Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 07, July-2020

Estimation of Effort in Software Projects using Genetic Programming

Mukesh Mahadev K
Department of CSE
B.M.S College of Engineering
Bengaluru, India

Dr. G. Gowrishankar
Department of CSE
B.M.S College of Engineering
Bengaluru, India

***Abstract*— Software effort estimation refers to the estimation of effort that is required in given software project. It starts at the proposal stage and can sometimes continue till the last stages of a software project. Projects normally have a budget, and continual cost estimation is necessary to ensure that spending is in line with the budget. There is need of finding a good model which can establish an accurate relationship between the software a project and cost drivers. It is important for project managers and the researchers working in the domain to explore, analyze and understand the strengths and weaknesses of various software cost estimation methods. This paper focusses on using Genetic Programming for software effort estimation. The implementation involves evolution of individuals for obtaining best results over several generations. Metrics are chosen to evaluate the model based on the literature survey. Standard software engineering datasets are used in this project so that suitability and possible relations that arise could be realized. K-fold validation is used to sum up with more reliable values of evaluation. The design, implementation and result presentation are completed successfully and recorded clearly.**

Keywords*— *Software effort estimation, genetic programming, DEAP.

I. INTRODUCTION

Precise evaluation of software development activities has important consequences in software development management. The software development team will be under considerable pressure if the manager's estimate is too low, the team tries to finish the product quickly and therefore the resulting software may not be completely functional or tested. As a result, the software can contain errors that need to be resolved in a later part of the software process in which the cost of corrective maintenance is greater. In contrary, if the estimation of the manager is too high then the project will be exposed to too many resources and if the organization develops contract software, then too high estimate may lead to failure of secure contract. The software project managers have to control project budgets and be able to calculate the costs of a software development [1].

The principal components of project costs are: Effort costs (the costs of paying software engineers), Hardware costs, Training costs. The cost of effort is the dominant cost. It is very difficult to estimate and control, because it has the most substantial impact on total costs. Software cost estimation is continuous, starting at the stage of the proposal and continuing throughout the life of the project. Projects normally have a budget and a continuous cost estimate is needed to ensure that the expenditure is in line with the budget. Effort can be measured in staff-hours or in staff-months (referred to as man-hours or man-months) [2].

There are many techniques for software cost estimation, some of the most popular approaches are: Non-Algorithmic, Algorithmic. Mathematical equations are used in algorithmic cost estimation techniques. Mathematical equations involved in the algorithmic technique depend on historical data, research and use features such as number of lines of code (SLOC), number of functions, and programming language used in the project, risk assessments, skill-levels, design methodology etc. There have been extensive studies of algorithmic processes and numerous models such as COCOMO models have been developed. Methods such as expert judgment and an analog estimation are contained in the non-algorithms. Some of those non-algorithm-based methods focus on evolutionary methods such as neural networks, genetic programming and fuzzy logic [2][3].

II. LITERATURE SURVEY

Bardsiri, Amid Khatibi et al [4], introduced some of the most important software effort estimation datasets and focused on five datasets. Desharnais, Canadian financial (CF), Maxwell, International software benchmarking standard group (ISBSG) and IBM data processing services (DPS). PROMISE and ISBSG repositories are among the most popular datasets, have so far been used in numerous studies. Thorough statistical analysis of these datasets was performed and was visualized using the above mentioned five datasets. A complete comparison was made to help select a suitable dataset. There was a tangible difference seen between the various repositories when compared with the results and figures of each repository. Study showed that the most crucial component in software effort prediction is the metrics suite and not the algorithm.

Mohammad Azzeh et al[5], made comparative studies using seven datasets. Q-Q plots for each of the seven datasets was generated for all features in order to investigate if their values were normally distributed. As these plots showed quite small deviations from the straight line but was considered to show the characteristic normality. Desharnais, China and Albrecht show the characteristic of normal distribution. Desharnais dataset has more normality than Albrecht dataset.

Ayesha Saeed, Wasi Haider Butt et al [6], surveyed 10 effort estimation models, most of these estimation models used public dataset. These models were efficient and, in some way, validating through empirical data. Azzeh, M suggested Combination of Support vector machine and radial basis neural networks. Rijwani, resolved effort estimation problem through ANN. Non-linear Regression Model along with Multi-layer perceptron (MLP) neural network model was proposed by A.B. Nassif. Classical and fuzzy analogy-based models and ensembles were presented. This survey paper showed that machine learning methods are the most frequently used models. MRE and MMRE are the most frequently used accuracy metrics. The results show that there is no single perfect method which is suitable for all the datasets.

