Genetic Programming, Neural Networks and Linear Regression in Software Project Estimation

Javier *Dolado*[β] and Luis *Fernández*[ε]
[β]: University of the Basque Country, San Sebastián, Spain; *dolado@si.ehu.es*
[ε]: European University of Madrid, Madrid, Spain; *lufern@dpris.esi.uem.es*

Abstract

The estimations made at the planning phase are critical in providing the managers with accurate information about what is believed to be the course of the project. The improvement in the software process requires improvement in the estimations as an essential prerequisite. The estimation of effort is the most sought after variable to be quantified, since it drives the management process. There are several models in the literature for effort estimation, many of them being derived from statistical regression. New techniques coming from the artificial intelligence field are being introduced, such as neural networks -NN-. Here a new technique named Genetic Programming -GP- is tested in the software estimation process. GP is used as a probabilistic technique for deriving equations, following the paradigm of the "survival of the fittest". The results of GP, NN and linear regression -LR-are compared on several datasets. Furthermore, they are compared on different samples.

1 Estimation in the Software Process

Part of the success of the activities in software management lies in the ability to make accurate predictions. Process improvement depends in part on the improvements achieved in the estimations made at early stages of the software process. These estimations, in turn, depend on the amount and quality of the data gathered. The most common method to make predictions is to identify the underlying relationships among the data by means of linear regression and classical statistics. New methods have appeared attempting to capture the nonlinearities that can exist in the data, such as neural networks, but the relationships can not be represented symbolically. Here, GP will generate the symbolic equations, overcoming some of the problems of NN.

Therefore, we are concerned with the ability to estimate of different, but complementary techniques. Genetic Programming is introduced as a new technique that can help to discover symbolically the nonlinearities of the data. In a second

analysis, every dataset is divided into two subsets, one for model building and other for prediction evaluation. The datasets used are those of Belady, Boehm, Albrecht and Gaffney, Kemerer (all publicly available) and other from our environment. The data relate "project effort" to LOC or to Function Points. In this study, the important thing is not the interpretation of the underlying relationships for extrapolation to other environments, but to understand the use of different methods for the purposes of estimation.

2 The Methods of Analysis

The mathematical methods used to infer relationships in the software estimation process have been varied, the classical technique of *multiple linear regression* being the most common. A comparison of the different methods used for making estimations can be found in [8], [14]. Next, we briefly review those used in the present work.

2.1 Statistical Analysis by Linear Regression -LR-

LR is a technique of classical statistics for model building. The output variable is assumed to be linearly related to the input variables. However, in case of suspecting that other functions can better fit the data, some transformations are allowed in the variables to achieve a linear model. The most common transformations are to take the logarithm of the independent and/or of the dependent variables. The goodness of fit of the model built is usually evaluated by means of the R^2 (and adjusted R^2) and the analysis of variance. Furthermore, when constructing a model some examination has to be made of the effect of outliers on the function estimated. Since building a model for a dataset is different to sampling, the two strategies followed here are: a) building a model with all points; b) building a model on the sample (66% of the data points) and evaluating it on the rest (34% of the points). The main criteria for validation is to obtain acceptable values of PRED(0.25) and MMRE in the validation dataset (34%). The samples have been selected randomly.

2.2 Neural Networks

Artificial neural networks -NN- are nets of processing elements that are able to learn the mapping existent between input and output data [9], [20]. It has been proved that some types of NN are universal approximators, and that a two-level NN is able to approximate any function. They have already been used previously in the software engineering field [7], [20]. The motivation for using NN here is that the equation relating the input variable (LOC of Function Points) and output variable (Effort) can be left unspecified, allowing to model possibly unknown relationships to the estimators. NN in this context act as nonlinear regression models. On the negative side, the manipulation of the mapping learned is very

limited, since it is not possible to meaningfully reason about the weights of the processing elements.

The performance of a NN depends on the architecture and parameters of the net. The NN used here for prediction are two-layer networks with two nodes in the hidden layer. Other structures tested included more layers or more neurons, but they tended to overfit the training data.

