Multi-objective Software Effort Estimation

Federica Sarro*, Alessio Petrozziello† and Mark Harman*
University College London, London, United Kingdom*
University of Portsmouth, Portsmouth, UK†
f.sarro@ucl.ac.uk, alessio.petrozziello@port.ac.uk, mark.harman@ucl.ac.uk

ABSTRACT

We introduce a bi-objective effort estimation algorithm that combines Confidence Interval Analysis and assessment of Mean Absolute Error. We evaluate our proposed algorithm on three different alternative formulations, baseline comparators and current state-of-the-art effort estimators applied to five real-world datasets from the PROMISE repository, involving 724 different software projects in total. The results reveal that our algorithm outperforms the baseline, state-of-the-art and all three alternative formulations, statistically significantly ($p < 0.001$) and with large effect size ($\hat{A}_{12} \geq 0.9$) over all five datasets. We also provide evidence that our algorithm creates a new state-of-the-art, which lies within currently claimed industrial human-expert-based thresholds, thereby demonstrating that our findings have actionable conclusions for practicing software engineers.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management

Keywords

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

1. INTRODUCTION

Effort estimation is a critical activity for planning and monitoring software project development in order to deliver the product on time and within budget [9, 51, 88]. The competitiveness (and occasionally the survival) of software organisations depends on their ability to accurately predict the effort required for developing software systems; both over- or under-estimates can negatively affect the outcome of software projects [59, 63, 84, 88]. Several algorithmic approaches have been proposed in literature to support software engineers in improving the accuracy of their estimations. These methods often produce a point estimate of the effort required to develop a new project.

ICSE'16 May 14-22, 2016, Austin, TX, USA

ACM ISBN 978-1-4503-3900-1/16/05.

DOI: http://dx.doi.org/10.1145/2884781.2884830

Very few previous studies have accounted for the inherent uncertainty of the estimates produced [4, 6, 76, 83, 85, 86]. Some previous work has instead investigated the over-confidence and/or under-confidence of the prediction given by expert judgement [35, 36, 37, 38, 40, 41, 67]. Existing surveys on estimation practice [37, 67] suggest that human effort estimates are over-optimistic and there is a strong over-confidence in their accuracy.

We introduce a multi-objective evolutionary approach that seeks to build a robust estimation model by simultaneously maximising the estimation accuracy and minimising the uncertainty associated with the estimation model itself. We named this approach Confidence Guided Effort Estimator (CoGEE). We use the familiar sum of absolute error, abs(real effort - estimated effort), as one objective to guide our algorithm, combining this with the (less widely-known and less widely-used) confidence interval. The confidence interval is an estimated range of values that are likely (according to the chosen interval range) to include the estimated effort.

We report the results of four sets of experiments on five publicly available datasets to compare and evaluate our approach against candidate competitors: baseline estimators [82], state-of-the-art estimators [39, 50, 90], alternative single and multi-objective formulations, and currently claimed best industrial practice based on human judgment [67]. In our evaluation we follow recent best practice to assess prediction systems [82, 90] and evolutionary approaches [3, 31].

Our new bi-objective effort estimation algorithm outperforms baseline estimators (a sanity check), state-of-the-art techniques (case-based reasoning, linear regression, regression trees) and also three alternatives that we implemented in order to assess the degree to which multi-objectivity plays a role in the performance of our algorithm. These claims have been tested using a non-parametric (Wilcoxon) test for statistical significance which reveals that the results are significant ($p < 0.001$), after applying the Bonferroni correction for multiple statistical testing (the most conservatively cautious of all corrections). Furthermore, in all cases, our bi-objective algorithm outperforms these candidate competitors with a large effect size, as measured using the Vargha-Delaney non-parametric effect size measure ($\hat{A}_{12} \geq 0.9$).

We also compare both the estimation error produced by our algorithm (and the current state-of-the-art) and the budget overruns that would accrue from using them against two claimed thresholds for industrial best estimation practice. The results are very encouraging, suggesting that CoGEE moves median expected state-of-the-art performance within at least one, and sometimes both thresholds.

The rest of the paper is organised as follows. Section 2 gives some background on Software Development Effort Estimation. Section 3 describes our proposal, CoGEE, for multi-objective effort estimation. The research questions and experimental method we used to address them are described in Section 4. The results of the empirical study are reported in Section 5. The study validity is discussed in Section 6. Section 7 reports on related work, while final remarks are presented in Section 8.

2. SOFTWARE EFFORT ESTIMATION

Software development effort estimation is the process of predicting the most realistic amount of effort (usually expressed in terms of person-hours or person-month) required to develop or maintain a software project based on information collected in the early stage of a software project. Expert estimation remains the dominant strategy when estimating software development effort in practice [37]. Research results have focused on the construction of formal software effort estimation models to support the engineers in the estimation process. The first estimation models were based on regression analysis. Since then, different model building approaches have been investigated, including approaches based on analogy-based techniques (e.g., Case-Based Reasoning) [2, 47, 49, 81], machine learning techniques (e.g., Support Vector Regression, Bayesian Network [58]), search-based approaches (e.g., Genetic Programming, Tabu Search [24]), and combinations of two or more of these models (e.g., [15, 16, 50, 52]).

Formal estimation approaches usually exploit training data about past projects to build an estimation model which is then used to predict the effort for a new project. Such a model takes as input a set of predictors (e.g., manager experience, team experience) and returns a scalar value that represents the effort estimated to develop a new software system having the characteristics captured by the predictors. This model can be described by the following equation:

$$EstimatedEffort = c_1\ op_1\ f_1\ ...\ c_n\ op_{2n-1}\ f_n\ op_{2n}\ C \quad (1)$$

where f_i represents the value of the i^{th} project feature and c_i its coefficient, C represents a constant, while op_i represents the i^{th} mathematical operator of the model.

The way in which the predictors are used is specific to the prediction approach. For example, when using a linear regression technique, the predictors are combined through a linear combination, while Case-Based Reasoning exploits the predictors to identify the most similar past projects.

Several measures have been proposed to evaluate the accuracy of a prediction model. Generally they are based on the Absolute Error (i.e, $|RealEffort - EstimatedEffort|$).

The most popular are MMRE and Pred(25) [14]. The former is the Mean of *Magnitude of Relative Error* (MRE), where MRE [14] is defined as:

$$MRE = |RealEffort - EstimatedEffort|/RealEffort \quad (2)$$

MRE is calculated for each project whose effort has to be estimated and MMRE is used to have a cumulative measure of the error. The *Prediction at level l* – Pred(l) – [14] measures the percentage of the estimates whose error is less than $l\%$ and l is usually set at 25. It is defined as follows:

$$Pred(25) = k/N \quad (3)$$

where N is the total number of projects and k is the number of observations whose MRE is less than or equal to 0.25.

These measures have been criticised [25, 45, 53, 71, 79, 87] as being biased towards underestimations and can behave very differently when comparing prediction models, thereby motivating the use of other (more standardised) measures, such as the *Mean Absolute Error* (MAE) and the *Standardized Accuracy* (SA), recently recommended [55, 82] to compare the performance of prediction models. MAE is unbiased (towards over or underestimation) and is defined as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|RE_i - EE_i| \quad (4)$$

where N is the number of projects used for evaluating the performance, and RE_i and EE_i are the actual and estimated effort, respectively, for the project i.
SA is based on MAE and it can be defined as follows:

$$SA = (1 - MAE_{P_j}/MAE_{rguess}) \cdot 100 \quad (5)$$

where MAE_{P_j} is the MAE of the approach P_j being evaluated and MAE_{rguess} is the MAE of a large number (e.g., 1000 runs) of random guesses. Thus, SA represents how much better P_j is than random guessing. A value close to zero means that the prediction model P_j is practically useless, performing little better than a mere random guess [82]. Effort estimation techniques seek to minimise MAE; whereas they seek to maximise SA.

3. BI-OBJECTIVE ESTIMATION

Our approach to software effort estimation uses Search Based Software Engineering (SBSE), an increasingly prevalent approach to software engineering [13, 26, 28, 30] in which software engineering problems are reformulated as search problems within the search space that can be explored using computational search algorithms.

