not change the overall conclusion that CoGEE outperformed the other three evolutionary-based techniques (RQ3).

The results for RQ4 show that the median project overrun for CoGEE$_{NSGAII}$ lies within the bounds of current claims for industrial best practice for all five datasets, and the distribution of its MRE lies below this thresholds for three out of five datasets. These are the same results of the original study, thus confirming the same conclusions. It is possible that the current reluctance to take-up intelligent effort estimation within the industry is due to the risk aversion and reticence to risk overrun. The result of RQ4 shows that CoGEE can produce estimations with a lower risk of cost overrun, as a result of optimisation for CI, thus, mitigates the managers' natural concern for underestimates and consequent budget overrun.

Furthermore, the newly investigated questions, RQ5 and RQ6, shed light on the role played by different types of MOEAs and their efficacy in terms of running time when coded with two different languages (i.e., `R` and `Java`). In the original study an `R` implementation of NSGA-II was used to realise CoGEE, while in this replication we also used GA, NSGA-II, NSGA-III, SPEA2, MOCell and IBEA (all realised in `Java` based on the `JMetal` framework). We did not find any statistical significant differences among the accuracy of the estimation models built by using these NSGA-II and the other MOEAS (except for IBEA which was significantly worse for all datasets studied). On the other end, although SPEA2 and NSGA-III produced models of comparable quality to NSGA-II, they were slower. This is in line with the findings of Khare et al. [57] who compared the performance of NSGA-II and SPEA2 in a different domain. They found that SPEA2 scales well in terms of maintaining the diversity of the solutions in the Pareto Front; however, it suffers in running time [57]. The higher running time of SPEA2 is due to the maintenance of a separate archive population and a higher volume of computations that it needs in order to rank the solutions for selection since it uses an additional factor (i.e., domination strength [116]) to determine the ranks of the solutions. Similarly, previous work showed that NSGA-III was found to be worse than NSGA-II for some problems [49], and generally slower when used with fewer objectives or smaller instances [18]. On the other hand, IBEA, which also uses an additional computation of a quality indicator (i.e., Hypervolume) to rank its solutions, fails to produce good quality solutions for the problem investigated herein. Sayyad et al. [90] showed that IBEA could be very slow to converge, although it delivers better scalability in a higher number of objectives in comparison with NSGA-II. Finally, we have also found that using the `JMetal` version of CoGEE allows us to decrease the running time by over 99.8% with respect to using its `R` counterpart. Based on the results we obtained for RQ5 and RQ6, we can conclude that the effectiveness of CoGEE is generally not limited to nor dependent on the choice of the multi-objective

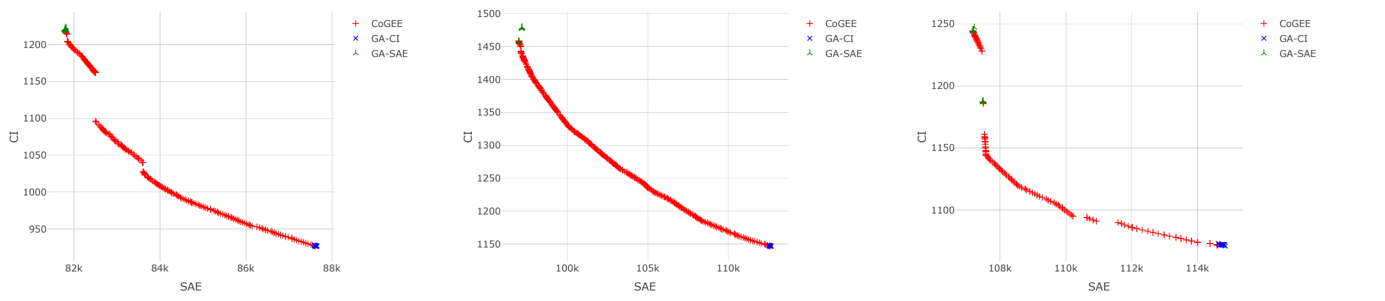
11. For sake of space we make all plots available on-line [106]

12. The original study used `R` with `nsgr2r` library version 1.0, while this study uses the `Java JMetal` framework version 5.4

TABLE 11: RQs5-6. Results of the Wilcoxon test ($\hat{A}_{12}$ effect sizes in brackets) comparing the distribution of the running time obtained over 30 runs by all the MOEA variants and the original R version (CoGEE$_{NSGAII-R}$).

