Genetic Programming for Effort Estimation: an Analysis of the Impact of Different Fitness Functions

Filomena Ferrucci, Carmine Gravino, Rocco Oliveto, Federica Sarro
DMI, University of Salerno, via Ponte don Melillo, 84084 Fisciano (SA), Italy
{fferrucci, gravino, roliveto, fsarro}@unisa.it

***Abstract—Context*: The use of search-based methods has been recently proposed for software development effort estimation and some case studies have been carried out to assess the effectiveness of Genetic Programming (GP). The results reported in the literature showed that GP can provide an estimation accuracy comparable or slightly better than some widely used techniques and encouraged further research to investigate whether varying the fitness function the estimation accuracy can be improved. *Aim*: Starting from these considerations, in this paper we report on a case study aiming to analyse the role played by some fitness functions for the accuracy of the estimates. *Method*: We performed a case study based on a publicly available dataset, i.e., Desharnais, by applying a 3-fold cross validation and employing summary measures and statistical tests for the analysis of the results. Moreover, we compared the accuracy of the obtained estimates with those achieved using some widely used estimation methods, namely Case-Based Reasoning (CBR) and Manual StepWise Regression (MSWR). *Results*: The obtained results highlight that the fitness function choice significantly affected the estimation accuracy. The results also revealed that GP provided significantly better estimates than CBR and comparable with those of MSWR for the considered dataset.**

***Keywords*-Software Development Effort Estimation; Genetic Programming; Empirical Studies.**

I. Introduction

Effort estimation is a critical activity for planning and monitoring software project development and for delivering the product on time and within budget. Significant over or under-estimates can be very expensive for a company and the competitiveness of a software company heavily depends on the ability of its project managers to accurately predict in advance the effort required to develop software systems. Several approaches have been proposed to estimate software development effort. Among them, data-driven approaches exploit data from past projects to estimate the effort for a new project under development [1] [2] [3] [4]. These data consist of information about some relevant factors (named cost drivers) and the effort actually spent to develop the projects. Usually a data-driven method tries to explain the relation between effort and cost drivers building an estimation model (equation) that is used to estimate the effort for a new project. Widely used and studied data-driven approaches are Linear and StepWise Regression (LR and SWR), and Case Based-Reasoning (CBR) [5].

Recently the use of search-based methods has been suggested to address the software development effort estimation problem [6] [7]. Such a problem can be formulated as an optimisation problem where we have to identify the estimation model which provides the best predictions. In the literature some attempts have been reported on the use of Genetic Programming (GP) for building software development effort estimation models [8] [9] [10] [11]. All the studies showed that GP provided models with estimation accuracy comparable with the ones obtained employing some widely used estimation techniques. However, several crucial design choices need to be made when using GP, such as population size, maximum number of generations, genetic operator rates, and fitness function [6] [8]. Special relevance has the choice of the fitness function to guide the search towards a solution able to provide accurate estimates. The common fitness function analysed in the previous studies was based on the Mean Magnitude of Relative Error (MMRE) [12] that represents the most widely used evaluation criterion for assessing the accuracy of a software prediction model. However, it was observed that employing MMRE as fitness function had the effect to degrade a lot of other measures that are usually employed to complement the analysis of the estimation accuracy [8]. Moreover, it has also been argued that the choice of the criterion for establishing the best model can be a managerial issue. In particular, a project manager could prefer to use MMRE as the criterion for judging the quality of the prediction, while another might prefer to use another criterion, just for example Pred(25) [12]. Thus, further research was solicited to analyse which measures is the most appropriate as fitness function.

Based on these considerations we have carried out an empirical analysis to provide an insight on the use of GP for effort estimation and in particular to analyse how the estimation accuracy of GP is affected by the use of different fitness functions. To this end we experimented with different fitness functions based on widely recognised indicators used to evaluate the accuracy of the estimates (i.e., MMRE, MdMRE, Pred(25), MEMRE, and MdEMRE [12] [13]) and combinations of them (i.e., Pred(25) and MMRE, Pred(25) and MdMRE). The empirical study was based on a publicly available dataset, i.e., Desharnais [14], which has been widely and recently used to evaluate and

978-0-7695-4195-2/10 $26.00 © 2010 IEEE
DOI 10.1109/SSBSE.2010.20

compare estimation methods, see e.g., [4] [8] [15] [16] [17]. We also performed a comparison of the effectiveness of GP with widely used estimation methods, namely Manual SWR (MSWR) and CBR. The analysis of the results is based on widely used summary measures (i.e., MMRE, MdMRE, Pred(25), MEMRE, and MdEMRE [12] [13]) and on statistical test. To the best of our knowledge, only summary measures, including MMRE and Pred(25), have been used to compare estimations in the case studies carried out so far with the use of GP [8] [9] [10].

The rest of the paper is organised as follows. Section II introduces GP. The experimental method is described in Section III, while the results are reported in Section IV. Case study validity is discussed in Section V. Section VI reports on related work, while final remarks and directions for future work are presented in Section VII.

II. Genetic Programming

Genetic Programming (GP) [18] belongs to the family of evolutionary algorithms that, inspired by the theory of natural evolution, simulates the evolution of species emphasising the law of survival of the strongest to solve, or approximately solve, optimisation problems. Thus, these algorithms create consecutive populations of individuals, considered as feasible solutions for a given problem, to search for a solution which gives the best approximation of the optimum for the problem under investigation. To this end, a fitness function is used to evaluate the goodness (i.e., fitness) of the solutions represented by the individuals and genetic operators based on selection and reproduction are employed to create new populations (i.e., generations). The elementary evolutionary process of these algorithms is composed by the following steps:

- $\mathbf{S}_1$: a random initial population is generated and a fitness function is used to assign a fitness value to each individual;
- $\mathbf{S}_2$: according to their fitness value some individuals are selected to form the parents and new individuals are created by applying genetic operators (i.e., crossover and mutation). In particular, the crossover operator combines two individuals (i.e., parents) to form one or two new individuals (i.e., offspring), while the mutation operator is employed to randomly modify an individual. Then, to determine who will survive among the offspring and their parents a survivor selection is applied according to the individuals' fitness values;
- $\mathbf{S}_3$: step $\mathbf{S}_2$ is repeated until stopping criteria hold.