In the SBSE literature there has been more than 15 years of work on the use of search-based approaches for software effort estimation (e.g., [11, 18, 19, 20, 21, 22, 56, 77]). A recent survey of work on SBSE for effort prediction in particular (and software project management in general) can be found in the work of Ferrucci et al. [24]. However, all previous search-based effort estimation approaches seek to produce point estimates. Furthermore, only two previous studies [66, 74] concerned multi-objective formulations of effort estimation. Both of these two previous multi-objective studies focused on point estimates and aimed to analyse the tradeoff among different accuracy measures for the single overall objective of producing the most accurate point estimate.

By contrast, our multi-objective approach treats the accuracy of the point estimate as one of the two objectives, and the confidence interval as the other. In the following we describe our proposed approach, using the standard [26, 27, 29, 31] three 'key ingredients' of any SBSE approach: representation, fitness function, and computational search algorithm. Since our formulation is a bi-objective formulation we also describe how we handle multi-objective search.
Representation: Feasible solutions to the problem defined in Section 2 are estimation models described by Equation 1. Such a model can be encoded as a Genetic Algorithm individual using an expression syntax tree and randomly choosing the coefficients $c_i \in R$ and $op_i \in \{+, -, *\}$, while the factors values depend on the training data and do not

change during the evolution process. It is worth noting that the equations feasible for the effort estimation problem are those providing positive value for *EstimatedEffort*. The initial population was generated by building 100 random trees of fixed depth.

Fitness: To evaluate the fitness of each chromosome we employed a multi-objective function to simultaneously minimise the estimates accuracy and the estimates distribution uncertainty.

Many different indicators can be used to evaluate the accuracy of the estimates (see Section 2). Previous work [20, 57] showed that the use of different measures can impact both the fitting and the predictive performance of the models built by GA: relative measures (e.g., MMRE, MEMRE) often affect negatively the overall model accuracy, while absolute measures (e.g., SAE) seem to not have any detrimental effect. Thus, in this paper, we choose to employ the Sum of Absolute Error (SAE) [14] as fitness function.

On the other hand, fewer measures have been suggested in the literature to assess the uncertainty of the effort estimates and no previous work has been carried out so far to investigate their effectiveness as a fitness function. Therefore, in this paper we measure the estimate uncertainty by employing the confidence interval[1] associated with the estimation model to assess the uncertainty of the mean value of the distribution of absolute errors produced by the model as follows:

$$\frac{\phi(p, df) * std(AbsoluteErrors)}{\sqrt{n}} \tag{6}$$

where n is the size of the sample, $std(AbsoluteErrors)$ is the standard deviation of the distribution of absolute errors produced by the estimation model, and $\phi(p, df)$ is the quantile function [32]. This function returns a threshold value x, below which random draws from the given cumulative distribution function (c.d.f.) would fall p percent of the time:

$$Q(p) = inf\{x \in R : p \leq F(x)\} \tag{7}$$

for a probability $0 < p < 1$. Confidence intervals are usually calculated so that this percentage is 95%; while 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.

Handling Multiple Objectives: In our case, the two objectives are measured on orthogonal scales so we use Pareto optimality, which 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 x_1 is strictly better than x_2 in at least one objective."

Using Pareto optimality we can plot the set of solutions found to be non-dominating (and therefore equally viable).

Computational Search: As a ranking method, we employed a widely used Multi-objective evolutionary algorithm, namely NSGAII [17]. We experimented with two widely used selection operators, i.e., roulette wheel selector and tournament selector [54], whereas the crossover and mutation operators are specific for our solution encoding. We used the roulette wheel selector [54] to choose the individuals for reproduction, while we employed the tournament selector [54] to determine the individuals that are included in the next generation (i.e., survivals). The former assigns a roulette slice to each chromosome according to its fitness value. In this way, even if candidate solutions with a higher fitness have more chances to be selected, there is still a chance that they may remain unselected. On the contrary, using the tournament selector only the best n solutions (usually $n \in [1, 10]$) are copied straight into the next generation.

Crossover and mutation operators were defined to preserve well-formed equations in all offspring [20]. To this end, we used a single point crossover which randomly selects the same point in each predictive model expression tree and swaps the subtrees corresponding to the selected node. Since both trees are cut at the same point, the trees resulting after the swapping have the same depth as compared to those of parent trees. We used a mutation operator that selects a node of the tree and randomly changes the associated value. Mutation can affect internal nodes (i.e., operators) or leaves (i.e., coefficients) of the tree. In particular, when mutation involves internal nodes, a new operator $op_i' \in \{\{+, -, *\} \setminus op_i\}$ is randomly generated and assigned to the node, while if the mutation involves a leaf, a new coefficient $c_i' \in R$ is assigned to the node. It is worth noting that also the mutation operator we used preserves the syntactic structure of the equation. Crossover and mutation rate were fixed to 0.5 and 0.1, respectively, since in previous work recommended crossover rates ranged from 0.45 to 0.95 and mutation rates ranged from 0.06 to 0.1 [20, 33]. The evolutionary process terminates after 250 generations.

4. EMPIRICAL STUDY DESIGN

This section presents the design of the empirical study we carried out to get an insight into the use of multi-objective effort estimation. We first present the research questions we aim to answer and then the data and techniques we experimented with, and the evaluation criteria we used to assess the results.

4.1 Research Questions

A novel effort estimation approach must outperform baseline methods. Thus, the first research question we aim to answer represents a sanity check of our proposal:

RQ1 Sanity Check: Is the proposed approach CoGEE suitable for effort estimation? To answer this question we compare our algorithm with three common baseline benchmarks used in the context of effort estimation (i.e., Mean and Median Effort and Random Guessing) and described in Section 4.4. If the investigated estimation method does not outperform the results achieved using these baselines, it cannot be transferred to industry [61].

If a proposed estimation method outperforms the benchmark, then the next natural question to ask is whether it outperforms state-of-the-art techniques currently proposed in the effort estimation literature. If not, then there would be no reason to adopt it, since it could not be guaranteed to advance the state-of-the-art. This motivates RQ2:

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

[1]Please note that an important difference between Confidence Interval (CI) and Prediction Interval (PI) is that PI refers to the uncertainty of an estimate, while CI usually refers to the uncertainty associated with the parameters of an estimation model or distribution. The confidence level of a PI refers to the expected (or subjective) probability that the real value is within the predicted interval, while the confidence level of a model assesses the uncertainty of the mean value of a distribution of effort values.

To answer RQ2 we compared our proposal (CoGEE) to three widely-used and well-studied state-of-the-art effort estimation methods, namely LR [90], CART [8], and CBR [81], as detailed in Section 4.4. We compare against three different approaches, in order to improve the scientific evidence that our proposed approach does, indeed, advance the state-of-the-art. For different datasets, one or other of these techniques may produce the best effort estimates, but among the three of them, they typically can be expected to perform better than other techniques in the literature [34, 39, 50, 62]. Since CBR is configurable, depending on the number of cases used, and this can influence performance, we also report results for three different choices.

If we find that our multi-objective algorithm can outperform the state-of-the-art, then we have scientific evidence to suggest that it should be adopted. However, this would not provide scientific evidence that it is the *multi-objective* nature of our approach that confers the improvements in estimation accuracy we observe. Therefore, in RQ3, we investigate whether there is evidence that the use of the two objectives we have chosen leads to any improvements we might have observed in answering RQ2.