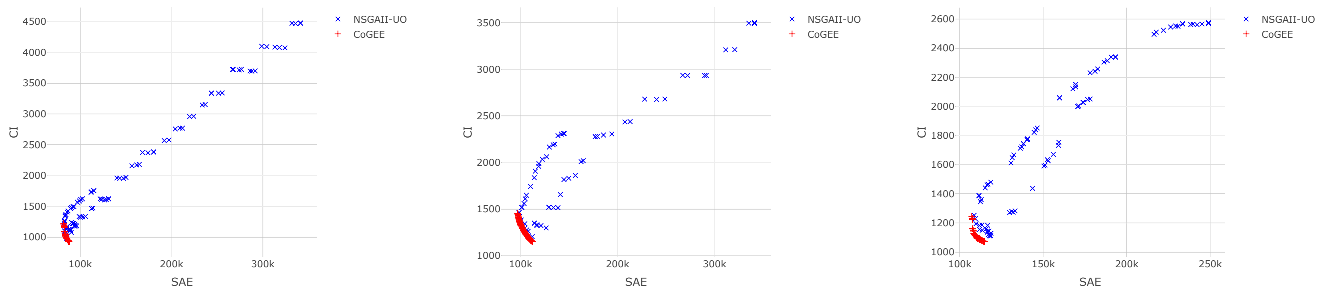
CoGEE$_{NSGAII}$ vs	China	Desharnais	Finnish	Maxwell	Miyazaki
CoGEE$_{IBEA}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
CoGEE$_{MOCell}$	1.000 (0.07)	1.000 (0.01)	0.001 (0.63)	1.000 (0.13)	1.000 (0.00)
CoGEE$_{NSGAIII}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (0.98)	<0.001 (1.00)	<0.001 (1.00)
CoGEE$_{SPEA2}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
CoGEE$_{NSGAII-R}$	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)

Fig. 3: RQ3.1. Pareto Fronts for CoGEE$_{NSGAII}$ (CoGEE) vs. GA-SAE and GA-CI on the Finnish dataset. The non-dominated solutions for CoGEE and the best solution for GA-SAE and GA-CI for each of the 30 independent runs are plotted.



(a) Fold 1 (b) Fold 2 (c) Fold 3

Fig. 4: RQ3.2. Pareto Fronts obtained by CoGEE$_{NSGAII}$ (CoGEE) and NSGAII-UO for the Finnish dataset (one randomly selected run among the 30 performed).



(a) Fold 1 (b) Fold 2 (c) Fold 3

evolutionary algorithm (RQ5), however different MOEAs exhibit different running times (RQ6), and the framework used to realise them have a significant impact on their speed (RQ6).

6 Threats to Validity

Since this study is a replication and an extension of the research performed by Sarro et al. [89] it shares some of the threats to validity and implements further mitigations.

Construct validity deals with the degree to which the predictor and response variable measures what they suppose to be measuring [71]. To mitigate possible threats arising from the predictors and target variables used to build the prediction models we used five publicly available datasets which were collected from real-world projects and contain reliable measures of software effort and size, such as Function Point, which are still in use in industrial settings and widely used in research studies [2], [27], [87]. Moreover, this data has been widely used in previous empirical studies to validate estimation models [10], [47], [48], [87]. Further, to mitigate possible threats in measuring the accuracy performance of our models we used robust evaluation measures such as the Mean Absolute Error and the Standard Accuracy, which are widely used and recommended in previous work [64], [88], [95]. To compare the evolutionary algorithms, we followed best practice by using a complementary set of Pareto Front quality indicators, which fairly compare all algorithms against a common reference front [43], [67].

Conclusion validity or internal validity represents to what extend it is valid to draw conclusions about the causal relation between the predictor and response variables [71]. In other words, is it valid to draw conclusions based on the results of the tests? In this regard, following the best practice, we chose standardized measures [97] to compare the results of our experiments and carefully applied the statistical significance tests with correction for multiple tests [6]. The new independent implementation strengthens the internal validity of the study.

Since the team of researchers involved in this replication study partially overlaps with the authors of the original study, potential researcher bias may arise in answering the RQs replicated from the original study (i.e., RQs 1–4). To minimise this threat, only the author who was not involved in the original study, was responsible for answering these questions. Specifically, the first author had access only to the original paper and the data used in the original study in order to replicate it, and he was responsible for coding the approaches and experiments, executing them and analysing the results from scratch, based on the experimental design explained in the paper. At the end of the replication the first author produced a report with the experimental results of the replicated RQs 1–4, and was given access to the original source code and raw results for comparison purposes. Throughout the replication, the first author sought consultation from the authors of the original study only on matters that needed some clarifications beyond the information provided in the original paper (e.g., additional information on the algorithmic setting which were unclear/missing from the original paper). Since RQ5 and RQ6 are new research questions, which were not included in the original study, no mitigation measures were needed for these goals.

We alleviated the threats to external validity using multiple datasets from different contexts with a high diversity in the type of features and the number of instances, which also mitigate the threats related to the number of observations. The datasets, coming from different contexts, might be characterised by some specific projects and human factors, such as development process, developer experience, tools and techniques used, and different time and budget, that makes it hard to claim that our results generalise beyond the subjects that we studied [89]. The external validity of the original study has been further strengthened herein by using four additional MOEAs with CoGEE, each one coming from a different family of multi-objective optimisation algorithms. To compare the results of CoGEE$_{NSGAII}$ with industrial practice we adopted the thresholds of industrial prediction from a survey of industry practices carried out in 2003 [75], thus one should take this into consideration while generalizing this result to other periods.

7 Related Work

Sarro et al. [89] classified previous work on robust effort estimation with confidence intervals into two broad categories: (i) those that produced confidence interval for point estimates during the estimation process, and (ii) those that produce probabilities of the predefined intervals prior to the effort estimation.

