Single and Multi Objective Genetic Programming for Software Development Effort Estimation

Federica Sarro
University of Salerno
Via Ponte Don Melillo,
84084 Fisciano (SA), Italy
{fsarro@unisa.it}

Filomena Ferrucci
University of Salerno
Via Ponte Don Melillo,
84084 Fisciano (SA), Italy
{fferrucci@unisa.it}

Carmine Gravino
University of Salerno
Via Ponte Don Melillo,
84084 Fisciano (SA), Italy
{gravino@unisa.it}

ABSTRACT

The idea of exploiting Genetic Programming (GP) to estimate software development effort is based on the observation that the effort estimation problem can be formulated as an optimization problem. Indeed, among the possible models, we have to identify the one providing the most accurate estimates. To this end a suitable measure to evaluate and compare different models is needed. However, in the context of effort estimation there does not exist a unique measure that allows us to compare different models but several different criteria (e.g., MMRE, Pred(25), MdMRE) have been proposed. Aiming at getting an insight on the effects of using different measures as fitness function, in this paper we analyzed the performance of GP using each of the five most used evaluation criteria. Moreover, we designed a Multi-Objective Genetic Programming (MOGP) based on Pareto optimality to simultaneously optimize the five evaluation measures and analyzed whether MOGP is able to build estimation models more accurate than those obtained using GP. The results of the empirical analysis, carried out using three publicly available datasets, showed that the choice of the fitness function significantly affects the estimation accuracy of the models built with GP and the use of some fitness functions allowed GP to get estimation accuracy comparable with the ones provided by MOGP.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management - *Cost Estimation*.

General Terms

Management, Measurement.

Keywords

Genetic Programming, Multi Objective Search, Effort Estimation, Empirical Study.

1. INTRODUCTION

Effort estimation is a critical activity for planning and monitoring software project development and for delivering the product on time and within budget. Several methods have been proposed to address the problem. In particular, data-driven approaches exploit data from past projects, consisting of both factor values that are related to effort and the actual effort to develop the projects, to construct an estimation model that is used to predict the effort for a new project under development [3][28]. In this class, we can include the search-based approaches [15]. These are meta-heuristics able to find optimal or near optimal solutions to problems characterized by large space that have turned out to be effective in solving numerous optimization problems in several contexts. Examples of search-based methods are Simulated Annealing, Tabu Search, Genetic Algorithms, and Genetic Programming [15]. The idea of exploiting these methods to estimate development effort is based on the observation that the effort estimation problem can be formulated as an optimization problem. As a matter of fact, among the possible estimation models, we have to identify the best one, i.e., the one providing the most accurate estimates.

The investigations carried out so far on the use of search-based approaches for effort estimation have mainly focused on the use of Genetic Programming (GP) providing promising results [5][12][13][21][27]. Nevertheless, the design of these techniques deserves to be further explored and empirically analyzed. In particular, a crucial design choice is the definition of the fitness function which indicates how a solution is suitable for the problem under investigation driving the search towards optimal solutions. For the effort estimation problem the fitness function should be able to assess the accuracy of estimation models. It is worth noting that several different accuracy measures have been proposed for assessing the effectiveness/accuracy of effort prediction models. Among them the Mean Magnitude of Relative Error (MMRE) and the Prediction at level 25 (Pred(25)) represent the most widely used measures [8]. Each measure focuses the attention on a specific aspect, as a matter of fact "Pred(25) measures how well an effort model performs, while MMRE measures poor performance" [23].

It could be argued that the choice of the criterion for assessing predictions and establishing the best model can be a managerial issue: a project manager could prefer to use Pred(25) as the criterion for judging the quality of a model, while another might prefer to use another criterion, just for example MMRE [14].

On the other hand, in order to get a more reliable assessment of estimation methods, several evaluation criteria (i.e., MMRE, Pred(25), MdMRE, MEMRE) covering different aspects of models performances (e.g., underestimating or overestimating, success or poor performance) are usually jointly used [9][14][22].