There is evidence in the literature on multi-objective optimisation [5, 72] that multi-objective formulations can outperform single objective formulations, *even* when compared against the specific single objective targeted by a single objective formulation. When this happens, it provides evidence that the multiple objectives are, in a sense, 'sympathetic' to the targeted single objective; they help guide the search towards desired single objective *even better* and focusing solely on the single objective itself. 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. Hitherto, this possibility has not been investigated for software project effort estimation, thereby motivating RQ3:

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

First, we seek to establish whether the two objectives we consider together outperform each when considered individually. Therefore, we compare two variants of the effort estimation formulated as a single-objective problem in which each optimise one of the two objectives used by our approach, i.e., GA-SAE, where the goal is to minimize the Sum of Absolute Error and GA-CI, where the goal is to minimize the Confidence Interval:

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

When we optimise the Sum of Absolute Error (SAE) we are implicitly searching for a compromise between underestimates and overestimates, because the sum of absolute error is the sum of underestimates and over estimates. However, these are clearly two contrasting goals that our formulation combines to give SAE. For completeness, we therefore separate out these two components of SAE, to investigate whether they should be separately optimised (using an implementation we call NSGAII-UO) or whether it is sufficient to combine them as a single objective (SAE):

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

If our proposed approach (CoGEE) satisfies all of the evaluation criteria covered by RQs, 1, 2 and 3, then we will have strong scientific evidence that it outperforms the state-of-the-art, and also other candidate alternative formulations. However, in order to have real world impact, it will also be necessary to outperform current industrial practice. There is little reliably consistent scientific evidence concerning the actual estimate accuracy of current industrial practice. Therefore, our CoGEE technique may have to outperform current *beliefs* about industrial practice, as reported by practising software engineers, in order to promote wider industrial uptake (of our approach and of automated effort estimation more generally). This motivates our final research question, in which we evaluate the performance of our proposed estimation technique against the claims for the performance of current industry best practice in effort estimation:

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

To answer RQ4 we compare the performance of our CoGEE (and other state-of-the-art estimators) against claims made for best human-expert-based results currently achievable in industry [37, 67]. We investigate the magnitude of relative error (compared to claimed industrial best practice), and, because industrialists tend to be more concerned with under estimated results (rather than overestimated results) [59], 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.

4.2 Datasets

To carry out the empirical study we exploited five publicly available datasets included in the PROMISE repository [64], namely China, Desharnais, Finnish, Maxwell and Miyazaki.

These datasets represent an interesting sample of industrial software projects collected from a single company or several software companies. The datasets cover a diversity of application domains and projects' characteristics. In particular, they differ for: *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, such as number of developers involved in the project and their experience, technologies used, size in terms of Function Points [48], etc.); *technical characteristics* (software projects developed in different programming languages and for different application domains, ranging from telecommunications to commercial information systems); *involved companies* (the Desharnais dataset is within-company (WC), the others are cross-company (CC)); *geographical locations* (software projects coming from China, Canada, Finland). Furthermore all these datasets have been widely used in previous research work to evaluate effort estimation methods. Table 1 summarises the descriptive statistics of the features of the datasets we considered, while further details are provided in Appendix A to allow readers to assess whether the results we gathered can scale up to their own contexts.

4.3 Validation and Evaluation

To verify whether a method gives useful estimations of the actual development effort, a validation process is required. To this end, we performed a multiple-fold cross validation, partitioning the whole dataset into training sets, for model building, and test sets, for model evaluation. Indeed, when the accuracy of the model is computed using the same dataset employed to build the prediction model, the accu-

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
		Inquiry	0	952	61.60	105.42
		File	0	2955	91.23	210.27
		Interface	0	1572	24.23	85.04
		Effort	26	54620	3921	6481
Desharnais	WC	TeamExp	0	4	2.30	1.33
(77 projects)		ManagerExp	0	4	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	23490	4903.95	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
Miyazaki	CC	SCRN	0	281	33.69	47.24
(48 projects)		FORM	0	91	22.38	20.55
		FILE	2	370	34.81	53.56
		Effort	896	253760	13996	36601.56
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	3.023	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	8223.2	10500

Table 1: Descriptive statistics of the dataset.

racy evaluation is considered optimistic [9]. To apply the multiple-fold cross validation, we partitioned a dataset in three test sets (the observations were sampled uniformly at random, without replacement), and then for each test set we considered the remaining observations as training set. This procedure was applied to each dataset, thus obtaining: for China a test set of 167 observations and two of 166; for Desharnais a test set of 25 observations and two of 26; for Finnish a test set of 12 observations and two of 13; for Maxwell a test set of 20 observations and two of 21; for Miyazaki three test sets of 16 observations.

Concerning the evaluation of the estimates obtained with the analysed estimation methods, we used the Mean Absolute Error and the Standardized Accuracy (see Section 2). To establish if the estimations of one method were significantly better than the estimations provided by another method, we tested the statistical significance of the absolute errors achieved with different estimation methods [45]. To check for statistical significance we used the Wilcoxon Signed Rank Test [12], since the Shapiro test [73] showed that many of our samples came from non-normally distributed populations, making the T-test unsuitable. The Wilcoxon test is a safe test to apply (even for normally distributed data), since it raises the bar for significance, by making no assumptions about underlying data distributions. In particular, we tested the following Null Hypothesis: "The absolute errors provided by the prediction model P_i are significantly less that those provided by the prediction model P_j.", and 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 it has been previously noted in advice on statistical testing of randomised algorithms [3], it is inadequate to merely show statistical significance alone; we also need to know whether the effect size is worthy of interest. To assess whether the effect size is worthy of interest we employed a non-parametric effect size measure, namely the Vargha and Delaney's A_{12} statistic [3], since not all samples were normally distributed. Indeed, as suggested in recent best practice [3, 82], it is better, in cases such as ours, to use a standardised measure rather than a pooled measure such as the Cohen's d effect size. Given a performance measure M, the A_{12} statistic measures the probability that running algorithm A yields better M-values than running another algorithm B, based on the following formula $\hat{A}_{12} = (R_1/m - (m+1)/2)/n$, 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. If the two algorithms are equivalent, then $\hat{A}_{12} = 0.5$. Given the first algorithm performing better than the second, $\hat{A}_{12}$ is considered small for $0.6 \leq \hat{A}_{12} < 0.7$, medium for $0.7 < \hat{A}_{12} < 0.8$, and large for $\hat{A}_{12} \geq 0.8$, although these thresholds are somewhat arbitrary. In this case we are always interested in *any* improvement in predictive performance, so no transformation of the $\hat{A}_{12}$ metric is needed [69].

To assess the performance of the multi-objective optimisation algorithms we carried out a quantitative assessment by employing three solution set quality indicators, namely Contributions (I_C), Hypervolume (I_{HV}), and Generational Distance (I_{GD}) [23]. To compute these indicators, we normalised the fitness values to avoid unwanted scaling effects, and we used the set of non-dominated solutions found by the union of all the approaches compared as Reference Set (RS)[46].

The I_C quality indicator is the simplest measure. It measures the proportion of solutions given by an algorithm, A, that lie on the reference front (i.e., RS) [23]. The higher this proportion, the more A contributes to the best solutions found by the approaches compared, and so the better is the quality of its solutions. I_C is a simple and intuitive measure, but it is affected by the number of solutions produced, unfavourably penalising algorithms that might produce 'few but excellent' solutions. This is why we also consider two other measures of solution quality, I_{HV} and I_{GD}.

The I_{HV} quality indicator [91] calculates the volume (in the objective space) covered by members of a non-dominated set of solutions from an algorithm of interest. The larger this volume, the better the algorithm, because the more it captures of the non-dominated solution space. Zitzler demonstrates [92] that this hypervolume measure is also strictly 'Pareto compliant'. That is, the hypervolume of A is higher than B if the Pareto set of A dominates that of B. By using a volume rather than a count, this measure is also less susceptible to bias when the numbers of points on the compared fronts are very different.

The I_{GD} quality indicator [89] computes the average distance between the set of solutions, S, from the algorithm measured and the reference set, RS. The distance between S and RS in an n-objective space is computed as the average n-dimensional Euclidean distance between each point in S and its nearest neighbouring point in RS. We can think of I_{GD} as the distance between the front S and the reference front RS in the n-dimensional problem objective space.

Due to the stochastic nature of evolutionary algorithms, best practice requires the use of careful deployment of inferential statistical testing to assess the differences in the performance of the algorithms used [3, 31]. We therefore performed 30 independent runs per algorithm, per fitness function measure, per project to allow for such statistical testing, correcting for multiple statistical tests.