With respect to other evolutionary methods, GP is characterised by the fact that individuals are computer programs (e.g., mathematical expressions) usually encoded as a tree where the leaves are terminals (e.g., operands) and the internal nodes are functions (e.g., mathematical operators). The initial population is usually generated building random trees of fixed or variable depth or a combination of them. The crossover and mutation operators are defined exchanging parent subtrees and making random changes in trees, respectively. Note that at each generation these operators are applied with a certain probability, named crossover rate and mutation rate. The stopping criterion for the evolutionary process is usually based on a maximum number of generations. This stopping criterion can be combined with other criteria to reduce the computation time. For example, the search process can be stopped after a certain number of generations or after some number of generations that do not provide an improvement in the fitness value.

III. Case study planning

This section presents the design of the case study we carried out to get an insight in the use of GP for effort estimation. The research goals of our study can be outlined as follows:

- $\mathbf{RG}_1$: Analysing the impact of different fitness functions on the accuracy of the estimation models built with GP.
- $\mathbf{RG}_2$: Comparing the estimates achieved by applying GP with the estimates obtained using widely and successfully employed estimation methods.

To address research goals $\mathbf{RG}_1$ we experimented with several fitness functions as reported and discussed in Section III-B. The second research goal ($\mathbf{RG}_2$) aims to get an insight on the estimation accuracy of GP and understand the actual effectiveness of the technique with respect to other effort estimation methods. For this reason, we first verified whether the estimates obtained with GP were characterised by significantly better accuracy than the simply mean and median of effort of past projects. Indeed, if the investigated estimation method does not outperform the results achieved by using the mean or median effort it cannot be transferred to industry [19] [20]. For a software company it could be more useful to simply use the mean or the median effort of past projects rather than dealing with complex computations of estimation methods. Moreover, we compared the estimations achieved using GP with those obtained by using MSWR [20] and CBR [4] to have other benchmarks to assess the effectiveness of GP.

A. Dataset Selection

To carry out the empirical study we exploited an industrial dataset comprising 81 software projects. This dataset was derived from a Canadian software house by Jean-Marc Desharnais [14]. It is one of the largest, publicly available, datasets and it has been widely and recently used to evaluate estimation methods, see e.g., [4] [8] [15] [16] [17].

Table I reports the description of the eleven variables (nine independent and two dependent) included in the Desharnais dataset. In our analysis we considered as dependent variable the total effort while we did not consider the length of the code [8]. Moreover, we excluded from the analysis four

Table I
PROJECT FEATURES OF THE DESHARNAIS DATASET

Feature	Description	Type
TeamExp	The team experience measured in years	Discrete
ManagerExp	The manager experience measured in years	Discrete
Entities	The number of the entities in the system data model	Discrete
Transactions	The number of basic logical transactions in the system	Discrete
AdjustedFPs	The adjusted Function Points	Continuous
RawFPs	The raw Function Points	Continuous
Envergure	A complex measure derived from other factors defining the environment	Discrete
Language	The language used to develop the system	Categorical
YearEnd	The project year finished	Discrete
Effort	The actual effort measured in person hours (dependent variable)	Discrete
Length	The length of the code (dependent variable)	Discrete

Table II
DESCRIPTIVE STATISTICS OF THE DESHARNAIS DATASET FACTORS

Variable	Min	Max	Mean	Std. Dev.
TeamExp	0	4	2.30	1.33
ManagerExp	0	4	2.65	1.52
Entities	7	386	121.540	86.11
Transactions	9	661	162.940	146.08
AdjustedFPs	73	1127	284.480	182.26
RawFPs	62	1116	282.39	186.36
Envergure	5	52	27.24	8.600
Effort	546	2349	4903.95	4188.19

projects that had missing values. The same choice has been done in other studies (e.g., [4] [17] [15]). Categorical (or nominal) variables (i.e., Language and YearEnd) were also excluded from the analysis, as done in [15].

B. Setting of the Experimented GP-based Method

In this section we present the design choices we made in defining the GP-based estimation method experimented in our case study.

1) Solution Representation: In the context of effort estimation a solution consists of an estimation model described by an equation of this type:

$$\mathit{Effort} = c_1\ op_1\ f_1\ op_2\ ...\ op_{2n-2}\ c_n\ op_{2n-1}\ f_n\ op_{2n}\ C \quad (1)$$

where f_i represents the value of the i^{th} project feature and c_i its coefficient, C represents a constant, while op_i represents the i^{th} mathematical operator of the model. It is worth noting that the equations feasible for the effort estimation problem are those providing positive value for *Effort*.

To encode such a solution we used a binary tree containing features and coefficients as leafs and mathematical operators as internal nodes. In particular, we took into account the following mathematical operators $\{+, -, \cdot, exp, ln\}$.

According to [21] the initial population is generated by building $10V$ random trees of fixed depth, where V is the number of features, aiming at achieving a good compromise between the running time of GP and the accuracy of the estimates.

2) Fitness Function: The fitness function guides the search for the best estimation model. To this end, a suitable fitness function should be able to determine whether an estimation model leads to better predictions than another. In the literature, a large number of different prediction accuracy measures have been proposed. The most widely used are MMRE and Pred(25) [12]. The former is the mean of Magnitude of Relative Error (MRE), where MRE [12] is defined as:

$$\mathit{MRE} = \frac{|\mathit{Effort}_{real} - \mathit{Effort}_{estimated}|}{\mathit{Effort}_{real}} \quad (2)$$

where Effort_{real} and $\mathit{Effort}_{estimated}$ are the actual and the estimated efforts, respectively. MRE is calculated for each project whose effort has to be estimated and MMRE is used to have a cumulative measure of the error. Another cumulative measure widely employed is the Median of MRE (MdMRE) which is less sensitive to extreme values [22].

The *Prediction at level l* – Pred(l) – [12] is another useful indicator that measures the percentage of the estimates whose error is less than *l%* and *l* is usually set at 25. It can be defined as:

$$Pred(25) = \frac{k}{N} \quad (3)$$

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

Kitchenham *et al.* [13] suggest also the use of the Magnitude of Relative Error relative to the Estimate (EMRE). The EMRE has the same form of MRE, but the denominator is the estimate, giving thus a stronger penalty to under-estimates:

$$\mathit{EMRE} = \frac{|\mathit{Effort}_{real} - \mathit{Effort}_{estimated}|}{\mathit{Effort}_{estimated}} \quad (4)$$

As well as for MRE, we can also calculate the mean EMRE (MEMRE) and median EMRE (MdEMRE).