CoGEE falls in the first category as it optimises the confidence interval while building the model, thus prediction models produced by it optimise both the accuracy of the point estimates and the confidence interval associated with them [89].

7.1 Confidence Interval for Point Estimates

The first study in the first category, proposed a bootstrap-based model for tuning an analogy based effort estimation model [5], however as pointed out by Jørgensen [52] this work confuses the Prediction Interval (PI) with the confidence interval of the mean effort. In a subsequent study, Jørgensen and Sjoeberg [54] introduced an approach assuming that the estimation accuracy of earlier software projects is a prediction for the effort PI of new projects. They evaluated their approach with two empirical studies to provide insight into when it is appropriate to use the proposed approach, regression-based approaches, or software professionals' judgement. The results showed that with a sufficiently high number of observations the regression model will result in more accurate effort PIs. On the other hand, when there are few samples and outliers are presented, the proposed approach may lead to better effort PIs. Finally, they suggest supporting and training software professionals to reduce their bias towards overconfidence may be the optimal solution.

Braga et al. [11] introduced a machine learning method which provides the estimation of the effort and a corresponding confidence interval. Their method for computing the confidence interval is independent of the probability distribution of the errors in the training set, unlike previous work in which the confidence intervals for predictions were computed by assuming that the errors follow some probability distribution. They evaluated the proposed approach on two datasets. The results showed that the proposed method was able to build robust confidence intervals.

Song et al. [102] introduced Relevance Vector Machine (RVM), a probabilistic model based on Bayesian inference, to construct Prediction Intervals (PI) with a confidence level. Song et al. in a recent study [103] stated that RVM sometimes provided too wide PIs that were not informative, therefore proposed a new PI estimator based on RVM called Synthetic Bootstrap ensemble of Relevance Vector Machines (SynB-RVM). This estimator adopts Bootstrap re-sampling to produce multiple RVM models based on modified training bags whose replicated data projects are replaced by their synthetic counterparts. Results showed that SynB-RVM's hit rates and relative widths are no worse than the other compared methods that can provide uncertain estimations. They evaluated their method using 11 datasets (four of which are publicly available) including Maxwell, on which they attained standard accuracy of 0.54 with the best performing variant of SynB-RVM, while this value for the best variant of CoGEE is 0.56.

Mensah et al. [72] proposed a model that automatically determines categorical intervals. They introduce a categorical level of effort (high, moderate, and low) based on a quantile discretisation of the effort values on the training set, and provide this level with each estimation alongside their point estimate. This level contributes to the easy interpretation of the model.

Ezghari and Zahi [30] introduced an improvement on Fuzzy Analogy-based Software Effort Estimation model which employs successfully fuzzy logic with approximate reasoning theory to handle imprecision and reasoning under uncertainty. However, their technique had a detrimental effect on SA for datasets like China, suspecting that the distribution of the samples in this dataset does not allow the uncertainty management parameters to reach an acceptable optimisation degree.

7.2 Probabilities for Predefined Intervals

Only two works fall in the second category. Sentas et al. [91] investigated an ordinal regression technique to model the probability of correctly classifying a new project to one of the predefined cost categories, each of which corresponds to an effort interval. The work by Bakir et al. [7] differs from the previous studies in that the effort intervals are not predefined manually but determined by clustering analysis. They evaluated their approach using seven public datasets. The results of point estimations were comparable to the previous work, but estimation hit around $90-100\%$, which was higher than those obtained in the previous studies.

8 Conclusion

This paper is a replication of a previously published study [89], where a bi-objective software effort estimation algorithm was proposed and evaluated. The original work showed that the proposed technique, namely CoGEE, outperforms state-of-the-art techniques and produces human competitive results [89].

In this replication, we carried out a large empirical study in order to answer the same research questions by means of the same experimental design extended by the use of a recent state-of-the-art baseline benchmark, four additional variants of multi-objective evolutionary algorithms under the hood of CoGEE, execution time evaluation of the different variants, and a completely new and independent implementation of CoGEE based on a popular `Java` Evolutionary Computation framework (i.e., `JMetal` [78]), which runs significantly faster than the original `R` version.

Our replication shows that the results are consistent with the original study to a large extent and there are small differences, which are justified by the stochastic nature of the algorithms used, and do not affect the overall findings. According to our results, CoGEE$_{NSGAII}$ outperforms all benchmarks considered including state-of-the-art techniques. Moreover, it provides estimates within (and even close to the lower) thresholds of industrial best practice. Therefore, confirming CoGEE provides human-competitive results.

The analysis of the Pareto Fronts showed that SPEA2 either dominated the other Fronts or produced solutions with the same quality. However, it seems that the diversity of the solutions on the Front produced by SPEA2 contributed to the lower SA value of this MOEA in comparison with the others on four of the datasets. We also found that among the five MOEAs compared, NSGA-II, NSGA-III and MOCell produce comparable results in terms of solution quality but 100 times quicker than SPEA2. On the other hand, IBEA took 10 times longer than NSGA-II, NSGA-III and MOCell to run. We conclude that by using CoGEE with NSGA-II, NSGA-III, and MOCell, one can produce human competitive results within a minute.