4.4 Benchmarks

Random Guessing. Random guessing is a naïve benchmark suggested to assess the usefulness of a prediction system [82]. It randomly assigns the y value of another case to the target case. More formally, it is defined as: predict a y for the target case t by randomly sampling (with equal probability) over all the remaining $n-1$ cases and take $y = r$ where r is drawn randomly from $1...n^r = t$ [82]. Any prediction system should outperform random guessing since an inability to predict better than random implies that the prediction system is not using any target case information.

Mean (Median) Effort. These are two baseline benchmarks commonly used for effort estimation techniques. Specifically, the mean (median) of the past project efforts is used as predicted effort for a new project.

Linear Regression. We used the Automatically Transformed Linear Model (ATLM) recently proposed as a suitable approach for comparison against novel software effort estimation methods [90]. Despite its simplicity ATLM performs well over a range of different project types and requires no parameter tuning; it is also deterministic, meaning that results obtained are amenable to replication [90].

Case-Based Reasoning. CBR is a branch of artificial intelligence that has been successfully used in Software Engineering for prediction and reuse type applications [78]. Given a new software project (i.e., target project) — characterised by its set of features — the past projects relevant to solve it are retrieved from a case base of past projects. These relevant cases are identified by using a similarity function that measures the distance between the target case and the other cases based on the values for the n features of these projects. The effort values of the k most similar projects (i.e., analogies) are then used as final prediction for the new project. The choice of k is left to the user and has been a matter of some debate [42]. We used ANGEL [80] to obtain CBR predictions. It is a tool introduced by Shepperd and Schofield to estimate the development effort of a software project. It supports the Euclidean distance measure between vectors and we used this metric to compute project similarity, while the final estimation was computed as the mean effort of the k nearest analogies. We report results of each of the choices of k, between $k = 1$ and $k = 3$ analogies.

Classification and Regression Trees. CART are machine learning methods to build prediction models by recursively partitioning the data and fitting a simple prediction model within each partition [8]. The partitioning can be represented graphically with a decision tree. Decision trees where the dependent variable takes a finite set of values are called classification trees, while decision trees where the dependent variable takes continuous values are called regression trees. In our work, regression trees were generated using the R package `rpart`[2].

Genetic Algorithms. We considered two variants of the effort estimation formulated as a single-objective problem and one formulated as a multi-objective problem. These variants differ only in the objective function: (1) GA-SAE, where the goal is to minimize the Sum of Absolute Errors; (2) GA-IC, where the goal is to minimize the Confidence Interval associated to the mean of the absolute errors; (3) NSGAII-UO, where the goal is to simultaneously minimise over- and under-estimates. The single-objective GAs were implemented by using the R package `GA`[3]. The multi-objective algorithms, NSGAII-UO and CoGEE, were implemented using the R package `nsga2R`[4]. All these algorithms use the same solution encoding and setting (see Section 3).

[2]https://cran.r-project.org/web/packages/rpart/index.html

China	SA	Desharnais	SA	Finnish	SA	Maxwell	SA	Miyazaki	SA
CoGEE	0.48	CoGEE	0.47	CART	0.52	CoGEE	0.56	CoGEE	0.90
GA-SAE	0.48	LR	0.46	CoGEE	0.45	GA-SAE	0.56	LR	0.76
GA-CI	0.45	GA-SAE	0.45	GA-CI	0.45	CART	0.51	GA-CI	0.66
CART	0.40	CART	0.38	LR	0.42	CBR3	0.51	GA-SAE	0.66
CBR3	0.40	CBR3	0.34	CBR3	0.41	GA-CI	0.48	NSGAII-UO	0.60
Median	0.38	Median	0.33	GA-SAE	0.41	CBR2	0.47	CBR2	0.56
CBR2	0.35	CBR2	0.32	CBR2	0.38	NSGAII-UO	0.41	CBR3	0.56
CBR1	0.29	CBR1	0.27	CBR1	0.31	LR	0.38	CBR1	0.55
Mean	0.25	Mean	0.26	NSGAII-UO	0.19	Median	0.33	Median	0.49
LR	0.23	GA-SAE	0.09	Mean	0.17	Mean	0.27	CART	0.46
NSGAII-UO	-1.73	NSGAII-UO	0.08	Median	0.14	CBR1	0.26	Mean	0.30

Table 2: RQs1-2: Standard Accuracy (SA) values achieved by our approach CoGEE, the baseline (Mean and Median Effort) and state-of-the-art (CBR1-2, LR, and CART) techniques over the five datasets. For completeness, SA results are also included for the other three alternative evolutionary algorithms considered later (in answer to RQ3): GA-CI, GA-SAE, and NSGAII-UO.

5. RESULTS

This section presents our results in answer to RQs1-4.

5.1 RQ1. Sanity Check

The analysis of SA (see Table 2) suggests that the estimations obtained with CoGEE are better than those achieved by using Mean, Median, and Random estimates.

Table 3 shows the results of the Wilcoxon test (together with the corresponding $\hat{A}_{12}$ effect size) to compare the statistical significance and effect size of the improvements over the baseline due to CoGEE. The first row of the table, for each dataset, presents the results that compare our proposed approach, CoGEE, with the accuracy provided by Mean, Median, and Random baseline estimates. For completeness, Table 3 also compares the performance of the other evolutionary approaches we investigate subsequently in RQ3. Table 3 reveals that the improvements over the baseline we observed in Table 2 for our proposed approach are significant ($p < 0.001$) and with large effect size ($\hat{A}_{12} \geq 0.97$). The results remain significant after correcting for multiple statistical testing. This inferential statistical analysis confirms that our approach significantly outperforms the baseline, thereby passing the sanity check set by RQ1.

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

The analysis of the SA measure (see Table 2) reveals that our proposed algorithm, CoGEE, not only outperforms the baseline, but also the different state-of-the-art techniques against which we compare it. Indeed, the SA values provided by CoGEE are always higher than those provided by CBR and LR on all the datasets we considered and higher than those provided by CART on 4 out 5 datasets. These

[3]https://cran.r-project.org/web/packages/GA/index.html
[4]https://cran.r-project.org/package=nsga2R

Dataset	Technique	Mean	Median	Random
China	CoGEE	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	NSGAII-UO	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
	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)
Desharnais	CoGEE	<0.001 (1.00)	<0.001 (0.97)	<0.001 (1.00)
	NSGAII-UO	1.00 (0.00)	1.00 (0.00)	<0.001 (0.73)
	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)
Finnish	CoGEE	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	NSGAII-UO	0.32 (0.53)	0.002 (0.70)	<0.001 (0.93)
	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)
Maxwell	CoGEE	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	NSGAII-UO	<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)
Miazaky	CoGEE	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
	NSGAII-UO	<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)

Table 3: RQ1. Results of the Wilcoxon test ($\hat{A}_{12}$ effect size in brackets) performed on the Mean of Absolute Errors provided by our algorithm, CoGEE, compared to baseline comparators: Mean, Median, and Random estimates. For completeness, sanity check comparators are also included for the other three alternative algorithms considered later (in answer to RQ3): GA-CI, GA-SAE, and NSGAII-UO.

CoGEE vs.	CBR1	CBR2	CBR3	LR	CART
China	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
Desharnais	<0.001 (1.00)	<0.001 (0.97)	<0.001 (0.97)	0.98 (0.13)	<0.001 (0.97)
Finnish	<0.001 (0.93)	<0.001 (0.93)	<0.001 (0.93)	<0.001 (0.93)	1.00 (0.03)
Maxwell	<0.001 (0.97)	<0.001 (0.97)	<0.001 (0.97)	<0.001 (0.97)	<0.001 (0.97)
Miyazaki	<0.001 (0.97)	<0.001 (0.97)	<0.001 (0.97)	<0.001 (0.97)	<0.001 (0.97)

Table 4: RQ2. Results of the Wilcoxon test (with $\hat{A}_{12}$ effect sizes in brackets) which compare the Mean and Median of the Absolute Errors for our algorithm, CoGEE, to those for the state-of-the-art techniques, CBR1–3, LR, and CART.

observations are confirmed by the inferential statistical analysis, the results of which are presented in Table 4; the improvement of our algorithm over the three state-of-the-art techniques is significant ($p < 0.001$) and the effect size is large ($\hat{A}_{12} \geq 0.9$) in 23 out 25 cases. The results remain significant after correcting for multiple statistical testing. We also verified that CoGEE remains the best algorithm when we use the Median of Absolute Error (usually less sensitive to extreme values than the mean) as evaluation criterion[5].