To address research goal $\mathbf{RG}_1$ we experimented with each of the above accuracy measures as fitness function to analyse the impact on the estimation accuracy of the constructed models. Moreover, the observation that different accuracy measures take into account different aspects of predictions accuracy [13] suggested us to investigate also the effectiveness of some combinations of those accuracy measures. In particular, we also experimented with $\frac{Pred(25)}{MMRE}$ and $\frac{Pred(25)}{MdMRE}$ as fitness functions[1].

3) Evolutionary Process: The evolutionary process we experimented with employed two widely used selection operators, i.e., roulette wheel selector and tournament selector [18], whereas the crossover and mutation operators are specific for our solution encoding.

In particular, we used the roulette wheel selector [18] to choose the individuals for reproduction, while we employed the tournament selector [18] to determine the individuals that are included in the next generation (i.e., survivals). The former assigns a roulette slice to each chromosome according to its fitness value. In this way, even if candidate

[1]Thus, if MMRE (MdMRE, MEMRE, or MdEMRE) was used as fitness function the GP goal was to find the solution having the lowest MMRE (MdMRE, MEMRE, or MdEMRE) value. Otherwise, if Pred(25) ($\frac{Pred(25)}{MMRE}$ or $\frac{Pred(25)}{MdMRE}$) was used as fitness function the GP goal was to find the solution having the highest Pred(25) ($\frac{Pred(25)}{MMRE}$ or $\frac{Pred(25)}{MdMRE}$) value.

solutions with a higher fitness have more chance to be selected, there is still a chance that they may be not. On the contrary, using the tournament selector only the best n solutions (usually $n \in [1, 10]$) are copied straight into the next generation.

Crossover and mutation operators were defined to preserve well-formed equations in all offspring. To this end, we used a single point crossover which randomly selects the same point in each tree and swaps the subtrees corresponding to the selected node. Since the two trees are cut at the same point, the trees resulting after the swapping have the same depth as compared to those of parent trees. Concerning the mutation, we employed an operator that selects a node of the tree and randomly changes the associated value. The mutation can affect internal node (i.e., operators) or leaves (i.e., coefficients) of the tree. In particular, when the mutation involves internal node, a new operator $op'_i \in \{\{+, -, \cdot, exp, ln\} \setminus op_i\}$ is randomly generated and assigned to the node, while if the mutation involves a leaf a new coefficient $c'_i \in R$ is assigned to the node. It is worth noting that also the employed mutation preserves the syntactic structure of the equation. Crossover and mutation rate were fixed to 0.5 and 0.1, respectively, since in previous works recommended crossover rate ranged from 0.45 to 0.95 [21] and mutation rate ranged from 0.06 to 0.1 [23].

According to [21] the evolutionary process is stopped after $1000V$ trials, where V is the number of features or if the fitness value of the best solution does not change after $100V$ trials[2].

Since GP does not give the same solution each time it is executed, we performed 10 runs and among the 10 solutions we retained as final prediction model the one that had the fitness value closest to the average value achieved on the training sets in the 10 runs.

C. Validation Method and Evaluation Criteria

In order to verify whether or not a method gives useful estimations of the actual development effort a validation process is required. To this end, we performed a multiple-fold cross validation, partitioning the whole dataset into training sets, for model building, and test sets, for model evaluation. Indeed, when the accuracy of the model is computed using the same dataset employed to build the prediction model, the accuracy evaluation is considered optimistic [5]. Cross validation is widely used in the literature to validate effort estimation models when dealing with medium/small datasets (see, e.g. [1] [2]). In particular, to apply the multiple-fold cross validation, we partitioned the dataset in 3 randomly test sets (one containing 25 observations and two containing 26 observations), and then for each test set we considered the remaining observations as training set to build the estimation model. The three folds are given in Table III to allow for replications of our study.

[2]Since we focused on seven features (see section III-A) we executed GP using a population of 70 (i.e., 10*# features) individuals and the generation process was stopped after 7000 (1000*# features) generations or when the best results did not change after 700 (100*# features) generations.

Table III
THE 3 FOLDS EMPLOYED IN OUR STUDY

	Project Id
Fold 1	11, 19, 24, 77, 01, 26, 78, 02, 05, 12, 22, 23, 35,
(25 observations)	40, 58, 61, 68, 69, 29, 50, 13, 81, 49, 70, 65.
Fold 2	10, 15, 30, 41, 42, 43, 21, 03, 47, 63, 56, 62, 74,
(26 observations)	31, 52, 37, 57, 73, 76, 34, 27, 33, 72, 79, 54, 80.
Fold 3	04, 08, 09, 14, 16, 17, 18, 25, 32, 36, 39, 45, 51,
(26 observations)	53, 55, 59, 60, 67, 71, 06, 07, 20, 28, 46, 48, 64.

Concerning the evaluation of the estimates obtained with the analysed estimation methods, we used several summary measures, namely MMRE, MdMRE, Pred(25), MEMRE and MdEMRE [12][13]. According to [12], a good effort estimation model should have an MMRE less than 0.25, to denote that the mean estimation error should be less than 25%, and a Pred(25) greater than 0.75, meaning that at least 75% of the predicted values should fall within 25% of their actual values. Moreover, we complemented these indicators with the analysis of the boxplots of the absolute residuals, as suggested in [13] [24] [25]. The use of boxplots is widely used in exploratory data analysis since they summarise the data (using five values, i.e., median, upper and lower quartiles, minimum and maximum values, and outliers) through a visual representation [13]. In the context of effort estimation, boxplots are generally used to represent in a visual fashion the amount of the error for a given estimation method. To this end, the spread of the absolute residuals, calculated as $|Effort_{real} - Effort_{estimated}|$, can be graphically rendered.

The analysis of summary measures and boxplots gives only an indication on which is the estimation method that globally gives best effort estimations. In order to establish if one of the estimation methods provides better results than the others it is necessary to test the statistical significance of the obtained results. For this reason we tested the statistical significance of the absolute residuals achieved with different estimation methods [13] [22] [26]. Such an analysis aims at verifying that the estimations of one method are significantly better than the estimations provided by another method. Since (i) the absolute residuals for all the analysed estimation methods were not normally distributed (as confirmed by the Shapiro test [27] for non-normality), and (ii) the data was naturally paired, we used the Wilcoxon Test [28] setting the confidence limit at $\alpha = 0.05$.

IV. ANALYSIS AND INTERPRETATION OF THE RESULTS

The following subsections present and discuss on the results achieved in the empirical study. In subsection IV-C we also compare the results with the ones obtained in the literature exploiting GP on the same dataset.