From both points of view search-based methods represent an opportunity. Indeed, they let use as fitness function any measure able to evaluate some properties of interest [16], thus allowing a project manager to select his/her preferred accuracy measure so that the search for the model is driven by such a criterion. On the other end, some search-based techniques have been conceived to address also multi-objective optimization problems where two or more different objectives can be simultaneously considered to guide the search, thus allowing us to take into account several evaluation criteria.

Nevertheless, to the best of our knowledge all the studies carried out so far on search-based techniques for effort estimation exploited only single objective search. Moreover, all those studies used GP with MMRE [5][21] or Mean Square Error (MSE) [12][13][27] as fitness function, except for [14] where some evaluation criteria were considered. This preliminary investigation showed that the employed fitness function can influence the overall accuracy of the models built with GP and some criteria can degrade a lot the performance in terms of the other measures.

Aiming at better understand these effects that should be known to project managers, in this paper we further investigated the use of different fitness functions with GP. Moreover, we exploited a Multi-Objective Genetic Programming (MOGP) based on Pareto optimality [6] to simultaneously optimize all the evaluation measures and analyzed whether MOGP is able to build estimation models more accurate than those obtained using GP. To this end, we experimented GP and MOGP on three publicly available datasets (i.e., Desharnais [11], Finnish [28], and Miyazaki [24]) included in the PROMISE repository [25], performing a 3-fold cross validation and using both summary measures and statistical significance tests as evaluation criteria.

The remainder of the paper is organized as follows. Section 2 provides the description of the employed techniques. Section 3 summarizes the design of the empirical study we performed and discusses the threats to its validity. Results are presented and discussed in Section 4, while related work is reported in Section 5. Final remarks conclude the paper.

2. GENETIC PROGRAMMING FOR EFFORT ESTIMATION

Genetic Programming (GP) [20] belongs to the family of evolutionary algorithms that, inspired by the theory of natural evolution, simulates the evolution of species emphasizing the law of survival of the strongest to solve, or approximately solve, optimization problems. 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). With respect to other evolutionary methods, GP is characterized by the fact that individuals are computer programs (e.g., mathematical expressions) usually encoded as a tree where leaves are terminals (e.g., operands) and internal nodes are functions (e.g., mathematical operators).

The elementary evolutionary process of GP is composed by the following steps:

Step1. [population generation] the initial population is usually generated building random trees of fixed or variable depth or a combination of them;

Step2. [fitness assignment] a fitness function is used to assign a fitness value to each individual;

Step3. [reproduction phase] 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) and evaluated using the fitness function;

Step4. [replacement phase] to determine the individuals that will be included in the next generation (i.e., survivals) a selection based on individual's fitness value is applied;

Step5. [stopping criteria] steps 2, 3, and 4 are repeated until stopping criteria hold.

In the following we detailed the design choices we made for tailoring GP and MOGP to the effort estimation problem.

2.1 Single Objective GP for Effort Estimation

In the context of effort estimation, a solution consists of an estimation model described by an equation of this type:

$$EstimatedEffort = c_i\ op_i\ f_i \ldots\ c_n\ op_{2n-1} f_n\ op_{2n}\ C \qquad (1)$$

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. In particular, the mathematical operators {+,-,*,exp, ln} were took into account. It is worth noting that the equations feasible for the effort estimation problem are only those providing positive value for *EstimatedEffort*.

The initial population (*Step1*) is generated by building m=10V random trees of fixed depth, according to [17] that suggests to use a population of 10V individuals, where V is the number of features, in order to achieve a good compromise between the running time and the accuracy of the estimates.

As for *Step2*, since each measure that has been proposed as a means of evaluating some properties of interest can be used as fitness function [16] one of the measures suggested for the evaluation of estimation model accuracy [8][19] can be employed to this end. The most common ones are based on the Magnitude of Relative Error (MRE) and the Magnitude of Relative Error relative to the Estimate (ERE), defined as:

$$MRE = (|EFreal-EFpred|)/EFreal \qquad (2)$$

$$ERE = (|EFreal-EFpred|)/EFpred \qquad (3)$$

where EFreal and EFpred are the actual and the predicted efforts, respectively. As we can note ERE has the same form of MRE, but the denominator is the estimate, giving thus a stronger penalty to under-estimates. Calculating the MRE and ERE values of each observation in the dataset and aggregating them using the Mean and the Median, gave rise to MMRE and MEMRE, and MdEMRE and MdEMRE, respectively. Such criteria allow a project manager to measure poor performance of an estimation model but can be skewed by a single large mistake [23]. However, MdMRE and MdEMRE are less sensitive to extreme values than MMRE and MEMRE. On the other end, a project manager that would to assess how well an effort model performs can use the Prediction at level l [4] criterion [23]. Indeed, it is is defined as

$$Pred(l) = k/n \qquad (4)$$

where k is the number of observations whose MRE is less than or equal to l, and n is the total number of observations in the validation set. Generally, a value of 25 for the level l is chosen. In other words, Pred(25) is a quantification of the predictions whose error is less than 25%.

Once the fitness is assigned, the Roulette Wheel Selector is employed to select individuals for the reproduction phase. This selector assigns a roulette slice to each chromosome according to its fitness value. In this way, even if candidate solutions with a higher fitness have more chance to be selected, there is still a chance that they may be not. Applying crossover and mutation operators with a certain probability (i.e., crossover rate and mutation rate) to the selected individuals an offspring is produced (*Step3*). As for the genetic operators crossover and mutation operators specific for the solution encoding were employed in order to preserve well-formed equations in all offspring. In particular, a single point crossover which randomly selects in each tree a node placed at the same depth and swaps the subtrees corresponding to the selected point was used. 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, an operator that selects a node of the tree and randomly changes the associated value was employed. 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 $\{\{+,-,*,\exp,\ln\}-op_i\}$ 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 the employed mutation preserves the syntactic structure of the equation. Crossover and mutation rate were fixed to 0.5 and 0.1, respectively. As for the replacement phase (*Step4*) the Tournament Selector is employed, where the chromosomes are ranked by their fitness value and the best *m* ones are copied straight into the next generation.

With regard to the stopping criteria (*Step5*), the evolutionary process is stopped after 1000V, where V is the number of features contained in the dataset or if the best solution does not change after 100V generations [17].

2.2 Multi Objective GP for Effort Estimation

The Multi Objective Genetic Programming (MOGP) we designed is an adaptation to GP of the NSGA-II[1] algorithm [10], based on the concept of Pareto optimality [6]. The employed MOGP is very similar to the GP described in Section 2.1, with crucial variations in Steps 2 and 4. In particular, the main difference respect to GP is that in MOGP an objective vector is considered instead of a single function and the fitness assignment procedure is based on the dominance deep according to NSGA-II [10]. This algorithm decomposes the population into several fronts, as follows:

1. all the solutions are ranked using the non-dominance concept[2] counting;
2. all non-dominated solutions of the population are assigned to rank 1, then they are removed from the population;
3. iteratively, non-dominated solutions are determined and assigned rank 2.

Steps 1-3 are iterated until the population is empty. Then the solutions are ranked again according to a crowding distance, namely the difference between the left and right neighbors or infinity if there are no neighbors. The use of the crowding distance is crucial to preserve the diversity in the solutions fronts, since computing the distance between a given solution and its nearest neighbors allows NSGA-II to approximate the density of the obtained solution. So, solutions with higher crowding distance are considered better solutions, as they introduce more diversity in the population. Once all the solutions are ranked by both dominance deep and crowing distance, the same crossover and mutation operators employed for GP are applied to produce an offspring (*Step3*). Then a tournament selector is applied and the best *m* solutions (in terms of dominance and crowding) are copied straight into the next generation (*Step4*). The algorithm is stopped according to the same criteria used for GP (*Step5*). The final result of the above algorithm is a set of solutions, that are all equivalent in the sense that although each of the elements of the objective vector have different values, no solution in this set (named Pareto front) can be considered superior to any other. To select a final solution from the Pareto front a decision maker is usually used [2][6]. In particular, we employed an “a priori” decision maker [6] which provides a complete order between the Pareto optimal solutions according to the following expression: Pred(25)/(MMRE+MEMRE+MdMRE+MdEMRE).