5.3 RQ3. Does Multi-objectivity Help?

Table 5 shows the quality indicators of the Pareto fronts obtained from the 30 runs by using the proposed bi-objective algorithm (i.e., CoGEE) and the single objective algorithms GA-CI and GA-SAE. We can observe that for the three data sets China, Finnish and Miyazaki our proposed approach produces comfortably better results according to any quality indicator (such as hypervolume, or contribution to or distance from the reference Pareto front typically used in such experiments [23, 68, 70]). For the remaining two datasets, Desharnais and Maxwell, the two single objective competitors occasionally provide solutions on or close to the Pareto front produced by our algorithm.

The results of the Wilcoxon test (see Table 6) executed to

[5]All our results, including these, are available at http://www0.cs.ucl.ac.uk/staff/F.Sarro/projects/CoGEE/

Dataset	Technique	I_{GD}	I_{HV}	I_C
China	CoGEE	0.00	0.31	0.50
	GA-SAE	0.06	0.45	0.16
	GA-CI	0.00	0.23	0.33
Desharnais	CoGEE	0.00	0.94	0.34
	GA-SAE	0.00	0.95	0.58
	GA-CI	0.34	0.53	0.08
Finnish	CoGEE	0.11	0.36	0.33
	GA-SAE	0.00	0.13	0.33
	GA-CI	0.41	0.17	0.33
Maxwell	CoGEE	0.01	0.58	0.71
	GA-SAE	0.07	0.42	0.19
	GA-CI	0.16	0.34	0.10
Miyazaki	CoGEE	0.00	1.00	1.00
	GA-SAE	0.11	0.27	0.00
	GA-CI	0.10	0.3	0.00
China	CoGEE	0.00	0.9	0.93
	NSGAII-UO	0.07	0.71	0.07
Desharnais	CoGEE	0.03	0.67	0.39
	NSGAII-UO	0.00	0.75	0.61
Finnish	CoGEE	0.00	0.90	0.67
	NSGAII-UO	0.06	0.63	0.33
Maxwell	CoGEE	0.03	0.72	0.53
	NSGAII-UO	0.09	0.69	0.47
Miyazaki	CoGEE	0.00	0.99	0.96
	NSGAII-UO	0.08	0.34	0.04

Table 5: RQ3. Mean of the quality indicators (I_{GD}, I_{HV}, I_C) computed on the Pareto fronts of the considered evolutionary approaches over 30 runs.

compare the quality indicators of the Pareto fronts obtained by CoGEE are significantly better than the ones provided by GA-CI and GA-SAE in 23 out 30 of the comparisons (and never worse in 6 cases) often with medium or large effect size. These observations remain after Bonferroni correction for multiple statistical tests. Thus, we conclude that our proposed algorithm, CoGEE, outperforms the single objective competitors, GA-SAE and GA-CI (RQ3.1).

To answer RQ3.2 we compared the Pareto fronts obtained by the proposed algorithm, CoGEE, and a multi-objective algorithm that minimises under and over estimates (NSGAII-UO). From the Pareto fronts we observe that CoGEE produces better results than NSAGAII-UO (Table 5). In fact, the competitor, NSGAII-UO, never produces a solution that dominates any solution on the Pareto front produced by CoGEE. The Wilcoxon test (see Table 6) executed to compare the quality indicators of the Pareto fronts obtained by CoGEE and NSGAII-UO confirms that CoGEE significantly outperforms NSGAII-UO in 9 out 15 cases (and never worse in the other 6), often with medium or large effect size. These observations remain after Bonferroni correction. As a final check, we confirmed that CoGEE remains the best performing algorithm when we use under and over estimation as the evaluation criterion (though space does not permit us to include the corresponding Pareto fronts[5]).

5.4 RQ4. Comparison to Industrial Practices

In Figure 1(a) we show box plots for the Magnitude of Relative Error (MRE) obtained in the prediction for each project in a given dataset by using our proposed multi-objective approach, CoGEE, and the three state-of-the-art techniques (i.e., CBR, LR, and CART). On the figure, we plot two dotted lines that denote desirable thresholds within which we would like to see these errors lie. These two thresholds, set at 1.3 and 1.89, denote predictions of project effort which lie within 30% and 40% of the true value. The reason for choosing these two thresholds derives from evidence that industrial practice based on human judgment hopes/claims

Dataset	Technique	I_{GD}	I_{HV}	I_C
China	CoGEE vs. GA-SAE	0.440 (0.49)	0.010 (0.60)	0.010 (0.42)
	CoGEE vs. GA-CI	0.830 (0.71)	0.010 (0.66)	0.010 (0.71)
	CoGEE vs. NSGAII-UO	<0.001 (0.26)	<0.001 (0.73)	<0.001 (0.94)
Desharnais	CoGEE vs. GA-SAE	0.010 (0.39)	<0.001 (0.33)	0.997 (0.39)
	CoGEE vs. GA-CI	0.005 (0.39)	<0.001 (0.88)	0.040 (0.66)
	CoGEE vs. NSGAII-UO	0.680 (0.56)	0.180 (0.56)	0.780 (0.44)
Finnish	CoGEE vs. GA-SAE	<0.001 (0.32)	0.130 (0.55)	0.640 (0.69)
	CoGEE vs. GA-CI	0.430 (0.51)	<0.001 (0.65)	0.920 (0.56)
	CoGEE vs. NSGAII-UO	<0.001 (0.70)	<0.001 (0.82)	<0.001 (0.71)
Maxwell	CoGEE vs. GA-SAE	<0.001 (0.08)	<0.001 (0.63)	<0.001 (0.98)
	CoGEE vs. GA-CI	<0.001 (0.25)	<0.001 (0.70)	<0.001 (0.92)
	CoGEE vs. NSGAII-UO	0.094 (0.03)	0.700 (0.72)	0.470 (0.53)
Miazaky	CoGEE vs. GA-SAE	<0.001 (0.25)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE vs. GA-CI	<0.001 (0.25)	<0.001 (1.00)	<0.001 (1.00)
	CoGEE vs. NSGAII-UO	<0.001 (0.26)	<0.001 (0.95)	<0.001 (0.95)

Table 6: RQ3. 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 our algorithm, CoGEE, to the ones obtained by the other evolutionary approaches over 30 runs.

to produce predictions within these tolerances. The evidence for these thresholds comes from a survey of current industry practices by Molkken and Jørgensen [67].

As can be seen from Figure 1(a) the magnitude of relative error lies comfortably within both thresholds. Indeed, as the box plots show, in all but one case, the entire distribution of estimation errors for our proposed approach, CoGEE, lies within both thresholds. The same cannot be said for Case-Based Reasoning, one of the state-of-the-art techniques. Although these box plots show the Median of Magnitude Relative Error (MdMRE) this value should not be used to compare techniques against one another as, like MMRE, it can be misleading [25, 45, 53, 71, 79, 87]. We present the box plots simply to depict the distribution of relative errors for each technique and their relationship to these two industry thresholds.

There is also evidence from industry [59] that managers are far more concerned about underestimated project effort (and thereby underestimated project duration), than they are about overestimates. While an overestimate may give rise to missed opportunities, an underestimate, and consequent project overrun, can have far more pernicious consequences. A project manager may therefore be interested to see the distribution of the magnitude of underestimated predictions for each technique. The industrial thresholds we used of 1.3 and 1.89 are, in fact, derived from the current industrial claims concerning project overrun. As such, they better indicate the threshold within which the set of all underestimates must lie (in order to be competitive and actionable for industrial uptake), than they indicate a threshold for the magnitude of relative error.

Perhaps the risk aversion and reticence to risk overrun might be a contributory factor to the current lack of take-up of effort estimation within the industry. Therefore, in order to address the managers' natural disinclination for underestimates and consequent budget overrun, we report box plots for the distribution of overrun project budgets that would be expected from each effort prediction approach in Figure 1(b).