2.3 Genetic Programming

GP is an extension of the genetic algorithm technique, originated after the work of Koza [12]. It is included in the set of methods named *evolutionary computation* techniques [2], characterised by the fact that the solution is achieved by means of a cycle of generations of candidate solutions that are pruned by the criteria 'survival of the fittest'. GP has been used in a variety of fields, including the automated synthesis of circuits, nonlinear system identification in chemical process engineering (identifying relevant variables), symbolic regression, etc. [16], [17], [19]. Here, genetic programming is used as an automated symbolic regression to derive the equations.

The name GP comes from the similarities with the paradigm of 'natural selection' in the biological area. There, schematically, populations of specific species evolve according to random mutations of the genes and to the 'appropriateness' to the environment of the species. In the case of system identification (symbolic regression) the idea is also simple: to generate randomly a set of initial equations that relate the input and the output variables and select the equations according to the principle 'survival of the fittest'. The process is repeated until an acceptable solution is obtained.

GP is a nonparametric method since it does not make any assumption about the distribution of the data, and derives the equations according only to fitted values. The use that is made here can be considered as an alternative to Curve Estimation (and Linear Regression), and will also allow us to compare the linear equations with those derived automatically. Since a set different equations are derived in different runs, only those that give the best results in the evaluation data are reported. Parameters of GP have been set heuristically.

The equation reported and compared is the one that gives the best fit in the last generation. This equation is applied to the evaluation data. Each run had an initial population of 25 to 50 randomly generated equations. The number of generations in each run varied, but the best results were obtained with 3 to 5 generations (works reported in the literature use a higher number of generations). The number of equations that remained from one generation to the next represented 10% of the previous one. The new equations of the new generation were formed by:

a) *crossover*: two equations exchange parts, preserving the syntax of the mathematical expression;

b) *mutation*: a term of the equation (function, variable or constant) is changed randomly.

In general, GP has worked as expected in providing equations "fitted to the data". The algorithm that evolves a solution for symbolic regression is as follows. The software for GP has been provided by the Newcastle Symbolic Optimisation Group (runs in Matlab 4.2.c).

```
Genetic Programming Algorithm
    Generate initial population (of size N) of equations
    While there are generations to run do
        Evaluate fitness of each equation
        For each equation in the population
          select randomly one of
          a) Mutation with probability Pm
          b) Crossover with probability Pc
          c) Direct reproduction with probability (1- Pm -Pc)
          Add the new equation to the new population
        Endfor
endwhile
```

Figure 1. The algorithm.

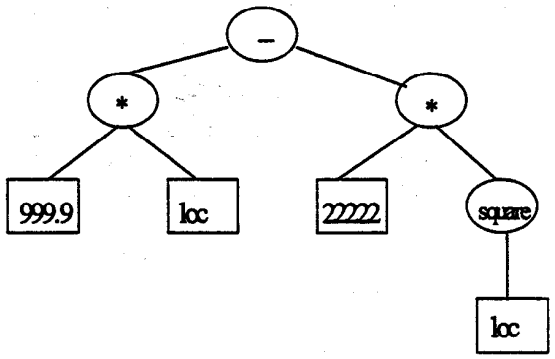


Figure 2. Tree representation of an equation

Each equation in the population is represented as a tree. Figure 2 represents the equation $Effort = 999.9 \cdot loc - 22222 \cdot loc^2$. The operation of *crossover* takes a subtree of two equations and interchange them to form new members of the populations. *Mutation* randomly modifies a subtree of an equation.

The functional set used is: +, -, *, /, ^, sqrt, square, log, exp. Probabilities of mutation and crossover are in the range of Pm=0.8 and Pc=0.2. One of the problems of GP is that in order to generate simpler expressions some limit to the trees has to be set. Here the best results were obtained by limiting the number of generations. The fitness measure in the present work is the *mean squared error* of the predicted values of the equation. In the implementation used, regression constants in the equations are determined using the Levenberg-Marquardt method of non-linear least-square optimisation.

2.4 Evaluation of the techniques

The criteria for comparing the three methods on the datasets is to compute the values for PRED(0.25) and MMRE $\leq$ 0.25 [5]. In the case of multiple linear regression an assessment is made by the R^2 and other statistical criteria in order to select the best equation.