A. Influence of the Fitness Function

In this section we report the results related to the first research goal and obtained employing different fitness functions, i.e., MMRE, Pred(25), MdMRE, MEMRE, MdEMRE, $\frac{Pred(25)}{MMRE}$, and $\frac{Pred(25)}{MdMRE}$.

To get an insight on the use of these fitness functions, we first analysed the ability of the obtained estimation models to fit data considering the results obtained on the training sets and then we analysed their predictive capability considering the results obtained on the test sets.

Table IV reports on the average summary measures obtained on the training sets. We can observe that the use of MMRE and MdEMRE as fitness functions provided the worst results, whereas the best results were achieved by using Pred(25), MdMRE and $\frac{Pred(25)}{MMRE}$ as fitness function. However, there is no clear winner among them. Furthermore, the use of MEMRE and $\frac{Pred(25)}{MdMRE}$ provided an MMRE value worse than the ones obtained using other fitness functions (e.g., Pred(25)).

Interesting considerations can be made by observing the relationship between the fitness function and the summary measures used to evaluate the estimation model. In particular, the results achieved on the training set (see Table IV) suggest that almost all the fitness functions are able to guide the search to get the best value for the considered summary measure (as highlighted in bold face in Table IV). Indeed, GP with MMRE obtained the best value for MMRE (0.51). This happens also for GP with Pred(25) that gets 0.50 as Pred(25) that is the best value obtained with the considered fitness functions. This does not hold for GP based on MdEMRE and $\frac{Pred(25)}{MdMRE}$ since other fitness functions are able to get better results for MdEMRE and $\frac{Pred(25)}{MdMRE}$ values, respectively. Moreover, the use of some summary measures as fitness functions decreased the value of the other summary measure values. In particular, using MMRE as fitness function, the estimation accuracy in terms of the other summary measures was poor, since MEMRE and MdEMRE values were very high and Pred(25) was very low. This confirms the observation of Burgess of Lefley [8] for MMRE. However, such a phenomenon can be also observed for MEMRE and MdEMRE, whose use determines an increasing of the MMRE value. This does not hold for the use of the other fitness functions (i.e., Pred(25), MdMRE, and $\frac{Pred(25)}{MMRE}$) that were able to provide good fitness value without decreasing so much the other measures. This observation is also confirmed by graphically comparing the trend of the values of summary measures MMRE, Pred(25), MdMRE, MEMRE, and MdEMRE achieved by GP during the evolution process. As an example, in Figure 1 we can observe that using MMRE as fitness function the values of the Pred(25) and MEMRE became worst during the evolution process, while as we can see in Figure 2 this did not happen using the MdMRE as fitness function.

Table IV
RESULTS ON TRAINING SET USING DIFFERENT FITNESS FUNCTIONS

Fitness Function	MMRE	Pred(25)	MdMRE	MEMRE	MdEMRE
MMRE	**0.51**	0.26	0.44	0.82	0.63
Pred(25)	0.68	**0.50**	0.28	0.42	0.31
MdMRE	0.68	0.44	**0.28**	0.39	0.33
MEMRE	0.85	0.42	0.35	**0.36**	0.32
MdEMRE	1.14	0.32	0.57	0.46	**0.34**
$\frac{Pred(25)}{MMRE}$	0.59	0.48	0.31	0.43	0.32
$\frac{Pred(25)}{MdMRE}$	0.75	0.48	0.29	0.43	0.32

Table V
RESULTS ON TEST SET USING DIFFERENT FITNESS FUNCTIONS

Fitness Function	MMRE	Pred(25)	MdMRE	MEMRE	MdEMRE
MMRE	**0.58**	0.23	0.44	0.84	0.63
Pred(25)	0.68	**0.43**	0.33	**0.38**	**0.31**
MdMRE	0.67	**0.43**	**0.32**	**0.38**	0.32
MEMRE	0.91	0.38	0.36	0.39	0.35
MdEMRE	1.32	0.26	0.62	0.47	0.44
$\frac{Pred(25)}{MMRE}$	0.64	0.39	0.33	0.44	0.34
$\frac{Pred(25)}{MdMRE}$	0.87	0.36	0.40	0.47	0.39

Another interesting observation can be made related to the number of iterations (i.e., generations) performed by GP during the evolution process. In particular, we observed that GP was able to find a solution in a relative low number of generations with all the fitness functions. We also observed that the maximum number of generations was rarely achieved and the evolutionary process was generally stopped because the best solution found did not change after a fixed number of generations. In particular, we compared the trend of the fitness value of the best solution obtained with the trend of the average fitness value of the whole population for all the considered fitness functions. The analysis suggested that less than 1,000 iterations are needed to GP to converge. As an example, when MMRE is used as fitness function the analysis highlighted that after about 700-800 generations the two curves were identical indicating that the best solution found cannot be improved. Thus, we can state that the stopping criteria we used is sufficient for GP to converge and the convergence is not influenced by the fitness function employed.

Table V reports the summary measures related to the accuracy achieved by the models constructed by GP with the analysed fitness functions on the test sets. First of all, we can observe that the summary measures were not much more worse than the ones achieved on the training sets. However, we can observe that none of the exploited fitness functions was able to provide summary measure values that satisfy thresholds provided in [12]. Indeed, Pred(25) value was always less than 0.75 and MMRE and MdMRE values were always greater than 0.25. Moreover, we can observe that the predictive capacity of the estimation models is very similar to the ones achieved on the training sets. Indeed, the use of Pred(25), MdMRE, and $\frac{Pred(25)}{MMRE}$ provided the best results, while the worst summary measure were achieved by using MMRE and MdEMRE as fitness functions. Finally, despite the use of MEMRE and $\frac{Pred(25)}{MdMRE}$ provided results similar to Pred(25) except for the MMRE value that was

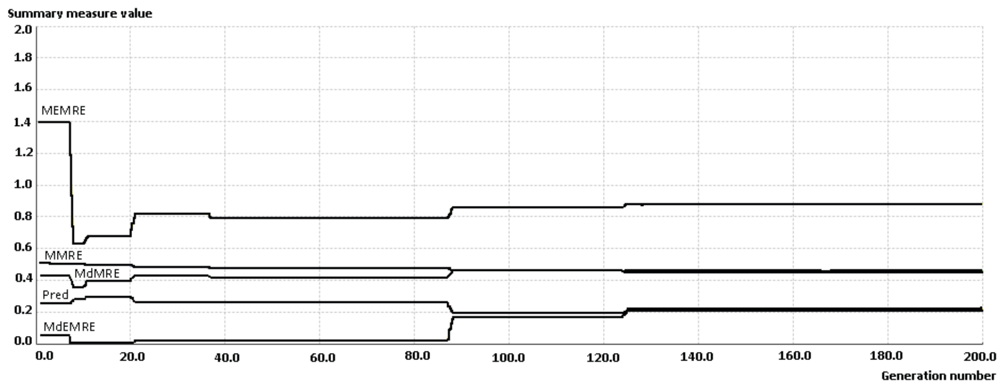

Figure 1. An excerpt of the trend of summary measures when MMRE is used as fitness function.