As can be seen from the results, the median expected budget overrun for our approach lies within the claimed best results obtained from industrial practice, for all data sets except the China (where it is very close to the upper bound). In two of the five datasets the entire distribution of overrun values expected from our estimation algorithm lie within the upper bound, while in the other two, the vast majority of the distribution of overruns lies within this bound.

By contrast, the current state-of-the-art techniques yield median expected overrun values that lie outside the currently-claimed industrial upper abound.

We therefore find evidence to support the claim that our proposed estimation algorithm, CoGEE, moves the state-of-the-art that can be expected from automated estimators within the bounds of current claims for industrial best practice. This may prove to be an important finding, because it provides evidence that our new multi-objective approach can advance the claimed state-of-best-practice as well as the known scientific state-of-the-art.

6. THREATS TO VALIDITY

Several factors can bias the validity of empirical studies. In this section we discuss the validity of our empirical study based on three types of threats, namely construct, conclusion, and external validity.

To satisfy construct validity a study has "to establish correct operational measures for the concepts being studied" [44]. This means that the study should represent to what extent the predictor and response variables precisely measure the concepts they claim to measure [60]. Thus, the choice of the features and how to collect them represents a crucial aspect. We tried to mitigate such a threat by using real-world data coming from five publicly available datasets [64] widely used to empirically evaluate effort estimation methods.

With regards to the conclusion validity, we carefully applied the statistical tests, verifying all the required assumptions and correcting for multiple statistical testing. We also used datasets of different sizes to mitigate the threats related to the number of observations in each dataset. To reduce conclusion instability [65], we followed recent best practice to assess prediction systems [82, 90] and evolutionary approaches [3, 31, 69].

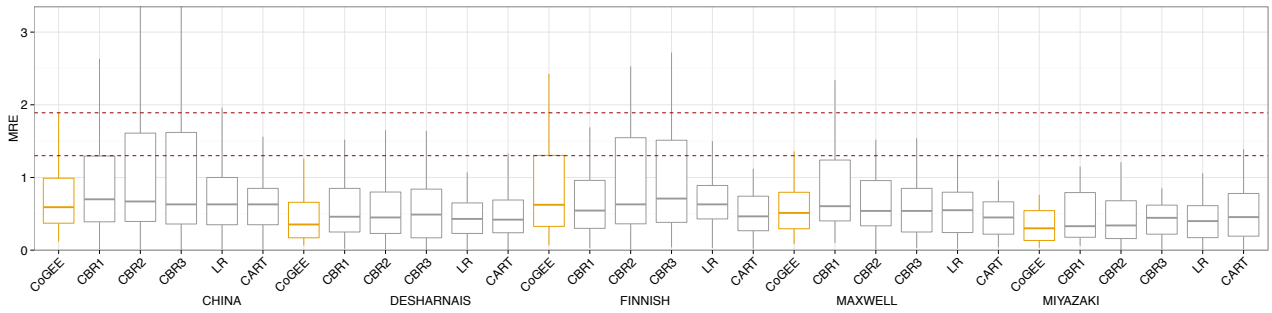
To mitigate threats to external validity we used datasets containing projects related to different contexts that might be characterised by some specific project and human factors, such as development process, developer experience, tools and technologies used, time and budget constraints [10]. Despite we used a set of subjects that has such a degree of diversity, we cannot claim that our results generalise beyond the subjects studied. Moreover, the industrial prediction thresholds used in our study come from a survey of industry practices carried out in 2003 [67], thus they may not generalise to other periods.

7. RELATED WORK

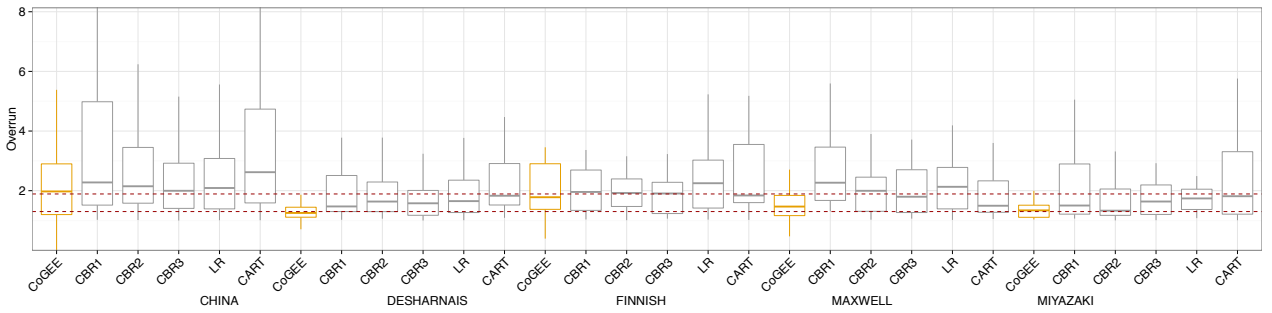
A comprehensive review of work exploiting evolutionary approaches for effort estimation can be found elsewhere [24]. In this section we summarise the main work investigating robust effort estimates with confidence intervals by highlighting the difference with the approach we proposed herein.

These studies can be classified into two broad categories: (i) those that produce confidence intervals for point estimates during the estimation process and (ii) those that produce probabilities of predefined intervals before the estimation process. Our approach, CoGEE, falls in the first category, since it builds estimation models that optimise both the accuracy of the point estimates and the confidence intervals during its evolutionary process.

Angelis and Stamelos [1] report the first empirical investigation of estimation models based on prediction intervals



(a) Magnitude of Relative Error



(b) Overrun

Figure 1: RQ4. Comparison with claimed optimal industrial practice when the magnitude of relative error (a) and overrun (b) of each project in a given dataset are considered. These results provide evidence that our multi-objective approach can move the current state-of-the-art for automated effort estimation within current claims for human-expert-based industrial best practice.

in the context of software development effort estimation. They compared the effort prediction intervals derived from a bootstrap-based model with the ones obtained by using regression based effort estimation models. However, this study displays a confusion of terms, and a critique was consequently made by Jørgensen [35] to clarify the ambiguity. The same authors also investigated statistical simulation techniques for calculating confidence intervals for project portfolios [85]. In a subsequent study [40], Jørgensen described the uncertainty of the estimate through an effort Prediction Interval (PI) and introduced an approach that is based on the assumption that the estimation accuracy of earlier software projects predicts the effort PI of new projects. The approach has been evaluated with two empirical studies to provide insight into when to use the proposed approach, regression-based approaches, or software professionals' judgment. Braga et al. [7] introduced a method based on machine learning which gives the estimation of the effort together with a confidence interval for it. They used robust confidence intervals, which do not depend on the form of probability distribution of the errors in the training set. They evaluated the proposed approach on two datasets. The results showed that the proposed method was able to build robust confidence intervals.

The first study that falls in the second category used multinomial logistic regression for modeling productivity intervals [75]. In this study Sentas et al. also investigated predefined intervals of productivity in a Bayesian belief network to support expert opinion. Subsequently, the same authors [76] investigated ordinal regression to model the probabilities of both effort and productivity intervals. Bibi et al. [6] also provided an empirical comparison between models producing point estimates and models producing predefined interval estimates. Bakir et al. [4] proposed a new approach that converts effort estimation into a classification problem to classify new software projects in one of the effort classes, each of which corresponds to an effort interval. Differently from the previous studies, the effort intervals are not predefined manually but determined by clustering analysis. Moreover, Bakir et al. used classification algorithms instead of regression-based methods. The approach, evaluated on 7 public datasets, provided point estimations comparable to those in literature, but estimation hit around 90–100%, which is higher than those obtained in previous studies.

8. CONCLUSION

This paper has introduced and evaluated a bi-objective software project effort estimation algorithm. The primary novelty of the algorithm lies in its incorporation of confidence intervals to guide a multi-objective evolutionary algorithm. Our results indicate that the new algorithm outperforms the state of the art, moving it to within current claimed thresholds for industrial human-expert-based best practice in effort estimation. Our results also provide evidence that it is the multi-objective nature of our approach that conveys this significantly improved performance. As well as the inherently attractive performance improvements, we believe developers and their managers may also find the provision of confidence intervals on effort estimations useful, since they bound the uncertainty of the estimation.