In conclusion, the replication study has increased the confidence in the findings of the original study by producing confirmatory results on the effectiveness of the multi-objective optimisation, and in particular optimising for low uncertainty of the predictions, in Software Effort Estimation. It has also further mitigated potential internal and external threats to the validity of the original study by adding more dimensions to the experimental design including: benchmarking against a new robust state-of-the-art baseline, use of four additional variants of multi-objective evolutionary algorithms, and an independent implementation, execution, and analysis of the experiments. We have made CoGEE's source code publicly available in order to facilitate its adoption. We also provide the data used in our study to allow for replications and extensions [105], [106].

Acknowledgments

The work of Vali Tawosi, Federica Sarro, Alessio Petrozziello and Mark Harman at UCL is supported by the ERC Advanced fellowship grant no. 741278 (EPIC).

References

[1] CRAN - Package nsga2R. URL: https://cran.r-project.org/web/packages/nsga2R/index.html.

[2] International function point user group, ifpug. URL: https://www.ifpug.org/european-parliament-recommends-ifpug-methodology-for-pricing-software-development/.

[3] LP4EE: Linear Programming for Effort Estimation, 2018 (accessed March 27, 2020). URL: https://github.com/fedsar/LP4EE.

[4] R: The R Project for Statistical Computing, Version 3.5.3, 2019 (accessed March 27, 2020). URL: https://www.r-project.org/.

[5] Lefteris Angelis and Ioannis Stamelos. A simulation tool for efficient analogy based cost estimation. *Empirical software engineering*, 5(1):35–68, 2000.

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

[7] Ayşe Bakır, Burak Turhan, and Ayşe Bener. A comparative study for estimating software development effort intervals. *Software Quality Journal*, 19(3):537–552, 2011.

[8] Marcio de Oliveira Barros. An analysis of the effects of composite objectives in multiobjective software module clustering. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 1205–1212, 2012.

[9] Barry W Boehm. Software engineering economics. *IEEE transactions on Software Engineering*, (1):4–21, 1984.

[10] Michael F. Bosu and Stephen G. Macdonell. Experience: Quality benchmarking of datasets used in software effort estimation. *J. Data and Information Quality*, 11(4), August 2019. URL: https://doi.org/10.1145/3328746, `doi:10.1145/3328746`.

[11] Petronio L Braga, Adriano LI Oliveira, and Silvio RL Meira. Software effort estimation using machine learning techniques with robust confidence intervals. In *7th international conference on hybrid intelligent systems (HIS 2007)*, pages 352–357. IEEE, 2007.

[12] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[13] Colin J Burgess and Martin Lefley. Can genetic programming improve software effort estimation? a comparative evaluation. *Information and software technology*, 43(14):863–873, 2001.

[14] Yongtao Cao, Byran J Smucker, and Timothy J Robinson. On using the hypervolume indicator to compare pareto fronts: Applications to multi-criteria optimal experimental design. *Journal of Statistical Planning and Inference*, 160:60–74, 2015.

[15] Jeffrey C Carver. Towards reporting guidelines for experimental replications: A proposal. In *1st international workshop on replication in empirical software engineering*, pages 2–5. Citeseer, 2010.

[16] Yair Censor. Pareto optimality in multiobjective problems. *Applied Mathematics and Optimization*, 4(1):41–59, 1977.

[17] Sunita Chulani, Barry Boehm, and Bert Steece. Bayesian analysis of empirical software engineering cost models. *IEEE Transactions on Software Engineering*, 25(4):573–583, 1999.

[18] Guillermo Campos Ciro, Frédéric Dugardin, Farouk Yalaoui, and Russell Kelly. A nsga-ii and nsga-iii comparison for solving an open shop scheduling problem with resource constraints. *IFAC-PapersOnLine*, 49(12):1272–1277, 2016.

[19] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems Second Edition*. Springer Science, 2nd edition, 2007.

[20] Jacob Cohen. *Statistical power analysis for the behavioral sciences*. Routledge, 2013.

[21] Anna Corazza, Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, Federica Sarro, and Emilia Mendes. How effective is tabu search to configure support vector regression for effort estimation? In *Proceedings of the 6th international conference on predictive models in software engineering*, pages 1–10, 2010.

[22] Anna Corazza, Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, Federica Sarro, and Emilia Mendes. Using tabu search to configure support vector regression for effort estimation. *Empirical Software Engineering*, 18(3):506–546, 2013.

[23] Fabio QB Da Silva, Marcos Suassuna, A César C França, Alicia M Grubb, Tatiana B Gouveia, Cleviton VF Monteiro, and Igor Ebrahim dos Santos. Replication of empirical studies in software engineering research: a systematic mapping study. *Empirical Software Engineering*, 19(3):501–557, 2014.

[24] Kalyanmoy Deb and Himanshu Jain. 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):577–601, 2013.

[25] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[26] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[27] Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, and Federica Sarro. Web effort estimation: Function point analysis vs. cosmic. *Information and Software Technology*, 72:90 – 109, 2016. `doi:https://doi.org/10.1016/j.infsof.2015.12.001`.