Rohit Kumar Sachan, Ayush Nigam et al [7], used simplified form of genetic algorithm which was used for optimizing values. They proposed an experiment in which COCOMO model is used, the value of A and B of the basic COCOMO model was optimized using simplified Genetic Algorithm. In this experiment NASA dataset was used. The computed values of effort for basic COCOMO model was 48.8360 and for simplified GA was 6.7125. The result showed that effort estimated by the proposed simplified GA is better as compared to basic COCOMO effort.

Y. Shan, R. I. McKay [8], used Grammar Guided Genetic Programming (GGGP) to fit nonlinear models to a dataset of past projects, their aim was to determine appropriate metrics sets and improve the prediction. 5 GP runs were generated, on the same training and testing datasets, 5 linear and log-log equations were also derived using standard regression techniques. The training error of log regression was larger than linear regression. The mean square error (MSE) of GP was better compared to Linear and Log regression. The GP model does not generalize perfectly so there is some loss of accuracy on the test sets, it is still far more accurate in testing error than the other two regression model. The results showed that GP models perform better than traditional regression models.

Colin Burgess and Martin Lefley [9], This project enlists the importance of software cost estimation such as the effects of better resource allocation due to improvement in accuracy of effort prediction models. It provides a clear explanation of software estimation metrics and provides reasons for choosing some of them. It discusses about the usage of Artificial Neural Network for effort prediction by stressing upon the problems faced when configuring neural networks and how it tends to counteract their performance in terms of accuracy. The results of this paper indicate MMRE values ranging between 37.95 and 52.12. It compares these values with that of artificial neural network and prove genetic programming yields better results. It also compares these two prediction systems based on two factors: Transparency of solution and Ease of configuration. It states that GP can produce very transparent solutions in the sense that solution is an algebraic expression. It illustrates that the configuration of ANN and GP requires a great level of expertise. It concludes by highlighting the need of further investigation, particularly to explore the effects of various parameters on the models in terms of improving robustness and accuracy.

DEAP is a framework that combines the flexibility and power of the Python programming language. It contains transparent Evolutionary Computing components which facilitates rapid prototyping and testing of new Evolutionary Algorithm ideas. It encourages creativity through simplicity and explicit algorithms.15 examples of OpenBEAGLE plus 20 more examples are implemented by DEAP with 10 times less lines of code. This framework includes more than 35 applications [10].

III. GENETIC PROGRAMMING

Genetic Programming (GP) is an extension of Genetic Algorithm (GA). The basic idea of GA is based on Darwinian theory of evolution, which says that genetic operations between chromosomes eventually leads to fitter individuals which are more likely to survive. Thus, over a long period of time, the population of the species as a whole improves. Genetic Algorithm is a random search algorithm based on natural selection and genetic mechanism it starts from a set of random generation of initial solution called "population". Each individual in the population is a solution to the problem. Chromosome is a string of symbols, such as a binary string, these chromosomes constantly evolve in the subsequent iterations. Crossover and mutation operations are performed on previous generation of chromosomes to produce offspring. Crossover operation involves combining the characteristics of the parents in the new generation individuals. Crossover reflects the idea of information exchange. Mutation involves changing the value of one of the string structure data(chromosome) for the selected individual. Genetic algorithm selects part of the offspring and eliminates some offspring according to the fitness value to form the new generation and to maintain a constant population size. "Fitness" is to measure the chromosomes in each generation as good or bad. The optimal or sub-optimal solutions of the problem is the best chromosomes that are produced after some generations. The fitness function is also known as the evaluation function. The fitness function is always non-negative and is determined according to the objective function. It is the standard for distinguishing the individuals of the population [11].

GP adds more advantages to the ones offered by other evolutionary algorithms is considered one of the evolutionary algorithms that hold all the advantages offered by evolutionary algorithms and adds several more. One of the advantages of GP is, it is not prone to local optimal values like the neural networks. GP is a global search technique and it is less likely to get stuck in the local optimum. GP has the capability to perform the task of feature extraction algorithm and eliminate unrelated features, of the Modelling problem. It represents a tree in which less important features would appear deeper in the tree and more important features can appear near the root[5]. GP can operate on portion of data to extract significant rules.[6]

A. Represntation of GP

In GP, programs are usually represented as a variable sized tree structure. This type of representation allows a variety of models to be developed. A tree consists of nodes and terminals. In every terminal node, there is an operand and in every node there is a function. Trees can be easily evaluated in a recursive manner. This way we can evolve mathematical models in a very simple way [14].

