Software Effort Estimation as a Multi-objective Learning Problem

Leandro L. Minku and Xin Yao, The University of Birmingham

Ensembles of learning machines are promising for software effort estimation (SEE), but need to be tailored for this task to have their potential exploited. A key issue when creating ensembles is to produce diverse and accurate base models. Depending on how differently different performance measures behave for SEE, they could be used as a natural way of creating SEE ensembles. We propose to view SEE model creation as a multi-objective learning problem. A multi-objective evolutionary algorithm (MOEA) is used to better understand the trade-off among different performance measures by creating SEE models through the simultaneous optimisation of these measures. We show that the performance measures behave very differently, presenting sometimes even opposite trends. They are then used as a source of diversity for creating SEE ensembles. A good trade-off among different measures can be obtained by using an ensemble of MOEA solutions. This ensemble performs similarly or better than a model that does not consider these measures explicitly. Besides, MOEA is also flexible, allowing emphasis of a particular measure if desired. In conclusion, MOEA can be used to better understand the relationship among performance measures and has shown to be very effective in creating SEE models.

Categories and Subject Descriptors: D.2.9 [**Software Engineering**]: Management—*Cost estimation*; I.2.6 [**Artificial Intelligence**]: Learning—*Connectionism and Neural Nets*; *Concept Learning*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms: Experimentation, Algorithms, Management

Additional Key Words and Phrases: Software effort estimation, ensembles of learning machines, multi-objective evolutionary algorithms

ACM Reference Format:
Minku, L., and Yao, X. 2011. Software Effort Estimation as a Multi-objective Learning Problem. ACM Trans. Softw. Eng. Methodol. V, N, Article A (January YYYY), 32 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Estimating the cost of a software project is a task of strategic importance in project management. Both over and underestimations of cost can cause serious problems to a company. For instance, overestimations may result in a company losing contracts or wasting resources, whereas underestimations may result in poor quality, delayed or unfinished softwares. The major contributing factor for software cost is effort [Agarwal et al. 2001].

This work is supported by two EPSRC grants (Nos. EP/D052785/1 and EP/J017515/1).

Authors' address: L. Minku and X. Yao are with the Centre of Excellence for Research in Computational Intelligence and Applications, School of Computer Science, The University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

Human made effort estimations tend to be strongly affected by effort-irrelevant and misleading information [Jrgensen and Grimstad 2011]. Moreover, developers tend not to improve their effort estimation uncertainty assessment even after feedback about their estimates is provided [Gruschke and Jørgensen 2008]. An alternative to human made effort estimations is to use automated effort estimators. Models for estimating software effort can be used as decision support tools, allowing investigation of the impact of certain requirements and development team features on the cost/effort of a project to be developed.

A major difficulty in performing software estimation is the lack of project data that include size and detailed quantitative counts on the elements of UML models used, e.g., the number of analysis classes, number of relationship types, number of associated attributes, use-case point, etc. As a result, even though UML models are commonly used, it is still very difficult to do software estimation based on UML models constructed in the early stage of software development. Hence, it is very difficult to further test, evaluate and enhance very promising estimation approaches that are based on the UML models constructed in the early state of requirements analysis [Tan et al. 2009; Mohagheghi et al. 2005; Tan et al. 2006]. The analysis of the effectiveness of estimation models based on limited project data thus becomes an important field to be investigated. Different automated methods for software cost or Software Effort Estimation (SEE) have been proposed [Jorgensen and Shepperd 2007].

In the present work, we look into a type of machine learning method which has recently attracted attention from the SEE community [Braga et al. 2007; Kultur et al. 2009; Kocaguneli et al. 2009; Minku and Yao 2011], namely ensembles of learning machines. Ensembles are sets of learners trained to perform the same task and combined with the aim of improving predictive performance [Chen and Yao 2009]. When combining learning models in an attempt to get more accurate predictions, it is commonly agreed that these base models should behave differently from each other. Otherwise, the overall prediction will not be better than the individual predictions [Brown et al. 2005; L. I and Whitaker 2003]. This behaviour matches intuition: if models that make the same mistakes are combined into an ensemble, the ensemble will make the same mistakes as the individual models and its performance will be no better than the individual performances. On the contrary, ensembles composed of diverse models can compensate the mistakes of certain models through the correct predictions performed by other models. So, diversity refers to the predictions/errors made by the models. Two models are said to be diverse if they make different errors on the same data points [Chandra and Yao 2006]. Different ensemble learning approaches can be seen as different ways to generate diversity among the base models.

Even though ensembles have been shown to be promising for SEE, it is necessary to tailor them for this estimation task. Simply using any general purpose ensemble approach from the literature does not necessarily improve SEE in comparison to single learning machines [Minku and Yao 2011]. Minku and Yao [2012] showed that combining the power of ensembles to local learning through the use of bagging ensembles of regression trees outperforms several other learning machines for SEE in terms of Mean Absolute Error (MAE). This combination is a way to tailor ensembles for SEE, but Minku and Yao [2012] also show that more improvements may still be achieved if additional tailoring is performed. Very recently, Kocaguneli et al. [2011] proposed an ensemble method claimed to outperform single learners for SEE. However, their method is not fully automated, as it needs manual/visual inspection of extensive experiments to create the ensemble. Section 2 explains related work on machine learning and in particular ensembles for SEE.

Much of the SEE work involves empirical evaluation of models and several different measures of performance can be used for that. Examples of measures are two popular measures based on the magnitude of the relative error, namely Mean Magnitude of the Relative Error (MMRE) and percentage of estimations within N% of the actual value (PRED(N)), and a measure based on the logarithm of the residuals recommended by Foss et al. [2003], namely Logarithmic Standard Deviation (LSD). Different measures can behave differently and it is highly unlikely that there is a "single, simple-to-use, universal goodness-of-fit kind of metric" [Foss et al. 2003], i.e., none of these measures is totally free of problems.

The relationship among different measures is not yet well understood. For example, it is not known to what extent a lower MMRE could be related to a higher LSD and vice versa, or how different are the MMREs associated to the same PRED value. Depending on how differently these performance measures behave, it may be possible to use them as a natural way to generate diversity in ensembles for SEE. With that in mind, this paper aims at answering the following research questions:

— RQ1: What is the relationship among different performance measures for SEE?
— RQ2: Can we use different performance measures as a source of diversity to create SEE ensembles? In particular, can that improve the performance in comparison to models created without considering these measures explicitly? Existing models do not necessarily consider the performance measures in which we are interested explicitly. For example, Multi-layer Perceptrons (MLPs) are usually trained using Backpropagation, which is based on the mean squared error. So, they can only improve MMRE, PRED and LSD indirectly. We would like to know whether creating an ensemble considering MMRE, PRED and LSD explicitly leads to more improvement in the SEE context.
— RQ3: Is it possible to create models that emphasize particular performance measures should we wish to do so? For example, if there is a certain measure that we believe to be more appropriate than the others, can we create a model that particularly improves performance considering this measure? This is useful not only for the case where the software manager has sufficient domain knowledge to chose a certain measure, but also (and mainly) if there are future developments of the SEE research showing that a certain measure is better than others for a certain purpose.

In order to answer these questions, we formulate the problem of creating SEE models as a multi-objective learning problem that considers different performance measures explicitly as objectives to be optimised. This formulation is key to answer the research questions because it allows us to use a Multi-objective Evolutionary Algorithm (MOEA) [Wang et al. 2010] to generate SEE models that are generally good considering all the pre-defined performance measures. This feature allows us to use plots of the performances of these models to understand the relationship among the performance measures and how differently they behave. Once these models are obtained, the models that perform best for each different performance measure can be determined. These models behave differently from each other and can be used to form a diverse ensemble that provides an ideal trade-off among these measures. Choosing a performance measure is not an easy task. By using our ensemble, the software manager does not need to chose a certain performance measure, as an ideal trade-off among different measures is provided. As an additional benefit of this approach, each of the models that compose the ensemble can also be used separately to emphasize a specific performance measure if desired. The performance measures used by the MOEA in this work are MMRE, PRED(25) and LSD. Section 3 explains MOEAs and section 4 explains our proposed approach to use MOEAs for creating SEE models.

Our analysis shows that the different performance measures behave very differently when analysed at their individual best level and sometimes present even opposite behaviour (RQ1). For example, when considering nondominated (section 3) solutions, as

MMRE is improved, LSD tends to get worse. This is an indicator that these measures can be used to create diverse ensembles for SEE. We then show that MOEA is successful in generating SEE ensemble models by optimising different performance measures explicitly at the same time (RQ2). The ensembles are composed of nondominated solutions selected from the last generation of the MOEA, as explained in section 4. These ensembles present performance similar or better than a model that does not optimize the three measures concurrently. Furthermore, the similar or better performance is achieved considering all the measures used to create the ensemble, showing that these ensembles not only provide a better trade-off among different measures, but also improve the performance considering these measures. We also show that MOEA is flexible, allowing us to chose solutions which emphasize certain measures, if desired (RQ3).

The base learners generated by the MOEA in this work are Multi-layer Perceptrons (MLPs) [Bishop 2005], which have been showing success in the SEE literature for not being restricted to linear project data [Tronto et al. 2007]. Even though we generate MLPs in this work, MOEAs could also be used to generate other types of base learners, such as RBFs, RTs and linear regression equations. An additional comparison of MOEA to evolve MLPs was performed against nine other types of models. The comparison shows that the MOEA-evolved MLPs were ranked first more often in terms of five different performance measures, but performed less well in terms of LSD. They were also ranked first more often for the data sets likely to be more heterogeneous. It is important to note, though, that this additional comparison is not only evaluating the MOEA and the multi-objective formulation of the problem, but also the type of model being evolved (MLP). Other types of models could also be evolved by the MOEA and could possibly provide better results in terms of LSD. As the key point of this paper is to analyse the multi-objective formulation of the creation of SEE models, and not the comparison among different types of models, the experimentation with different types of MOEA and different types of base models is left as future work.

This paper is organised as follows. Section 2 presents related work on machine learning and ensembles for SEE. Section 3 explains MOEAs and the type of MOEA used in this work. Section 4 explains our approach for creating SEE models (including ensembles) through MOEAs. In particular, section 4.1 explains the multi-objective formulation of the problem of creating SEE models. Section 5 explains the experimental setup for answering the research questions and evaluating our approach. Section 6 explains the data sets used in the experiment. Section 7 provides an analysis of the relationship among different performance measures (RQ1). Section 8 provides an evaluation of the MOEA's ability of creating ensembles by optimising several different measures at the same time (RQ2). Section 9 shows that MOEAs are flexible, allowing us to create models that emphasize particular performance measures, if desired (RQ3). Section 10 shows that there is still room for improvement in the choice of MOEA models to be used for SEE. Section 11 complements the evaluation by checking how well the Pareto ensemble of MLPs performs in comparison to other types of models. Section 12 presents the conclusions and future work.

2. MACHINE LEARNING FOR SEE

Several different methods for automating software cost/effort estimation have been proposed [Jorgensen and Shepperd 2007]. For example, Shepperd and Schofield [1997] present a landmark study using estimation by analogy, which is a type of case-based reasoning. The features and effort of completed projects are stored and then the effort estimation problem is regarded as the one of finding the most similar projects in terms of Euclidean distance to the one for which an estimation is required. The approach was evaluated on nine data sets and obtained in general better MMRE and PRED(25) than more traditional stepwise regression models. Chulani et al. [1999] presented another study using a Bayesian approach to combine a priori information based on expert knowledge to a linear regression model based on log transformation of the data. The proposed approach outperforms linear regression models based on log transformed data in terms of PRED(20), PRED(25) and PRED(30) on one data set and its extended version. No comparison with Shepperd and Schofield [1997]'s work was given.

More recently, effort estimators based on machine learning approaches such as MultiLayer Perceptrons (MLPs), Radial Basis Function networks (RBFs) and Regression Trees (RTs) [Srivasan and Fisher 1995; Wittig and Finnie 1997; Heiat 2002; Baskeles et al. 2007; Tronto et al. 2007; Kultur et al. 2009; Braga et al. 2007] have received increased attention [Jorgensen and Shepperd 2007]. MLPs are artificial neural networks widely used in the machine learning literature. They are composed of at least three layers of neurons, where neurons of a certain layer are connected to all neurons of the next layer. Neurons compute a function of the weighted sum of their inputs. An example of function frequently used is sigmoid. MLPs can approximate any continuous function and learning consists of adjustments to the connection weights. RBFs are artificial neural networks whose neurons of the first layer compute a radial basis function, allowing them to perform local learning. Learning consists of calculating the centres of the neurons of the first layer, the width of their radial basis function, and adjusting the weights of the connections of the output neurons. RTs are easy-to-understand structures that provide if-then rules to perform regression based on the values of the input attributes. There are several different types of RTs and an example of RT for effort estimation is given in figure 1 [Minku and Yao 2011]. Learning consists of determining which attributes to split and based on what values.