[28] Jose Javier Dolado. A validation of the component-based method for software size estimation. *IEEE Transactions on Software Engineering*, 26(10):1006–1021, 2000.

[29] Jose Javier Dolado. On the problem of the software cost function. *Information and Software Technology*, 43(1):61–72, 2001.

[30] Soufiane Ezghari and Azeddine Zahi. Uncertainty management in software effort estimation using a consistent fuzzy analogy-based method. *Applied Soft Computing*, 67:540–557, 2018.

[31] Filomena Ferrucci, Carmine Gravino, Rocco Oliveto, and Federica Sarro. Using tabu search to estimate software development effort. In *International Workshop on Software Measurement*, pages 307–320. Springer, 2009.

[32] Filomena Ferrucci, Carmine Gravino, Rocco Oliveto, and Federica Sarro. Genetic programming for effort estimation: an analysis of the impact of different fitness functions. In *2nd International Symposium on Search Based Software Engineering*, pages 89–98. IEEE, 2010.

[33] Filomena Ferrucci, Carmine Gravino, Rocco Oliveto, Federica Sarro, and Emilia Mendes. Investigating tabu search for web effort estimation. In *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 350–357. IEEE, 2010.

[34] Filomena Ferrucci, Carmine Gravino, and Federica Sarro. How multi-objective genetic programming is effective for software development effort estimation? In *International Symposium on Search Based Software Engineering*, pages 274–275. Springer, 2011.

[35] Filomena Ferrucci, Mark Harman, Jian Ren, and Federica Sarro. Not going to take this anymore: multi-objective overtime planning for software engineering projects. In *Proc. of 35th International Conference on Software Engineering, ICSE'13*, pages 462–471, 2013. `doi:10.1109/ICSE.2013.6606592`.

[36] Filomena Ferrucci, Mark Harman, and Federica Sarro. Search-based software project management. In *Software Project Management in a Changing World*, pages 373–399. Springer, 2014.

[37] Carlos M Fonseca, Joshua D Knowles, Lothar Thiele, and Eckart Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. In *Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, volume 216, page 240, 2005.

[38] Tron Foss, Erik Stensrud, Barbara Kitchenham, and Ingunn Myrtveit. A simulation study of the model evaluation criterion mmre. *IEEE transactions on software engineering*, 29(11):985–995, 2003.

[39] Bronius Grigelionis. *Student's t-distribution and related stochastic processes*. Springer, 2013.

[40] Yann-Gaël Guéhéneuc and Foutse Khomh. Empirical software engineering. In *Handbook of Software Engineering*, pages 285–320. Springer, 2019.

[41] G. Guizzo, G. M. Fritsche, S. R. Vergilio, and A. T. R. Pozo. A Hyper-Heuristic for the Multi-Objective Integration and Test Order Problem. In *Proc. of GECCO'15*, 2015.

[42] Giovani Guizzo, Federica Sarro, Jens Krinke, and Silvia Regina Vergilio. Sentinel: A hyper-heuristic for the generation of mutant reduction strategies. *IEEE Transactions on Software Engineering*, 2020.

[43] Giovani Guizzo, Silvia R Vergilio, Aurora TR Pozo, and Gian Mauricio Fritsche. A multi-objective and evolutionary hyper-heuristic applied to the integration and test order problem. *Applied Soft Computing*, 56:331–344, 2017.

[44] Mark Harman, S Afshin Mansouri, and Yuanyuan Zhang. Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys (CSUR)*, 45(1):1–61, 2012.

[45] Mark Harman, Phil McMinn, Jerffeson Teixeira De Souza, and Shin Yoo. Search based software engineering: Techniques, taxonomy, tutorial. In *Empirical software engineering and verification*, pages 1–59. Springer, 2010.

[46] Geoffrey W Hill. Algorithm 396: Student's t-quantiles. *Communications of the ACM*, 13(10):619–620, 1970.

[47] Ali Idri, Fatima azzahra Amazal, and Alain Abran. Analogy-based software development effort estimation: A systematic mapping and review. *Information and Software Technology*, 58:206–230, 2015.

[48] Ali Idri, Mohamed Hosni, and Alain Abran. Systematic literature review of ensemble effort estimation. *Journal of Systems and Software*, 118:151–175, 2016.

[49] Hisao Ishibuchi, Ryo Imada, Yu Setoguchi, and Yusuke Nojima. Performance comparison of nsga-ii and nsga-iii on various many-objective test problems. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 3045–3052. IEEE, 2016.

[50] Hisao Ishibuchi, Yusuke Nojima, and Tsutomu Doi. Comparison between single-objective and multi-objective genetic algorithms: Performance comparison and performance measures. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1143–1150. IEEE, 2006.

[51] Himanshu Jain and Kalyanmoy Deb. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: handling constraints and extending to an adaptive approach. *IEEE Transactions on evolutionary computation*, 18(4):602–622, 2013.