B. Units

Two well-known evaluation criteria are used to assess the degree of accuracy to which the estimated efforts match actual efforts:

- MMRE: MMRE is the mean of absolute percentage errors.

$$\text{MMRE} = \sum_{i=1}^{n}\left(\frac{|E-\hat{E}|}{E}\right)\frac{100}{N}$$

  Where E is the actual effort and $\hat{E}$ is the predicted effort, and N is the number of projects.
- PRED (25): Pred(25) is the percentage of predictions that fall within 25 percent of the actual value [15].

IV. SYSTEM DESIGN

The system consists of four stages:

- Data Input.
- Genetic Programming Configuration.
- Genetic Programming execution
- Evaluation

A. Data Input

The datasets used in the project commonly called “Desharnais”, “China”, “Albrecht”. They contain the information related to software projects derived from different software projects. They are publicly available datasets and can be used to evaluate performance of new software cost estimation models. The dataset is in attribute-relation file format (.arff) format. Parsing of these files is done using parser packages that are available in python. The datasets also consist of some features which are not required for estimation of effort are eliminated. Only some of the input variables are used in the effort estimation. The dataset is parsed and the input is made available as numeric values for further use.

B. Genetic Programming Configuration

User has to perform preparatory steps for a genetic programming which are used in configuration and execution of programs(individuals). User defined primitives and terminals are contained in a primitive set, which are primarily used in the algorithm. The five steps which are treated as preparatory steps are mentioned below:

- Primitive Set: This set consists of the operators and input variables which are found as nodes of individual programs. The operators used in the project are basic mathematical operators such as addition, multiplication, subtraction and division. The terminals are classified in two types: the constants and the arguments. The constants remain the same for the entire evolution while the arguments are the program inputs. The input variables of the dataset act as the terminals. The input variables are all numeric values and can be operated on by basic mathematical operators to build expressions.
- Fitness function: The fitness is implemented considering one of the metrics that are used for evaluation of software cost estimation techniques. The individuals of the population are evaluated based on this function.
- Parameters: Size of the population and the number of generations are the main parameters that are decided for efficient results. They are varied according to the convergence of the resulting accuracy.

C. Genetic Programming Execution

The genetic programming execution is the execution of the algorithm with the given configuration parameters. The execution is the looping of a set of genetic operations. The set of steps involved can be listed as:

- Population Initialization: The population is a set of individual programs or tress that can parsed to get an expression.
- Evolution through algorithm execution: The algorithm is executed as per the provided parameters. The execution of the algorithm includes fitness evaluation, selection, crossover, mutation, reproduction. The reproduction stage incorporates the new individual into next generation. The number of times the algorithm is executed is dependent on a parameter passed to an execution function.

D. Evaluation

The evaluation of the predicted results is done based empirically evident software cost estimation metrics. The evaluation metric is chosen after substantial exploration of the literature and is discussed in the introduction section of this report. The detailed information about the selection of the metric can be found in part in the implementation section as well.

Workflow

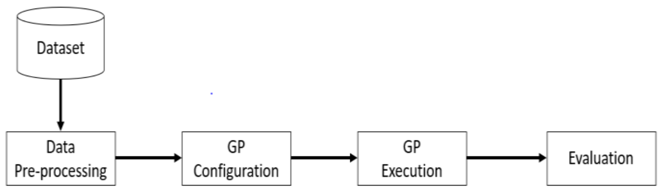

Figure 1: Pipeline of Execution

The Figure 1, Pipeline of Execution shows the complete workflow of execution on different stages. The dataset is obtained from a repository of publicly available software engineering datasets. The dataset goes though the pre-processing stage where some the features are selected for carrying out the estimation procedure. The subsequent phase is the configuration of the Genetic Programming. The primitive set i.e. the set of operators and operands which act as nodes for individuals of a population are selected. The GP execution involves looping through the set of genetic operators such as selection, crossover and mutation, to obtain new generation of individuals. The individuals are evaluated constantly for fitness and fitter individuals are propagated to next generations. The evaluation phase involves the calculation of the metrics which are chosen for evaluation of the estimation model.

V. IMPLEMENTATION

The project is implemented in Python programming language and popular evolutionary algorithm framework called DEAP. The following sections will describe about the implementation in details.

DEAP (Distributed Evolutionary Algorithms in Python): DEAP is a framework used for realizing all the evolutionary computation process. The DEAP framework is built over the Python programming language that provides the essential components that make each and every step of genetic programming implementation easy.

DEAP is made up of two main components: a creator and a toolbox. The 'creator' helps in creating classes that represent the individual programs. New classes can also be built from any built-in python datatype like list, set, 'PrimitiveTree'. 'ToolBox' is a container to store primitive set and genetic operators. The users have several options to populate the toolbox to implement the genetic programming with flexibility.

