Prediction of fault count data using genetic programming

Wasif Afzal, Richard Torkar and Robert Feldt
Blekinge Institute of Technology,
S-372 25 Ronneby, Sweden
{*waf,rto,rfd*}*@bth.se*

Abstract

Software reliability growth modeling helps in deciding project release time and managing project resources. A large number of such models have been presented in the past. Due to the presence of these many number of models, their inherent complexity and accompanying assumptions; the selection of suitable models becomes a challenging task. This paper presents empirical results of using genetic programming (GP) for modeling software reliability growth based on weekly fault count data of three different industrial projects. The goodness of fit (adaptability) and predictive accuracy of the evolved model is measured using five different statistics in an attempt to present a fair evaluation. The results show that the GP evolved model has statistically significant goodness of fit and predictive accuracy.

1. Introduction

Software has become a key element in the daily life of individuals and societies as a whole. We are increasingly dependent on software and because of this ever-increasing dependency, software failures can lead to hazardous circumstances. Ensuring that the software is of high quality is thus a high priority. A key element of software quality is software reliability, defined as the ability of a system or component to perform its required functions under stated conditions for a specific period of time [11]. If the software frequently fails to perform according to user-specified behavior, other software quality factors matters less [19].

Software reliability growth models (SRGMs), which are based on the time domain,[1] describe the behavior of software failures with respect to time. Specifically, reliability growth modeling performs curve fitting of observed time-based failure data by a pre-specified model formula, where the parameters of the model are found by statistical techniques such as e.g. the maximum likelihood method [20]. The model then estimates reliability or predicts future reliability by different forms of extrapolation [15]. After the first software reliability growth model was proposed by Jelinski and Moranda in 1972 [12], there have been numerous reliability growth models following it. These models come under different classes [14], e.g. exponential failure time class of models, Weibull and Gamma failure time class of models, infinite failure category models and Bayesian models. The existence of a large number of models requires a user to select and apply an appropriate model. For practitioners, this may be an unmanageable selection problem and there is a risk that the selected model is unsuitable to the particulars of the project in question. Some models are complex with many parameters. Without extensive mathematical background, practitioners cannot determine when it is applicable and when the model diverges from reality. Even if the dynamics of the testing process are well known, there is no guarantee that the model whose assumptions appear to best suit these dynamics will be most appropriate [21]. Moreover, these *parametric* software reliability growth models are often characterized by a number of assumptions, e.g. an assumption that once a failure occurs, the fault, which causes the failure, is immediately removed and the fault removal process will not introduce new faults. These assumptions are often unrealistic in real-world situations (see e.g. [28]), therefore, causing problems in the long-term applicability and validity of these models. Under this scenario, what becomes significantly interesting is to have modeling mechanisms that can exclude the pre-suppositions about the model and is based entirely on the fault data. In this respect, genetic programming (GP) can be used as an effective tool because, being a *non-parametric* method, GP does not conceive a particular structure for

[1] There are also software reliability growth models based on the coverage of a testing criterion.

the resulting model and GP also does not take any assumptions about the distribution of the data.

In this paper, we present an experiment where we apply GP to evolve a model based on weekly fault count data. We use five different statistics to evaluate the adaptability and predictive ability of the GP-evolved model on three sets of fault data that corresponds to three projects carried out by a large telecommunication company. The results of the experiment indicate that software reliability growth modeling is a suitable problem domain for GP as the GP evolved model gives statistically significant results for goodness of fit and predictive accuracy on each of the data sets.

The remainder of this paper is organized as follows. Section 2 describes related work, including differences between this study and previous works. Section 3 presents a brief introduction to genetic programming. In Section 4 we detail our research method along with a discussion of evaluation measures used. Section 5 and 6 comprises of experimental setup and results respectively. Validity evaluation is given in Section 7 while the discussion and future work appears in Section 8.

2. Related work

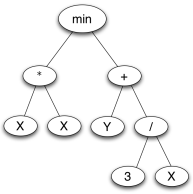
Within the realm of machine learning algorithms, there has been work exploring the use of artificial neural networks for software reliability growth modeling (e.g. [24]), but our focus here is on the research done using GP for software reliability growth modeling.