- *Mean magnitude of relative error*, MMRE, is defined as $MMRE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{e_i - \hat{e}_i}{e}\right|$, where e is a real value of a variable in a project, $\hat{e}$ is its estimate and n is the number of projects. Thus if the MMRE is small, then we have a good set of predictions. A usual criteria for accepting a model as good is that the model has a MMRE $\leq$ 0.25.
- *Prediction at Level l* -PRED(l)- , where l is a percentage, is defined as the quotient of number of cases in which the estimates are within the l absolute limit of the actual values divided by the total number of cases. For example PRED(0.1) = 0.9 means that 90% of the cases have estimates within the 10% range of their actual values. A standard criteria for considering a model as acceptable is PRED(0.25) $\geq$ 0.75. This means that at least 75% of the estimates are within the range of the 25% of the actual values. Some authors relax this requirement.
- *Coefficient of multiple determination,* R^2 and *adjusted* R^2, are some usual measures in regression analysis, denoting the percentage of variance accounted for by the independent variables used in the regression equations.
- *mean squared error*, as defined for regression models is $mse = \frac{1}{n-2}\sum_{i=1}^{n}(e_i - \hat{e}_i)^2$.

3 The Datasets

In all datasets the dependent variable is the effort measured in man-months or in man-hours.

3.1 Dataset1 -DT1-. Belady's data

This dataset is taken from [3] (33 points). The independent variable is LOC.

3.2 Dataset2 -DT2-. Boehm's data

The data was obtained from the book [4]. The independent variable used is "adjusted delivered source instructions". An independent evaluation made by Conte et al. [5] provides the following values of the evaluation of the model: Basic

Cocomo PRED(0.25)=27% and MMRE = 0.6; Intermediate (2) Cocomo gives PRED(0.25)=0.76 and MMRE = 0.19.

3.3 Dataset3 -DT3-. Albrecht and Gaffney's plus Kemerer's data

This dataset is a combination of the datasets [1] (Albrecht and Gaffney, 24 points) and [11] (Kemerer, 15 points). This combination was used in the article of Matson et al. [15], and it has been used here with the purpose of having a dataset with a number of points sufficient enough for dividing the dataset. The independent variable is "function points".

The results of Shepperd and [18] with respect to DT3 is as follows: the data is disaggregated into the two subsets, Kemerer on one hand and Albrecht and Gaffney on the other. Albrecht and Gaffney's data obtains PRED(0.25)=33% and MMRE = 0.62 when using *analogy* and PRED(0.25)=0.33 and MMRE = 0.9 when using linear regression. Kemerer's data obtains PRED(0.25)=40% and MMRE = 0.62 when using *analogy* and PRED(0.25)=0.13 and MMRE = 1.07 when using linear regression.

Matson et al. [15] concluded, cautiously, that the appropriate model to be fitted could be the log-log model (power curve). The main criterion followed was the *mse*. However, they stated that aggregation of the two datasets should only be made after careful examination of the environments. Our purpose here is only to test methods, and not to extrapolate the equation to other environments.

3.4 Dataset4 -DT4-. Academic Environment

This dataset has been obtained with student's projects (48 points). The independent variable is LOC. 15 points are used for validation. Although the variable "function points" was also computed, it did not provide any predictive value.

3.4 Dataset5 -DT5-

This dataset is an approximation to the data used by Matson et al. [15], since the original dataset is not available. As it is not a real dataset, the analysis is provided in Appendix A.

4 The Application of the Methods to the Datasets

The following tables report the results of the application of the three methods (GP, NN and MLR) to every dataset in the two situations: a) using the same data points for model building and model evaluation, and b) using a sample for model building and the rest for model evaluation.

4.1 Predictions Using the Whole Datasets for Model Building

Table 1 indicates that GP obtains better results than NN and LR in datasets DT3 and DT4, and worse in DT1 and DT2. NN presents the same situation. GP surpasses NN in all cases. Surprisingly, GP was unable to find the power curves of DT1 and DT2, although there was a reasonable amount of runs (44 and 32 respectively). The corresponding plots are Figure 3 through Figure 6.

GP finds in DT1 two very different equations. Dashed line in Figure 3 is an equation with worse evaluation resuls. In DT2 the simple regression or NN cannot improve the results of Basic or Intermediate Cocomo. In DT3, GP is able to overcome the problems of aggregating the two datasets, providing PRED(0.25)=33.3% and MMRE = 0.684. It is peculiar how GP approximates a curve in Figure 5 which minimizes the *mse*, although giving a surprising shape. A second GP equation seems more plausible.