Acknowledgement

The research is funded by the Dynamic Adaptive Automated Software Engineering Programme Grant (EP/J017515) and supported by two Microsoft Azure Research Grants (Sarro 2014, Petrozziello 2015).

Appendix A: Datasets

In this appendix we provide details on the datasets used in our study (descriptive statistics are shown in Table 1).

The *China* dataset includes data of 499 projects. We employed, as independent variables, the elements used to calculate Function Points (i.e., `Input`, `Output`, `Inquiry`, `File`, `Interface`) and `Effort` as dependent variable.

Desharnais is an industrial dataset comprising 81 software projects derived from a Canadian software company. We considered the total effort as dependent variable, but not the length of the code. We also excluded from our analysis the categorical variables (i.e., `Language` and `YearEnd`) and four projects that have missing values, as done in previous works (e.g., [43, 81]). Therefore, we 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 contains data from 38 projects developed by 9 different Finnish companies. Each project is described by a dependent variable, the Effort expressed in person-hours, and five independent variables. Among them we decided to not consider the PROD variable since it represents the productivity expressed in terms of Effort and size. The independent variables we employed are: `HW` (i.e., the type of harware), `FP` (i.e., Function Points), `AR` and `CO`.

The *Maxwell* dataset contains 62 projects developed for one of the biggest commercial banks in Finland. We employed 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 with the Desharnais dataset, we did not use categorical variables.

The *Miyazaki* dataset is composed by 48 projects developed by 20 different software companies of the Fujitsu Large Systems Users Group. 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.

9. REFERENCES

[1] L. Angelis and I. Stamelos. A simulation tool for efficient analogy based cost estimation. *EMSE*, 5(1):35–68, 2000.

[2] L. Angelis, I. Stamelos, and M. Morisio. Building A software cost estimation model based on categorical data. In *Proc. of METRICS'01*, pages 4–15, 2001.

[3] A. Arcuri and L. C. Briand. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *STVR*, 24(3):219–250, 2014.

[4] A. Bakir, B. Turhan, and A. Bener. A comparative study for estimating software development effort intervals. *SQJ*, 19(3):537–552, 2011.

[5] M. Barros. An analysis of the effects of composite objectives in multiobjective software module clustering. In *Proc. of GECCO '12*, pages 1205–1212, 2012.

[6] S. Bibi, I. Stamelos, and E. Angelis. Software Cost Prediction with Predefined Interval Estimates. In *Proc. of SMEF'04*, pages 237–246, 2004.

[7] P. Braga, A. Oliveira, and S. Meira. Software effort estimation using machine learning techniques with robust confidence intervals. In *Proc. of HIS'07*, pages 352–357, 2007.

[8] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.

[9] L. C. Briand and I. Wieczorek. Software resource estimation. *Encyclopedia of Software Engineering*, pages 1160–1196, 2002.

[10] L. C. Briand and J. Wüst. Modeling development effort in object-oriented systems using design properties. *IEEE TSE*, 27(11):963–986, 2001.

[11] C. J. Burgess and M. Lefley. Can genetic programming improve software effort estimation? a comparative evaluation. *IST*, 43(14):863–873, 2001.

[12] J. Cohen. *Statistical power analysis for the behavioral sciences*. Lawrence Earlbaum Associates, 2nd edition, 1988.

[13] T. E. Colanzi, S. R. Vergilio, W. K. G. Assuncao, and A. Pozo. Search based software engineering: Review and analysis of the field in Brazil. *JSS*, 86(4):970–984, 2013.

[14] D. Conte, H. Dunsmore, and V. Shen. *Software engineering metrics and models*. Benjamin/Cummings Publishing Company, Inc., 1986.

[15] A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, F. Sarro, and E. Mendes. How effective is tabu search to configure support vector regression for effort estimation? In *Proc. of PROMISE'10*, pages 4:1–4:10, 2010.

[16] A. Corazza, S. D. Martino, F. Ferrucci, C. Gravino, F. Sarro, and E. Mendes. Using tabu search to configure support vector regression for effort estimation. *EMSE*, 18(3):506–546, 2013.

[17] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE TEC*, 6:182–197, 2002.

[18] J. J. Dolado. A validation of the component-based method for software size estimation. *IEEE TSE*, 26(10):1006–1021, 2000.

[19] F. Ferrucci, C. Gravino, R. Oliveto, and F. Sarro. Using tabu search to estimate software development effort. In *Proc. of MENSURA'09*, pages 307–320.

LNCS 5891, Springer, 2009.

[20] F. Ferrucci, C. Gravino, R. Oliveto, and F. Sarro. Genetic programming for effort estimation: An analysis of the impact of different fitness functions. In *Proc. of SSBSE'10*, pages 89–98, 2010.

[21] F. Ferrucci, C. Gravino, R. Oliveto, F. Sarro, and E. Mendes. Investigating tabu search for web effort estimation. In *Proc. of EUROMICRO-SEAA'10*, pages 350–357, 2010.

[22] F. Ferrucci, C. Gravino, and F. Sarro. How multi-objective genetic programming is effective for software development effort estimation? In *Proc. of SSBSE'11*, pages 274–275, 2011.

[23] F. Ferrucci, M. Harman, J. Ren, and F. Sarro. Not going to take this anymore: Multi-objective overtime planning for software engineering projects. In *Proc. of ICSE'13*, 2013.

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

[25] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion MMRE. *IEEE TSE*, 29(11):985–995, 2003.

[26] F. G. Freitas and J. T. Souza. Ten years of search based software engineering: A bibliometric analysis. In *Proc. of SSBSE'11*, pages 18–32, 2011.

[27] M. Harman. The current state and future of search based software engineering. In *Proc. of FOSE'07*, pages 342–357, 2007.

[28] M. Harman, Y. Jia, and Y. Zhang. Achievements, open problems and challenges for search based software testing (keynote). In *Proc. of ICST'14*, 2014.

[29] M. Harman and B. F. Jones. Search based software engineering. *IST*, 43(14):833–839, 2001.

[30] M. Harman, A. Mansouri, and Y. Zhang. Search based software engineering: Trends, techniques and applications. *ACM Computing Surveys*, 45(1):11:1–11:61, 2012.

[31] M. Harman, P. McMinn, J. Teixeira de Souza, and S. Yoo. Search based software engineering: Techniques, taxonomy, tutorial. In *LASER*, pages 1–59, 2010.

[32] G. W. Hill. Algorithm 396: Student's t-quantiles. *Commun. ACM*, 13(10):619–620, 1970.

[33] S.-J. Huang and N.-H. Chiu. Optimization of analogy weights by genetic algorithm for software effort estimation. *JSS*, 48(11):1034–1045, 2006.

[34] R. Jeffery, M. Ruhe, and I. Wieczorek. A comparative study of cost modelling techniques using public domain multi-organisational and company-specific data. In *Proc. of ESCOM'2000*, 2000.

[35] M. Jørgensen. Comments on 'A simulation tool for efficient analogy based cost estimation'. *EMSE*, 7(4):375–376, 2002.

[36] M. Jørgensen. The ignorance of confidence levels in minimum-maximum software development effort interval. *LNSE*, 2(4):327–330, 2004.

[37] M. Jørgensen. A review of studies on expert estimation of software development effort. *JSS*, 70(1-2):37–60, 2004.

[38] M. Jørgensen and K. Moløkken. Combination of software development effort prediction intervals: Why, when and how? In *Proc. of SEKE'02*, pages 425–428, 2002.

[39] M. Jørgensen and M. Shepperd. A systematic review of software development cost estimation studies. *IEEE TSE*, 33(1):33–53, 2007.

[40] M. Jørgensen and D. Sjöberg. An effort prediction interval approach based on the empirical distribution of previous estimation accuracy. *IST*, 45(3):123 – 136, 2003.