Studies reporting the use of GP for software reliability modeling are few and recent. Costa et al. [5] presented the results of two experiments exploring GP models based on time and test coverage. The authors compared the results with other traditional and non-parametric artificial neural network (ANN) models. For the first experiment, the authors used 16 data sets containing time-between-failure (TBF) data from projects related to different applications. The models were evaluated using five different measures, four of these measures represented different variants of differences between observed and estimated values. The results from the first experiment, which explored GP models based on time, showed that GP adjusts better to the reliability growth curve. Also GP and ANN models converged better than traditional reliability growth models. GP models also showed lowest average error in 13 out of 16 data sets. For the second experiment, which was based on test coverage data, a single data set was used. This time the Kolmogorov-Smirnov test was also used for model evaluation. The results from the second experiment showed that all metrics were always better for GP and ANN models. The authors later extended GP with boosting techniques for reliability growth modeling [22] and reported improved results. A similar study by Zhang and Chen [29] used GP to establish a software reliability model based on mean time between failures (MTBF) time series. The study used a single data series and used six different criteria for evaluating the GP evolved model. The results of the study also confirmed that in comparison with the ANN model and traditional models, the model evolved by GP had higher prediction precision and better applicability.

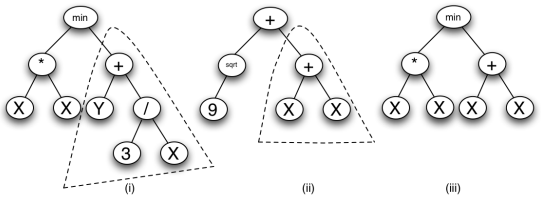
There are several ways in which the present work differs from the aforementioned studies. Firstly, none of the previous studies used data sets consisting of weekly fault count data. In this study, our aim is to use the weekly fault count data as a means to evolve the reliability growth model using GP. Secondly, we have avoided performing any pre-processing of data to avoid chances of incorporating bias. Thirdly, we remain consistent with using 2/3 of the data to build the model and use the rest 1/3 of the data for model evaluation for all of our data sets. Perhaps the most important difference between prior studies and the study in this paper is that previous studies have focused on comparative accuracy with traditional and ANN models, rather than on the suitability of the approach of using GP for building software reliability growth models which is the focus of this study. In an attempt to provide a fair evaluation, we also remain consistent with using the same set of five different statistics for evaluating GP evolved models for all the data sets.

3. Background to genetic programming

GP is an evolutionary computation technique (first results reported by Smith [25] in 1980) and is an extension of genetic algorithms. Genetic algorithms are search methods based on the principles of natural selection and genetics [8]. As compared with genetic algorithms, the population structures (individuals) in GP are not fixed length character strings, but programs that, when executed, are the candidate solutions to the problem. GP is a systematic, domain-independent method for getting computers to solve problems automatically starting from a high-level statement of what needs to be done [23]. Programs are expressed in GP as syntax trees, with the nodes indicating the instructions to execute and are called functions, while the tree leaves are called terminals which may consist of independent variables of the problem and random constants. In Figure 1(a), variables x, y and constant 3 are the terminals while *min*, $*$, $+$ and $/$ are the functions. There are five preparatory steps for a basic GP [2]: specifying the set



(a) Tree structured representation showing $min(x*x, y+3/x)$.



(b) Crossover in GP, dotted line showing the subtrees exchanged between (i) first parent and (ii) second parent, to create offspring program (iii).

Figure 1. Tree structured representation and crossover operation in GP.

of terminals, the set of functions, the fitness measure for measuring the fitness of individuals in the population, parameters for controlling the run and the termination criterion. The first two steps define the search space that will be explored by GP. The fitness measure guides the search in promising areas of the search space and is a way of communicating problems requirements to GP. The fitness evaluation of a particular individual is determined by the correctness of the logical output produced for all of the fitness cases [1]. The last two steps are administrative. The control parameters limit and control how the search is performed like setting the population size and probabilities of performing the genetic operations. The termination criterion specifies the ending condition for the GP run and typically includes a maximum number of generations [2].

GP iteratively transforms a population of computer programs into a new generation of programs using various genetic operators. Typical operators include crossover, mutation and reproduction. The crossover operator recombines randomly chosen parts from two selected programs and creates new program(s) for the new population (Figure 1(b)). The mutation operator selects a point in a parent tree and generates a new random sub-tree to replace the selected subtree, while the reproduction operator simply replicates a selected individual to a new population.

3.1. Genetic programming as a symbolic regression technique

The evolution of software reliability growth models using GP is an example of a symbolic regression problem. Symbolic regression is an error-driven evolution as it aims to find a function, in symbolic form, that fits (or approximately fits) data from an unknown curve [13]. In simpler terms, symbolic regression finds a function whose output matches some target values. GP is well-suited for symbolic regression problems as it does not make any assumptions about the structure of the function.