[52] Magne Jørgensen. Comments on'a simulation tool for efficient analogy based cost estimation', by l. angelis and i. stamelos, published in empirical software engineering, 5, 35-68 (2000). *Empirical Software Engineering*, 7(4):375, 2002.

[53] Magne Jørgensen. A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70(1-2):37–60, 2004.

[54] Magne Jørgensen and Dag I. K. Sjoeberg. An effort prediction interval approach based on the empirical distribution of previous estimation accuracy. *Information and software Technology*, 45(3):123–136, 2003.

[55] Natalia Juristo and Omar S Gómez. Replication of software engineering experiments. In *Empirical software engineering and verification*, pages 60–88. Springer, 2010.

[56] Gada Kadoda, Michelle Cartwright, and Martin Shepperd. Issues on the effective use of cbr technology for software project prediction. In *International Conference on Case-Based Reasoning*, pages 276–290. Springer, 2001.

[57] Vineet Khare, Xin Yao, and Kalyanmoy Deb. Performance scaling of multi-objective evolutionary algorithms. In *International conference on evolutionary multi-criterion optimization*, pages 376–390. Springer, 2003.

[58] Barbara A Kitchenham, Lesley M Pickard, Stephen G. MacDonell, and Martin J. Shepperd. What accuracy statistics really measure. *IEE Proceedings-Software*, 148(3):81–85, 2001.

[59] Ekrem Kocaguneli, Tim Menzies, Jacky Keung, David Cok, and Ray Madachy. Active learning and effort estimation: Finding the essential content of software effort estimation data. *IEEE Transactions on software engineering*, 39(8):1040–1053, 2012.

[60] Ekrem Kocaguneli, Tim Menzies, and Jacky W Keung. On the value of ensemble effort estimation. *IEEE Transactions on Software Engineering*, 38(6):1403–1416, 2011.

[61] Ekrem Kocaguneli, Ayse Tosun, and Ayse Bener. Ai-based models for software effort estimation. In *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 323–326. IEEE, 2010.

[62] Marcel Korte and Dan Port. Confidence in software cost estimation results based on mmre and pred. In *Proceedings of the 4th international workshop on Predictor models in software engineering*, pages 63–70, 2008.

[63] John R Koza et al. *Genetic programming*. MIT press Cambridge, 1994.

[64] William B Langdon, Javier Dolado, Federica Sarro, and Mark Harman. Exact mean absolute error of baseline predictor, marp0. *Information and Software Technology*, 73:16–18, 2016.

[65] William B Langdon and Riccardo Poli. *Foundations of genetic programming*. Springer Science & Business Media, 2013.

[66] Martin Lefley and Martin J Shepperd. Using genetic programming to improve software effort estimation based on general data sets. In *Genetic and Evolutionary Computation Conference*, pages 2477–2487. Springer, 2003.

[67] M. Li, T. Chen, and X. Yao. How to evaluate solutions in pareto-based search-based software engineering? a critical review and methodological guidance. *IEEE Transactions on Software Engineering*, pages 1–1, 2020. `doi:10.1109/TSE.2020.3036108`.

[68] Chris Lokan. What should you optimize when building an estimation model? In *11th IEEE International Software Metrics Symposium (METRICS'05)*, pages 10–pp. IEEE, 2005.

[69] Katrina Maxwell and K Maxwell. *Applied statistics for software managers*. Prentice Hall PTR Englewood Cliffs, 2002.

[70] Steve McConnell. *Software estimation: demystifying the black art*. Microsoft press, 2006.

[71] Emilia Mendes, Ian Watson, Chris Triggs, Nile Mosley, and Steve Counsell. A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering*, 8(2):163–196, 2003.

[72] Solomon Mensah, Jacky Keung, Michael Franklin Bosu, and Kwabena Ebo Bennin. Duplex output software effort estimation model with self-guided interpretation. *Information and Software Technology*, 94:1–13, 2018.

[73] Nikolaos Mittas, Ioannis Mamalikidis, and Lefteris Angelis. A framework for comparing multiple cost estimation methods using an automated visualization toolkit. *Information and Software Technology*, 57:310–328, 2015.

[74] Y Miyazaki, M Terakado, K Ozaki, and H Nozaki. Robust regression for developing software estimation models. *Journal of Systems and Software*, 27(1):3–16, 1994.

[75] Kjetil Moløkken-Østvold and Magne Jørgensen. A review of surveys on software effort estimation. In *2003 ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 2003)*, pages 220–230, 2003.

[76] Floréal Morandat, Brandon Hill, Leo Osvald, and Jan Vitek. Evaluating the design of the r language. In *European Conference on Object-Oriented Programming*, pages 104–131. Springer, 2012.

[77] Antonio J Nebro, Juan J Durillo, Francisco Luna, Bernabé Dorronsoro, and Enrique Alba. Mocell: A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems*, 24(7):726–746, 2009.

[78] Antonio J Nebro, Juan J Durillo, and Matthieu Vergne. Redesigning the jmetal multi-objective optimization framework. In *Proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation*, pages 1093–1100, 2015.

[79] Geoffrey Neumann, Mark Harman, and Simon Poulding. Transformed vargha-delaney effect size. In *International Symposium on Search Based Software Engineering*, pages 318–324. Springer, 2015.