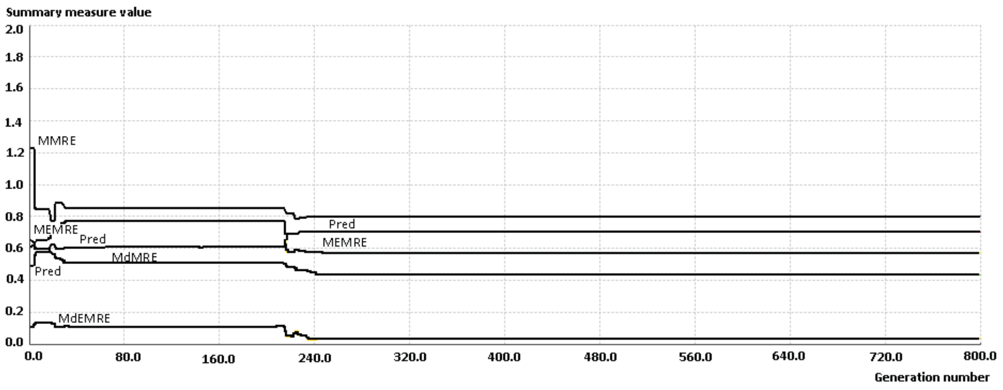

Figure 2. An excerpt of the trend of summary measures when MdMRE is used as fitness function.

worse. As in the case of training sets we also noted that the model employing MMRE (MdEMRE) as fitness function improved the estimation accuracy in terms of MMRE (MdEMRE) but decreased the accuracy in terms of the others summary measures. This did not happen using the other fitness functions (i.e., Pred(25), MdMRE, and $\frac{Pred(25)}{MMRE}$).

The boxplots in Figure 3 confirm the results obtained in terms of summary measures. Indeed, the median of $\frac{Pred(25)}{MMRE}$ is more close to zero than those of MMRE, MEMRE, MdEMRE, and $\frac{Pred(25)}{MdMRE}$. On the other hand, the median of MdMRE and Pred(25) is very close to the one of $\frac{Pred(25)}{MMRE}$. Moreover, even if the box length and tails of MMRE and $\frac{Pred(25)}{MdMRE}$ are close to the ones of Pred(25), MdMRE, and $\frac{Pred(25)}{MMRE}$ they have more outliers that are more far from the box than the ones of Pred(25), MdMRE, and $\frac{Pred(25)}{MMRE}$.

The indications given by summary measures and boxplots are confirmed also by using statistical tests. In particular, we performed the Wilcoxon test to verify the following null hypothesis: "*the use of f_i as fitness function does not provide better results than using f_j*", where f_i and f_j are two experimented fitness functions. The results shown in Table VI reveal that the use of MdEMRE provided the worst accuracy, i.e., all the other fitness functions provided better results than it. Moreover, the use of Pred(25), MdMRE, and $\frac{Pred(25)}{MMRE}$ provided statistically significant better estimates than the use of MMRE, MEMRE, and $\frac{Pred(25)}{MdMRE}$. This is especially interesting taking into account that previous studies on the use of Genetic Programming employed only MMRE as fitness function [8] [10] [16].

Table VI
RESULTS OF THE WILCOXON TESTS COMPARING FITNESS FUNCTIONS

$<$	MMRE	Pred(25)	MdMRE	MEMRE	MdEMRE	$\frac{Pred(25)}{MMRE}$	$\frac{Pred(25)}{MdMRE}$
MMRE	-	0.989	0.986	0.486	**0.032**	0.995	0.735
Pred(25)	**0.011**	-	0.286	**0.001**	**7.6e-5**	0.190	**0.000**
MdMRE	**0.014**	0.715	-	**0.001**	**9.9e-5**	0.201	**0.000**
MEMRE	0.516	0.999	0.999	-	**0.000**	0.984	0.708
MdEMRE	0.969	1	1	0.999	-	0.999	0.999
$\frac{Pred(25)}{MMRE}$	**0.005**	0.811	0.800	**0.016**	**0.001**	-	**0.010**
$\frac{Pred(25)}{MdMRE}$	0.996	0.990	0.999	0.293	**0.003**	0.990	-

B. Comparison with other Effort Estimation Methods

In this section we report the results related to the second research goal and obtained by comparing the estimates provided by GP with the ones provided by Mean, Median, MSWR, and CBR. In particular, since the previous analysis revealed that GP performs better when Pred(25), MdMRE,

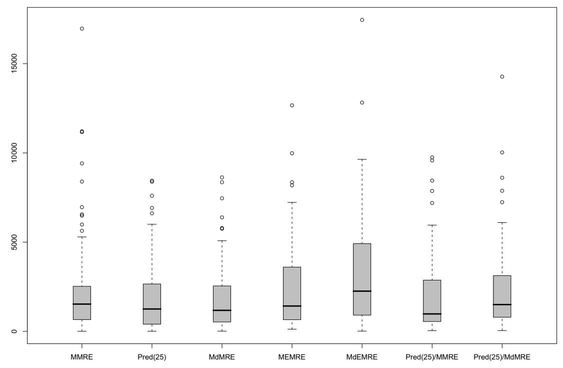

Figure 3. Boxplots of absolute residuals related to the application of the models constructed by GP with the analysed fitness functions on the test sets.

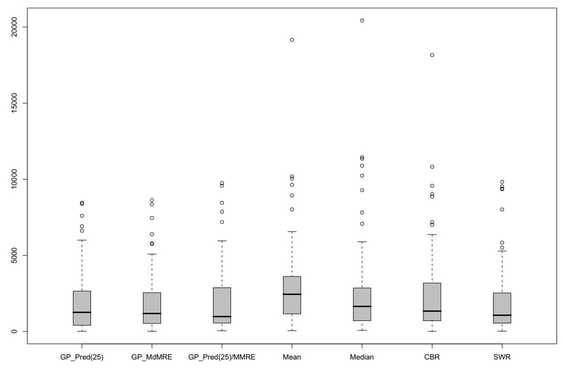

Figure 4. Boxplots of absolute residuals obtained with the analysed estimation methods.

and $\frac{Pred(25)}{MMRE}$ are used as fitness function, we exploited them (reported in the following as $GP_{Pred(25)}$, GP_{MdMRE} and, $GP_{\frac{Pred(25)}{MMRE}}$, respectively) for the comparison.