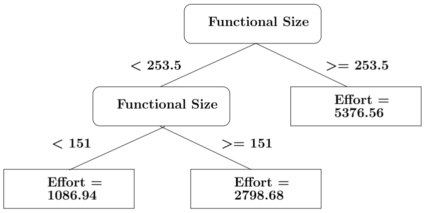


Fig. 1: An example plot of RT for SEE [Minku and Yao 2011].

The motivation behind the use of such approaches is that they make no or minimal assumptions about the function being modelled and the data used for training. For instance, Tronto et al. [2007] showed that MLPs improve SEE over conventional linear models because they are not restricted to linear functions, being able to model observations that lie far from the best straight line. Earlier work also reported favourably over MLPs [Wittig and Finnie 1994]. Dejaeger et al. [2012] argued that techniques such as ordinary least squares regression based on log transformed data perform in general better than some types of regression trees and neural networks analysed in their study. Nine data sets and several performance measures were used in their study.

Another type of approaches that has been recently attracting the attention of the SEE community are ensembles of learning machines [Braga et al. 2007; Kultur et al. 2009; Kocaguneli et al. 2009; Minku and Yao 2011]. Minku and Yao [2011] showed that a bagging [Breiman 1996] ensemble of MLPs performs similarly to REPTrees [Hall et al. 2009] for SEE in terms of MMRE and PRED(25) based on thirteen data sets. They explained that additional tailoring is necessary so that ensembles can improve SEEs on these performance measures. Minku and Yao [2012] showed that combining the power of ensembles to local learning through the use of bagging ensembles of REPTrees outperforms several other learning machines for SEE in terms of MAE. Combining ensembles and locality is a way to tailor ensembles for SEE, but Minku and Yao [2012] also show that more improvements may still be achieved if additional tailoring is performed. Kocaguneli et al. [2011] proposed an ensemble method that outperforms single learners for SEE. Their method combines several types of so called *solo-methods* (combinations of single learners and preprocessing techniques) to perform SEE. They reported that the ensemble presents less instability than solo-methods when ranked in terms of the total number of $wins$, $losses$ and $wins - losses$ considering several different performance measures and twenty data sets. These observations confirmed earlier results reported in the ensemble learning literature that ensembles generally perform better than its component learners. They also reported that the ensembles obtained less $losses$ than other methods. As an additional contribution, their extensive study showed that the non-linear approaches CART (a type of RT) and estimation by analogy *based on log transformed data* can outperform other methods such as linear regression based on log transformed data.

However, their approach [Kocaguneli et al. 2011] has high implementation complexity and is not fully automated. It requires an extensive experimentation procedure using several types of single learners and preprocessing techniques for creating the ensemble. It consists of selecting the "best" solo-methods in terms of $losses$ and stability to compose the ensemble, by manually/visually checking and comparing their stability. The manual/visual checking process is needed because it is necessary not only to determine what solo-methods have the lowest number of $losses$ (that by itself could be automated), but also to check whether these are the same as the ones comparatively more stable and what level of stability should be considered as comparatively superior or not.

Differently from Kocaguneli et al. [2011]'s approach, our approach is fully automated. Once developed, a tool using our MOEA can be easily run to learn a model that uses data from a specific company. It is worth noting that parameters choice of our approach can be automated, as long as the developer of the tool embeds on it several different parameter values to be tested on a certain percentage of the completed projects of the company. More importantly, we can create ensembles in such a way to encourage both accuracy and diversity, which is known to be beneficial for ensembles [Brown et al. 2005; L. I and Whitaker 2003]. Kocaguneli et al. [2011]'s approach is mainly focused on accuracy, providing no guarantee that the base learners perform diversely. As explained by Chandra and Yao [2006], ensembles need their base models to be both accurate and diverse in order to perform well. However, there is a trade-off between accuracy and diversity of base models. So, if we focus only on improving their accuracy, they are likely to lack diversity, reducing the accuracy of the ensemble as a whole.

3. MOEAS

MOEAs are population-based optimisation algorithms that evolve sets of candidate solutions by optimising two or more possibly conflicting objectives. Candidate solutions are generated/evolved through evolutionary operators such as crossover and mutation in rounds called generations. The evolutionary process is frequently guided by the concept of dominance, which is defined as follows. Consider a multi-objective optimisation problem consisting of N objectives $f_i(x)$ to be minimized, where x is a p dimensional vector containing p design or decision variables [Srinivas and Deb 1994]. A solution $x^{(1)}$ *dominates* a solution $x^{(2)}$ iff:

$$f_i(x^{(1)}) \leq f_i(x^{(2)})\ \forall i \wedge \exists i \mid f_i(x^{(1)}) < f_i(x^{(2)})$$

This concept can easily be generalized to problems involving maximization.

The set of optimal solutions (nondominated by any other solution) is called *Pareto front*. Even though the true Pareto front is very difficult to be found, a MOEA can find a set of acceptable solutions nondominated by any other solution in the last generation. We will refer to these solutions as *Pareto solutions*.

There exist several different MOEAs. As examples we can cite the conventional and well known Nondominated Sorting Genetic Algorithm II (NSGA-II) [Deb et al. 2002]; the improved Strength Pareto Evolutionary Algorithm (SPEA2) [Zitzler et al. 2002]; the two-archive algorithm [Praditwong and Yao 2006], which has been used in the software engineering context for software module clustering by Praditwong et al. [2011]; and the Harmonic Distance MOEA (HaD-MOEA) [Wang et al. 2010].

Once a representation, evolutionary operators and objective functions are defined, our proposed approach can be used in combination to any MOEA. We chose HaD-MOEA as the algorithm to be used in our experiments due to its simplicity and advantages over NSGA-II as explained in section 3.1. This paper does not intend to show that a particular MOEA performs better or worse than another. For that reason, we leave the evaluation of other MOEAs to evolve SEE models as future work. In the current paper, we concentrate on the multi-objective formulation of the problem, how to use a multi-objective approach to solve it, and to provide a better understanding of different performance measures.

3.1. HaD-MOEA

HaD-MOEA [Wang et al. 2010] is a MOEA that improves upon NSGA-II. Even though NSGA-II is a conventional and well known MOEA, it does not perform well when the number of objectives increases [Khare et al. 2003]. Wang et al. [2010] explained that two key problems in NSGA-II are its measure of crowding distance and the method for selecting solutions based on it.

The crowding distance is used by NSGA-II to select solutions that cover the whole objective space well. This is a critical step in NSGA-II [Deb et al. 2002], as it helps maintaining the diversity of the population, which in turn helps the search process to find solutions with better quality. However, NSGA-II uses the 1-norm distance between the two nearest neighbours of a solution as the crowding distance measure. Wang et al. [2010] showed that this measure does not reflect well the actual crowding degree of a given solution. The problem with this measure is inherent from 1-norm's own definition. So, Wang et al. [2010] proposed to use the harmonic distance to overcome this problem in their algorithm HaD-MOEA.

Also, NSGA-II selects solutions based on the crowding distance calculated considering only the solutions belonging to the same nondominated front. This obviously does not reflect the real crowding distance considering all solutions selected so far, which is the real crowding of the solutions. So, in HaD-MOEA, after sorting the solutions in the intermediate population into a number of fronts, if some solutions are to be selected from the same front, the crowding distance is calculated based on both the solutions belonging to the same front and all the previously selected solutions.

HaD-MOEA's pseudo-code is shown in algorithm 1. Our implementation was based on the Meta-heuristic Optimisation Framework for Java Opt4J [Lukasiewycz et al. 2011]. As in other evolutionary algorithms, parents are selected from the population to produce offspring based on evolutionary operators (line 3). In HaD-MOEA, parents selection is typically done using tournament selection [Miller and Goldberg 1995].

ALGORITHM 1: HaD-MOEA

```
**Input**: population size $\alpha$, number of generations $G$.
**Output**: $P_g$.
1  Initialize initial population $P_1 = \{x_1, x_2, ..., x_\alpha\}$;
2  **for** $g \leftarrow 1$ *to* $G$ **do**
3      Generate offspring population $Q_g$ from $P_g$ with size $\alpha$;
4      Combine parent and offspring population $R_g = P_g \cup Q_g$;
5      Sort all solutions of $R_g$ to get all nondominated fronts $F = fast\_nondominated\_sort(R_g)$,
       where $F = (F_1, F_2, ...)$;
6      Set $P_{g+1} = \{\}$ and $i = 1$;
7      **while** *the population size* $|P_{g+1}| + |F_i| < N$ **do**
8          Add the $i$th nondominated front $F_i$ to $P_{g+1}$;
9          $i = i + 1$;
10     **end**
11     Combine $F_i$ and $P_{g+1}$ to a temporary vector $T$;
12     Calculated the harmonic crowding distance of individuals of $F_i$ in $T$;
13     Sort $F_i$ according to the crowding distance;
14     Set $T = \{\}$;
15     Fill $P_{g+1}$ with the first $\alpha - |P_{g+1}|$ elements of $F_i$;
16 **end**
```

The population of parents and offspring is then combined (line 4) and the individuals are separated into nondominated fronts (line 5). Each front is composed of individuals that are not dominated by any individual of any subsequent front. They are used to choose the individuals that compose the next population (lines 7-15).

As each population has a pre-defined size α, it is usually not possible to include all the individuals from a certain front in the new population. In order to determine which individuals should be included, HaD-MOEA calculates their harmonic distance considering both this front and the individuals already included in the new population (lines 11 and 12). The algorithm then selects the individuals with the largest distances (line 15), which represent less crowded regions of the solution space. In this way, diversity is encouraged and preserved.

4. USING HAD-MOEA FOR CREATING SEE MODELS

As explained by Harman and Clark [2004] ,"[m]etrics, whether collected statically or dynamically, and whether constructed from source code, systems or processes, are largely regarded as a means of evaluating some property of interest". For example, functional size and software effort are metrics derived from the project data. Performance measures such as MMRE, PRED and LSD are metrics that represent how well a certain model fits the project data, and are calculated based on metrics such as software effort.

Harman and Clark [2004] explain that metrics can be used as fitness functions in search based software engineering, being able to guide the force behind the search for optimal or near optimal solutions in such a way to automate software engineering tasks. For example, metrics can be used in the design of process, architecture and infrastructure, and test data. In the context of software cost/effort estimation, genetic programming has been applied using mean squared error (MSE) as fitness function [Dolado 2000; 2001; Shan et al. 2002].

In the present work, we innovatively formulate the problem of creating SEE models as a multi-objective learning problem. This formulation was preliminary presented in [Minku 2011]. We use different performance measures as objectives to be optimised simultaneously for generating SEE models. Differently from other formulations, that allows us to get a better understanding of different performance measures, to create well performing SEE ensembles and to emphasize different performance measures if desired, as explained later in section 5.

The algorithm used in this work was HaD-MOEA (section 3.1). When using MOEAs for a particular problem, the objectives, the representation of solutions and the evolutionary operators need to be carefully designed. In this section, we explain our proposed approach to create SEE models using MOEA. Section 4.1 explains how we formulate the problem of creating SEE models as a multi-objective optimisation problem. Section 4.2 comments on the type of SEE models generated. Section 4.3 explains the representation of the models and the evolutionary operators used. Section 4.4 explains which solutions produced by HaD-MOEA are used in the analyses.

4.1. Multi-objective Formulation of the Problem

Considering a set of T projects, the metrics used as objectives to be optimised when creating models in this work are defined as follows:

— Mean Magnitude of the Relative Error:

$$MMRE = \frac{1}{T}\sum_{i=1}^{T} MRE_i,$$

where $MRE_i = |\hat{y}_i - y_i|/y_i$; $\hat{y}_i$ is the predicted effort; and y_i is the actual effort.