It is important to collect an appropriate set of data points for symbolic regression which acts as fitness cases for a problem. For our study, we used the weekly fault count data as fitness cases (see Appendix A). The independent variable in our case was the week number while the corresponding dependent variable was the count of faults.

4. Research method

In this section we outline the research method used in this paper. We describe the data sets used, the formulated hypotheses and a description of the evaluation measures.

4.1. Fault count data sets

The data sets used in this study are based on the weekly fault count data collected during the testing of three large-scale software projects at a large telecom company. The motivation for selecting the fault count data from an industrial context is to be representative of real-world problem domain. The projects are targeted towards releases of three mature systems that have been on the market for several years. These projects followed an iterative development process which means that within each iteration, a new system version, containing new functionality and fixes of previously discovered faults, is delivered to test. These iterations occurred on weekly basis or even more frequently, while testing of new releases proceeded continuously. In this scenario, it becomes important for project managers to estimate the current reliability and to predict the reliability ahead of time, so as to measure the quality impact with continuous addition of new functionality and fixes of previously discovered faults. The three projects are similar in size i.e. they have approximately half a million lines of code. There are, however, minor differences with respect to the projects duration. The

first project lasted 26 weeks, whereas the second and third projects lasted 30 and 33 weeks respectively. Appendix A shows the data sets used in the study, but due to the proprietary nature of data, the number of faults are multiplied by a factor and are given for illustrative purposes only. Nevertheless, we believe that making the data sets available allows the research community to replicate results and to perform additional studies. The *results* of the evaluation measurements in the rest of the paper are, however, based on original data sets.

We used $2/3$ of the data in each data set for building the model and $1/3$ of the data for evaluating the model according to the five different statistics (Subsection 4.3). This implies that we are able to make predictions on several weeks constituting $1/3$ of the data.

4.2. Hypothesis

The purpose of this experiment is to evaluate the predictive accuracy and goodness of fit of GP in modeling software reliability using weekly fault count data collected in an industrial context. In order to formalize the purpose of the experiment, we define the following hypotheses:

H_{0-acc}: GP model does not produce significantly accurate predictions.

H_{1-acc}: GP model produces significantly accurate predictions.

H_{0-gof}: GP model does not fit significantly to a set of observations.

H_{1-gof}: GP model fits significantly to a set of observations.

In order to test the above hypotheses, we use five measures for evaluating the goodness of fit and predictive accuracy as detailed in the next section.

4.3. Evaluation measures

It is usually recommended to use more than one measure to determine model applicability, as in [21], because reliance on a single measure can lead to making incorrect choices. The deviation between observed and the fitted values was, in our case, measured using a goodness-of-fit test. We selected two measures for determining the goodness of fit, the Kolmogorov-Smirnov (K-S) test and the Spearman's rank correlation coefficient. For measuring predictive accuracy, we used prediction at level l, mean magnitude of relative error (MMRE) and a measure of prediction stability. What follows is a brief description of each of these measures.

Kolmogorov-Smirnov The K-S test is a commonly used statistical test for measuring goodness of fit [3, 18]. The K-S test is a distribution-free test for measuring general differences in two populations. The statistic J for the two-sided two-sample K-S test is given by,

$$J = \frac{mn}{d} \max_{-\infty<t<+\infty} \{ \mid F_m(t) - G_n(t) \mid \} \tag{1}$$

Where $F_m(t)$ and $G_n(t)$ are the empirical distribution functions for the two samples respectively, m and n are the two sample sizes and d is the greatest common divisor of m and n. The null hypothesis of interest here is that the two samples have the same probability distribution and represents the same population.

$$H_0 : [F(t) = G(t), \text{for every } t] \tag{2}$$

We have used the significance level $\alpha = 0.05$ and if the K-S statistic J is greater or equal than the critical value J_α, the null hypothesis is rejected in favor of the alternate hypothesis, otherwise we conclude that the two samples have the same distribution. For detailed description of the test, see [10].

Spearman's rank correlation coefficient Spearman's rank correlation coefficient, ρ, is the non-parametric counterpart of the parametric linear correlation coefficient, r. Spearman's rank correlation coefficient ρ is independent of the assumption of linear relationship and bivariate normal population distribution as required in the case of linear correlation coefficient. Moreover, the data for each variable is converted into ordinal values (ranks) such that for n number of values in each variable, we have n pairs of rank. The rank correlation coefficient is then given by,

$$\rho = 1 - \frac{6\sum_{i=1}^{n} D_i^2}{n(n^2-1)} \tag{3}$$

where D_i is the difference between the i^{th} pair of ranks.