Table 1. Note: (n1,n2): number of runs in GP and NN, respectively

		GENETIC PROGRAMMING	NN	CURVE ESTIMATION
DT1 (44,12)	*Equation*	$5.331\cdot10^{-5}\cdot loc / 0.987\cdot\left(\sqrt{loc}-3.881\cdot10^{-2}\cdot\left(-2.708\cdot10^{-2}\cdot loc\right)\right)$		$0.003067\cdot loc^{1.060791}$
	Pred(0.25)	24.2	18.18	33.33
	Mmre	0.848	1.2733	0.6258
DT2	*Equation*	$-655.4+0.4894\cdot(1.691\cdot10^{4}+loc)^{0.7106}$		$0.001852\cdot loc^{1.10839}$
	Pred(0.25)	15.9	12.70	17.46
(32,10)	*Mmre*	3.227	5.8576	1.1336
DT3 (39,21)	*Equation*	A) $fp/\sqrt{\lvert fp-2311\rvert}$ B) $1.139\cdot10^{-4}\cdot fp^{2}$		$0.001267\cdot fp^{1.576990}$
	Pred(0.25)	A) 33.3 B) 15.4	17.95	7.69
	Mmre	0.684 1.586	2.123	1.18
DT4	*Equation*	$121.1+8.211\cdot10^{-2}\cdot loc+2.946\cdot10^{-7}\cdot loc^{2}$		$1.995953\cdot loc^{0.643862}$
	Pred(0.25)	43.8	43.75	37.99
(38,21)	*Mmre*	0.433	0.4211	0.4375

4.2 Models using a subset of the data for model building

4.2.1 Dataset1. Belady's data

GP works well except for Sample3. NN obtains better results in the three samples, comparatively. It can be observed from the following tables that samples do not significantly improve or worsen the results of DT1 in Table 4. Curves in Figure 7

represent economies of scale as well as diseconomies of scale, including quadratic, power and other types.

Table 2

DATASET1		GENETIC PROGRAMMING	NN	CURVE ESTIMATION
Sample1	*Pred (0.25)*	18.2	27.27	0
(8,18)	*Mmre*	1.912	1.6960	1.4124
Sample2	*Pred (0.25)*	A) 27.3 B) 36.4	36.36	18.18
(9,9)	*Mmre*	A) 2.760 B) 8.036	2.0549	4.6568
Sample3	*Pred (0.25)*	27.3	27.27	36.36
(9,9)	*Mmre*	1.911	3.0705	0.6266

4.2.1 Dataset2. Boehm's data

Sampling does not seem to have a clear effect on the evaluations. Curve estimations improve a little. GP appears to be very dependent on the sample (see Figure 8). Quadratic equations are found by GP, and the linear regression finds power equations.

Table 3

DATASET2		GENETIC PROGRAMMING	NN	CURVE ESTIMATION (LR)
Sample1	*Pred(0.25)*	14.3	14.29	14.29
(11,10)	*Mmre*	13.293	9.4686	1.0985
Sample2	*Pred(0.25)*	19	23.81	28.57
(12,14)	*Mmre*	2.954	5.0499	0.6345
Sample3	*Pred(0.25)*	9.5	14.29	23.81
(13,13)	*Mmre*	0.987	5.8588	1.1671

4.2.3 Dataset3. Albrecht and Gaffney's plus Kemerer's data

Sampling does not seem to have any clear benefit on the evaluations. Genetic Programming works slightly better than linear regression. GP finds quadratic as well as other rarer equations. Linear regression shows cubic, quadratic and exponential equations (Figure 9).

Table 4

DATASET3		GENETIC PROGRAMMING	NN	CURVE ESTIMATION (LR)
Sample1	*Pred(0.25)*	15.4	15.38	15.38
(11,23)	*Mmre*	0.614	1.1738	0.6151
Sample2	*Pred(0.25)*	15.4	15.38	0.0769
(13,10)	*Mmre*	1.829	3.6960	12.8257
Sample3	*Pred(0.25)*	30.8	23.08	23.08
(9,22)	*Mmre*	1.4444	1.6320	3.3374

4.2.4 Dataset4. Academic environment

Similar results to Table 1. GP appears to be very dependent on the sample (see Figure 10). Linear regression derives power equations and GP derives a variety of equations .