— Percentage of estimations within 25% of the actual values:

$$PRED(25) = \frac{1}{T}\sum_{i=1}^{T} \begin{cases} 1, & \text{if } MRE_i \leq \frac{25}{100} \\ 0, & \text{otherwise} \end{cases} .$$

— Logarithmic Standard Deviation:

$$LSD = \sqrt{\frac{\sum_{i=1}^{T}\left(e_i + \frac{s^2}{2}\right)^2}{T-1}},$$

where s^2 is an estimator of the variance of the residual e_i and $e_i = \ln y_i - \ln \hat{y}_i$.

MMRE and LSD are objectives to be minimized, whereas PRED(25) is to be maximized. In order to avoid possible infinite LSD averages due to negative estimations, any negative estimation was replaced by the value one when calculating $\ln \hat{y}$. MMRE and PRED(25) are popular metrics in the SEE literature, as illustrated by table 2 of Dejaeger et al. [2012]'s work, whereas LSD was recommended by Foss et al. [2003] as being more reliable especially for multiplicative models. These measures were chosen because, even though all of them were initially designed to represent how well a model performs, they can behave very differently from each other, as illustrated in section 7. This is potentially very useful for maximizing diversity among ensemble members, as explained in section 4.4.

During the MOEA evolving procedure, the objective values are calculated using a set of projects with known effort which will be referred to as the training set. When evaluating the results of the approaches, the performance measures are calculated over the test set.

Several different performance measures for SEE can be found in the literature. Most of them are calculated over the prediction error ($y_i - \hat{y}_i$) [Menzies and Shepperd 2012]. The Mean Absolute Error (MAE) is the mean of the absolute prediction error, providing an unbiased measure that does not favour under or overestimates. Sometimes median measures are used to avoid influence of extreme values. For these reasons, the evaluation analysis of our approach also considers its MAE, Median Absolute Error (MdAE) and Median MRE (MdMRE).

4.2. SEE Models Generated

The models generated by the MOEA in this work are Multi-Layer Perceptrons (MLPs) [Bishop 2005], which are widely used artificial neural networks as explained in section 2. As explained in section 1, MLPs can improve SEE over conventional linear models because they are not restricted to linear functions, being able to model observations that lie far from the best straight line [Tronto et al. 2007]. Even though we evolve MLPs in this work, MOEAs could also be used to generate other types of models for SEE, such as RBFs, RTs and linear regression equations. As the key point of this paper is to analyse the multi-objective formulation of the creation of SEE models, and not the comparison among different types of models, the experimentation of MOEAs to create other types of models is left as future work.

4.3. Representation and Evolutionary Operators

The MLP models were represented by a real value vector of size $n_i \cdot (n_h + 1) + n_h \cdot (n_o + 1)$, where n_i, n_h and n_o are the number of inputs, hidden nodes and output nodes, respectively. This real value vector is manipulated by the HaD-MOEA to generate SEE models. Each position of the vector represents a weight or the bias of a node. The value one summed to n_h and n_o in the formula above represents the bias. The number of input nodes corresponds to the number of project independent variables and the number of output nodes is always one for the SEE task. The number of hidden nodes is a parameter of the approach.

The crossover and mutation operators were inspired by Chandra and Yao [2006]'s work, which also involves evolution of MLPs. Let w^{p_1}, w^{p_2} and w^{p_3} be three parents. One child w^c is generated with probability P_c according to the following equation:

$$w^c = w^{p_1} + N(0, \sigma^2)(w^{p_2} - w^{p_3}),$$

where w is the real value vector representing the individuals and $N(0, \sigma^2)$ is a random number drawn from a Gaussian distribution with mean zero and variance σ^2.

An adaptive procedure inspired by simulated annealing is used to update the variance σ^2 of the Gaussian at every generation [Chandra and Yao 2006]. This procedure allows the crossover to be initially explorative and then become more exploitative. The variance is updated according to the following equation:

$$\sigma^2 = 2 - \left(\frac{1}{1 + e^{(\text{anneal\_time} - \text{generation})}} \right),$$

where *anneal_time* is a parameter meaning the number of generations for which the search is to be explorative, after which σ^2 decreases exponentially until reaching and keeping the value of one.

Mutation is performed elementwise with probability Pm according to the following equation:

$$w_i = w_i + N(0, 0.1),$$

where w_i represents a position of the vector representing the MLP and $N(0, 0.1)$ is a random number drawn from a Gaussian distribution with mean zero and variance 0.1.

The offspring individuals receive further local training using Backpropagation [Bishop 2005], as in Chandra and Yao [2006]'s work.

4.4. Using the Solutions Produced by HaD-MOEA

The solutions produced by HaD-MOEA are innovatively used for SEE in two ways in this work. The first one is a *Pareto ensemble* composed of the *best fit Pareto solutions*. These solutions are the ones with the best train performance considering each objective separately. So, the ensemble will be composed of the Pareto solution with the best train LSD, best train MMRE and best train PRED(25). The effort estimation given by the ensemble is the arithmetic average of the estimations given by each of its learners. So, each performance measure can be seen as having "one vote", providing a fair and ideal trade-off among the measures when no emphasis is given to a certain measure over the others. This avoids the need for a software manager to decide on a certain measure to be emphasized.

It is worth noting that this approach to create ensembles focuses not only on accuracy, but also on diversity among base learners, which is known to be a key issue when creating ensembles [Brown et al. 2005; L. I and Whitaker 2003]. Accuracy is encouraged by using a MOEA to optimise MMRE, PRED and LSD simultaneously. So, the base learners are created in such a way to be generally accurate considering these three measures at the same time. Diversity is encouraged by selecting only the best fit Pareto solution according to each of these measures. As shown in section 7, these measures behave very differently from each other. So, it is likely that the MMRE of the best fit Pareto solution according to LSD will be different from the MMRE of the best fit Pareto solution according to MMRE itself. The same is valid for the other performance measures. Models with different performance considering a particular measure are likely to produce different estimations, being diverse.

The second way to use the solutions produced by HaD-MOEA is to use each best fit Pareto solution by itself. These solutions can be used when a particular measure is to be emphasized.

It is worth noting that HaD-MOEA automatically creates these models. The Pareto solutions and the best bit Pareto solutions can be automatically determined by the algorithm. There is no need for involving manual/visual checking.

5. EXPERIMENTAL STUDIES

The experiments were designed with the aim of answering research questions RQ1-RQ3. In order to answer RQ1, we show that plots of the Pareto solutions can be used to provide a better understanding of the relationship among different performance measures. They can show that, for example, when increasing the value of a certain measure, the value of another measure may decrease and by how much. Our study shows the very different and sometimes even opposite behaviour of different measures. This is an indicator that these measures can be used to create diverse ensembles for SEE (section 7).

In order to answer RQ2, the following comparison was made (section 8):

- Pareto ensemble vs Backpropagation MLP (single MLP created using Backpropagation). This comparison was made to show the applicability of MOEAs to generate SEE ensemble models. It analyses the use of MOEA, which considers several performance measures at the same time, against the non-use of a MOEA.

The results of this comparison show that MOEA is successful in generating SEE ensemble models by optimising different performance measures explicitly at the same time. These ensembles present performance similar or better than a model that does not optimize the three measures concurrently. Furthermore, the similar or better performance is achieved considering all the measures used to create the ensemble, showing that these ensembles not only provide a better trade-off among different measures, but also improve the performance considering these measures.

In order to answer RQ3, the following comparison was made (section 9):

- Best fit Pareto MLP vs Pareto ensemble. This comparison allows us to check whether it is possible to increase the performance considering a particular measure if we would like to emphasize it. This is particularly useful when there is a certain measure that we believe to be more appropriate than the others.

The results of this comparison reveal that MOEAs are flexible in terms of providing SEE models based on a multi-objective formulation of the problem. They can provide both solutions considered as having a good trade-off when no measure is to be emphasized and solutions that emphasize certain measures over the others if desired. If there is no measure to be emphasized, a Pareto ensemble can be used to provide a relatively good performance in terms of different measures. If the software manager would like to emphasize a certain measure, it is possible to use the best fit Pareto solution in terms of this measure.

Additionally, the following comparisons were made to test the optimality of the choice of best fit MLPs as models to be used (section 10):

- Best Pareto MLP in terms of test performance vs Backpropagation MLP, and Pareto ensemble composed of the best Pareto MLPs in terms of each test performance vs Backpropagation MLP. This comparison was made to check whether better results could be achieved if a better choice of solution from the Pareto front was made. Please note that choosing the best models based on their test performance was done for analysis purpose only and could not be done in practice.

The results of this comparison show that there is still room for improvement in terms of Pareto solution choice.

The comparisons to answer the research questions as outlined above show that it is possible and worth considering SEE models generation as a multi-objective learning problem and that a MOEA can be used both to provide a better understanding of different performance measures, to create well performing SEE ensembles and to create SEE models that emphasize particular performance measures.

In order to show how the solutions generated by the MOEA to evolve MLPs compare to other approaches in the literature, an additional round of comparisons was made against the following methods (section 11):

- Single learners: MultiLayer Perceptrons (MLPs) [Bishop 2005]; Radial Basis Function networks (RBFs) [Bishop 2005]; Regression Trees (RTs) [Zhao and Zhang 2008]; and Estimation by Analogy (EBA) [Shepperd and Schofield 1997] based on log transformed data.
- Ensemble learners: Bagging [Breiman 1996] with MLPs, with RBFs and with RTs; Random [Hall et al. 2009] with MLPs; and Negative Correlation Learning (NCL) [Liu and Yao 1999b; 1999a] with MLPs.

These comparisons do not evaluate the multi-objective formulation of the problem by itself, but a mix of the MOEA to the type of models being evolved (MLP). According to very recent studies [Kocaguneli et al. 2011; Minku and Yao 2011; 2012], REPTrees, bagging ensembles of MLPs, bagging ensembles of REPTrees and EBA based on log transformed data can be considered to be among the best current methods for SEE. The implementation used for all the opponent learning machines but NCL was based on Weka [Hall et al. 2009]. The regression trees were based on the REPTree model. We recommend the software Weka should the reader wish to get more details about the implementation and parameters. The software used for NCL is available upon request.

The results of these comparisons show that the MOEA-evolved MLPs were ranked first more often in terms of MMRE, PRED(25), MdMRE, MAE and MdAE, but performed less well in terms of LSD. They were also ranked first more often for the ISBSG (cross-company) data sets, which are likely to be more heterogeneous. It is important to emphasize, though, that this additional comparison is not only evaluating the MOEA and the multi-objective formulation of the problem, but also the type of model being evolved (MLP). Other types of models could also be evolved by the MOEA and could possibly provide better results in terms of LSD.

The experiments were based on the PROMISE data sets explained in section 6.1, the ISBSG subsets explained in section 6.2 and a data set containing the union of all the ISBSG subsets (orgAll). The union was used in order to create a data set likely to be more heterogeneous than the previous ones.

Thirty rounds of executions were performed for each data set from section 6. In each round, for each data set, 10 projects were randomly picked for testing and the remaining were used for the MOEA optimisation process/training of approaches. Holdout of size 10 was suggested by Menzies et al. [2006] and allows the largest possible number of projects to be used for training without hindering the testing. For the data set sdr (described in section 6.1), half of the projects were used for testing and half for training, due to the small size of the data set. The measures of performance used to evaluate the approaches are MMRE, PRED(25) and LSD, which are the same performance measures used to create the models (section 4.1), but calculated on the test set. It is worth noting that MMRE and PRED using the parameter 25 were chosen for being popular measures, even though raw values from different papers are not directly comparable because they use different training and test sets, besides possibly using different evaluation methods. In addition, we also report MdMRE, MAE and MdAE. The absolute value of the Glass's Δ effect size [Rosenthal 1994] was used to evaluate the practical significance of the changes in performance when choosing between the Pareto ensemble and an opponent approach:

$$\Delta = \frac{|M_a - M_p|}{SD_p},$$

where M_p and M_a are the performances obtained by the Pareto ensemble and an opponent approach, and SD_p is the standard deviation obtained by the Pareto ensemble. As the effect size is scale-free, it was interpreted based on Cohen [1992]'s suggested categories: small (≈ 0.2), medium (≈ 0.5) and large (≈ 0.8). Medium and large effect sizes are of more "practical" significance.

The parameters choice of the opponent approaches was based on five preliminary executions using several different parameters (table I). The set of parameters leading to the best MMRE for each data set was used for the final thirty executions used in the analysis. MMRE was chosen for being a popular measure in the literature. The experiments with the opponent approaches were also used by Minku and Yao [2011].