We use hypothesis testing to determine the strength of relationship between observed and estimated model values. If the absolute value of the computed value of ρ exceeds the critical values of ρ for $\alpha = 0.05$, we conclude that there is a significant relationship between the observed and estimated model values. Otherwise, there is not sufficient evidence to support the conclusion of a significant relationship between the two distributions.

Prediction at level *l* Prediction at level l, $pred(l)$, represents the count of the number of predictions within

$l\%$ of the actuals. We have used the standard criterion for considering a model as acceptable which is $pred(0.25) \geq 0.75$ which means that at least 75% of the estimates are within the range of 25% of the actual values [6].

Mean magnitude of relative error Mean magnitude of relative error (MMRE) is the most commonly used accuracy statistic and is defined as,

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} \mid \frac{e_i - e_i^{'}}{e_i} \mid \quad (4)$$

where e_i is the actual fault count data and $e_i^{'}$ is the estimated value of the fault count data. If we have a small MMRE, then we have a good set of predictions. Conte et al. [4] consider MMRE ≤ 0.25 as acceptable for effort prediction models; we use the same custom measure for our study as well.

Measure of prediction stability The predictions of a model should not vary significantly and should remain stable to denote the maturity of the model. We use here a good rule of thumb given in [27] for prediction stability which says that a prediction is stable if the prediction in week i is within 10% of the prediction in week $i-1$.

5. Experimental setup

In this study we used MATLAB version 7.0 [17] and GPLAB version 3.0 [9] (a GP toolbox for MATLAB).

5.1. Control parameter selection for GP

GPLAB allows for different choices of tuning control parameters. We were able to adjust the control parameters after certain amount of experimentation. We experimented with different function sets and terminal sets by fixing the rest of the control parameters like population size, number of generations and sampling strategy. Initially we experimented with a minimal set of functions by keeping the terminal set containing the independent variable only. We incrementally increased the function set with additional functions and later on also complemented the terminal set with a random constant. For each data set, the best model having the best fitness was chosen from all the runs of the GP system with different variations of function and terminal sets. The function set for project 1 and project 3 data sets were the same, while a slightly different function set for project 2 gave the best fitness. The GP programs were evaluated according to the sum of absolute differences between the obtained and expected results in all fitness cases,

$$\sum_{i=1}^{n} \mid e_i - e_i^{'} \mid \quad (5)$$

where e_i is the actual fault count data, $e_i^{'}$ is the estimated value of the fault count data and n is the size of the data set used to train the GP models. The control parameters that were chosen for the GP system are shown in Table 1.

Table 1. Main control parameters used for the GP system.

Control Parameter	*Value*
Population size	30
Number of generations	200
Termination condition	200 generations
Function set (for project 1 & 3)	$\{+,-,*,sin,cos,log\}$
Function set (for project 2)	$\{+,-,*,/,$ $sin,cos,log\}$
Terminal set	$\{x\}$
Tree initialization	ramped half-and-half
Initial maximum number of nodes	28
Maximum number of nodes after genetic operations	512
Genetic operators	crossover, mutation, reproduction
Selection method	lexictour
Elitism	replace

6. Results

In this section, we describe the results of the evaluation measurements to assess the adaptability and predictive accuracy of the GP evolved model.

6.1. Adaptability of the model

Table 2 shows the statistic J for the K-S test performed on the validation fault count data ($1/3$ of the original data set) and the estimated fault count data provided by the GP evolved model for each of the data sets. The critical values J_α for $\alpha = 0.05$ are also given. We see that in each data set, $J < J_\alpha$; therefore the null hypothesis for K-S test statistic (Eq. 2) holds. This suggests that the estimated fault count data, as provided by

Table 2. Results of applying Kolmogorov-Smirnov test.

	J	$J_{\alpha=0.05}$	*Sample size*	$J < J_\alpha$
Project 1	0.40	0.70	10	$\surd$
Project 2	0.27	0.64	11	$\surd$
Project 3	0.10	0.70	10	$\surd$

Table 3. Results of applying Spearman's correlation coefficient test.

	ρ	$r_{\alpha=0.05}$	*Sample size*	$\rho > r_{\alpha=0.05}$
Project 1	0.99	0.56	10	$\surd$
Project 2	0.93	0.54	11	$\surd$
Project 3	1.00	0.56	10	$\surd$

the GP model, fits quite well to the set of observations in all three data sets.