1 We have chosen NSGA-II since it has been shown that it outperformed simpler algorithms, such as NSGA or VEGA [26]. Furthermore, even if it produces results similar to SPEA2 on some MOPs [29], its time complexity (i.e., $O(n^2)$ [10]) is better than SPEA2 (i.e., $O(n^3)$ [29]).

2 A solution A is said to dominate a solution B if and only if A is at least equal to B in all objectives, and excels B in at least one objective.

3. CASE STUDY PLANNING

This section presents the design of the empirical study we carried out to address the following research questions:

(RQ1) How the choice of the fitness function impact on the accuracy of the estimation models built with GP?

(RQ2) Is MOGP able to build estimation models more accurate than those obtained using GP?

3.1 Datasets

To carry out the empirical study we exploited three publicly available datasets, namely Desharnais, Finnish, and Miyazaki, included in the PROMISE repository [25]. The first one is an industrial dataset comprising 81 software projects derived from a Canadian software house [11]. The Finnish dataset contains industrial data about 38 projects developed by different Finnish companies [28]. Also the Miyazaki dataset is composed by data projects provided by different software companies, comprising data collected from 48 systems in 20 companies by Fujitsu [24]. All these datasets have been widely and recently used to evaluate estimation methods (see e.g., [5][24][28]). The descriptive statistics of the employed features for each dataset are shown in Table 1. We can observe that these datasets are a good sample since they are both single and cross company and differ for number of observations and feature characteristics.

Table 1. Descriptive statistics of the employed datasets

Dataset	Variable	Min	Max	Mean	Std.Dev
Desharnais	TeamExp	0	4	2.30	1.33
	ManagerExp	0	7	2.65	1.52
	Entities	7	387	120.55	86.11
	Transactions	9	886	177.47	146.08
	AdjustedFPs	73	1127	298.01	182.26
	RawFPs	62	1116	282.39	186.36
	Envergue	5	52	27.45	10.53
	Effort	546	23490	4903.95	4188.19
Finnish	HW	1	3	1.26	0.64
	AR	1	5	2.24	1.50
	FP	65	1,814	763.58	510.83
	CO	2	10	6.26	2.73
	Effort	460	25,65	7,678.29	7,135.28
Miyazaki	SCRN	0	281	33.69	47.24
	FORM	0	91	22.38	20.55
	FILE	2	370	20.55	53.56
	Effort	896	253.76	13,996	36,601.56

3.2 Setting of GP and MOGP

To address our first research question we experimented the proposed GP and MOGP using as fitness function the measures described in Section 2.1. This allowed us to analyze how this choice impacts on the estimation accuracy of the constructed models. In particular, we experimented GP with the following fitness functions:

- GP1, that maximizes 1/MMRE as fitness function;
- GP2, that maximizes Pred(25) as fitness function;
- GP3, that maximizes 1/MdMRE as fitness function;
- GP4, that maximizes 1/MEMRE as fitness function;
- GP5, that maximizes 1/MdEMRE as fitness function;

As for MOGP we exploited as objectives all the considered summary measures, i.e., MMRE, Pred(25), MdMRE, MEMRE, MdEMRE. The other settings of the experimented algorithms are reported in Table 2. It is worth mentioning that they reflect in some way the size of the datasets, as suggested in [14][17]. Moreover, we verified that they fit also to our cases comparing the trend of the fitness value of the current best solution with the trend of the average fitness value of the whole population. As an example, using GP with MMRE as fitness function on the Desharnais dataset the analysis highlighted that after about 700-800 generations the two curves were identical indicating that the best solution found cannot be improved. Moreover, we also observed that the evolutionary process was generally stopped because the best solution did not change after a fixed number of generations. Thus, we can be confident that the setting we used is sufficient for the algorithms to converge.

Table 2. Setting of the employed GP and MOGP

Parameter	Desharnais	Finnish	Miyazaki
Population Size	70	40	30
Generation Number	<=7000	<=4000	<=3000
Crossover Rate	0.5	0.5	0.5
Mutation Rate	0.1	0.1	0.1

Finally, to take into account the non determinist nature of GP and MOGP (i.e., they cannot give the same solution each time they are executed), we performed 10 runs and among the ten solutions we retained as final prediction model the one that had objective values closest to the average value achieved in the 10 runs on training sets.