The MLP's learning rate, momentum and number of hidden nodes used by HaD-MOEA were chosen so as to correspond to the parameters used by the opponent Backpropagation MLPs and are presented in table II. These parameters were tunned to provide very good results for the opponent Backpropagation MLPs, but were not specifically tunned for our proposed approach. The number of generations, also shown in table II, is the number of epochs used by the opponent Backpropagation MLPs divided by the number of epochs for the offspring Backpropagation. This value was chosen so that each MLP at the end of the evolutionary process is potentially trained with the same total number of epochs as the opponent Backpropagation MLPs. The remain-

Table I: Parameter values for preliminary executions.

Approach	Parameters
MLP	Learning rate = $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ Momentum = $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ # epochs = $\{100, 500, 1000\}$ # hidden nodes = $\{3, 5, 9\}$
RBF	# clusters = $\{2, 3, 4, 5, 6\}$ Minimum std. deviation for the clusters = $\{0.01, 0.1, 0.2, 0.3, 0.4\}$
REPTree	Minimum total weight for instances in a leaf = $\{1, 2, 3, 4, 5\}$ Minimum proportion of the data variance at a node for splitting to be performed = $\{0.0001, 0.001, 0.01, 0.1\}$
Ensembles	# base learners = $\{10, 25, 50\}$ All the possible parameters of the adopted base learners, as shown above
NCL	Penalty strength = $\{0.3, 0.4, 0.5\}$

Table II: Parameter values used in the HaD-MOEA.

Data Set	Learning rate	Momentum	# generations	# hidden nodes
Cocomo81	0.3	0.5	200	9
Sdr	0.5	0.2	20	9
Nasa	0.1	0.1	100	9
Desharnais	0.1	0.1	100	9
Nasa93	0.4	0.5	20	5
Org1	0.1	0.2	200	9
Org2	0.1	0.1	100	5
Org3	0.1	0.3	100	5
Org4	0.2	0.3	200	9
Org5	0.2	0.3	200	9
Org6	0.1	0.1	20	3
Org7	0.5	0.3	20	5
OrgAll	0.1	0.5	20	9

ing evolutionary parameters were fixed for all data sets and were not intended to be optimal. In summary, these are:

- Tournament size: 2. Tournament is a popular parent selection method. A tournament size of 2 is commonly used in practice because it often provides sufficient selection pressure on the most fit individuals [Legg et al. 2004].
- Population size: 100. This value was arbitrarily chosen.
- Number of epochs used for the Backpropagation applied to the offspring individuals: 5. This is the same value as used by Chandra and Yao [2006].
- Anneal_time: number of generations divided by 4, as in [Chandra and Yao 2006].
- Probability of crossover: 0.8. Chosen between 0.8 and 0.9 (the value used by Wang et al. [2010]) so as to reduce the MMRE in five preliminary executions for cocomo81. We decided to check whether 0.8 would be better than 0.9 because 0.9 can be considered as a fairly large probability.
- Probability of mutation: 0.05. Chosen between 0.05 and 0.1 (the value used by Wang et al. [2010]) so as to reduce the MMRE in five preliminary executions for cocomo81. We decided to check whether 0.05 would be better than 0.1 because the value 0.1 can be considered large considering the size of each individual of the population in our case.

A population size arbitrarily reduced to 30 and number of epochs for the offspring Backpropagation reduced to zero (no Backpropagation) were used for additional MOEA executions in the analysis. Unless stated otherwise, the original parameters summarized above were used.

6. DATA SETS

The analysis presented in this paper is based on five data sets from the PRedictOr Models In Software Engineering Software (PROMISE) Repository [Shirabad and Menzies 2005] and eight data sets based on the International Software Benchmarking Standards Group (ISBSG) Repository [ISBSG 2011] Release 10, as in Minku and Yao [2011]. The data sets were chosen to cover a wide range of problem features, such as number of projects, types of features, countries and companies. Sections 6.1 and 6.2 provide their description and processing.

6.1. PROMISE Data

The PROMISE data sets used in this study are: cocomo81, nasa93, nasa, sdr and desharnais. Cocomo81 consists of the projects analysed by Boehm to introduce COCOMO [Boehm 1981]. Nasa93 and nasa are two data sets containing Nasa projects from the 1970s to the 1980s and from the 1980s to the 1990s, respectively. Sdr contains projects implemented in the 2000s and was collected at Bogazici University Software Engineering Research Laboratory from software development organisations in Turkey. Desharnais' projects are dated from late 1980s. Table III provides additional details and the next subsections explain their features, missing values and outliers.

Table III: PROMISE data sets.

Data Set	# Projects	# Features	Min Effort	Max Effort	Avg Effort	Std Dev Effort
Cocomo81 (effort in person-months)	63	17	5.9	11,400	683.53	1,821.51
Nasa93 (effort in person-months)	93	17	8.4	8,211	624.41	1,135.93
Nasa (effort in person-months)	60	16	8.4	3,240	406.41	656.95
Sdr (effort in person-months)	12	23	1	22	5.73	6.84
Desharnais (effort in person-hours)	81	9	546	23,940	5,046.31	4,418.77

6.1.1. Features. Cocomo81, nasa93 and nasa are based on the COCOMO [Boehm 1981] format, containing as input features 15 cost drivers, the number of lines of code and the development type (except for nasa, which does not contain the latter feature). The actual effort in person-months is the dependent variable. Sdr is based on COCOMO II [Boehm et al. 2000], containing as input features 22 cost drivers and the number of lines of code. The actual effort in person-months is the dependent variable. The data sets were processed to use the COCOMO numeric values for the cost drivers. The development type was transformed into dummy variables.

Desharnais follows an independent format, containing as input features the team experience in years, the manager experience in years, the year the project ended, the number of basic logical transactions in function points, the number of entities in the system's data model in function points, the total number of non-adjusted function points, the number of adjusted function points, the adjustment factor and the programming language. Actual effort in person-hours is the dependent variable.

6.1.2. Missing Values. The only data set with missing values is desharnais. In total, it contains only 4 in 81 projects with missing values. So, these projects were eliminated.

6.1.3. Outliers. The literature shows that SEE data sets frequently have a few outliers, which may hinder the SEEs for future projects [Seo et al. 2008]. In the current work, outliers were detected using k-means. This method was chosen because it has shown to improve performance in the SEE context [Seo et al. 2008]. K-means is used to divide the projects into clusters. The silhouette value for each project represents the similarity of the project to the other projects of its cluster in comparison to projects of the other clusters, ranging from -1 (more dissimilar) to 1 (more similar). So, the average silhouette value can be used to determine the number of clusters k. After applying k-means to the data, clusters with less than a certain number n of projects or projects with negative silhouette values are considered outliers.

We used $n = 3$, as in Seo et al. [2008]'s work. The number of clusters k was chosen among $k = \{2, 3, 4, 5\}$, according to the average silhouette values. As shown in table IV, the highest average silhouette values were always for $k = 2$ and were very high for all data sets (between 0.8367 and 0.9778), indicating that the clusters are generally homogeneous. The number of outliers was also small (from none to 3), representing less than 5% of the total number of projects, except for sdr. The projects considered as outliers were eliminated from the data sets, apart from the outlier identified for sdr. As this data set is very small (only 11 projects), there is not enough evidence to consider the identified project as an outlier.

Table IV: PROMISE data sets – outliers detection. The numbers identifying the outlier projects represent the order in which they appear in the original data set, starting from one.

Data Set	K	Average Silhouette	Outliers	Number of outliers / Total data set size
Cocomo81	2	0.9778	None	0.00%
Nasa93	2	0.9103	42, 46, 62	3.23%
Nasa	2	0.9070	2, 3	3.33%
Sdr	2	0.9585	9	8.33%
Desharnais	2	0.8367	9, 39, 54	3.70%

6.2. ISBSG Data

The ISBSG repository contains a large body of data about completed software projects. The release 10 contains 5,052 projects, covering many different companies, several countries, organisation types, application types, etc. The data can be used for several different purposes, such as evaluating the benefits of changing a software or hardware development environment; improving practices and performance; and estimation.

In order to produce reasonable SEE using ISBSG data, a set of relevant comparison projects needs to be selected. We preprocessed the data set to use projects that are compatible and do not present strong issues affecting their effort or sizes, as these are the most important variables for SEE. With that in mind, we maintained only projects with:

- Data quality and function points quality A (assessed as being sound with nothing being identified that might affect their integrity) or B (appears sound but there are some factors which could affect their integrity / integrity cannot be assured).
- Recorded effort that considers only the development team.
- Normalised effort equal to total recorded effort, meaning that the reported effort is the actual effort across the whole life cycle.
- Functional sizing method IFPUG version 4+ or NESMA.

— No missing organisation type field. Projects with missing organisation type field were eliminated because we use this field to create different subsets, as explained in the next paragraph.

The preprocessing resulted in 621 projects.

Table V: ISBSG data – organisation types used in the study.

Organisation Type	Id	# Projects
Financial, Property & Business Services	Org1	76
Banking	Org2	32
Communications	Org3	162
Government	Org4	122
Manufacturing; Transport & Storage	Org5	21
Ordering	Org6	22
Billing	Org7	21

After that, with the objective of creating different subsets, the projects were grouped according to organisation type. Only the groups with at least 20 projects were maintained, following ISBSG's data set size guidelines. The resulting organisation types are shown in table V.

Table VI contains additional information about the subsets. As we can see, the productivity rate of different companies varies. A 7-way 1 factor Analysis of Variance (ANOVA) [Montgomery 2004] was used to determine whether the mean productivity rate for all different subsets are equal or not. The factor considered was organisation type, with seven different levels representing each of the organisation types, and each level containing its corresponding projects as the observations. ANOVA indicates that there is statistically significant difference at the 95% confidence interval (p-value $< 2.2e{-}16$).

Table VI: ISBSG subsets.

Id	Unadjusted Function Points				Effort				Productivity			
	Min	Max	Avg	Std Dev	Min	Max	Avg	Std Dev	Min	Max	Avg	Std Dev
Org1	43	2906	215.32	383.72	91	134211	4081.64	15951.03	1.2	75.2	12.71	12.58
Org2	53	499	225.44	135.12	737	14040	3218.50	3114.34	4.5	55.1	15.05	9.94
Org3	3	893	133.24	154.42	4	20164	2007.10	2665.93	0.3	43.5	17.37	9.98
Org4	32	3088	371.41	394.10	360	60826	5970.32	8141.26	1.4	97.9	18.75	16.69
Org5	17	13580	1112.19	2994.62	762	54620	8842.62	11715.39	2.2	52.5	23.38	14.17
Org6	50	1278	163.41	255.07	361	28441	4855.41	6093.45	5.6	60.4	30.52	17.70
Org7	51	615	160.10	142.88	867	19888	6960.19	5932.72	14.4	203.8	58.10	61.63

The next sections explain how the features were selected, how to deal with the missing values and outliers.

6.2.1. Features. The ISBSG suggests that the most important criteria for estimation purposes are the functional size; the development type (new development, enhancement or re-development); the primary programming language or the language type (e.g., 3GL, 4GL); and the development platform (mainframe, midrange or PC). As development platform has more than 40% missing feature values for two organisation types, the following criteria were used as features:

— Functional size.
— Development type.

Table VII: ISBSG subsets – outliers detection. The numbers identifying the outlier projects represent the order in which they appear in the original data set, starting from one.

Id	K	Average Silhouette	Outliers	Number of outliers / Total subset size
Org1	2	0.9961	38	1.32%
Org2	2	0.9074	None	0.00%
Org3	2	0.8911	80, 91, 103, 160	2.47%
Org4	2	0.8956	4, 10, 75, 89, 104	4.10%
Org5	2	0.9734	20	4.76%
Org6	3	0.8821	4	4.55%
Org7	3	0.8898	None	0.00%

— Language type.

The normalised work effort in hours is the dependent variable. Due to the preprocessing, this is the actual development effort across the whole life cycle.

6.2.2. Missing Values. The features "functional size" and "development type" have no missing values. The feature "language type" is missing in several subsets, even though it is never missing in more than 40% of the projects of any subset.

So, an imputation method based on k-Nearest Neighbours (k-NN) was used so that this feature can be kept without having to discard the projects in which it is missing. K-NN imputation has shown to be able to improve SEEs [Cartwright et al. 2003]. It is particularly benefic for this area because it is simple and does not require large data sets. Another method, based on the sample mean, also presents these features, but k-NN has shown to outperform it in two SEE case studies [Cartwright et al. 2003].