We additionally calculated the Spearman's rank correlation coefficient ρ for determining the relationship between actual and estimated model values (Table 3). At significance level $\alpha = 0.05$, computed values of ρ exceeds the critical values $r_{\alpha=0.05}$ for every data set. This indicates that there is a strong relationship between actual values and estimated model values.

Based upon the results of applying Kolmogorov-Smirnov and Spearman's rank correlation coefficient, we are able to reject the null hypothesis, $H_{0-\text{gof}}$ in support of the alternative hypothesis, $H_{1-\text{gof}}$.

6.2. Measuring predictive accuracy

Table 4 presents the results of measuring $pred(0.25)$ for the three data sets where e_i denotes the actual fault count data and $e_i^{'}$ is the estimated value of the fault count data. In all the data sets, the measurement $pred(0.25) \geq 0.75$ holds true. The bold values in Table 4 illustrate the cases when the model underestimates the actual fault count data.

We also calculated the MMRE for each of the data sets. The MMRE values for the three data sets were 0.0992, 0.06558 and 0.0166, respectively. Each of these values satisfy the criterion of MMRE ≤ 0.25, therefore we have confidence that we have a good set of predictions. For evaluating the prediction stability, we calculated whether the prediction in week i is within 10% of the prediction in week $i-1$. The results (Table 5) indicate that the predictions are indeed stable.

The results of applying $pred(l)$, MMRE and the measure of prediction stability show that the GP model is able to produce significantly accurate predictions. We

Table 4. Testing for $\mathbf{pred(0.25) \geq 0.75}$.

$25\%\ of\ e_i$	$e_i^{'}$	$e_i^{'}$ *within range of* $25\%\ of\ e_i$?
Project 1		
25 ± 6.25	25	$\surd$
27 ± 6.75	26.23	$\surd$
30 ± 7.5	27.53	$\surd$
33 ± 8.25	28.83	$\surd$
34 ± 8.5	30.10	$\surd$
35 ± 8.75	31.28	$\surd$
36 ± 9	32.38	$\surd$
40 ± 10	33.44	$\surd$
40 ± 10	34.51	$\surd$
41 ± 10.25	35.58	$\surd$
Project 2		
69 ± 17.25	75.82	$\surd$
70 ± 17.5	77.30	$\surd$
74 ± 18.5	74.69	$\surd$
78 ± 19.5	**76.40**	$\surd$
79 ± 19.75	84.14	$\surd$
83 ± 20.75	88.64	$\surd$
85 ± 21.25	94.28	$\surd$
93 ± 23.25	96.48	$\surd$
102 ± 25.5	**93.36**	$\surd$
109 ± 27.25	**102.56**	$\surd$
110 ± 27.5	**102.91**	$\surd$
Project 3		
153 ± 38.25	**148.54**	$\surd$
162 ± 40.5	**159.07**	$\surd$
173 ± 43.25	**167.06**	$\surd$
180 ± 45	**174.67**	$\surd$
184 ± 46	**181.04**	$\surd$
190 ± 47.5	**189.07**	$\surd$
196 ± 49	196.18	$\surd$
204 ± 51	**203.80**	$\surd$
208 ± 52	**207.65**	$\surd$
210 ± 52.5	216.32	$\surd$

Table 5. Testing for prediction stability.

Prediction in week i	10% *of the prediction in week* $i-1$	*Prediction stability*
Project 1		
25	–	–
26.23	25 ± 2.5	$\surd$
27.53	26.23 ± 2.62	$\surd$
28.83	27.53 ± 2.75	$\surd$
30.10	28.83 ± 2.88	$\surd$
31.28	30.10 ± 3.01	$\surd$
32.37	31.28 ± 3.12	$\surd$
33.44	32.37 ± 3.23	$\surd$
34.50	33.44 ± 3.34	$\surd$
35.57	34.50 ± 3.45	$\surd$
Project 2		
75.81	–	–
77.30	75.81 ± 7.58	$\surd$
74.69	77.30 ± 7.73	$\surd$
76.39	74.69 ± 7.46	$\surd$
84.14	76.39 ± 7.63	$\surd$
88.64	84.14 ± 8.41	$\surd$
94.28	88.64 ± 8.86	$\surd$
96.48	94.28 ± 9.42	$\surd$
93.35	96.48 ± 9.64	$\surd$
102.56	93.35 ± 9.33	$\surd$
102.91	102.56 ± 10.25	$\surd$
Project 3		
148.53	–	–
159.06	148.53 ± 14.85	$\surd$
167.06	159.06 ± 15.90	$\surd$
174.66	167.06 ± 16.70	$\surd$
181.04	174.66 ± 17.46	$\surd$
189.07	181.04 ± 18.10	$\surd$
196.18	189.07 ± 18.90	$\surd$
203.80	196.18 ± 19.61	$\surd$
207.65	203.80 ± 20.38	$\surd$
216.32	207.65 ± 20.76	$\surd$