[41] M. Jørgensen, K. H. Teigen, and K. MoløKken. Better sure than safe? over-confidence in judgement based software development effort prediction intervals. *JSS*, 70(1-2):79–93, 2004.

[42] G. Kadoda, M. Cartwright, and M. Shepperd. Issues on the effective use of cbr technology for software project prediction. In *Case-Based Reasoning Research and Development*, LNCS v. 2080, pages 276–290. 2001.

[43] G. Kadoda and M. Shepperd. Using simulation to evaluate predictions techniques. In *Proc. of Int. Software Metrics Symposium*, pages 349–358. IEEE press, 2001.

[44] B. Kitchenham, L. Pickard, and S. Pfleeger. Case studies for method and tool evaluation. *IEEE Software*, 12(4):52–62, 1995.

[45] B. Kitchenham, L. M. Pickard, S. G. MacDonell, and M. J. Shepperd. What accuracy statistics really measure. *IEEE Proc. Software*, 148(3):81–85, 2001.

[46] J. D. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical Report 214, ETH Zurich, 2006.

[47] E. Kocaguneli, T. Menzies, A. Bener, and J. Keung. Exploiting the essential assumptions of analogy-based effort estimation. *IEEE TSE*, 38(2):425–438, 2012.

[48] E. Kocaguneli, T. Menzies, J. Hihn, and B. H. Kang. Size doesn't matter?: On the value of software size features for effort estimation. In *Proc. of PROMISE'12*, pages 89–98, 2012.

[49] E. Kocaguneli, T. Menzies, J. Keung, D. Cok, and R. Madachy. Active learning and effort estimation: Finding the essential content of software effort estimation data. *IEEE TSE*, 39(8):1040–1053, 2013.

[50] E. Kocaguneli, T. Menzies, and J. W. Keung. On the value of ensemble effort estimation. *IEEE TSE*, 38(6):1403–1416, 2012.

[51] E. Kocaguneli, A. Misirli, B. Caglayan, and A. Bener. Experiences on developer participation and effort estimation. In *Proc. of EUROMICRO-SEAA'11*, pages 419–422, 2011.

[52] E. Kocaguneli, A. Tosun, and A. Bener. Ai-based models for software effort estimation. In *Proc. of EUROMICRO-SEAA'10*, pages 323–326, 2010.

[53] M. Korte and D. Port. Confidence in software cost estimation results based on mmre and pred. In *Proc. of PROMISE'08*, pages 63–70, 2008.

[54] J. R. Koza. *Genetic Programming*. MIT Press, 1992.

[55] W. B. Langdon, J. Dolado, F. Sarro, and M. Harman. Exact mean absolute error of baseline predictor, MARP0. *IST*, 73:16 – 18, 2016.

[56] M. Lefley and M. J. Shepperd. Using genetic

programming to improve software effort estimation based on general data sets. In *Proc. of GECCO'03*, pages 2477–2487, 2003.

[57] C. Lokan. What should you optimize when building an estimation model? In *Proc. of METRICS'05*, page 34, 2005.

[58] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd, and S. Webster. An investigation of machine learning based prediction systems. *JSS*, 53(1):23–29, 2000.

[59] S. McConnell. *Software Estimation: Demystifying the Black Art*. Microsoft Press, 2006.

[60] E. Mendes, S. Counsell, N. Mosley, C. Triggs, and I. Watson. A comparative study of cost estimation models for web hypermedia applications. *EMSE*, 8(23):163–196, 2003.

[61] E. Mendes and B. Kitchenham. A comparison of cross-company and within-company effort estimation models for web applications. In *Proc. of EASE'04*, pages 47–55, 2004.

[62] E. Mendes and N. Mosley. Further investigation into the use of cbr and stepwise regression to predict development effort for web hypermedia applications. In *Proc. of Int. Symposium on Empirical Software Engineering*, pages 79–90, 2002.

[63] T. Menzies, Z. Chen, J. Hihn, and K. Lum. Selecting best practices for effort estimation. *IEEE TSE*, 32(11):883–895, 2006.

[64] T. Menzies, M. Rees-Jones, R. Krishna, and C. Pape. The promise repository of empirical software engineering data, 2015.

[65] T. Menzies and M. Shepperd. Special issue on repeatable results in software engineering prediction. *EMSE*, 17(1):1–17, 2012.

[66] L. L. Minku and X. Yao. Software effort estimation as a multiobjective learning problem. *ACM TOSEM*, 22(4):35, 2013.

[67] K. Molkken and M. Jörgensen. A review of surveys on software effort estimation. In *Proc. of ISESE'03*, pages 223–230, 2003.

[68] S. Nejati and L. C. Briand. Identifying optimal trade-offs between cpu time usage and temporal constraints using search. In *Proc. of ISSTA'14*, pages 351–361, 2014.

[69] G. Neumann, M. Harman, and S. M. Poulding. Transformed vargha-delaney effect size. In *Proc. of SSBSE'15*, pages 318–324, 2015.

[70] R. Olaechea, D. Rayside, J. Guo, and K. Czarnecki. Comparison of exact and approximate multi-objective optimization for software product lines. In *Proc. of SPLC'14*, pages 92–101, 2014.

[71] D. Port and M. Korte. Comparative studies of the model evaluation criterions mmre and pred in software cost estimation research. In *Proc. of ESEM'08*, pages 51–60, 2008.

[72] K. Praditwong, M. Harman, and X. Yao. Software module clustering as a multi-objective search problem. *IEEE TSE*, 37(2):264–282, 2011.

[73] P. Royston. An extension of Shapiro and Wilk's W test for normality to large samples. *Applied Statistics*, 31(2):115–124, 1982.

[74] F. Sarro, F. Ferrucci, and C. Gravino. Single and multi objective genetic programming for software development effort estimation. In *Proc. of ACM SAC'12*, pages 1221–1226, 2012.

[75] P. Sentas, L. Angelis, and I. Stamelos. Multinomial logistic regression applied on software productivity prediction. In *9th Panhellenic Conf. in Inf.*, 2003.

[76] P. Sentas, L. Angelis, I. Stamelos, and G. Bleris. Software productivity and effort prediction with ordinal regression. *IST*, 47(1):17–29, 2005.

[77] Y. Shan, R. I. Mckay, C. J. Lokan, and D. L. Essam. Software project effort estimation using genetic programming. In *Proc. of CCS'02*, pages 1108–1112, 2002.

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

[79] M. Shepperd, M. Cartwright, and G. Kadoda. On building prediction systems for software engineers. *EMSE*, 5(3):175–182, 2000.

[80] M. Shepperd and C. Schofield. Estimating Software Project Effort using Analogies. *IEEE TSE*, 23(11):736 –743, 1997.

[81] M. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE TSE*, 23(11):736–743, 2000.

[82] M. J. Shepperd and S. G. MacDonell. Evaluating prediction systems in software project estimation. *IST*, 54(8):820–827, 2012.

[83] D. L. Shrestha and D. P. Solomatine. Machine learning approaches for estimation of prediction interval for the model output. *Neural Networks*, 19(2):225 – 235, 2006.

[84] I. Sommerville. *Software Engineering*. Pearson, 9th edition, 2010.

[85] I. Stamelos and L. Angelis. Managing uncertainty in project portfolio cost estimation. *IST*, 43(13):759–768, 2001.

[86] I. Stamelos, L. Angelis, P. Dimou, and E. Sakellaris. On the use of bayesian belief networks for the prediction of software productivity. *IST*, 45(1):51–60, 2003.

[87] E. Stensrud, T. Foss, B. Kitchenham, and I. Myrtveit. A further empirical investigation of the relationship between MRE and project size. *EMSE*, 8(2):139–161, 2003.

[88] A. Trendowicz. *Software Project Effort Estimation: Foundations and Best Practice Guidelines for Success*. Springer, 2014.

[89] D. A. V. Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis, 1998.

[90] P. A. Whigham, C. A. Owen, and S. G. Macdonell. A baseline model for software effort estimation. *ACM TOSEM*, 24(3):20:1–20:11, 2015.

[91] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE TEC*, 3(4):257 –271, 1999.

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