The analysis of summary measures (see Table VII) suggests that the estimations obtained with GP are better than those achieved by using Mean and Median of effort. The boxplots of absolute residuals shown in Figure 4 confirm these results. Indeed, the median of GP boxplots are more close to zero than the one of the boxplots of Median and Mean of effort. Furthermore, GP boxplots have less ouliers (and less far from their boxes) and their box length and tails are less skewed than those of the boxplots of Mean and Median of effort. Moreover, Table VIII shows the results of the Wilcoxon test to statistically compare the accuracy provided by GP with the accuracy provided by Mean and Median. This test revealed that the absolute residuals obtained with GP are significantly better than those obtained by using the Mean and Median of effort.

Concerning the comparison with MSWR and CBR,

Table VII
COMPARISON BASED ON SUMMARY MEASURES

Method	MMRE	Pred (25)	MdMRE	MEMRE	MdEMRE
$GP_{Pred(25)}$	0.68	0.43	0.33	0.38	0.31
GP_{MdMRE}	0.67	0.43	0.32	0.38	0.32
$GP_{\frac{Pred(25)}{MMRE}}$	0.64	0.39	0.33	0.44	0.34
Mean	1.21	0.22	0.54	0.62	0.51
Median	0.83	0.35	0.41	0.74	0.42
CBR	0.72	0.35	0.42	0.55	0.41
MSWR	0.62	0.40	0.36	0.47	0.38

Table VIII
COMPARISON BASED ON WILCOXON TESTS

$<$	$GP_{Pred(25)}$	GP_{MdMRE}	$GP_{\frac{Pred(25)}{MMRE}}$	Mean	Median	CBR	MSWR
$GP_{Pred(25)}$	-	0.286	0.190	**9.3e-6**	**0.001**	**0.007**	0.100
GP_{MdMRE}	0.715	-	0.201	**1.1e-5**	**0.002**	**0.008**	0.077
$GP_{\frac{Pred(25)}{MMRE}}$	0.811	0.800	-	**1.1e-5**	**0.001**	**0.021**	0.133
Mean	1	1	1	-	0.952	0.999	1
Median	0.998	0.998	0.999	0.050	-	0.976	0.998
CBR	0.993	0.992	0.979	**0.001**	**0.025**	-	0.963
MSWR	0.900	0.923	0.868	**2.3e-5**	**0.002**	**0.037**	-

the analysis of the summary measures (see Table VII) demonstrates that GP (i.e., $GP_{Pred(25)}$, GP_{MdMRE} and, $GP_{\frac{Pred(25)}{MMRE}}$) achieved better results than CBR and comparable results with MSWR. This result is also confirmed by the analysis of boxplots (see Figure 4). In particular, box length and tails of GP boxplots have a median more close to zero than CBR boxplot and very close to the median of MSWR boxplot, while the box length and tails of the GP boxplots are similar to the ones of MSWR and CBR boxplots. Furthermore, we can observe that outliers of the CBR boxplot are more far from its box than those of the other boxplots. As designed we also tested whether there was statistically significant difference between estimates obtained with GP and those obtained with MSWR and CBR. The results suggest that the absolute residuals obtained with GP are significantly less than those achieved with CBR (see Table VIII), while no statistically significant difference was found between the estimates provided by GP and MSWR.

C. Comparison with Burgess and Lefley's Case Study

Burgess and Lefley [8] also assessed the use of GP for estimating software development effort in a case study that exploited the Desharnais dataset [14]. The GP parameters they used are: a population of 1000 individuals, 500 generations, and 10 executions (see Table IX). They exploited only one fitness function designed to minimise MMRE and employed canonical genetic operators. They also used a tree-based representation for the solutions. However, differently from our proposal, during the evolutionary process trees with different depths can be produced. As for cross validation, they employed the hold-out which is the simplest kind of cross validation (i.e., one-fold), where the dataset is split into a training set used to build the estimation model and a test used to validate it. In particular a training set containing 63 observations and a test set containing 18 observations were considered[3]. Thus, they employed also the 4 observations we excluded from the analysis due to the presence of missing

[3]Note that this split is not publicly available.

Table IX
COMPARISON OF THE SETTING OF GP-BASED APPROACHES

	Burgess and Lefley [8]	Our approach
Population size	1000	70
Number of generations	500	¡=7000
Number of executions	10	10
Tree depth	Variable	Fixed
Evolutionary approach	Canonical	Canonical
Fitness function	MMRE	MMRE, Pred(25), MdMRE, MEMRE, MdEMRE $\frac{Pred(25)}{MMRE}$ $\frac{Pred(25)}{MdMRE}$

Table X
RESULTS ACHIEVED BY BURGESS AND LEFLEY [8]

	Method	MMRE	#MRE<0.25	Pred(25)
Burgess and Lefley [8]	Genetic Programming (GP)	0.45	4	0.23
	Linear LSR (LR)	0.46	10	0.56
	2 nearest neighbours (CBR_2)	1.62	8	0.44
	5 nearest neighbours (CBR_5)	1.68	8	0.44

values. However, hold-out procedure can be biased since the prediction performance may heavily depend on how the dataset is split (what are the data points in training set and in test set, respectively). Therefore, in our case study we used a 3-fold cross validation which less suffers of such bias. Moreover, differently from our work, they did not perform statistical tests to verify differences in the distribution of the absolute residuals or MRE values.

