Journal of Theoretical and Applied Information Technology
31st October 2017. Vol.95. No 20
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A STUDY TO IMPROVE THE SOFTWARE ESTIMATION USING DIFFERENTIAL EVOLUTION ALGORITHM WITH ANALOGY

[1] THAMARAI. I , [2]DR. MURUGAVALLI. S

[1] Research Scholar, Sathyabama University, Department of Computer Science, Chennai-119, India

[2] Research Supervisor, Sathyabama University, Department of Computer Science Chennai-119, India

E-mail: [1] ilango.thamarai@gmail.com, [2]murugavalli26@rediffmail.com

ABSTRACT

Software effort estimation is the process of calculating the effort required to develop a software product based on the input parameters that are partial in nature. Estimation requires information about project scope, resources available, process requirements and many other factors associated with a software product development. Inaccurate estimation leads to financial loss and delay in the projects. Due to the intangible nature of software, most of the software estimation process is not reliable. But there is a strong relationship between effort estimation and success of project management activities. Hence the aim of the research is to propose a new software estimation model that combines the Analogy concept with Differential Evolution Algorithm that is more efficient than the existing methods. Several methods for software effort estimation are discussed in this paper including the widely used and metrics used for evaluation. The use of Differential Evolution in the estimation is dealt in detail. A new model for estimation using Differential Evolution Algorithm called DEAPS is proposed and its advantages discussed. The proposed model is checked with the popular data sets namely Desharnais datasets, Albrecht dataset and COCOMO dataset. The results are compared with previous findings and the results clearly show that the proposed method is better than the existing methods. The proposed Model can be used to minimize the errors in the software estimation which is a crucial step in the software development process. Hence the financial loss and delay in the completion of project may be avoided

Keywords: *Software Effort Estimation Methods, Algorithmic and Non-Algorithmic Models, Evolutionary Computational Methods, Differential Evolution Algorithm*

1. INTRODUCTION

Software project planning is one of the most important activities in a software development process. It is a most important task but most difficult and complicated step in the software product development. Planning largely depends on the effort estimation and this requires many parameters such as size, number of persons, schedule, etc. It is a difficult and complicated task. Linda M. Laird (2006) enumerates the reasons for the inaccurate effort estimation [1]. In [2], M. Jorgenson and D.I.K. Sjoberg (2001) demonstrated how the software effort is affected by the client's expectations about cost. Ning Nan and Donald E. Harter (2009) emphasize the role of budget and schedule pressure [3]. In [4], Magne Jorgenson and Stein Grimstad (2011) made a detailed study on how irrelevant information can affect the estimation of software. According to their research, the field settings that led to the irrelevant information have a very small impact than the artificial settings for doing experiments. Tim Menzies et al (2013) suggested to create clusters and proved that the best are that which are near the source data but not from the same source as the test data [5]. The general principles of software effort estimation were explored by Ekrem Kocaguneli et al (2012) to guide the design for estimation [6]. Many factors are considered to estimate the software cost and effort and the most important factors are size of the project, number of persons and schedule. Their research emphasizes that the estimation can be improved by dynamic selection of nearest neighbor with small variance.

Prediction of software effort is a very difficult and complicated task. Software is intangible in nature and hence the measurement of progress is the software process is very difficult to access. Also the requirements of software project change continually which causes the change in estimation. Inaccurate estimation of effort is the main cause of software project failures. In [7], Magne Jorgensen et al (2009) suggested that the type of individual lesson learned processes may have effect on the accuracy of estimates. In [8], Magne Jorgenson and Martin Sheppard (2007) present a systematic review of various journals and concluded that the properties of dataset impact the result when evaluating the estimation. In [9], Karel Dejaegar et al (2012) present an overview of the literature related to software effort estimation. The paper gives detailed study with different processing steps and addresses many issues like quality of the data, missing values etc. In [10], Tim Menzies et al (2013) evaluated the lessons that are global to multiple projects and local to particular projects in software effort estimation. Nicolos Mittas and Lefteris Angelis (2012) proposed a statistical framework based on multiple comparison algorithms to rank several cost estimation models [11]. In [12], Mark Harman and Afshin Mansouri (2010) proposed the application of search based optimization methods for the software effort estimation. The advantages of using search based optimization are listed as robustness, scalability and powerfulness. In [13], Ekrem kocaguneli et al (2013) proposed a tool called QUICK TOOL that reduces the complexity in data interpretation. The tool is suitable for small data sets.