Table 5

DATASET4		GENETIC PROGRAMMING	NN	CURVE ESTIMATION (LR)
Sample1	*Pred (0.25)*	46.7	40.00	46.67
(12,18)	*Mmre*	0.34	0.3656	0.3073
Sample2	*Pred (0.25)*	53.3	46.67	40.00
(10,17)	*Mmre*	0.355	0.4127	0.3181
Sample3	*Pred (0.25)*	40.0	40.00	46.67
(9,22)	*Mmre*	0.541	0.5571	0.4856

4.3 The Effort-Size Relationship and the Methods to Discover It

The major question of "what is the shape of the effort-size relationship?" remains unsolved. There seems to be no pattern whether samples are chosen for model building or the whole dataset, observing Tables 1 through 5 and Figures 3 through 10. The work of Hu [10] only concluded that there was no linear model in the production function of software, and that the quadratic model is a candidate to describe the effort-size relationship.

Quadratic functions have appeared in Table 1 for datasets DT1 and DT3 and DT5, by means of Genetic Programming. The procedures used in curve estimation and linear regression do not have selected the quadratic functions due to the values provided for the R^2 and to the errors incurred on the statistical model. So, from this point of view, GP has caused the emergence of the equations by checking only the mean squared error (the fitness measure).

Neural networks and genetic programming have more flexibility in approximating models than classical statistics. However, from the point of view of making good predictions, no technique has been proved to be clearly superior. In the work of [6] the GP provided similar or better solutions than classical statistics, overcoming also the problems of NN, which did not provide symbolic explanation of the relationship.

Taken samples of the datasets have neither improved nor degraded the results in a clear way. Therefore, and from the practical point of view it would not matter

which way is followed for the analysis. GP, trying to minimize the error will show some uncommon curves.

5 Conclusions

The first and more evident conclusion is that *estimating effort is -really- a very difficult task*, as the values of the PRED(0.25) and MMRE variables indicate. All environments analyzed here have provided results far away from the ideal values. The quality of the data seems to be more important.

When using all points in every dataset for model building the results are not better than those obtained when using a sample for model building and the rest for evaluation. This contradicts the common assumption about using the same points for equation building and equation evaluation. This is probably due to the nature of the data points, that is the effort-size relationship. No generalization of this result should be made without explicit consideration of the environment.

From the values shown in the tables there is no great superiority of one method versus the others. GP is able to find rarer relationships than those that can be obtained with classical statistics, which can help to uncover some data trends. GP can be used as an alternative to linear regression, or as a complement to it. Neural networks provide no better values than the others. The use of different and complementary methods helps the estimator to assess the predictions. Other different datasets, such as those presented in [18], provide values of Pred(0.25) ranging between 13% and 51%, values of Mmre between 0.37 and 2.52, and even when analogy is used things remain not very good.

As an aside result it is observed that the academic environment is the one with best predictible results. In this case the variable used has been LOC. The independent variable used is responsible for that result, since in a previous work the prediction capabilities of the "Function Points" variable was null. Selection of the independent variables is an important factor in obtaining good estimations. The academic environment may be suspicious of lacking some of the situations of reality, but at least it can provide some type of "homogeneity".

Curiously, the results of this study tend to support the ideas of Lederer and Prasad [13] that "the use of complex statistics, software, and standards do not facilitate more accurate estimates". While not trying to be naïve (at all) about this issue, and acknowledging that the "estimating problem" is still open, the fact is that the evaluation neither of the algorithmic nor of the neural network models provide much confidence for generating predictions. But it is worth quoting the phrase from [15]: "function points, even in concert with other predictions, seem only moderately helpful in software development cost estimation".

Also, the conflicting methods used by different authors impede further clarification of the important issues of the production function definition for software. Either a sample of the dataset is used for model building or all datapoints are used. The first option is intended to be statistically more useful for prediction, but the second seems to be appropriate for finding a production function for software. Furthermore, the measures Pred(0.25) and Mmre have been used in this

study to choose the model, what conveys the idea of "ability to predict" as being of interest (not fitting a model to the data).