3.3 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 validation sets, for model evaluation. In particular, we partitioned the Desharnais dataset in 3 randomly validation sets (one containing 25 observations and two 26), and then for each validation set we considered the remaining observations as training set. The same procedure was applied also for Finnish and Miyazaki datasets obtaining for Finnish a validation set of 12 observations and two of 13, while for Miyazaki each validation set contains 16 observations.

Concerning the evaluation of the estimates obtained with the analyzed estimation methods we used all the summary measures described in Section 2.1, namely MMRE, MdMRE, Pred(25), MEMRE and MdEMRE. These measures give an indication on which is the method that globally provided the best estimates. To compare different estimation methods we tested also whether there was statistical significant difference among the absolute residuals [19]. Since (i) the absolute residuals for all the analyzed methods were not normally distributed, and (ii) the data was naturally paired, we used the Wilcoxon Test [7] setting the confidence at $\alpha= 0.05$. In particular, we verified the following null hypothesis: "the use of m_i does not provide better results than using m_j", where m_i and m_j are two experimented methods.

3.4 Case Study Validity

It is widely recognized that several factors can bias the validity of empirical studies. In this section we discuss on the validity of the empirical study based on three types of threats: construct validity, related to the agreement between a theoretical concept and a specific measuring device or procedure; conclusion validity, related to the ability to draw statistically correct conclusions; external validity, related to the ability to generalize the achieved results. As highlighted by Kitchenham *et al.* [18], in order 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. Thus, the choice of the features and how to collect them represent the crucial aspects. We tried to mitigate this threat by evaluating the proposed estimation methods on reliable project data coming from a public repository (i.e., PROMISE [25]) and previously used in many other empirical studies carried out to evaluate effort estimation methods (e.g., [5][12][25]). Concerning the conclusion validity we carefully applied the statistical tests, verifying all the required assumptions. Moreover, we used three medium size datasets to mitigate the threats related to the number of observations composing a dataset. Furthermore, each dataset that contains projects related to one context (e.g., the Desharnais dataset) might be characterized by some specific project and human factors, such as development process, developer experience, employed technologies, time, and budget constraints [1]. Thus, using only one dataset could represent an important external validity threat that we mitigated using three datasets which contain projects related to different companies. Taking into account data from different datasets, let us to be more confident in the generalization of the achieved results.

4. RESULTS

Table 3 reports on the average summary measures obtained applying the 3-fold cross-validation with GP and MOGP.

Concerning the application of GP, the results revealed that for all the considered datasets the best overall accuracy (i.e., taking into account all together the summary measures[3]) was achieved by the models built with GP2 and GP3 (i.e., using Pred(25) and MdMRE measures as fitness function, respectively). While the worst overall accuracy was obtained with the models built with GP1, GP4, and GP5 (i.e., using MMRE, MEMRE, and MdEMRE measures as fitness function, respectively). In particular, we observed that using MMRE (MEMRE) as fitness function improved the estimation accuracy in terms of MMRE (MEMRE) but decreased a lot the accuracy in terms of MEMRE (MMRE). These findings are confirmed by the Wilcoxon tests results reported in Table 4 which highlight that GP1, GP4, and GP5

[3] Let us recall that Pred(25) has to be maximized, while the others summary measures have to be minimized.

provided significantly worse results than GP2 and GP3 on the considered datasets. Thus, we can positively answer the research question RQ1, i.e., the choice of the fitness function impacts on the accuracy of the estimation models built with GP. In particular, using each evaluation measure as fitness function GP is able to get better values for the chosen criterion. Nevertheless, some measures (e.g., MMRE or MEMRE) should be carefully used as fitness functions since they can negatively affect the other measures and so the overall accuracy. Others (e.g., Pred(25) or MdMRE) do not exhibit such effects and seem to behave well.