Table X shows the average results Burgess and Lefley achieved executing 10 runs of their GP based estimation method. We can observe that their approach obtained an MMRE equals to 0.45 and a Pred(25) equals to 0.23. Even if GP did not outperform LR (Linear Regression) the results encouraged other research in this field. They observed that the use of MMRE degraded the other accuracy measures and suggested that the use of different functions could improve the accuracy of estimations. By using the same dataset but with a different validation method (3-fold vs hold-out) and validation criteria our study has confirmed the observation and the intuition of Burgess and Lefley. Indeed, first of all we have confirmed that MMRE as fitness function is not the best choice, since it allows us to get better MMRE values but not an overall good prediction accuracy as we have shown analysing other summary measures (i.e. Pred, MdMRE, MEMRE, MdEMRE), boxplot of absolute residuals, and statistical significance tests. We have also shown that this behaviour is common to other summary measures investigated in our paper as fitness function (MdEMRE, MEMRE, $\frac{Pred(25)}{MdMRE}$). Moreover, we have identified some accuracy measures (i.e. Pred, MdMRE, and $\frac{Pred(25)}{MMRE}$) that can be more promising as fitness functions since they do not exhibit this problem. Furthermore, we have confirmed the intuition of Burgess and Lefley, since we have identified GP settings with estimation accuracy very close to the one achieved with MSWR and significantly better than those obtained with CBR. Note that we used a different application of linear regression (i.e., MSWR) that is considered more solid [20].

V. VALIDITY EVALUATION

It is widely recognised that several factors can bias the validity of empirical studies. In this section we discuss the validity of the empirical study based on three types of threats, namely *construct*, *conclusion*, and *external* validity.

As highlighted by Kitchenham *et al.* [29], to satisfy construct validity a study has “to establish correct operational measures for the concepts being studied”. This means that the study should represent to what extent the predictor and response variables precisely measure the concepts they claim to measure [22]. Thus, the choice of the features and how to collect them represents the crucial aspects. We tried to mitigate such a threat by evaluating the employed estimation methods on a publicly available dataset [14]. Moreover, since the dataset is publicly available it has been previously used in many other empirical studies carried out to evaluate effort estimation methods, e.g., [4] [8] [15] [17].

Concerning the conclusion validity we carefully applied the statistical tests, verifying all the required assumptions. Moreover, we used a medium size dataset to mitigate the threats related to the number of observations composing the dataset. However, the employed dataset contains projects related to one context that might be characterised by some specific project and human factors, such as development process, developer experience, tools, technologies used, time, and budget constraints [30]. This represents an important external validity threat that can be mitigated only replicating the study taking into account data from other companies, thus getting a generalisation of the results.

VI. RELATED WORK

Besides the work of Burgess and Lefley [8] whose results have been reported and discussed in Section IV-C, some empirical investigations have been performed to assess the effectiveness of GP in estimating software development effort. In particular, GP was employed by Dolado [9] in order to automatically derive equations alternative to multiple linear regression. The aim was to compare the linear equations with those obtained automatically. GP was run a minimum of 15 times and each run had an initial population of 25 equations. Even if in each run the number of generations varied, the best results were obtained with three to five generations (as reported in the literature, usually more generations are used) and by using the Mean Square Error (MSE) [12] as fitness function. As dataset, 46 projects developed by academic students were exploited. It is worth noting that the main goal of Dolado work was not the assessment of GP but the validation of the component-based method for software sizing. However, he observed that GP provided similar or better values than regression equations.

Successively, Shepperd and Lefley [10] also assessed the effectiveness of GP and compared it with several estimation techniques such as LR, ANN (Artificial Neural Networks), and CBR. As for GP setting they applied the same choice of Burgess and Lefley [8] while a different dataset was exploited. This dataset is refereed as "Finnish Dataset" and included 407 observations and 90 features, obtained from many organizations. After a data analysis, a training set of 149 observations and a test set of 15 observations were obtained and used in the empirical analysis. Even if the results revealed that there was not a method that provided better estimations than the others, GP performed consistently well. However, the authors observed that GP and ANN were harder to configure than LR and companies have to weight the complexity of these methods against the small increases in accuracy to decide whether to use it to estimate development effort [10].

An evolutionary computation method, named Grammar Guided Genetic Programming (*GGGP*), was proposed in [11] to fit models, with the aim of improving the estimation of the software development effort. Data of 423 software projects from *ISBSG* (http://www.isbsg.org.au) database were employed to build the estimation models using GGGP and LR. The fitness function was designed to minimize MSE, an initial population of 1000 was chosen, the maximum number of generations was 200 and the number of executions was 5. The results revealed that GPPP performed better than LR in terms of MMRE and Pred(25).

VII. Conclusions and Discussion

The choice of the fitness function represents one of the main critical design choices in the use of GP. This is especially true in the context of effort estimation since it should guide the search to get a model with good estimation accuracy. Unfortunately, not a unique criterion exists to measure such an accuracy and then to define the corresponding fitness function. Indeed, although the identification of "the" software estimation accuracy measure is still an open issue, the research community has agreed that the prediction model accuracy assessment should be based on the comparison of different summary measures (e.g., MMRE, MdMRE, Pred(25), MEMRE, and MdEMRE) as well as on the use of more sophisticated techniques (e.g., boxplots of absolute residuals, statistical tests).

Starting from the observation by Burgess and Lefley [8] that the use of MMRE (the most widely used evaluation criterion) as fitness function could degrade a lot of the other accuracy measures, we carried out an empirical study whose main goal was to analyse how the use of different fitness functions affects the accuracy of GP for effort estimation. We also compared the estimations achieved using GP with those achieved by widely used estimation techniques, such as MSWR and CBR. The experimentation was performed by the Desharnais dataset [14].

The main result achieved in the case study is that the choice of the accuracy measures as fitness function significantly influenced the accuracy of estimations obtained with GP. In particular, the results of Wilcoxon test revealed that the use of Pred(25), MdMRE, and $\frac{Pred(25)}{MMRE}$ as fitness function provided statistically significant better estimates than the use of MMRE, MEMRE, MdEMRE, and $\frac{Pred(25)}{MdMRE}$. This also confirms and extends the observation made by Burgess and Lefley [8] on the use of MMRE as objective function. Nevertheless, the analysis also demonstrated that there are some measures, e.g., Pred(25), that when used as objective functions are able to guide towards estimation model with better accuracy since do not degrade a lot of all the other summary measures.

The idea that several factors should be taken into account by the fitness function suggested us to investigate the use of two combinations of summary measures as fitness function (namely $\frac{Pred(25)}{MMRE}$ and $\frac{Pred(25)}{MdMRE}$). In our case study $\frac{Pred(25)}{MMRE}$ is one of the fitness functions that gave the best results. Nevertheless, the combination only performs better than MMRE but it is not better than Pred(25). On the contrary $\frac{Pred(25)}{MdMRE}$ did not provide good results, maybe it was not able to capture complementary accuracy aspects. In any case, the use of more than one summary measure in the definition of the fitness function deserves to be further investigated by employing more sophisticated combinations and multi-objective optimisation approaches (e.g., the ones based on Pareto optimality) [31]. This will be part of our agenda of future work. Moreover, an empirical analysis could be carried out to verify whether acting on others GP design choice (e.g., population size, generation number, crossover and mutation rates) influences the GP performance aiming at choosing the most appropriate GP setting, as done in [32].