There remain many issues related to the application of the GP to the software engineering field to be researched. The powerful capabilities of the "genetic paradigm" could be used in combination with other more cognitive human abilities to solve questions about prediction in the software engineering field (such as analogy or other non-algorithmic methods). Moreover, it is important to define what the estimators want: either predictions without knowing the model (NN) or to build equations that give the best predictions. The questions of how to use the statistical tests hovers over any decision about the model to choose.

Acknowledgements.
The analysis by Genetic Programming has been possible thanks to the software of the Newcastle Symbolic Optimisation Research Group (http://lorien.ncl.ac.uk/sorg), to whom we are very grateful.

6 References

[1] A.J. Albrecht and J.R. Gaffney, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation", *IEEE Trans. Software Engineering*, vol. 9, no. 6, pp 639-648, 1983

[2] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary Computation: Comments on the History and Current State", *IEEE Trans. on Evolutionary Computation*, 1 No.1, April 1997, pp. 3-17

[3] L.A. Belady and M.M: Lehman, "The Characteristics of Large Systems", in *Research Directions in Software Technology*, P.Wegner et al. (eds.), pp. 106-138, MIT Press, Cambridge, Mass., 1979

[4] B.W. Boehm, *Software Engineering Economics*, Prentice-Hall, 1981

[5] S.D. Conte, H.E. Dunsmore and V.Y. Shen, *Software Engineering Metrics and Models*, Benjamin/Cummings, 1986

[6] J.J. Dolado, "A Validation of the Component-Based Method for Software Size Estimation", submitted.

[7] 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, pp. 281-289, 1997

[8] A. R. Gray and S. G. MacDonell, "A comparison of techniques for developing predictive models of software metrics", *Information and Software Technology*, 39, pp. 425-437, 1997

[9] M.T. Hagan, H.B. Demuth and M. Beale, *Neural Network Design*, PWS Publishing Company, 1996

[10] Q. Hu, Evaluating Alternative Software Production Functions, *IEEE Transactions on Software Engineering*, Vol. 23, N. 6, June 1997, pp. 379-387

[11] C.F. Kemerer, "An Empirical Validation of Software Cost Estimation Models", *Comm. ACM*, vol. 30, no. 5, pp. 416-429, 1987

[12] J.R. Koza, *Genetic Programming: On the Programming of Computers by Natural Selection*, MIT Press, 1992

[13] A.L. Lederer and J. Prasad, "A Causal Model for Software Cost Estimating Error", *IEEE Trans. on Software Engineering*, Vol. 24, no. 2, pp. 137-148, February, 1998-05-19

[14] S.G. MacDonell and A.R. Gray, "Alternative to Regression Models for Estimating Software Projects", *Proceedings of the IFPUG Fall Conference*, Dallas TX, IFPUG (1996) 279.1-279.15

[15] J.E. Matson, B.E. Barret and J.M. Mellichamp, "Software Development Cost Estimation using Function Points", *IEEE Trans. on Software Engineering* 20, No. 4 April, pp. 275-287, 1994

[16] B. McKay, J. Elsey, M. Willis and G. Barton, "Evolving Input-Output Models of Chemical Process Systems using Genetic Programming", *Proceedings of the IFAC '96*, San-Francisco, 1996

[17] B. McKay, M.J. Willis and G.W. Barton., "Steady-State Modelling of Chemical Process Systems using Genetic Programming", *Computers and Chemical Engineering*, 1997, 21 (9) pp. 981-996

[18] M. Shepperd and C. Schofield, "Estimating Software Project Effort Using Analogies", *IEEE Trans. on Software Engineering*, Vol. 23, no. 11, pp. 736-743, Novembre 1997

[19] M.Willis, H. Hiden, M. Hinchliffe, B. McKay and G. Barton, "Systems Modelling Using Genetic Programming", *Computers and Chemical Engineering*, 1997, Vol. 21 Supplement, pp. 1161-1166.