To improve the uncertainties in cost assessments, Magne Jorgensen (2005) provided evidence based guidelines in [14]. Some of the important guidelines include not to rely solely on unaided, intuition based uncertainty assessment process but to apply structured and explicit judgment based process. Thus it can be seen that many aspects has to be considered for Software Effort Estimation Models. Accurate effort estimation is critical for both developers and customers and is crucial for the success of the project. The effort estimation methods available so far have both merits and demerits. There is a strong relationship between effort estimation and the project management activities which affect the success of a software project. Hence developing an efficient software effort estimation model is an urgent problem and of great practical importance. The weaknesses in the existing effort estimation methods have provided the motivation to develop a new software estimation model that combines the Analogy concept with Differential Evolution Algorithm that is more efficient than the existing methods

The aim of this paper is to give a detailed description about various software effort estimation methods and to emphasize the use of the Evolutionary Computation Algorithms for software estimation. It proposes a model called DEAPS that combines the concept of Analogy with Differential Evolution Algorithm. The paper is organized as follows: Section 2 consists of brief discussion on Materials and Methods on software estimation methods. In Section 3, we give the results and Section 4 discuss about the results of the model. Section 5 is the conclusion and recommendation for future work.

2. SOFTWARE ESTIMATION METHODS

Many traditional methods are used for Software Estimation ranging from Expert Judgment method, Function Point method, COCOMO, SLIM Model, Case Based Reasoning Model to the recent methods that uses Neural Networks, Fuzzy Logic, Genetic Algorithm, Genetic Programming, Particle Swarm Optimization etc. The Software Effort Estimation models are primarily divided into four main divisions. The foremost is the Expert Judgment Method, where the effort is estimated by experts in the field. Algorithmic models are mainly based on the mathematical formulas. Some of the Algorithmic Models are FP (Function Point), COCOMO (Constructive Cost Model) and SLIM (Software Life cycle Management model). These models depends on various parameters like LOC (Lines of code), Complexity, Number of Interfaces etc.

The limitation of Algorithmic models led to the Non-algorithmic models. Case Based Reasoning (CBR) is a popular Non-algorithmic method. Analogy is a CBR method. Also with the advent of soft computing techniques, new methodology of evolutionary computation came into existence. In this, many Machine Learning methods are used including GA, GP and DE. These methods are discussed in the following sections. The research work by Mudasir Manzoor et al (2015) evaluates the performance of Re-UCP model and compares the results with the UCP and e-UCP method of software effort estimation [15]. The accuracy of results were validated by using MRE (Magnitude of Relative

Error), MMRE (Mean Magnitude Relative Error), MdMRE (Median of Magnitude Relative Error) tools to check the error rate and PRED(10%) and PRED(20%) to find out the accuracy of Re-UCP software effort estimation method. The observations made from the results are based on the comparison of Re-UCP, e-UCP and UCP models of software effort estimation. The deviation percentage calculated using Re-UCP justifies the improved performance of Re-UCP method of software effort estimation in comparison with UCP and e-UCP methods of software effort estimation.

Subimtsha and Kowski Rajan (2014) used different types of techniques including techniques such as Multilayered Perceptron Network, Radial Basis Function Neural Network, Support Vector Machines and Particle Swarm Optimization [16]. A simple technique like regression is found to be well suited for software effort estimation which is particularly interesting. The research by Surfyan Basri et al (2015) introduces a new change in the effort estimation approach that is able to use different estimation techniques for different states of software artifacts [17]. The outcome of this research is an effort estimation approach for software development phase using the extended version of the static and dynamic analysis techniques. All the approaches have their own merits and demerits. It should be noted that there is not a single method which can be said to be best for all situations.

Tuan Khanh Lee Do et al (2010) have proposed an approach to filter noise in the historical projects to increase the accuracy of Analogy based Estimation [18]. Noise refers to the data corruptions that cause a negative impact on the performance of an estimation model. Desharnais dataset, Maxwell dataset and ISBSG (International Software Benchmarking Standard Group) dataset were used. They introduced a metric called EID (Effort Inconsistency Degree) that is used to measure the degree that the effort of a project is inconsistent from similar projects. This has led to improvement of the accuracy of estimation by ABE. In [19], Hathaichanok Suwanjang and Nakornthip Prompoon (2012) proposed a framework for developing a model for software cost estimation based on a relational matrix of a project profile for their study. They chose a human resource management related company, which has many software modification projects. The model is based on the multiple regression analysis and analogy method. Tridas Mukhopadyay et al (1992) proposed a analogy based model called ESTOR (Estimator) based on the verbal protocols of a human expert [20]. The results were compared with Function Point and COCOMO models and proved to be more accurate.