As for the comparison between GP and MOGP, we can observe that in general the summary measures obtained with MOGP are better than the results achieved employing GP1, GP4, and GP5 on all the considered datasets (see Table 3). This is interesting since means that the use of multi-objective optimization allowed us to mitigate the negative influence of MMRE, MEMRE, and MdEMRE when used as single fitness function. Nevertheless, MOGP did not provide better results than GP2 and GP3.

Table 3. Summary measures achieved by GP and MOGP

Technique	MMRE	Pred(25)	MdMRE	MEMRE	MdEMRE
			Desharnais		
GP1	0.58	0.23	0.44	0.84	0.63
GP2	0.68	0.43	0.33	0.38	0.31
GP3	0.67	0.43	0.32	0.38	0.32
GP4	0.91	0.38	0.36	0.39	0.35
GP5	1.32	0.39	0.33	0.44	0.34
MOGP	0.71	0.35	0.34	0.39	0.34
			Finnish		
GP1	0.59	0.16	0.66	2.50	1.65
GP2	1.05	0.21	0.49	0.58	0.59
GP3	1.29	0.18	0.47	0.55	0.56
GP4	1.92	0.24	0.84	0.52	0.49
GP5	1.71	0.21	0.80	0.51	0.51
MOGP	0.92	0.13	0.60	1.02	0.76
			Miyazaki		
GP1	0.53	0.25	0.44	0.85	0.70
GP2	0.58	0.46	0.34	0.40	0.32
GP3	0.52	0.35	0.34	0.54	0.37
GP4	0.53	0.38	0.33	0.49	0.39
GP5	0.50	0.38	0.32	0.56	0.44
MOGP	0.51	0.33	0.33	0.55	0.45

Table 4. Wilcoxon test results comparing GP and MOGP

<	GP1	GP2	GP3	GP4	GP5
			Desharnais		
GP1	-	0.99	0.99	0.77	0.67
GP2	**0.02**	-	0.64	**0.01**	**0.04**
GP3	**0.01**	0.36	-	**0.01**	**0.03**
GP4	0.23	0.99	0.99	-	0.57
GP5	0.33	0.96	0.97	0.43	-
MOGP	**0.02**	0.78	0.63	**0.00**	**0.02**
			Finnish		
GP1	-	0.92	0.97	0.34	0.43
GP2	**0.08**	-	0.10	**0.01**	**0.02**
GP3	0.14	0.90	-	**0.01**	**0.02**
GP4	0.66	0.99	0.99	-	0.95
GP5	0.57	0.98	0.99	**0.04**	-
MOGP	**0.02**	0.95	0.78	0.13	0.24
			Miyazaki		
GP1	-	0.98	1.00	1.00	1.00
GP2	**0.02**	-	**0.08**	0.09	0.09
GP3	**0.00**	0.93	-	0.92	0.71
GP4	**0.00**	0.91	0.08	-	0.06
GP5	**0.00**	0.91	0.30	0.94	-
MOGP	**0.00**	0.92	0.42	0.94	0.65

The indications given by summary measures are confirmed also by using statistical tests. Indeed, the Wilcoxon test results (see Table 4) show that for all the datasets the absolute residuals obtained with MOGP were significantly better than those of GP1. Moreover, MOGP also provided significant better estimations than GP4 and GP5 on the Desharnais datasets, while for Finnish MOGP performed significantly better than GP4.

The above analysis suggests that we can partially positively answer our second research question (i.e., RQ2). Indeed, despite MOGP is able to address the drawback revealed by the use of GP using MMRE or MEMRE as fitness functions, providing better results than GP1, GP4, and GP5, it is not able to get better results of GP2 and GP3. Thus, the increasing of complexity determined by the use of MOGP seems to not be paid back by an improvement of performance respect to the use of GP using certain fitness functions (i.e., Pred(25) or MdMRE).