[20] G. Wittig and G. Finnie, Estimating Software Development Effort with Connectionist Models, *Information and Software Technology* Vol. 39, pp.469-476, 1997

Appendix A Analysis of an Approximation to a Dataset -DT5-

An analysis has been carried out with an approximation to the dataset described in [15]. We had a special interest in this dataset since the number of points is very large (104 points). Since the numerical values are not publicly available, a recovery procedure was designed, starting from the plots depicted in the article. The data thus obtained was an approximation to the original data. In order to respect the confidentiality of data but serving scientific interest of the software engineering community, we comment briefly the conclusions obtained.

As with the rest of the datasets used in the present study three different samples were extracted for model building and evaluation. Also a comparison of the methods was made using all the data points.

A.1 All data points

The original analysis (with the correct data) [15] found that the most reasonable equation was $\ln effort = 2.51 + 1.00 \cdot \ln fp$. The Mmre is 0.87 and the Pred(0.25)= 0.64. However, the value of Pred(0.25) seems to be computed on the transformed data and not on the original variables. This is corroborated by the fact that it is quite strange to find a data set with a Mmre above 0.8 with such level of prediction. Also, it seems that the equation is not complete or the coefficient 1.00 is incorrect, because undoing the transformation we obtain $effort = 12.3049 \cdot fp^{1.00}$, that is simply a linear equation (The transformation applied is $\ln y = \ln b_0 + b_1 \ln t \Rightarrow y = b_0 t^{b_1}$).

In our environment it has been discovered that:

- Genetic Programming finds two equations (26 runs) with similar results:
  a) $14.22 \cdot fp + 0.001113 \cdot fp^2$, that gives Pred(0.25)=27.9 and Mmre=1.031;
  b) $\left(\left(-0.1569 + 8.134 \cdot 10^{-5} \cdot fp\right) \cdot fp\right)^2 + 10.02 \cdot fp$, that gives Pred(0.25)=29.8 and Mmre=1.295;
- Neural Networks: Pred(0.25)=30.77 and Mmre=1.1448;
- Statistical Equation: $10.051811 \cdot fp^{1.032947}$, Pred(0.25)=27.88 and Mmre=0.8485;

The differences of [15] from the results of this study may be due to the approximation of the data, too. All values are worse that those attributed to the correct dataset.
The three samples provide an insignificant improvement in the evaluations.