According to Cartwright et al. [2003], "k-NN works by finding the k most similar complete cases to the target case to be imputed where similarity is measured by Euclidean distance". When $k > 1$, several different methods can be used to determine the value to be imputed, for example, simple average. For categorical values, vote counting is adopted. Typically, $k = 1$ or 2. As language type is a categorical feature, using $k = 2$ could cause draws. So, we chose $k = 1$. The Euclidean distance considered normalised data sets.

6.2.3. Outliers. Similarly to the PROMISE data sets (section 6.1), outliers were detected through k-means [Hartigan 1975] and eliminated. K was chosen among $k = \{2, 3, 4, 5\}$ based on the average silhouette values. The best silhouette values, their corresponding ks and the projects considered as outliers are shown in table VII. As with the PROMISE data sets, the silhouette values were high (between 0.8821 and 0.9961), showing that the clusters are homogeneous. The number of outliers varied from none to 5, representing always less than 5% of the total number of projects. None of the data sets were reduced to less than 20 projects after outliers elimination.

7. THE RELATIONSHIP AMONG DIFFERENT PERFORMANCE MEASURES

This section presents an analysis of the Pareto solutions with the aim of providing a better understanding of the relationship among MMRE, PRED(25) and LSD (RQ1). All the plots presented here refer to the execution among the thirty runs in which the Pareto ensemble obtained the median test MMRE, unless its test PRED(25) was zero. In that case, the non-zero test PRED(25) execution closest to the median MMRE solution was chosen. This execution will be called *median MMRE run*.

Figure 2 presents an example of Pareto solutions plot for Nasa93. We can see that solutions with better PRED(25) do not necessarily have better MMRE and LSD. The same is valid for the other performance measures. For example, a solution with relatively good MMRE may have very bad LSD, and a solution with good LSD may have very bad PRED(25). This demonstrates that model choice or creation based solely on one performance measure may not be ideal. In the same way, choosing a model based solely on MMRE when the difference in MMRE is statistically significant [Menzies et al. 2006] may not be ideal.

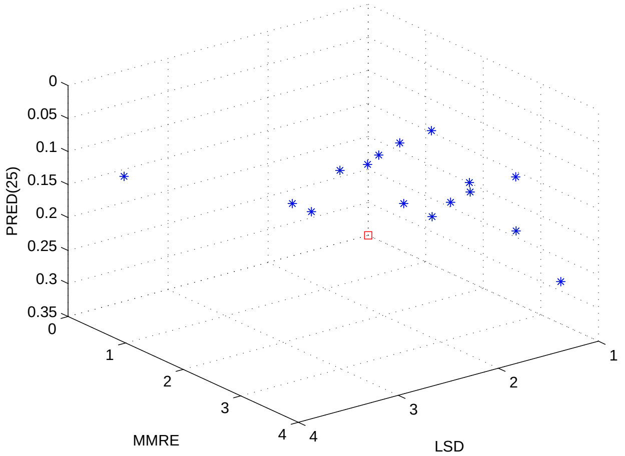


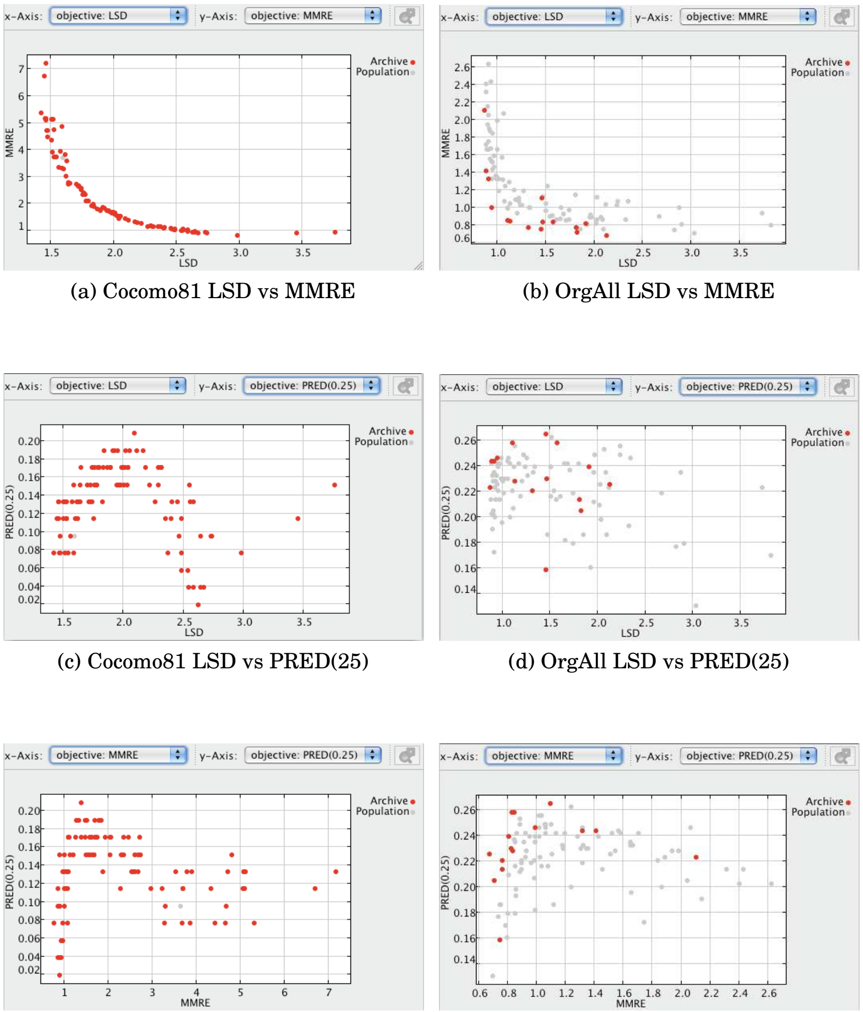
Fig. 2: An example plot of "Pareto solutions" (nondominated solutions in the last generation) for Nasa93. The red square represents the best position that can be plotted in this graph.

In order to better understand the relationship among the performance measures, we plotted graphs LSD vs MMRE, LSD vs PRED(25) and MMRE vs PRED(25) for the median MMRE run, for each PROMISE data set and for ISBSG OrgAll. Figure 3 shows representative plots for cocomo81 and orgAll. Other figures were omitted due to space restrictions and present the same tendencies. It is worth noting that, even though some Pareto solutions have worse performance than other solutions considering the two measures in the plots, they are still nondominated when all three objectives are considered. All the three objectives have to be considered at the same time to determine whether a solution is (non)dominated.

Considering LSD vs MMRE (figures 3a and 3b), we can see that as MMRE is improved (reduced), LSD tends to get worse (increased). This tendency is particularly noticeable for cocomo81, which contains more solutions in the Pareto front.

Considering LSD vs PRED(25) (figures 3c and 3d), we can see that solutions with similar PRED(25) frequently present different LSD. The opposite is also valid: solutions with similar LSD frequently present different PRED(25). As PRED(25) is the percentage of estimations within 25% of the actual effort, one would expect several solutions with different LSD to have the same PRED(25), as a big improvement is necessary to cause impact on PRED(25). The opposite is somewhat more surprising. It indicates that average LSD by itself is not necessarily a good performance measure and may be affected by a few estimations containing extreme values.

A similar behaviour is observed in the graphs MMRE vs PRED(25) (figure 3e and 3f), but it is even more extreme in this case: solutions with even more different MMRE present the same PRED(25).



(a) Cocomo81 LSD vs MMRE

(b) OrgAll LSD vs MMRE

(c) Cocomo81 LSD vs PRED(25)

(d) OrgAll LSD vs PRED(25)

(e) Cocomo81 MMRE vs PRED(25)

(f) OrgAll MMRE vs PRED(25)

Fig. 3: Plot of solutions in the last generation according to two of the three objectives. Points in red (dark points) represent the Pareto solutions. Points in light grey represent the other solutions in the population.

Overall, the plots show that, even though a certain solution may appear better than another in terms of a certain measure, it may be actually worse in terms of the other measures. As none of the existing performance measures has a perfect behaviour, the software manager may opt to analyse solutions using several different measures, instead of basing decisions on a single measure. For instance, s/he may opt for a solution which behaves better considering most performance measures. The analysis also shows that MMRE, PRED(25) and LSD behave differently, indicating that they may be useful for creating SEE ensembles. This is further investigated in section 8.

Moreover, considering this difference in behaviour, the choice of a solution by a software manager may not be easy. If the software manager has a reason for emphasizing a certain performance measure, s/he may choose the solution more likely to perform best for this measure. However, if it is not known what measure to emphasize, s/he may be interested in a solution which provides a good trade-off among different measures. We show in section 8 that MOEA can be used to automatically generate an ensemble that provides a good trade-off among different measures, so that the software manager does not necessarily need to decide on a particular solution or performance measure. Our approach is also robust, allowing the software manager to emphasize a certain measure should s/he wish to, as shown in section 9.

8. ENSEMBLES BASED ON CONCURRENT OPTIMISATION OF PERFORMANCE MEASURES

This section concentrates on answering RQ2. As explained in section 7, different measures behave differently, indicating they may provide a natural way to generate diverse models to compose SEE ensembles. In this section, we show that the best fit MOEA Pareto solutions in terms of each objective can be combined to produce an SEE ensemble (Pareto ensemble) which achieves good results in comparison to a traditional algorithm that does not consider several different measures explicitly. So, the main objective of the comparison presented in this section is to analyse whether MOEA can be used improve the performance over the non-use of MOEA considering the same type of base models. Comparison against other types of models is shown in section 11.

The analysis is done by comparing the Pareto ensemble of MLPs created by the MOEA as explained in section 4 against MLPs trained using Backpropagation [Bishop 2005], which is the learning algorithm most widely used for training MLPs. Each best fit solution used to create the Pareto ensemble is the one with the best train performance considering a particular measure. The Pareto ensemble represents a good trade-off among different performance measures, if the software manager does not wish to emphasize any particular measure.

Firstly, let's analyse the test performance average considering all data sets. Table VIII shows the average test performance and standard deviation for the Pareto ensemble and the Backpropagation MLP. The cells in light grey represent better averages (not necessarily statistically different). The table also shows the overall average and p-value of the Wilcoxon test used for the statistical comparison of all runs over multiple data sets. P-values less than 0.0167 (shown in dark grey) indicate statistically significant difference using Bonferroni corrections considering the three performance measures at the overall level of significance of 0.05. As we can see, the two approaches are statistically the same considering the overall MMRE, but different when considering LSD and PRED(25). In the latter case, the Pareto ensemble wins in 7 out of 13 data sets considering LSD and PRED(25), as shown by the light grey cells. This number of wins is similar to the number of losses, so additional analysis is necessary to better understand the Pareto ensemble's behaviour and check if it can be improved, as shown in the next paragraphs.

Hence, secondly, if we take a closer look, we can see that Backpropagation MLP frequently wins for the smallest data sets, whereas the Pareto ensemble tends behave better for the largest data sets. Considering LSD, the Pareto ensemble wins in 6 out of 8 large data sets. Considering MMRE, it wins in 5 out of 8 large data sets. Considering PRED(25), it wins in 7 out of 8 large data sets. So, we performed additional statistical tests to compare the behaviour of all the runs considering the data sets with less than 35 projects (sdr, org2, org5, org6, org7) and with 60 or more projects (cocomo81, nasa93, nasa, desharnais, org1, org3, org4, orgAll) separately.

Table VIIIb shows the overall averages and p-values for these two groups of data sets. We can see that there is statistically significant difference considering all performance measures for the large data sets, including MMRE. That together with the fact that the Pareto ensemble wins in most cases for these data sets indicates that it

Table VIII: Test performance average and standard deviation. The p-values of the Wilcoxon tests for the comparison over multiple data sets are also shown. P-values less than 0.0167 indicate statistically significant difference using Bonferroni corrections at the overall level of significance of 0.05 and are in dark grey.