[80] Adriano LI Oliveira. Estimation of software project effort with support vector regression. *Neurocomputing*, 69(13-15):1749–1753, 2006.

[81] Dan Port and Marcel Korte. Comparative studies of the model evaluation criterions mmre and pred in software cost estimation research. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 51–60, 2008.

[82] Kata Praditwong, Mark Harman, and Xin Yao. Software module clustering as a multi-objective search problem. *IEEE Transactions on Software Engineering*, 37(2):264–282, 2010.

[83] Paul Ralph, Nauman bin Ali, Sebastian Baltes, Domenico Bianculli, Jessica Diaz, Yvonne Dittrich, Neil Ernst, Michael Felderer, Robert Feldt, Antonio Filieri, Breno Bernard Nicolau de França, Carlo Alberto Furia, Greg Gay, Nicolas Gold, Daniel Graziotin, Pinjia He, Rashina Hoda, Natalia Juristo, Barbara Kitchenham, Valentina Lenarduzzi, Jorge Martínez, Jorge Melegati, Daniel Mendez, Tim Menzies, Jefferson Molleri, Dietmar Pfahl, Romain Robbes, Daniel Russo, Nyyti Saarimäki, Federica Sarro, Davide Taibi, Janet Siegmund, Diomidis Spinellis, Miroslaw Staron, Klaas Stol, Margaret-Anne Storey, Davide Taibi, Damian Tamburri, Marco Torchiano, Christoph Treude, Burak Turhan, Xiaofeng Wang, and Sira Vegas. Empirical standards for software engineering research, 2021. `arXiv:2010.03525`.

[84] Aurora Ramirez, José Raúl Romero, and Sebastian Ventura. A survey of many-objective optimisation in search-based software engineering. *Journal of Systems and Software*, 149:382–395, 2019.

[85] F. Sarro, F. Ferrucci, M. Harman, A. Manna, and J. Ren. Adaptive multi-objective evolutionary algorithms for overtime planning in software projects. *IEEE Transactions on Software Engineering*, 43(10):898–917, 2017. `doi:10.1109/TSE.2017.2650914`.

[86] Federica Sarro. Search-based approaches for software development effort estimation. In *Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement*, pages 38–43, 2011.

[87] Federica Sarro, Rebecca Moussa, Alessio Petrozziello, and Mark Harman. Learning from mistakes: Machine learning enhanced human expert effort estimates. *IEEE Transactions on Software Engineering*, 2020.

[88] Federica Sarro and Alessio Petrozziello. Linear programming as a baseline for software effort estimation. *ACM Trans. Softw. Eng. Methodol.*, 27(3), September 2018. URL: https://doi.org/10.1145/3234940, `doi:10.1145/3234940`.

[89] Federica Sarro, Alessio Petrozziello, and Mark Harman. Multi-objective software effort estimation. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 619–630. IEEE, 2016.

[90] Abdel Salam Sayyad, Joseph Ingram, Tim Menzies, and Hany Ammar. Optimum feature selection in software product lines: Let your model and values guide your search. In *2013 1st International Workshop on Combining Modelling and Search-Based Software Engineering (CMSBSE)*, pages 22–27. IEEE, 2013.

[91] Panagiotis Sentas, Lefteris Angelis, Ioannis Stamelos, and George Bleris. Software productivity and effort prediction with ordinal regression. *Information and software technology*, 47(1):17–29, 2005.

[92] Yin Shan, Robert I McKay, Chris J Lokan, and Daryl L Essam. Software project effort estimation using genetic programming. In *IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions*, volume 2, pages 1108–1112. IEEE, 2002.

[93] Martin Shepperd. Case-based reasoning and software engineering. In *Managing Software Engineering Knowledge*, pages 181–198. Springer, 2003.

[94] Martin Shepperd, Nemitari Ajienka, and Steve Counsell. The role and value of replication in empirical software engineering results. *Information and Software Technology*, 99:120–132, 2018.

[95] Martin Shepperd, Michelle Cartwright, and Gada Kadoda. On building prediction systems for software engineers. *Empirical Software Engineering*, 5(3):175–182, 2000.

[96] Martin Shepperd and Gada Kadoda. Using simulation to evaluate prediction techniques [for software]. In *Proceedings Seventh International Software Metrics Symposium*, pages 349–359. IEEE, 2001.

[97] Martin Shepperd and Steve MacDonell. Evaluating prediction systems in software project estimation. *Information and Software Technology*, 54(8):820–827, 2012.

[98] Martin Shepperd and Chris Schofield. Estimating software project effort using analogies. *IEEE Transactions on software engineering*, 23(11):736–743, 1997.

[99] Martin Shepperd, Chris Schofield, and Barbara Kitchenham. Effort estimation using analogy. In *Proceedings of IEEE 18th International Conference on Software Engineering*, pages 170–178. IEEE, 1996.

[100] Forrest Shull, Victor Basili, Jeffrey Carver, José Carlos Maldonado, Guilherme Horta Travassos, Manoel Mendonça, and Sandra Fabbri. Replicating software engineering experiments: addressing the tacit knowledge problem. In *Proceedings international*

symposium on empirical software engineering, pages 7–16. IEEE, 2002.