A. Data Parsing

The information about the dataset is discussed in detail in the introduction part. The data is made of only numeric attributes. The data format of the files is attribute-relation format file. The data is parsed using a built-in parser of the scipy.io package.

B. Data Pre-processing

Selection and Elimination of features, the datasets contain many features some of which can be used for effort estimation and some prove to be unnecessary for the predicting the effort. So, the pre-processing stage involves identification and elimination of features. This step proves to be important, as most of the features in the dataset are numeric, they may play a deciding role in the predicting the effort by the model. The chance of deviation of results could be the effect of including such features. Some features represent names and, such features are eliminated

There are four major steps in achieving the implementation of evolutionary computations using DEAP:

- Primitive set Configuration
- Individuals and population Initialization
- Genetic operations Setup
- Algorithm execution

The individual steps listed above are discussed in detail below.

- ➢ **Primitive Set Configuration:** The individuals of the genetic programming are the individual programs which are represented as trees. The trees are made up of nodes. The nodes consist of operands or operators. Operands are found on the terminal nodes and operators can be found in the non-terminal nodes. The set of the operators and operands together is called the 'Primitive Set'. The operands usually are either the input features in the datasets or constants. The choice of the operators depends on the operands. The datasets chosen for this project contain only numeric features, so simple mathematical operators are considered and added to the primitive set.
- ➢ **Individual and Population Initialization**: The individual trees and a population are created in a preliminary step in genetic programming. The DEAP framework provides a package to create individuals and population. The individuals are created 'creator' package. The individual creation and population initialization is done using the 'toolbox'. The creator package creates individuals using built-in types. 'Primitive Tree' is data type used for individuals. The population is made up of a fixed number of individuals which is specified by the user.
- ➢ **Genetic Operations Setup**: Genetic Operations form the core part of genetic algorithm. These operations function to produce newer off-springs which in-turn produce newer generations. The evolution of new generations is supported by fitter individuals in the current generation. This mechanism can be implemented easily with DEAP framework. The genetic operations are added successively to the toolbox. The elements in the toolbox are used during the execution of algorithm.
- ➢ **Genetic Algorithm Execution:** The iterative transformation of the population is executed inside the main generational loop of the run of genetic programming. The algorithms module in DEAP contains some specific algorithms in order to execute common evolutionary algorithms. The method used here are for convenience than reference as the implementation of every evolutionary algorithm may vary infinitely. Algorithm in this module use operators registered in the toolbox. Generally, the keyword used are mate() for crossover, mutate() for mutation, select() for selection and evaluate() for evaluation of fitness. DEAP provides a straight-forward function called eaMuPlusLambda() which takes in as parameters population, toolbox, number of individuals in the next generation (called 'Mu'), number of children to produce at each generation (called 'Lambda'), probability of crossover, probability of mutation, number of generations and etc.

VI. RESULT

Genetic Programming yields different results based on parameters values. The parameter values provided to the algorithm are as listed below.

Parameter	Value
Population size	300
Number of Generations	100
Mutation Probaliltity	0.5
Crossover Probability	0.5

Table 1: Genetic Programming Parameters

The results of the application of implementation on three different sets can be concluded using the below table.

Dataset Name	MMRE	Pred(25)
Desharnais	11.970	93.827
China	13.264	87.374
Albrecht	55.493	58.333

Table 2: Results

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 07, July-2020

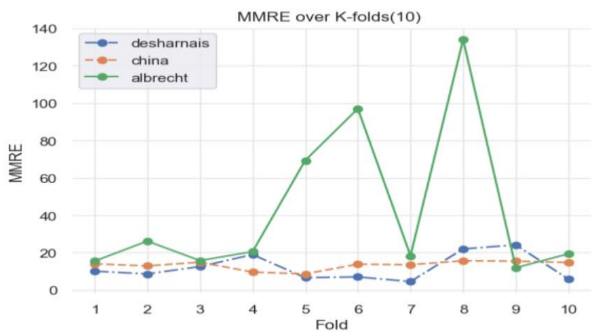

Figure 2: Graph showing the MMRE values over different folds of the K-fold validation

The graph provides a clear picture of how the MMRE values range between the different folds of the K-fold validation. The deviation of the MMRE values is seen to be more when the dataset is small (‘Albrecth’ dataset). The MMRE values for ‘Desharnais’ dataset show more deviation compared to the ‘China’ dataset which the biggest among the three. The graph hiddenly illustrates the relation of the deviation with the size of the datset.