Data Set	Pareto Ensemble			Backpropagation MLP		
	LSD	MMRE	PRED(25)	LSD	MMRE	PRED(25)
Cocomo81	1.9193+-0.7709	4.2374+-5.3865	0.1600+-0.1354	2.8237+-1.5270	2.7914+-1.6687	0.1300+-0.1179
Sdr	2.0944+-0.8367	8.6331+-19.0015	0.0833+-0.1295	1.5782+-0.4282	1.9254+-0.9617	0.1444+-0.1217
Nasa	1.4005+-0.9419	1.1140+-0.8244	0.3200+-0.1606	1.6676+-1.0735	1.0805+-0.9497	0.4267+-0.1530
Desharnais	1.1515+-1.3872	0.4701+-0.1903	0.4200+-0.1846	1.0291+-1.2157	0.4986+-0.1712	0.3333+-0.1373
Nasa93	1.2518+-0.4061	1.8031+-1.2951	0.1848+-0.1318	2.6262+-1.4871	1.7866+-0.8316	0.1970+-0.0927
Org1	1.1560+-1.3867	0.8739+-0.4551	0.2967+-0.1564	1.8601+-2.2647	1.2233+-0.9439	0.2600+-0.1653
Org2	0.8120+-0.2751	1.6575+-3.5308	0.2633+-0.1402	1.2455+-1.0130	1.0554+-0.3322	0.2200+-0.1186
Org3	0.7076+-0.2120	0.9449+-1.4016	0.2909+-0.1338	1.8770+-2.3732	3.2484+-9.6361	0.2333+-0.1485
Org4	1.1074+-1.1572	0.8596+-0.2613	0.2121+-0.1271	2.0374+-2.6682	1.1014+-0.6457	0.1848+-0.1157
Org5	6.4777+-5.4380	5.4348+-14.5110	0.2033+-0.0964	3.0275+-4.1660	1.2569+-0.7010	0.2133+-0.1106
Org6	1.4703+-1.5488	1.1507+-0.6037	0.1833+-0.1053	0.8470+-0.1909	1.0493+-0.4745	0.2067+-0.0944
Org7	2.1753+-1.5659	4.3080+-5.2502	0.1633+-0.1129	1.9786+-3.2286	1.3280+-0.8346	0.1833+-0.0913
OrgAll	1.1025+-0.9509	0.7313+-0.2866	0.2641+-0.1350	2.6782+-2.7064	2.0317+-2.5790	0.1744+-0.1068
Average	1.7559	2.4783	0.2343	1.9443	1.5675	0.2236
P-value	0.0076	0.2168	0.0036			

(a) Pareto ensemble vs Backpropagation MLP.

Data Set	Pareto Ensemble			Backpropagation MLP		
	LSD	MMRE	PRED(25)	LSD	MMRE	PRED(25)
Large	1.2246	1.3793	0.2686	2.0749	1.7202	0.2424
P-value	0.0000	0.0012	0.0003			
Small	2.6059	4.2368	0.1793	1.7354	1.323	0.1936
P-value	0.0001	0.0444	0.8449			

(b) Pareto ensemble vs Backpropagation MLP considering large and small data sets separately.

Data Set	Pareto Ensemble			Backpropagation MLP		
	LSD	MMRE	PRED(25)	LSD	MMRE	PRED(25)
Sdr	2.1160+-0.7330	4.1618+-2.7773	0.0944+-0.1132	1.5782+-0.4282	1.9254+-0.9617	0.1444+-0.1217
Org2	0.9022+-0.7834	0.5966+-0.2501	0.2833+-0.1416	1.2455+-1.0130	1.0554+-0.3322	0.2200+-0.1186
Org5	3.4037+-4.1020	1.5130+-1.6793	0.2833+-0.1234	3.0275+-4.1660	1.2569+-0.7010	0.2133+-0.1106
Org6	0.8075+-0.2065	0.7688+-0.3216	0.2800+-0.1126	0.8470+-0.1909	1.0493+-0.4745	0.2067+-0.0944
Org7	1.8411+-1.2749	1.9268+-1.7951	0.1700+-0.1088	1.9786+-3.2286	1.3280+-0.8346	0.1833+-0.0913
Average	1.8141	1.7934	0.2222	1.7354	1.3230	0.1936
P-value	0.1170	0.7166	0.0004			

(c) Pareto ensemble using adjusted parameters (reduced population size and no backpropagation) vs Backpropagation MLP for small data sets.

is likely to perform better than Backpropagation MLPs for large data sets. That is a very good achievement, especially considering that the Backpropagation MLPs were very well tuned for providing good MMRE. For the small data sets, the two approaches obtained statistically significantly different LSD and the Pareto ensemble was worse in 4 out of 5 small data sets. The two approaches were statistically equal in terms of MMRE and PRED(25) for the small data sets.

A possible reason for the worse behaviour for the smallest data sets is overfitting. To make this hypothesis more well grounded, we checked the MMREs obtained by each MLP of the Pareto ensemble for the median MMRE run of Org5. Org5 was chosen for being the data set in which the Pareto ensemble obtained the worst test MMRE, but not the one with the smallest number of projects, as only 12 projects may be too little for a significant analysis. The test MMREs were 1.9396, 1.9217 and 1.9613, whereas the corresponding train MMREs were 0.0998, 0.0976 and 0.1017, respectively. We can see that the test MMREs were drastically larger than the train MMREs. On the other hand, the three runs on and around the median test MMRE for the Backpropagation MLPs obtained 1.0403, 1.0433 and 1.1599 test MMRE, whereas the train MMREs were 0.3183, 1.0857 and 0.1176. So, the train MMREs were higher (worse) for the Backpropagation MLPs than for the best fit Pareto MLPs. That indicates that the MOEA may indeed be overfitting when the data sets are too small. Besides the fact that learning is inherently hard due to the small number of training examples, a possible cause for that is the parameters choice, as the parameters were not so well tuned for the MOEA.

So, thirdly, additional MOEA runs were performed in such a way to use the essence of the early stopping strategy to avoid overfitting [Finnoff et al. 1993]. This was done by reducing the population size from 100 to 30 and the number of epochs for the offspring Backpropagation from 5 to zero (no Backpropagation). The results are shown in table VIIIc. As we can see, the Pareto ensemble obtained similar LSD and MMRE to Backpropagation MLPs. PRED(25) was statistically significantly different and the Pareto ensemble won in 3 out of 5 data sets. The improvement in the overall average was of about 0.79 for LSD, 2.44 for MMRE and 0.04 for PRED(25). The test MMREs for the median MMRE run of the Pareto ensemble for Org5 were 0.8649, 0.8528 and 1.4543, whereas the corresponding train MMREs were 0.3311, 0.3116 and 0.5427. These train MMREs are closer to the corresponding test MMREs than when using the previous parameters configuration, indicating less overfitting.

Lastly, in order to check the performance of the Pareto ensemble for outlier projects, we have also tested it using test sets composed only of the outliers identified in section 6. The MMRE and PRED(25) obtained for the outlier sets were compared to the original test sets'. It is not possible to compare LSD because the number of outliers is too small and LSD's equation is divided by the number of projects minus one. So, even if the error obtained for the outlier sets is potentially smaller, the small number of examples increases LSD.

The results show that PRED(25) was always worse for the outlier sets than for the original test sets. That means that these outliers are projects to which the Pareto ensemble has difficulties in predicting within 25% of the actual effort. However, the MMRE is actually better for 3 out of 9 data sets that involve outliers. So, in about a third of the cases, the outliers are not projects to which the Pareto ensemble obtains the worst performances. For the other cases, the MMRE was usually less than 0.27 higher.

The analysis performed in this section shows that the use of MOEA for considering several performance measures at the same time is successful in generating SEE ensembles. The Pareto ensemble manages to obtain similar or better performance than Backpropagation MLP across data sets in terms of all the measures used as objectives by the MOEA. So, it is worth considering the creation of SEE models as a multi-objective problem and the Pareto ensemble can be used when none of the performance measures is to be emphasized over the others.

9. EMPHASIZING PARTICULAR PERFORMANCE MEASURES

This section concentrates on answering RQ3. As shown in section 8, MOEA can be used to automatically generate an ensemble that provides an ideal trade-off among different measures, so that the software manager does not need to decide on a particular solution or performance measure. The main objective of the comparison presented in the present section is to analyse whether we can further increase the test performance for each measure separately if we wish to emphasize this particular measure. This section provides a better understanding of the solutions that can be produced by the

MOEA and shows that our approach is robust to the case where the manager wishes to emphasize a certain measure. In order to do so, we check the performance of the best fit solution considering each objective separately. The best fit solution is the one with the best train performance considering a particular measure.

Tables IXa and IXb show the results of the comparisons between the best fit Pareto MLP in terms of each objective versus the Pareto ensemble for large and small data sets. The MOEA parameters for the small data sets are the ones with population size 30 and no Backpropagation. Dark grey color means statistically significant difference using Wilcoxon tests with Bonferroni corrections at the overall level of significance of 0.05 (statistically significant difference when p-value $< 0.05/(3 * 3)$). The number of times in which each approach wins against the Pareto ensemble is also shown and is in light grey when the approach wins more times.

Table IX: Test performance average of best fit Pareto MLPs against the Pareto ensemble. The adjusted MOEA parameters were used for the small data sets. The p-values of the Wilcoxon tests for the comparison over multiple data sets are also shown. P-values less than 0.0056 (in dark grey) indicate statistically significant difference using Bonferroni corrections at the overall level of significance of 0.05.

Data Set	LSD	MMRE	PRED(25)
Pareto ensemble	1.2246	1.3793	0.2686
Best LSD MLP avg	1.1356	2.1518	0.2587
Wins vs Pareto ens	6	0	3
P-value	0.0034	0.0000	0.3062
Best MMRE MLP avg	2.0070	0.8256	0.1974
Wins vs Pareto ens	0	7	0
P-value	0.0000	0.0000	0.0000
Best PRED MLP avg	1.4775	1.5084	0.2680
Wins vs Pareto ens	2	1	4
P-value	0.0000	0.0000	0.9240

(a) For large data sets.

Data Set	LSD	MMRE	PRED
Pareto ensemble avg	1.8141	1.7934	0.2222
Best LSD MLP avg	1.8432	2.5741	0.1913
Wins vs Pareto ens	3	0	2
P-value	0.1326	0.0000	0.0075
Best MMRE MLP avg	2.1598	1.3511	0.2113
Wins vs Pareto ens	0	5	3
P-value	0.0000	0.0000	0.5068
Best PRED MLP avg	2.0739	1.9623	0.2178
Wins vs Pareto ens	0	1	1
P-value	0.0000	0.0037	0.5438

(b) For small data sets.

The results of these comparisons indicate that using each best fit Pareto MLP considering a certain objective can sometimes improve the test performance considering this objective. Even though the test performance to be emphasized becomes equal or better than the Pareto ensemble, as shown by the statistical test and the number of wins, the performance considering the other measures gets equal or worse. It is worth noting, though, that the best fit Pareto MLPs are still nondominated solutions, providing acceptable performance in terms of all the measures in comparison to other solutions generated by the MOEA.

In addition to the statistical comparison, we are also interested in the effect size Δ (equation 5, section 5) of each best MLP in comparison to the Pareto ensemble, in terms of the performance measure for which the MLP performs best. Table X presents the minimum, maximum, average and standard deviation of the effect size Δ, as well as the number of data sets for which the effect size was small, medium or large. Even though using each best MLP separately can provide some improvement in performance, the improvements are usually small. Nevertheless, as it would be very easy to configure an approach to use the best MLPs instead of the Pareto ensemble, one may wish to use the best MLPs to emphasize a certain performance measure even considering that the improvement in performance is small.

Table X: Effect size for each best MLP against the Pareto ensemble, in terms of the performance measure for which the MLP performs best.

	Min Δ	Max Δ	Avg. Δ	Std. Δ	# Small	# Medium	# Large	# Medium+Large
Best LSD MLP	0.0257	0.3221	0.1395	0.0993	8	0	0	0
Best MMRE MLP	0.0046	0.7983	0.3462	0.3035	4	3	1	4
Best PRED MLP	0.0542	0.2953	0.1699	0.0987	8	0	0	0

(a) Large data sets

	Min Δ	Max Δ	Avg. Δ	Std. Δ	# Small	# Medium	# Large	# Medium+Large
Best LSD MLP	0.0294	0.4468	0.2009	0.1687	4	1	0	1
Best MMRE MLP	0.0567	0.5913	0.2665	0.2080	4	1	0	1
Best PRED MLP	0.0235	0.9502	0.4050	0.3977	2	1	2	3

(b) Small data sets

10. FURTHER ANALYSIS OF THE MODEL CHOICE

The main objective of the comparison presented in this section is to analyse whether our approach still has room for improvement in terms of model choice. If so, as future work, other methods for choosing models for the Pareto ensemble should be investigated with the aim of improving this approach further. In order to make this analysis, we used the best Pareto MLPs according to each test performance. These MLPs represent the best possible choice of solutions generated by the MOEA considering each objective separately. A Pareto ensemble composed of these MLPs was also formed. Tables XIa and XIb show the results of the comparisons. Again, the parameters used by the MOEA for the small data sets here are the ones with population size 30 and no Backpropagation. Wilcoxon tests with Bonferroni corrections at the overall significance level of 0.05 (statistically significant difference when p-value $< 0.05/(3*4)$) were used to aid the comparison.