can, thus reject the null hypothesis, H_{0-acc} in favor of the alternative, H_{1-acc}.

Figure 2 shows the comparison of actual and predicted fault count data for the three projects. The actual and predicted fault count data is multiplied by a constant factor due to proprietary concerns. The difference between the actual and predicted fault count is the least for data from project 3, which also has the best MMRE value of 0.0166. These charts show that the GP evolved curve is able to learn the pattern in failure count data and adapts reasonably well.

7. Validity evaluation

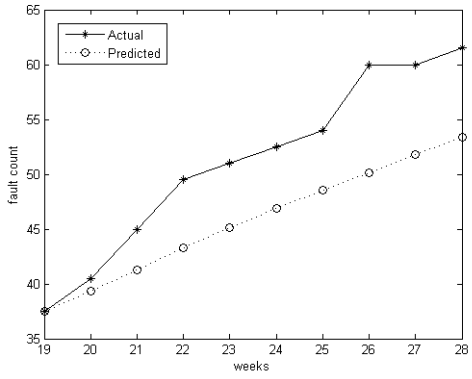
There can be different threats to the validity of experimental results.

Conclusion validity refers to the statistically significant relationship between the treatment and the outcome [26]. For K-S test and Spearman's rank correlation coefficient, we used hypothesis testing with 0.05 level of significance to identify significance of relationship between observed and estimated values. For $pred(0.25)$, MMRE and prediction stability, we used custom thresholds that have proven to be applicable in different predictive studies. One of the threats to conclusion validity is the use of MMRE which has been criticized in [7] for being unreliable. We have used an additional measure (Spearman's rank correlation coefficient) for measuring the strength of the relationship to minimize this threat.

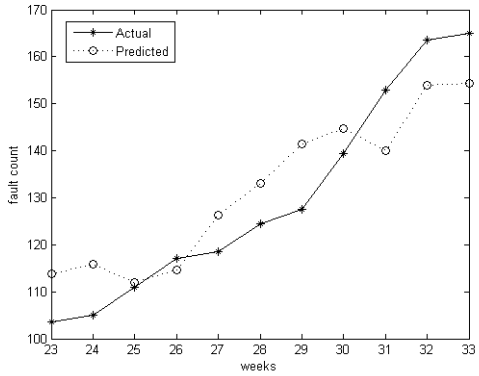
Internal validity refers to a causal relationship between treatment (independent variable) and outcome (dependent variable) [26]. In this paper, the GP algorithm is controlled by different parameters, all of which are configurable, therefore we are sure that there are no other influences affecting the independent variable with respect to causality.

Construct validity is concerned with the relationship between the theory and observation [26]. Our objective in this study is to measure the adaptability and predictive accuracy of GP evolved model. We used two measures for adaptability or goodness of fit and three for predictive accuracy. Also we used three data sets to have a reasonable representation of treatments.

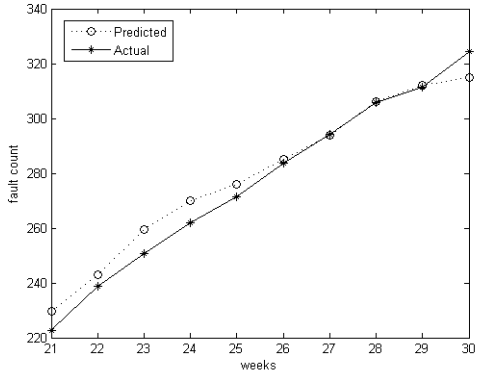
External validity is concerned with generalization of results outside the scope of the study. The experiment is conducted on three different data sets taken from an industrial setting. However, these projects are carried out by one organization following similar development methods. The generalizability of the research can be improved by experimenting with data sets taken from diverse projects employing different development methodologies. Also, as we described in related work



(a) Project 1—Predicted and actual fault count data.



(b) Project 2—Predicted and actual fault count data.