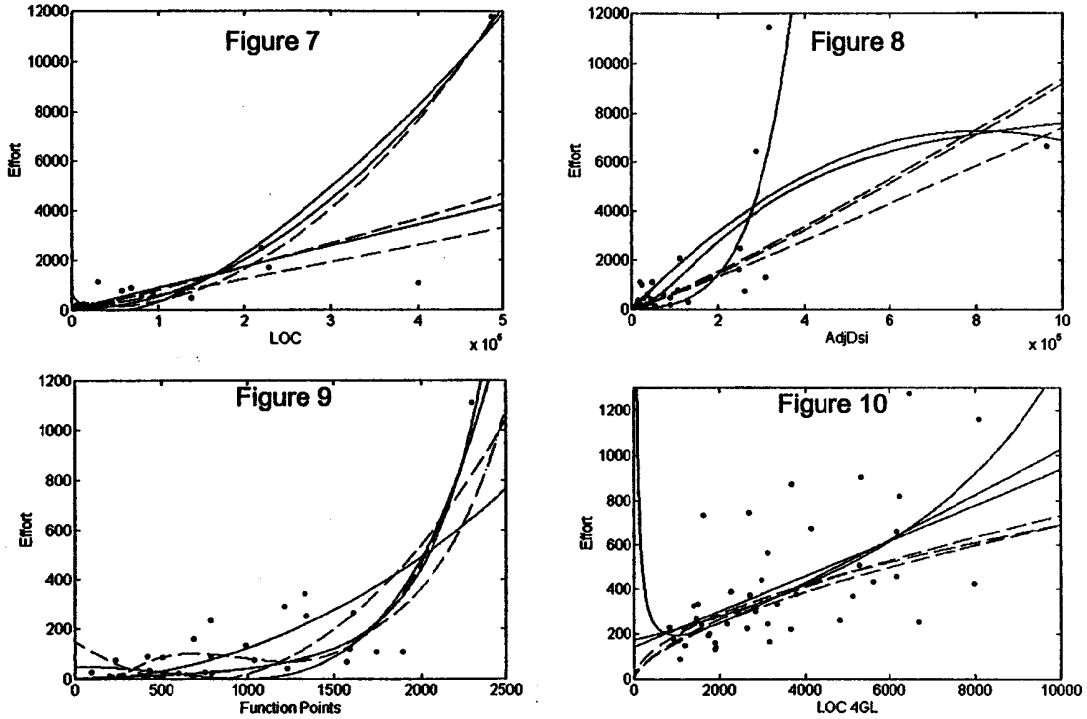
A.2 Evaluation on a sample

Table 6

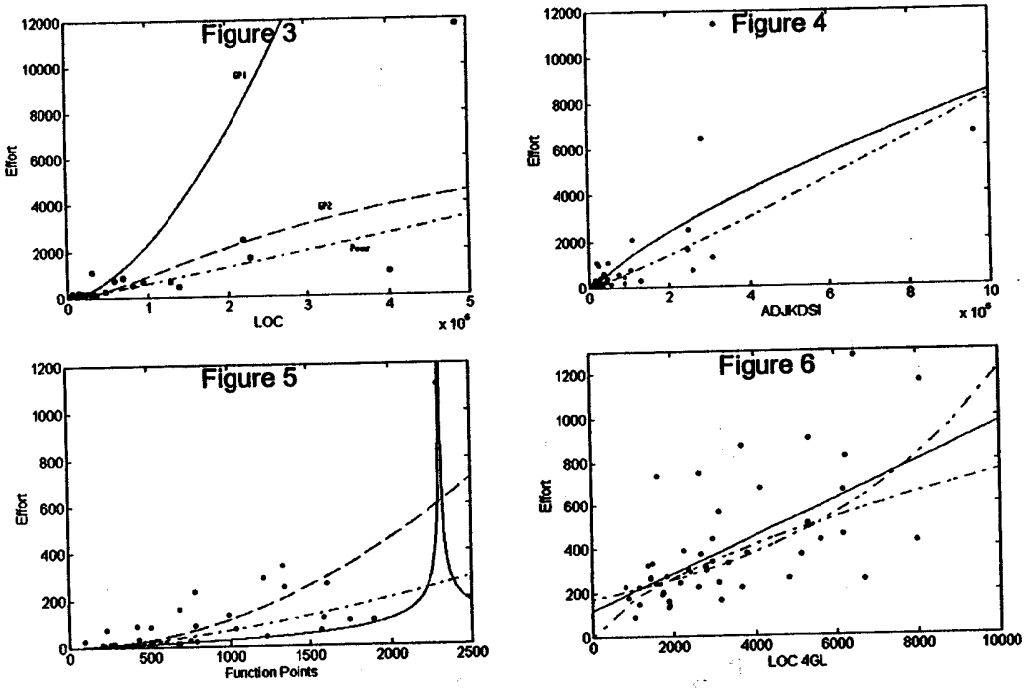
35 POINTS		GENETIC PROGRAMMING	NN	CURVE ESTIMATION (LR)
Sample1	*Pred(0.25)*	31.4	31.43	28.57
(9,40)	*Mmre*	1.199	1.1381	0.8363
Sample2	*Pred(0.25)*	31.4	25.71	25.71
(7,40)	*Mmre*	0.485	0.6045	0.494
Sample3	*Pred(0.25)*	28.6	31.43	20.0
(9,40)	*Mmre*	1.029	1.0621	0.804

Appendix B. Plots of the curves for DT1 to DT4.

Figures 3 to 10 in next pages.



Figures 7 to 10. These correspond to datasets DT1 through DT4 respectively when the equations are derived from a sample and the validation of the equations is made in the rest of the data points. The solid lines represent the curves obtained by Genetic Programming. The dashed lines represent the equations obtained by curve estimation and linear regression. There are three solid lines and three dashed lines in each figure, representing each one of the three samples.



Figures 3 to 6. These correspond to datasets DT1 through DT4, respectively, when the equations are derived from all data points. The solid lines represent the curves obtained by Genetic Programming. The dashed lines also plot Genetic Programming equations. The dash-dot lines represent the equations obtained by curve estimation and linear regression. The lines are those that have provided the best evaluation results.