We can see that there are test performance improvements in almost all cases, both when using the best Pareto MLPs by themselves and when combining them into a Pareto ensemble. In particular, for most results with statistically significant difference in the average LSD, MMRE or PRED(25), the best test performance approach wins more times than the Backpropagation MLPs. This analysis shows that, even though our approach can significantly reduce overfitting by reducing the population size and eliminating offspring Backpropagation for smaller data sets, simply choosing the best fit Pareto MLPs according to the train performance still does not necessarily lead to the best achievable performance. As future work, other strategies to chose models among the Pareto solutions should be investigated to improve the performance even further.

11. COMPARISON AGAINST OTHER TYPES OF MODELS

The main objectives of RQ2 and RQ3 were to show that MOEAs can be used to evolve models considering different performance measures at the same time, being able to

Table XI: Test performance average of best test performing Pareto MLPs against Backpropagation MLP. The adjusted MOEA parameters were used for the small data sets. The p-values of the Wilcoxon tests for the comparison over multiple data sets are also shown. P-values less than 0.0042 (in dark grey) indicate statistically significant difference using Bonferroni corrections at the overall level of significance of 0.05.

Data Set	LSD	MMRE	PRED(25)
Backprop MLP	2.0749	1.7202	0.2424
MLP Best LSD	0.8651	1.363	0.2994
Wins vs Backprop MLP	8	6	7
P-value	0.0000	0.0004	0.0000
MLP Best MMRE	1.4759	0.6423	0.31
Wins vs Backprop MLP	7	8	7
P-value	0.0031	0.0000	0.0000
MLP Best PRED	1.241	1.0735	0.437
Wins vs Backprop MLP	8	6	8
P-value	0.0000	0.0000	0.0000
Pareto ensemble	0.9951	0.9649	0.3272
Wins vs Backprop MLP	7	5	6
P-value	0.0000	0.0000	0.0000

(a) For large data sets.

Data Set	LSD	MMRE	PRED(25)
Backprop MLP	1.7354	1.323	0.1936
MLP Best LSD	1.3761	1.4272	0.2267
Wins vs Backprop MLP	5	2	4
P-value	0.0001	0.0812	0.0001
MLP Best MMRE	1.8198	1.0558	0.2496
Wins vs Backprop MLP	1	4	4
P-value	0.3141	0.0000	0.0000
MLP Best PRED	1.785	1.4286	0.3489
Wins vs Backprop MLP	2	2	5
P-value	0.5729	0.0043	0.0000
Pareto ensemble	1.5671	1.2636	0.2482
Wins vs Backprop MLP	3	2	3
P-value	0.1653	0.0011	0.0000

(b) For small data sets.

produce solutions that achieve good results in comparison to a traditional algorithm that does not use several different measures explicitly to generate the same type of models, besides being flexible to allow emphasizing different performance measures. The analyses explained in sections 8 and 9 answer RQ2 and RQ3. Nevertheless, even though it is not the key point of this paper, it is still interesting to know how well MOEAs to evolve MLPs behave in comparison to other types of model. Differently from the previous analysis, such a comparison mixes the evaluation of the MOEA to the evaluation of the underlying model being evolved (MLP).

In this section, we present a comparison of the Pareto ensemble to several different types of model besides MLPs: RBFs, RTs, EBA with log transformed data, Bagging with MLPs, Bagging with RBFs, Bagging with RTs, Random with MLPs and NCL with MLPs, as explained in section 5. For this comparison, the parameters used by the MOEA on the small data sets were adjusted to population size of 30 and no offspring Backpropagation. The performance measures used to evaluate the models are LSD, MMRE, PRED(25), MdMRE, MAE and MdAE.

The first step of our analysis consists in performing Friedman tests [Demšar 2006] for the statistical comparison of multiple models over multiple data sets for each performance measure. The null hypothesis is that all the models perform similarly according to the measure considered. The tests rejected the null hypothesis for the six performance measures with Holm-Bonferroni corrections at the overall level of significance of 0.05. The Friedman tests also provide rankings of the approaches across data sets, which show that Pareto ensemble, bagging +MLPs, log + EBA and RTs have the top half average ranks considering all measures but LSD. Models based on MLPs, including the Pareto ensemble, tend to be lower ranked considering LSD. A closer analysis of the MLPs revealed that they can sometimes make negative estimations, which have a strong impact on LSD. RTs, on the other hand, can never give negative estimations or estimations close to zero when the training data does not contain such effort values. So, learners based on RTs obtained in general better LSD. Bagging + RTs is the highest ranked approach in terms of both LSD and MAE.

It is important to note that MOEAs could also be used to evolve other types of structure than MLPs. However, the key point of this paper is the investigation of the multi-objective formulation of the problem of creating SEE models and obtaining a better understanding of the performance measures. The use MOEA to evolve other types of model such as RTs, which could improve LSD, is proposed as future work.

It is also interesting to verify the standard deviation of the Friedman ranking across different performance measures. The Pareto ensemble and log + EBA presented the median standard deviation, meaning that their average ranking across data sets does not vary too much when considering different performance measures, even though they are not the approaches that vary the least. Bagging + MLPs presented the lowest and bagging + RTs presented the highest standard deviation.

Nevertheless, simply looking at the ranking provided by the Friedman test is not very descriptive for SEE, as the models tend to behave very differently depending on the data set. Ideally, we would like to use the approach that is most suitable for the dataset in hands. So, as the second step of our analysis, we determine what approaches are ranked first according to the test performance on each data set separately. This is particularly interesting because it allows us to identify on what type of data sets a certain approach behaves better. Table XIIa shows the approaches ranked as first considering each data set and performance measure. Table XIIb helps us to see that the Pareto ensemble appears more often as the first than the other approaches in terms of all measures but LSD. Table XIIc shows that the Pareto ensemble is never ranked last more than twice considering all 13 data sets and it is only ranked worse twice in terms of MMRE. For the reason explained in the previous paragraphs, approaches based on MLPs are rarely ranked first in terms of LSD, whereas bagging + RTs and RTs are the approaches that appear most often as first in terms of this measure.

Table XIIa also reveals that the Pareto ensemble was first more often for the ISBSG data sets than for the PROMISE data sets. ISBSG data sets are considered as very heterogeneous. So, MOEAs might be particularly useful for more heterogeneous data. A possible reason for that is that the Pareto ensemble uses a global optimisation algorithm, which is usually better at identifying the most promising regions of the search space than local algorithms such as Backpropagation. More heterogeneous data sets may present search surfaces with several peaks, more difficult to tackle by local search algorithms.

As a third step of our analysis, the absolute value of the effect size Δ (equation 5, section 5) using MAE as the performance measure were calculated (table XIII). We report Δ based on MAE because this performance measure is symmetric/unbiased. As we can see, many of the effect sizes are medium or large. So, the choice of a certain approach instead of the Pareto ensemble can have a big impact on the performance of the SEEs.

Finally, it is also worth noting that, in terms of computational time, some approaches such as log + EBA are faster than the MOEA. However, we do not consider the differ-

Table XII: Approaches ranked as first.

Approach	LSD	MMRE	PRED(25)	MdMRE	MAE	MdAE
Cocomo81	RT	Bag+MLP	Bag+MLP	Bag+ MLP	Bag + MLP	Bag + MLP
Sdr	RT	RT	Bag+RT	RT	RT	RBF
Nasa	Bag+RT	RT	Bag+MLP	Bag + MLP	Bag +RT	Bag + RT
Desharnais	Bag+RT	Bag+MLP	Pareto Ens	Pareto Ens	Pareto Ens	Pareto Ens
Nasa93	RT	RT	RT	RT	RT	RT
Org1	Bag+RBF	Pareto Ens	Pareto Ens	Pareto Ens	Pareto Ens	Pareto Ens
Org2	Bag+RT	Pareto Ens	Pareto Ens	Pareto Ens	Pareto Ens	Pareto Ens
Org3	Pareto Ens	Pareto Ens	Log + EBA	Log + EBA	Log + EBA	Log + EBA
Org4	Bag+RBF	Pareto Ens	RT	RT	Pareto Ens	Pareto Ens
Org5	Bag+RT	Log + EBA	Bag+RBF	Rand + MLP	Bag + RT	RT
Org6	Bag+RBF	Pareto Ens	Pareto Ens	Pareto Ens	Bag + RBF	Pareto Ens
Org7	Bag+RT	Log + EBA	Log + EBA	Log + EBA	Bag + RBF	Pareto Ens
OrgAll	RT	Pareto Ens	Pareto Ens	Pareto Ens	Pareto Ens	Pareto Ens

(a) Approaches per data set.

Approach	LSD	MMRE	PRED(25)	MdMRE	MAE	MdAE
Pareto Ens	1	6	5	5	5	7
RT	4	3	2	3	2	2
Bag+RT	5	0	1	0	2	1
Bag+MLP	0	2	2	2	1	1
Log + EBA	0	2	2	2	1	1
Bag+RBF	3	0	1	0	2	0
Rand+MLP	0	0	0	1	0	0
RBF	0	0	0	0	0	1
Total	13	13	13	13	13	13

(b) Total number of times ranked as first per approach. Approaches never ranked as first are omitted. Values higher than 3 are highlighted in grey.

Approach	LSD	MMRE	PRED(25)	MdMRE	MAE	MdAE
Bag + MLP	0	0	0	0	1	0
MLP	1	0	1	0	0	0
RT	0	0	0	1	1	0
Bag + RT	0	1	1	0	0	1
Pareto Ens	1	2	0	1	1	1
Rand + MLP	2	1	1	2	1	1
Bag + RBF	0	3	3	2	0	2
RBF	1	2	4	3	4	3
NCL	8	4	3	4	5	5
Total	13	13	13	13	13	13

(c) Total number of times ranked as last per approach. Approaches never ranked as last are omitted. Values higher than 3 are highlighted in grey.

Table XIII: Effect size for each approach against the Pareto ensemble.

Approach	Min Δ	Max Δ	Avg. Δ	Std. Δ	# Small	# Medium	# Large	# Medium+Large
Bag+MLP	0.0296	0.9194	0.3775	0.2839494668	7	4	2	6
Bag+RBF	0.0046	1.3072	0.5007	0.4233246489	6	3	4	7
Bag+RT	0.0518	1.3383	0.4842	0.4195283097	7	3	3	6
Log + EBA	0.0199	1.4054	0.5333	0.4839488764	7	2	4	6
MLP	0.0202	1.5049	0.4155	0.4034613873	7	3	3	6
NCL	0.0170	2.4236	0.7194	0.6534745379	5	3	5	8
Rand	0.0720	1.5655	0.4615	0.4444167846	7	3	3	6
RBF	0.0393	1.5730	0.6207	0.5299600056	6	2	5	7
RT	0.0392	1.8151	0.5614	0.5203035024	4	6	3	9

ences as of practical significance, because our approach did not present high running time. The reasons for the low running time are (1) SEE data sets are usually small in comparison to other ML applications and (2) our results were obtained without running the MOEA for too many generations. As an example, one run for the largest, the second largest and the third largest data sets, which use 20, 100 and 200 generations, respectively, takes less than 1 minute to finish.

In summary, this section shows that MOEA-evolved MLPs are able to achieve competitive results in comparison to other approaches in the literature. The Pareto ensemble usually obtained comparatively good results in terms of all measures but LSD. Whether this is acceptable is problem (project) dependent and also depends on the magnitude of loss in terms of LSD compared to the gain in other measures. Moreover, the Pareto ensemble seems to perform better for more heterogeneous data sets such as ISBSG.

12. CONCLUSIONS

This paper proposes and demonstrates how to view the problem of creating SEE models through a different perspective, by tackling it as a multi-objective problem that considers different performance measures explicitly and simultaneously. Using a MOEA to solve this problem allows us to better understand different performance measures and to produce SEE models with good overall performance in comparison to a model that does not consider these measures explicitly.