(c) Project 3—Predicted and actual fault count data.

Figure 2. Actual and predicted fault count data for three projects.

(Section 2), the study is carried out with the objective of evaluating the suitability of GP for building software reliability growth models, rather than comparing the accuracy with traditional and artificial neural network models. We acknowledge that the generalizability of the research can be improved further by having such a comparison.

8. Discussion and future work

The hypothesis to be tested was that GP could be a suitable approach for evolving a SRGM based on fault count data. The results of applying the evaluation criteria, as described in Subsection 4.3, confirmed that GP represents a suitable approach for modeling software reliability growth based on fault count data, both in terms of goodness of fit and predictive accuracy. In terms of goodness of fit, the K-S test statistic for all three data sets showed that at significance level of 0.05, the GP model fits well to the set of observations. We also calculated the Spearman's rank correlation coefficient to determine the strength of the relationship between actual values and and estimated model values. The results showed that at significance level of 0.05, there exists a strong relationship between the two distributions. The results obtained are also promising in terms of predictive accuracy. The custom measures of MMRE ≤ 0.25 and $pred(0.25) \geq 0.75$, as indicative of a good prediction system, holds true in all the three data sets. However, we noted a considerable variation in MMRE values for the three validation data sets. This indicates the sensitivity of GP to changes in the training set and is indicative of the adaptive nature of GP algorithm to deal with heterogeneous data. To have a degree of confidence about the accuracy of future estimates, we resorted to a good rule of thumb for evaluating predictive stability (Subsection 4.3).

In our case, we had one independent and one dependent variable. Hence, the GP algorithm generated good models efficiently within the termination criterion of 200 generations. However, it is common that efficiency and effectiveness of GP drops if the data tables contain hundreds of variables as the GP algorithm then can take a considerable amount of time in isolating the key features [23].

While measures of goodness of fit and predictive accuracy are important, we agree with Mair et al. [16] that these measures are not enough for a practical utility of a prediction system. Therefore, the explanatory value (transparency of solution) and ease of configuration are also important aspects that require discussion. Since the output of a GP system is an algebraic expression, it has the potential of generating transparent solutions; how-

ever the solutions can become complex as the number of nodes in the GP solution increases. There is a tradeoff in having more accurate predictions and less simplicity of the algebraic expressions but we believe that this tradeoff is manageable as achieving accurate models within acceptable thresholds is possible. In terms of ease of configuration, we found that configuring GP control parameters requires considerable effort. Different facets need to be determined, e.g. evaluation function, genetic operators and probabilities, population size and termination criterion to name a few. The parameter tuning problem is time consuming because the control parameters are not independent but interact in complex ways and trying all possible combinations of parameters is practically infeasible [23].

An interesting future work is to explore the relationship between end-user (estimator) and the prediction system to assess if the combination outperforms the individual estimations by either estimator or GP prediction system. Another possible area of future research is to use a different evaluation function (e.g. correlation coefficient) or a multi-objective fitness function that combines both error-based fitness function and correlation coefficient. We also feel that the search for more robust control parameters for tuning the GP algorithm should continue.

9. Conclusions

This paper presented the results of using genetic programming for modeling software reliability growth based on weekly fault count data of three different industrial projects. The results have been evaluated in terms of goodness of fit and predictive accuracy. For evaluating goodness of fit, the K-S statistic and Spearman's rank correlation coefficient gives statistically significant results in favor of adaptability of GP evolved model. The resulting statistics for evaluating predictive accuracy are also encouraging with $pred(0.25)$, MMRE and measure of prediction stability offering results in favor of statistically significant prediction accuracy. However, GP is found to require high set-up times and some degree of experimentation in configuring control parameters. The algebraic expression can also get complex as the number of nodes in the GP solution increases. Therefore, we believe that the practitioners need to be aware of the apparent trade-off between ease of configuration, transparency of solutions and acceptable accuracy of predictions provided by the GP evolved model.

References

[1] T. Bäck, D. Fogel, and T. Michalewicz. *Evolutionary computation 1—basic algorithms and operators*. Taylor & Francis Group, New York, USA, 2000.

[2] E. K. Burke and G. Kendall. *Search methodologies--introductory tutorials in optimization and decision support techniques*. Springer Science and Business Media, New York, USA, 2005.

[3] C. Stringfellow and A. Amschler Andrews. An empirical method for selecting software reliability growth models. *Empirical Software Engineering*, 7(4):319–343, 2002.