The case study also highlighted that GP performed better than widely used estimation methods. In particular, the results showed that GP provided better results than CBR and MSWR in terms of summary measures. Moreover, GP significantly outperformed CBR.

It is clear that the results presented in the paper should be verified with other datasets. Moreover, the behaviour of other search-based algorithms could be investigated [31], such as Simulated Annealing and Tabu Search, only employed in a preliminary case study [16].

References

[1] L. Briand, K. El. Emam, D. Surmann, I. Wiekzorek, and K. Maxwell, "An assessment and comparison of common software cost estimation modeling techniques," in *Proceedings of International Conference on Software Engineering*. IEEE press, 1999, pp. 313–322.

[2] L. Briand, T. Langley, and I. Wiekzorek, "A replicated assessment and comparison of common software cost modeling techniques," in *Proceedings of International Conference on Software Engineering*. IEEE press, 2000, pp. 377–386.

[3] G. R. Finnie, G. E. Wittig, and J.-M. Desharnais, "A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models," *Journal of Systems and Software*, vol. 39, no. 3, pp. 281–289, 1997.

[4] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Transaction on Software Engineering*, vol. 23, no. 11, pp. 736–743, 2000.

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

[6] M. Harman and B. F. Jones, "Search based software engineering," *Information and Software Technology*, vol. 43, no. 14, pp. 833–839, 2001.

[7] J. Clarke, J. J. Dolado, M. Harman, R. Hierons, B. Jones, M. Lumkin, B. Mitchell, K. Rees, and M. Roper, "Reformulating software engineering as a search problem," *IEE ProceedingsSoftware*, vol. 150, p. 2003, 2003.

[8] C. J. Burgess and M. Lefley, "Can genetic programming improve software effort estimation? a comparative evaluation," *Information and Software Technology*, vol. 43, no. 14, pp. 863–873, 2001.

[9] J. J. Dolado, "A validation of the component-based method for software size estimation," *IEEE Transactions on Software Engineering*, vol. 26, no. 10, pp. 1006–1021, 2000.

[10] M. Lefley and M. J. Shepperd, "Using genetic programming to improve software effort estimation based on general data sets," in *Proceedings of Genetic and Evolutionary Computation Conference*, 2003, pp. 2477–2487.

[11] Y. Shan, R. I. Mckay, C. J. Lokan, and D. L. Essam, "Software project effort estimation using genetic programming," in *Proceedings of International Conference on Communications Circuits and Systems*. IEEE press, 2002, pp. 1108–1112.

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

[13] B. Kitchenham, L. M. Pickard, S. G. MacDonell, and M. J. Shepperd, "What accuracy statistics really measure," *IEE Proceedings Software*, vol. 148, no. 3, pp. 81–85, 2001.

[14] J. M. Desharnais, "Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction," Ph.D. dissertation, Unpublished Masters Thesis, University of Montreal, 1989.

[15] G. Kadoda and M. Shepperd, "Using simulation to evaluate predictions techniques," in *Proceedings of International Software Metrics Symposium*. IEEE press, 2001, pp. 349–358.

[16] F. Ferrucci, C. Gravino, R. Oliveto, and F. Sarro, "Using tabu search to estimate software development effort," in *Proceedings of Mensura 2009*. Lecture Notes in Computer Science 5891 Springer, 2009, pp. 307–320.

[17] M. Shepperd, C. Schofield, and B. Kitchenham, "Effort estimation using analogy," in *Proceedings of International Conference on Software Engineering*. IEEE press, 1996, pp. 170–178.

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

[19] E. Mendes and B. Kitchenham, "A comparison of cross-company and within-company effort estimation models for web applications," in *Proceedings of Conference on Evaluation and Assessment in Software Engineering*, 2004, pp. 47–55.

[20] E. Mendes and B.Kitchenham, "Further comparison of cross-company and within-company effort estimation models for web applications," in *Proceedings of International Software Metrics Symposium*. IEEE press, 2004, pp. 348–357.

[21] S.-J. Huang and N.-H. Chiu, "Optimization of analogy weights by genetic algorithm for software effort estimation," *Journal of Systems and Software*, vol. 48, no. 11, pp. 1034–1045, 2006.

[22] E. Mendes, S. Counsell, N. Mosley, C. Triggs, and I. Watson, "A comparative study of cost estimation models for web hypermedia applications," *Empirical Software Engineering*, vol. 8, no. 23, pp. 163–196, 2003.

[23] H. G. Cobb and J. J. Grefenstette, "Proceedings of the 5th international conference on genetic algorithms, icga, 1993," in *ICGA*. Morgan Kaufmann, 1993, pp. 523–530.

[24] B. Kitchenham, T. Foss, E. Stensrud, and I. Myrtveit, "A simulation study of the model evaluation criterion mmre," *IEEE Transaction on Software Engineering*, vol. 29, no. 11, pp. 985–995, 2003.

[25] I. Myrtveit, M. Shepperd, and E. Stensrud, "Reliability and validity in comparative studies of software prediction models," *IEEE Transactions on Software Engineering*, vol. 31, no. 5, pp. 380–39, 2005.

[26] E. Stensrud and I. Myrtveit, "Human performance estimating with analogy and regression models: an empirical validation," in *Proceedings of International Software Metrics Symposium*. IEEE press, 1996, pp. 205–.

[27] P. Royston, "An extension of Shapiro and Wilk's W test for normality to large samples," *Applied Statistics*, vol. 31, no. 2, pp. 115–124, 1982.

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

[29] B. Kitchenham, L. Pickard, and S. Pfleeger, "Case studies for method and tool evaluation," *IEEE Software*, vol. 12, no. 4, pp. 52–62, 1995.

[30] L. C. Briand and J. Wüst, "Modeling development effort in object-oriented systems using design properties," *IEEE Transaction on Software Engineering*, vol. 27, no. 11, pp. 963–986, 2001.

[31] M. Harman, "The current state and future of search based software engineering," in *Proceedings of Future of Software Engineering - International Conference on Software Engineering*, 2007, pp. 342–357.

[32] V. Garousi, "Empirical analysis of a genetic algorithm-based stress test technique," in *Proceedings of the 10th Conference on Genetic and Evolutionary Computation*. ACM press, 2008, pp. 1743–1750.