5. RELATED WORK

In the last years some empirical investigations have been performed to assess the effectiveness of GP in estimating software development effort. In particular, Burgess and Lefley [5] assessed the use of GP exploiting the Desharnais dataset and employing a fitness function designed to minimize MMRE. The average results achieved with GP were better than those obtained with Case-Based Reasoning (CBR) and Artificial Neural Network (ANN) and worse than the ones achieved with Linear Regression (LR). Moreover, the use of MMRE degraded other accuracy measures, thus suggesting the authors the intuition that the use of different functions could improve the effort estimates. In [14] by using the same dataset but with a different validation method (3-fold vs hold-out) the authors confirmed the intuition of Burgess and Lefley [5]. Indeed, MMRE was not the best choice as fitness function, since it allowed for better MMRE values but not an overall good prediction accuracy in terms of all the considered measures. Moreover, the study highlighted that other measures (e.g., Pred(25) and MdMRE) can be more promising as fitness functions since they did not exhibit this problem. The Desharnais and Miyazaki datasets were also used by Dolado [12] together with other datasets, where the employed GP exploited a fitness function designed to minimize the Mean Squared Error (MSE) [8]. The accuracy evaluation was carried out considering only summary measures (i.e., MMRE and Pred(0.25)), whereas it was not reported which kind of validation was employed. The results of their analysis revealed that GP obtained better values for Pred(25) than standard regression, but at the cost of obtaining in some cases slight worse MMRE values. As for comparison, the results on Desharnais dataset were comparable with those we achieved here, while they obtained slightly better estimates on Miyazaki. However, they also employed KLOC, an information that is not available at the time the prediction would be made and that could create a false impression as the efficacy of the prediction method [28]. Dolado and Fernandez [13] compared GP, Neural Networks (NN) and LR with the aim to investigate the use of different methods for the purpose of estimation. They exploited a fitness function designed to minimize MSE [8] and carried out a case study on three publicly available datasets and two datasets collected in their environment [13]. The models accuracy was evaluated only in terms of MMRE and Pred(25) revealing that the best model obtained with GP was better than NN (on all the datasets) and LR (on two out of five datasets). Successively, Lefley and Shepperd [21] also assessed the effectiveness of an evolutionary approach and compared it with several estimation techniques such as LR, ANN, and CBR. As for GP setting, they applied the same choice of Burgess and Lefley [5] but using a different dataset. 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 validation set of 15 observations were obtained applying a hold-out validation and used in the empirical analysis. Even if the results revealed that there was not a method that provides better estimations than the others, the evolutionary approach performed consistently well. An evolutionary computation method, named Grammar Guided Genetic Programming (GGGP), was proposed by Shan *et al.* [27], with the aim of improving the estimation of the software development effort. Data of software projects from ISBSG database was used to build the estimation models using GGGP and LR. The fitness function was designed to minimize MSE. The results revealed that GPPP performed better than Linear Regression in terms of MMRE and Pred(25).

6. CONCLUSIONS

We designed GP and MOGP for estimating software development effort and empirically analyzed the performance on three publicly available datasets. The results showed that the choice of the evaluation criteria employed in the definition of the fitness function affects the overall accuracy of GP. Indeed, using fitness functions based on Pred(25) or MdMRE provided significantly better results than the use of other criteria such as MMRE and MEMRE. GP with Pred(25) and MdMRE was also able to achieve comparable results with respect to those obtained employing a more sophisticated technique such as MOGP using as objectives all the five summary measures. Thus, MOGP seems to be not cost/effective in the context of effort estimation, and GP with Pred(25) or MdMRE could represent suitable choices for project managers. However, these results needs to be deepen also using more data. Furthermore, other multi-objective optimization approaches could be considered to verify whether there are improvements in the built estimation models.

7. REFERENCES

[1] Briand, L., Wust, J. Modeling Development Effort in Object-Oriented Systems Using Design Properties. IEEE Trans. Softw. Engineer. 27(11), 2001, 963–986.

[2] Bowman, M., Briand, L., Labiche, Y. Solving the Class Responsibility Assignment Problem in Object-oriented Analysis with Multi-Objective Genetic Algorithms. IEEE Trans. Softw. Engineer. 36(6), 2010, 817-837.

[3] Briand, L., Wieczorek, I. Software resource estimation. Encyclopedia of Software Engineering. 2002, 1160–1196.

[4] Briand, L., El. Emam, K., Surmann, D., Wiekzorek, I., Maxwell, K. An assessment and comparison of common software cost estimation modeling techniques. In Procs. of Conference on Software Engineering, 1999, 313–322.