[4] S. D. Conte, H. E. Dunsmore, and V. Y. Shen. *Software engineering metrics and models*. B/C, Inc, 1986.

[5] E. O. Costa, S. R. Vergilio, A. Pozo, and G. Souza. Modeling software reliability growth with genetic programming. In *ISSRE '05: Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering*, pages 171–180, Washington, DC, USA, 2005. IEEE Computer Society.

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

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

[8] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley Publishing Company, Inc., 1989.

[9] GPLAB—A genetic programming toolbox for MATLAB. `http://gplab.sourceforge.net`. (Last checked 20 April 2008).

[10] M. Hollander and D. A. Wolfe. *Non-parametric statistical methods*. John Wiley and Sons, Inc., 1999.

[11] IEEE Std 610.12-1990. *IEEE standard glossary of software engineering terminology*, 1990.

[12] Z. Jelinski and P. Moranda. Software reliability research. In *Statistical Computer Performance Evaluation, Ed. W. Freiberger*, pages 465–497. Academic Press, New York, USA, 1972.

[13] J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, 1992.

[14] M. R. Lyu. *Handbook of software reliability engineering*. IEEE Computer Society Press and McGraw-Hill, 1996.

[15] M. R. Lyu. Software reliability engineering: A roadmap. In *FOSE'07: 2007 Future of Software Engineering*, pages 153–170, Washington, DC, USA, 2007. IEEE Computer Society.

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

[17] The MathWorks, Inc. `http://www.mathworks.com`. (Last checked 20 April 2008).

[18] K. Matsumoto, K. Inoue, T. Kikuno, and K. Torii. Experimental evaluation of software reliability growth models. In *Eighteenth International Symposium on Fault-Tolerant Computing, FTCS-18, Digest of Papers*, pages 148–153, Jun. 1988.

[19] J. D. Musa. *Software reliability engineering: more reliable software faster and cheaper*. AuthorHouse, 2nd edition, 2004.

[20] I. J. Myung. Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, 47(1), 2003.

[21] A. P. Nikora and M. R. Lyu. An experiment in determining software reliability model applicability. In *Proceedings of the 6th International Symposium on Software Reliability Engineering*, pages 304–313, Oct. 1995.

[22] E. Oliveira, A. Pozo, and S. R. Vergilio. Using boosting techniques to improve software reliability models based on genetic programming. In *ICTAI '06: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, pages 643–650, Washington, DC, USA, 2006. IEEE Computer Society.

[23] R. Poli, W. B. Langdon, and N. F. McPhee. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008.

[24] R. Sitte. Comparison of software reliability growth predictions: neural networks vs. parametric recalibration. *IEEE Transactions on Reliability*, 48(3):285–291, Sept. 1999.

[25] S. F. Smith. *A learning system based on genetic adaptive algorithms*. PhD thesis, University of Pittsburgh, Pittsburgh, PA, USA, 1980.

[26] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[27] A. Wood. Predicting software reliability. *Computer*, 29(11), 1996.

[28] A. Wood. Software reliability growth models: assumptions vs. reality. In *ISSRE '97: Proceedings of the 8th IEEE International Symposium on Software Reliability Engineering*, Los Alamitos, CA, USA, 1997. IEEE Computer Society.

[29] Y. Zhang and H. Chen. Predicting for MTBF failure data series of software reliability by genetic programming algorithm. In *ISDA '06: Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06)*, pages 666–670, Washington, DC, USA, 2006. IEEE Computer Society.

A. Data sets used in the study

Project 1		Project 2		Project 3	
Week	Fault Count	Week	Fault Count	Week	Fault Count
1	9	1	15	1	3
2	9	2	18	2	12
3	24	3	24	3	18
4	24	4	30	4	30
5	27	5	39	5	60
6	27	6	60	6	93
7	39	7	69	7	138
8	45	8	72	8	186
9	54	9	87	9	210
10	54	10	126	10	240
11	54	11	129	11	258
12	57	12	132	12	279
13	57	13	144	13	297
14	57	14	147	14	312
15	57	15	156	15	348
16	66	16	162	16	357
17	66	17	171	17	372
18	69	18	171	18	399
19	75	19	174	19	414
20	81	20	180	20	429
21	90	21	186	21	459
22	99	22	192	22	486
23	102	23	207	23	519
24	105	24	210	24	540
25	108	25	222	25	552
26	120	26	234	26	570
27	120	27	237	27	588
28	123	28	249	28	612
		29	255	29	624
		30	279	30	630
		31	306		
		32	327		
		33	330		