As an answer to RQ1, we show that LSD and MMRE are performance measures with somewhat opposite behaviour. Moreover, models with similar LSD/MMRE are likely to present different PRED(25) and vice-versa. So, one may wish to choose a model that does not behave particularly bad for any intended performance measures. This is one of the motivations for our approach to use MOEAs, as MOEAs consider all these performance measures at the same time. This difference in behaviour also provides the second motivation for our approach: it can produce diverse ensembles, which are more likely to have increased performance.

The results above indicate that considering different performance measures explicitly when creating a model may be used to produce good ensembles. As an answer to RQ2, we show that indeed a MOEA can be used to create models by explicitly considering different performance measures at the same time. Pareto ensemble of MLPs produced by a MOEA generally obtained similar or better performance than Backpropagation MLPs considering both LSD, MMRE and PRED(25). The average performances were generally better for the data sets with 60 or more projects.

As an answer to RQ3, we show that MOEAs are also flexible, allowing the software manager to choose models that emphasize certain performance measures over the others, if s/he desires to do so. A MOEA can be used to produce and choose models that emphasize a particular measure without completely ignoring the performance using other measures, whereas the Pareto ensemble can be used as a good trade-off among measures, if the software manager does not wish to emphasize any particular measure.

As shown in our comparison of the Pareto ensemble against Backpropagation MLPs, which produce the same type of model as the MOEA, the Pareto ensemble has shown to be useful for both single and multi-company data sets. Our comparison against models of a different type than the base models evolved by the MOEA further shows that MOEAs may be particularly useful for more heterogeneous data sets. They can make types of models that would usually not be ranked first in terms of performance become ranked first through the Pareto ensemble.

Our proposed methodology still has room for improvements in terms of the choice of models to compose the Pareto ensemble. The investigation of new strategies is left as future work, as well as the experimentation of our approach using different MOEAs and creating different types of SEE models, and investigation on how much our approach would be affected by outliers in the training set used to build the models.

13. ACKNOWLEDGEMENTS

The authors would like to thank the SEBASE and DAASE project members, and especially Dr. Rami Bahsoon for the useful comments and discussions. This work was supported by EPSRC grants (No. EP/D052785/1 and No. EP/J017515/1).

REFERENCES

AGARWAL, R., KUMAR, M., MALLICK, Y. S., BHARADWAJ, R. M., AND ANANTWAR, D. 2001. Estimating software projects. *Software Engineering Notes 16,* 4, 60–67.

BASKELES, B., TURHAN, B., AND BENER, A. 2007. Software effort estimation using machine learning methods. In *ISCIS'07*. Ankara, 1–6.

BISHOP, C. M. 2005. *Neural Networks for Pattern Recognition*. Oxford University Press, United Kingdom.

BOEHM, B. 1981. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ.

BOEHM, B., ABTS, C., BROWN, A. W., CHULANI, S., CLARK, B. K., HOROWITZ, E., MADACHY, R., REIFER, D. J., AND STEECE, B. 2000. *Software Cost Estimation with COCOMO II*. Prentice-Hall, Englewood Cliffs, NJ.

BRAGA, P. L., OLIVEIRA, A., RIBEIRO, G., AND MEIRA, S. 2007. Bagging predictors for estimation of software project effort. In *IJCNN'07*. Orlando, 1595–1600.

BREIMAN, L. 1996. Bagging predictors. *Machine Learning 24,* 2, 123–140.

BROWN, G., WYATT, J., HARRIS, R., AND YAO, X. 2005. Diversity creation methods: A survey and categorisation. *Information Fusion 6*, 5–20.

CARTWRIGHT, M. H., SHEPPERD, M. J., AND SONG, Q. 2003. Dealing with missing software project data. In *METRICS'03*. Sydney, 154–165.

CHANDRA, A. AND YAO, X. 2006. Ensemble learning using multi-objective evolutionary algorithms. *Journal of Mathematical Modelling and Algorithms 5,* 4, 417–445.

CHEN, H. AND YAO, X. 2009. Regularized negative correlation learning for neural network ensembles. *IEEE TNN 20,* 12, 1962–1979.

CHULANI, S., BOHEM, B., AND STEECE, B. 1999. Bayesian analysis of empirical software engineering cost models. *IEEE TSE 25,* 4, 573–583.

COHEN, J. 1992. A power primer. *Psychological Bulletin 112*, 155–159.

DEB, K., PRATAP, A., AGARWAL, S., AND MEYARIVAN, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE TEC 6,* 2, 182–197.

DEJAEGER, K., VERBEKE, W., MARTENS, D., AND BAESENS, B. 2012. Data mining techniques for software effort estimation: A comparative study. *IEEE TSE 38,* 2, 375–397.

DEMŠAR, J. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research 7*, 130.

DOLADO, J. 2000. A validation of the component-based method for software size estimation. *IEEE TSE 26*, 1006–1021.

DOLADO, J. 2001. On the problem of the software cost function. *IST 43*, 61–72.

FINNOFF, W., HERGERT, F., AND ZIMMERMANN, H. G. 1993. Improving model selection by nonconvergent methods. *Neural Networks 6*, 771–783.

FOSS, T., STENSRUD, E., KITCHENHAM, B., AND MYRTVEIT, I. 2003. A simulation study of the model evaluation criterion mmre. *IEEE TSE 29,* 11, 985–995.

GRUSCHKE, T. M. AND JØRGENSEN, M. 2008. The role of outcome feedback in improving the uncertainty assessment of software development effort estimates. *ACM TOSEM 17,* 4, 20:1–20:35.

HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. 2009. The weka data mining software: An update. *SIGKDD Explorations 11,* 1, 10–18.

HARMAN, M. AND CLARK, J. 2004. Metrics are fitness functions too. In *METRICS'04*. Chicago, 172–183.

HARTIGAN, J. A. 1975. *Clustering Algorithms*. John Wiley & Sons, New York.

HEIAT, A. 2002. Comparison of artificial neural network and regression models for estimating software development effort. *Information and Software Technology 44*, 911–922.

ISBSG. 2011. The International Software Benchmarking Standards Group. `http://www.isbsg.org`.

JORGENSEN, M. AND SHEPPERD, M. 2007. A systematic review of software development cost estimation studies. *IEEE TSE 33,* 1, 33–53.

JRGENSEN, M. AND GRIMSTAD, S. 2011. The impact of irrelevant and misleading information on software development effort estimates: A randomized controlled field experiment. *IEEE TSE 37,* 5, 695–707.

KHARE, V., YAO, X., AND DEB, K. 2003. Performance scaling of multi-objective evolutionary algorithms. In *2nd International Conference on Evolutionary Multi-Criterion Optimization (EMO'03), Lecture Notes in Computer Science*, C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, Eds. Vol. 2632. Springer-Verlag, 376–390.

KOCAGUNELI, E., BENER, A., AND KULTUR, Y. 2009. Combining multiple learners induced on multiple datasets for software effort prediction. In *ISSRE'07*. Mysuru, India, 6p.

KOCAGUNELI, E., MENZIES, T., AND KEUNG, J. 2011. On the value of ensemble effort estimation. *IEEE TSE (inpress)*, 14p.

KULTUR, Y., TURHAN, B., AND BENER, A. 2009. Ensemble of neural networks with associative memory (ENNA) for estimating software development costs. *Knowledge-Based Systems 22*, 395–402.

L. I, . K. AND WHITAKER, C. J. 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning 51*, 181–207.

LEGG, S., HUTTER, M., AND KUMAR, A. 2004. Tournament versus fitness uniform selection. In *Proceedings of the Congress of Evolutionary Computation (CEC)*. 2144–2151.

LIU, Y. AND YAO, X. 1999a. Ensemble learning via negative correlation. *Neural Networks 12*, 1399–1404.

LIU, Y. AND YAO, X. 1999b. Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE TSMC - Part B: Cybernetics 29,* 6, 716–725.

LUKASIEWYCZ, M., GLA, M., REIMANN, F., AND HELWIG, S. 2011. Opt4j: The meta-heuristic optimisation framework for java. `http://opt4j.sourceforge.net`.

MENZIES, T., CHEN, Z., HIHN, J., AND LUM, K. 2006. Selecting best practices for effort estimation. *IEEE TSE 32,* 11, 883–895.

MENZIES, T. AND SHEPPERD, M. 2012. Special issue on repeatable results in software engineering prediction. *ESE 17*, 17:1–17.

MILLER, B. L. AND GOLDBERG, D. E. 1995. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems 9,* 3, 193–212.

MINKU, L. L. May 2011. Machine learning for software effort estimation. The 13th CREST Open Workshop Future Internet Testing (FITTEST) & Search Based Software Engineering (SBSE), `http://crest.cs.ucl.ac.uk/cow/13/slides/presentation_leandro.pdf`, `http://crest.cs.ucl.ac.uk/cow/13/videos/M2U00270Minku.mp4`.

MINKU, L. L. AND YAO, X. 2011. A principled evaluation of ensembles of learning machines for software effort estimation. In *PROMISE'11*. Banff, Canada, 10p.

MINKU, L. L. AND YAO, X. 2012. Ensembles and locality: Insight on improving software effort estimation. *Submitted to IST*, 38p.

MOHAGHEGHI, P., ANDA, B., AND CONRADI, R. 2005. Effort estimation of use cases for incremental large-scale software development. In *ICSE*. St. Louis, Missouri, 303–311.

MONTGOMERY, D. C. 2004. *Design and Analysis of Experiments* 6th Ed. John Wiley and Sons, Great Britain.

PRADITWONG, K., HARMAN, M., AND YAO, X. 2011. Software module clustering as a multi-objective search problem. *IEEE Transactions on Software Engineering 37,* 2, 264–282.

PRADITWONG, K. AND YAO, X. 2006. A new multi-objective evolutionary optimisation algorithm: the two-archive algorithm. In *International Conference on Computational Intelligence and Security (CIS'2006)*. Vol. 1. Guangzhou, China, 286–291.

ROSENTHAL, R. 1994. *The Handbook of Research Synthesis*. Vol. 236. Sage, New York.

SEO, Y.-S., YOON, K.-A., AND BAE, D.-H. 2008. An empirical analysis of software effort estimation with outlier elimination. In *PROMISE*. Leipzig, 25–32.

SHAN, Y., MCKAY, R. J., LOKAN, C. J., AND ESSAM, D. L. 2002. Software project effort estimation using genetic programming. In *ICCCAS & WESINO EXPO*. Vol. 2. Chengdu Sichuan, China, 1108–1112.

SHEPPERD, M. AND SCHOFIELD, C. 1997. Estimating software project effort using analogies. *IEEE TSE 23,* 12, 736–743.

SHIRABAD, J. S. AND MENZIES, T. 2005. The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada, http://promise.site.uottawa.ca/SERepository.

SRINIVAS, N. AND DEB, K. 1994. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation 2*, 221–248.

SRIVASAN, K. AND FISHER, D. 1995. Machine learning approaches to estimating software development effort. *IEEE TSE 21,* 2, 126–137.

TAN, H. B. K., ZHAO, Y., AND ZHANG, H. 2006. Estimating LOC for information systems from their conceptual data models. In *ICSE*. Shanghai, 321–330.

TAN, H. B. K., ZHAO, Y., AND ZHANG, H. 2009. Conceptual data model-based software size estimation for information systems. *ACM TOSEM 19,* 2, 4:1–4:37.

TRONTO, I. F. B., SILVA, J. D. S., AND SANT'ANNA, N. 2007. Comparison of artificial neural network and regression models in software effort estimation. In *IJCNN'07*. Orlando, 771–776.

WANG, Z., TANG, K., AND YAO, X. 2010. Multi-objective approaches to optimal testing resource allocation in modular software systems. *IEEE TR 59,* 3, 563–575.

WITTIG, G. E. AND FINNIE, G. R. 1994. Using artificial neural networks and function points to estimate 4GL software development effort. *Australian J. Information Systems 1,* 2, 87–94.

WITTIG, G. E. AND FINNIE, G. R. 1997. Estimating software development effort with connectionist models. *Information and Software Technology 39*, 469–476.

ZHAO, Y. AND ZHANG, Y. 2008. Comparison of decision tree methods for finding active objects. *Advances in Space Research 41*, 1955–1959.

ZITZLER, E., LAUMANNS, M., AND THIELE, L. 2002. SPEA2: Improving the strength pareto evolutionary algorithm. *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems (EUROGEN'2002)*, 95100.