[5] Burgess, C., Lefley, M. Can Genetic Programming Improve Software Effort Estimation: a Comparative Evaluation. Inform. and Softw. Technology, 43(14), 2001, 863–873.

[6] Coello, C., Van Veldhuizen, D., Lamont, G. Evolutionary Algorithms for Solving Multi-Objective Optimization Problems. 2000. Kluwer Academic Publishers.

[7] Cohen, J. Statistical power analysis for the behavioral sciences, 1998. 2nd ed. Lawrence Earlbaum Associates.

[8] Conte, D., Dunsmore, H., Shen, V. Software engineering metrics and models. 1996. The Benjamin/Cummings Publishing Company, Inc..

[9] Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Sarro, F., Mendes, E. How Effective is Tabu Search to Configure Support Vector Regression for Effort Estimation?. In Procs. of PROMISE, 2010, ACM NY, 4.

[10] Deb, K., Pratap, A. Apratap, S., Agarwal, S., Meyarivan, T. A fast and elitist multi-objective genetic algorithm: NSGA-II", IEEE Trans. on Evol. Comp., 6(2), 2002, 182-197.

[11] Desharnais, J. Analyse statistique de la productivitie des projets informatique a partie de la technique des point des function. 1989. Master Thesis, University of Montreal.

[12] Dolado, J.. On the problem of the software cost function. Inform. and Softw. Technology 43 (1), 2001, 61-72.

[13] Dolado, J., Fernandez, L.. Genetic programming, neural networks and linear regression in software project estimation. In Procs.of International Conference on Software Process Improvement, Research, Education and Training, 1998, 157-171.

[14] Ferrucci, F., Gravino, C., Oliveto, R., Sarro, F. Genetic Programming for Effort Estimation: an Analysis of the Impact of Different Fitness Functions. In Procs.of International Symposium on Search Based Software Engineering, 2010, 89-98.

[15] Harman, M. The Current State and Future of Search Based Software Engineering. In Workshop on the Future of Software Engineering, 2007, 342-357.

[16] Harman. M., Clark, J. Metrics are Fitness Functions too. In Procs. of International Symposium on Software Metrics, 2004, 58–69.

[17] Huang, S., Chiu, N. Optimization of analogy weights by genetic algorithm for software effort estimation. Journal of Systems and Software 48 (11), 2006, 1034-1045.

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

[19] Kitchenham, B., Pickard, L., MacDonell, S., Shepperd, M. What accuracy statistics really measure. IEEE Procs. Software 148(3), 2001, 81–85.

[20] Koza, J. Genetic Programming, 1992. MIT Press.

[21] Lefley, M., Shepperd, M. Using genetic programming to improve software effort estimation based on general data sets. In Procs. of Genetic and Evolutionary Computation Conference, 2003, 2477–2487.

[22] Mendes, E., Mosley, N. Bayesian Network Models for Web Effort Prediction: A Comparative Study. IEEE Trans. Software Eng., 34(6), 2008, 723-737 .

[23] Menzies, T., Chen, Z., Hihn, J., Lum, K. Selecting best practices for effort estimation. IEEE Trans. on Softw. Engineer. 32 (11), 2006, 883-895.

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

[25] PROMISE Repository of empirical software engineering data, http://promisedata.org/repository.

[26] Schaffer, J., Caruna, R., Eshelman, L., Das, R. A study of control parameters affecting online performance of genetic algorithms for function optimization. In Procs. of. Int. Conf. on Genetic Algorithms and Their Applications, 1989, 51-60.

[27] Shan, Y., Mckay, R., Lokan, C., Essam, D. Software project effort estimation using genetic programming. In Procs. of Conf. on Comms. Circ. and Systems, 2002, 1108–1112.

[28] Shepperd, M., Schofield, C. Estimating software project effort using analogies. IEEE Trans. Softw. Engineer., 23(11), 2000, 736–743.

[29] Zhang, Y., Harman , M., Mansouri, S. The Multi-Objective Next Release Problem. In *Procs. of Genetic and Evolutionary Computation Conference*, 2007, 1129-1136.