[101] Forrest J Shull, Jeffrey C Carver, Sira Vegas, and Natalia Juristo. The role of replications in empirical software engineering. *Empirical software engineering*, 13(2):211–218, 2008.

[102] Liyan Song, Leandro L Minku, and Xin Yao. The potential benefit of relevance vector machine to software effort estimation. In *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, pages 52–61, 2014.

[103] Liyan Song, Leandro L Minku, and Xin Yao. Software effort interval prediction via bayesian inference and synthetic bootstrap resampling. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 28(1):1–46, 2019.

[104] Erik Stensrud, Tron Foss, Barbara Kitchenham, and Ingunn Myrtveit. A further empirical investigation of the relationship between mre and project size. *Empirical software engineering*, 8(2):139–161, 2003.

[105] Vali Tawosi, Federica Sarro, Alessio Petrozziello, and Mark Harman. CoGEE, Java source code on GitHub. URL: https://github.com/SOLAR-group/cogee.

[106] Vali Tawosi, Federica Sarro, Alessio Petrozziello, and Mark Harman. On-line appendix to the paper Multi-Objective Software Effort Estimation: A Replication Study. https://solar.cs.ucl.ac.uk/os/cogee.html and https://figshare.com/s/eeca74f09a7f2271cecf.

[107] Chris Tofallis. A better measure of relative prediction accuracy for model selection and model estimation. *Journal of the Operational Research Society*, 66(8):1352–1362, 2015.

[108] David A Van Veldhuizen and Gary B Lamont. Multiobjective evolutionary algorithm research: A history and analysis. Technical report, Citeseer, 1998.

[109] Peter A Whigham, Caitlin A Owen, and Stephen G Macdonell. A baseline model for software effort estimation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(3):1–11, 2015.

[110] Hadley Wickham. *Advanced R*. Chapman and Hall/CRC, 2014.

[111] Gerhard Wittig and Gavin Finnie. Estimating software development effort with connectionist models. *Information and Software Technology*, 39(7):469–476, 1997.

[112] Fang Hon Yun. China: Effort estimation dataset, April 2010. URL: https://doi.org/10.5281/zenodo.268446, `doi:10.5281/zenodo.268446`.

[113] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

[114] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *International conference on parallel problem solving from nature*, pages 832–842. Springer, 2004.

[115] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *International conference on parallel problem solving from nature*, pages 832–842. Springer, 2004.

[116] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001.

[117] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[118] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert Da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation*, 7(2):117–132, 2003.

Vali Tawosi is a PhD student at University College London, under the supervision of Prof. Federica Sarro and Prof. Mark Harman. His research covers Predictive Analytics for Software Engineering and Search-Based Software Engineering with a focus on software effort estimation and agile methods. Web page: http://www0.cs.ucl.ac.uk/people/V.Tawosi.html

Federica Sarro is a Professor of Software Engineering at University College London. Her research covers Predictive Analytics for Software Engineering (SE), Empirical SE and Search-Based SE. On these topics she has published over 80 papers in peer-reviewed international venues including ICSE, FSE, TSE, TOSEM, EMSE. Federica has received several international awards for her research work including the IEEE TCSE Rising Star Award and the FSE'19 ACM Distinguished Paper Award. She has been elected and invited to serve on several steering, organisation, programme committees, programme and editorial boards of well-renowned venues such as ICSE, FSE, ASE, ACM TOSEM, IEEE TSE, IEEE TEVC. Web page: http://www0.cs.ucl.ac.uk/staff/F.Sarro/

Alessio Petrozziello works as Senior Manager of Data Science at Expedia Group. He also holds an Honorary Senior Research Fellow position at UCL. He was awarded a PhD in Computational Intelligence at the University of Portsmouth (UK) and his main research interest is in Recommender Systems. He specializes in building complex Distributed Machine Learning Frameworks & Pipelines, solving business needs at scale through custom Machine Learning solutions (from prototypes to fully fledged production-ready models). Moreover, he has experience in working in the areas of Learn-to-Rank, Deep Learning and Time Series Forecasting. Web page: http://alessiopetrozziello.github.io

Mark Harman works full time at Facebook London as a Research Scientist and also holds a part-time professorship at UCL. Previously, Mark was the manager of the Facebook team that deployed Sapienz to test mobile apps, which grew out of Majicke, a start up co-founded by Mark and acquired by Facebook in 2017. Mark co-founded the field Search Based Software Engineering (SBSE), and is also known for scientific research on source code analysis, software testing, app store analysis and empirical software engineering. He received the IEEE Harlan Mills Award and the ACM Outstanding Research Award in 2019 for this work. In addition to Facebook itself, Mark's scientific work is also supported by the European Research Council (ERC), with an advanced fellowship grant, and has also been regularly and generously supported by the UK Engineering and Physical Sciences Research Council (EPSRC), with regular grants, a platform and a programme grant. Web page: http://www0.cs.ucl.ac.uk/staff/M.Harman/