VII. CONCLUSION AND FUTURE WORK

Genetic Programming as an approach to software effort estimation has been explored and implemented. Systematic study of Genetic programming and its adaptation for the task of effort estimation has been carried out and reported clearly as part of the work. The steps include understanding of bio-inspired evolutionary algorithms, exploration for the metrics to evaluate the model, implementation of the algorithm, presentation of results. The implementation of the genetic programming model using the Metrics like Pred25 and MMRE were considered for evaluation of the prediction model. Different datasets with varying sizes were used to test the predictive model. The K-fold validation results in more reliable accuracy measures. The results indicate that the model built using GP shows good accuracy values. The size of the dataset is also a factor which decides the deviation of the accuracy values over different folds of validation. The amount of deviation is observed to be high when the dataset size is smaller compared to the bigger ones. The results warrant that GP can be further explored on its different characteristics for better results.

The results of the GP on smaller dataset are very low which can be further worked upon. The future works could aim to attain better configurations of the GP setup which give more accuracy. Comparisons with other predictive model building methods can be considered for future research works which yield a comparable accuracy. More robust metrics could be designed for evaluating the accuracy of the effort estimation models as the currently used ones. Comparative studies showing the difference in accuracy due to the change in configuration parameters of the GP will help future researchers in choosing them wisely.

REFERENCES

[1] K. Srinivasan and D. Fisher, "Machine learning approaches to estimating software development effort," in IEEE Transactions on Software Engineering, vol. 21, no. 2, pp. 126-137, Feb. 1995, doi: 10.1109/32.345828.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73. The Comparison of the Software Cost Estimating Methods. (With *Liming Wu)*.Link(https://www.computing.dcu.ie/~renaat/ca421/LWu1.html).

[3] G. Gabrani and N. Saini, "Effort estimation models using evolutionary learning algorithms for software development," 2016 Symposium on Colossal Data Analysis and Networking (CDAN), Indore, 2016, pp. 1-6, doi: 10.1109/CDAN.2016.7570916.

[4] Bardsiri, Amid Khatibi, Seyyed Mohsen Hashemi, and Mohammadreza Razzazi. "Statistical analysis of the most popular software service effort estimation datasets." *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 7.1 (2015): 87-96.

[5] Azzeh, M. A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation. *Empir Software Eng* 17, 90–127 (2012).

[6] Saeed, Ayesha, et al. "Survey of software development effort estimation techniques." Proceedings of the 2018 7th International Conference on Software and Computer Applications. 2018.

[7] Sachan R.K., Nigam A. et al. Optimizing Basic COCOMO Model Using Simplified Genetic Algorithm (2016) *Procedia Computer Science*, 89, pp. 492-498. https://doi.org/10.1016/j.procs.2016.06.107

[8] Y. Shan, R. I. McKay, C. J. Lokan and D. L. Essam, "Software project effort estimation using genetic programming," IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions, Chengdu, China, 2002, pp. 1108-1112 vol.2, doi: 10.1109/ICCCAS.2002.1178979.

[9] Burgess, C. J. and Martin Lefley. “Can genetic programming improve software effort estimation? A comparative evaluation.” *Inf. Softw. Technol.* 43 (2001): 863-873.

[10] Fortin, Félix-Antoine & De Rainville, François-Michel & Gardner, M.A. & Parizeau, Marc & Gagné, Christian. (2012). DEAP: Evolutionary algorithms made easy. Journal of Machine Learning Research, Machine Learning Open Source Software. 13. 2171-2175.

[11] S. Liu and X. Xu, "Distributed Database Query Based on Improved Genetic Algorithm," 2016 3rd International Conference on Information Science and Control Engineering (ICISCE), Beijing, 2016, pp. 348-351, doi: 10.1109/ICISCE.2016.84.

[12] F. S. Alaa and A. Al-Afeef, "A GP effort estimation model utilizing line of code and methodology for NASA software projects," 2010 10th International Conference on Intelligent Systems Design and Applications, Cairo, 2010, pp. 290-295, doi: 10.1109/ISDA.2010.5687251.

[13] J. K. Kishore, L. M. Patnaik, V. Mani and V. K. Agrawal, "Application of genetic programming for multicategory pattern classification", *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 242-258, September 2000.

[14] Willis, Mark & Hiden, Hugo & Marenbach, P. & McKay, Ben & Montague, Gary. (1997). Genetic programming: An introduction and survey of applications. 314 - 319. 10.1049/cp:19971199.

[15] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," in IEEE Transactions on Software Engineering, vol. 23, no. 11, pp. 736-743, Nov. 1997, doi: 10.1109/32.637387.