BRACE (Bootstrap based Analogy Cost Estimation) was proposed by Stamelos et al (2001) in [21]. It applies analogy based techniques and re-sampling methodologies. In [22]. Myrtveit and Stensrud (1999) have proved that both Analogy and Regression techniques improve the accuracy of estimation. Li. Y. F et al (2009) also have also made a detailed study of non-linear adjustment for Analogy based Estimation [23]. Jianfeng Wen et al (2009) proposed a new method of using Principal Components Analysis (PCA) to extract the features and use Pearson Correlation coefficients between them [24]. Three benchmark datasets, namely, Desharnais, COCOMO and NASA were used in experiments whose results showed significant improvement in prediction accuracy and reliability. Compared with the other two categories of the software estimation, namely, Expert Judgment and Algorithmic models, Analogy based effort estimation performed the best in 60% of the cases reported in published studies.

Some of the research favors the combination of more methods that has been proved successful. In [25], Chao-Jung Hsu et al (2010), integrated several software estimation methods and assigned linear weights for combinations. This model is very useful in improving estimation accuracy. In the work of Shama Kousar Jabeen and Arthi (2014), the results using three function point based effort estimation models are analyzed [26]. They have also compared MMRE, MdMRE, MRE values by training the dataset using neuro-fuzzy logic based machine learning approach which overcomes the problems present in the traditional methods. Reviews of some popular data mining techniques used in software effort estimation have been presented by Mohita Sharma and Neha Fotedar (2014) in [27]. Effort has been calculated on the basis of MMRE value. Hence it can be seen that the most commonly used metrics in the estimation of software are MRE, MMRE, MdMRE and Pred(0.25).

2.1 Evolutionary Computation Models

The use of Evolutionary Computation Model is suggested recently to estimate the software projects. They have the advantage of handling large search spaces. The basic idea is the Darwin's theory of evolution according to which the genetic operations between chromosomes lead to the survival of the fittest individuals.

2.1.1 GA (Genetic Algorithm)

GA is a search based optimization algorithm to get an optimal solution. It is an evolutionary computation method. The main issues related with GA are the representation of solution, selection of genetic operators and choosing the best fitness function. GA can efficiently search through the solution space of complex problem. In [28], GA is used for project selection in the form of two models using Analogy by Y.F.Li et al (2010). The steps involved are encoding, population generation, fitness function evaluation, cross over, mutation, elitism and stopping criteria. The two real world data sets Desharnais data set and Albrecht data set are used for the experiments. The first model uses GA to select appropriate projects subsets named as PSABE (Project Selection in Analogy Based Estimation). The second model which is based on feature weights is called FWABE. The results are better than the other software estimation models.

2.1.2 GP (Genetic Programming)

GP is also an evolutionary computation method that works on tree data structure. The Non continuous functions are very common in software engineering application and Genetic Programming can be effectively used in such situations. Using a tree based representation in Genetic Programming requires adaptive individuals and domain specific grammar. GP begin with a population of randomly created programs. Each program is evaluated based on fitness function. Unlike Genetic Algorithm, mutation operation is not needed in GP because the crossover operation provides point mutation at nodes. The process of selection and crossover of individual continues till the termination criteria are satisfied. Colin J.Burgess and Martin Lefley (2000) analyzed the potential of G.P in Software Effort Estimation in terms of accuracy and ease of use [29]. Their research was based on Desharnais data set of 81 software projects. The authors prove that the use of GP offer improvement in accuracy but this improvement depends on the measure and interpretation of data used in the project.

2.1.3 DE (Differential Evolution)

Differential Evolution is an important evolutionary computation method in recent days that can be used to improve the exploration ability. Differential evolution (DE) is a method that optimizes a problem, iteratively to improve a solution .Differential Evolution is similar to Genetic Algorithm, but it differs in the sense that distance and direction information from the current population is used to guide the search process. DE performs well than any other contemporary algorithm and it is proved that it offers good optimization due to higher number of local optima and higher dimensionality. There are many types of DE such as Simple DE, Population based DE, Compact DE, etc. In a simple DE algorithm, an initial population is created by random set of individuals. For each generation, three individuals say x1, x2 and x3 are selected. An off spring x′ off is generated by mutation as

$$x'off = x1+F(x2-x3)$$

F is a scale factor. Then crossover is performed based on some condition.

2.2 Analogy with Differential Evolution Algorithm

Analogy is the most accepted method for the estimation of software. In our proposed method, Differential Evolution Algorithm is used with Analogy to improve the accuracy of effort estimation.. Differential Evolution Algorithm which is an Evolutionary Computational Approach is based on the biological evolutions of organisms. Swagatham Das et al (2011) made a detailed study on Differential Evolution Algorithm [30]. The DE technique is a population based approach such as the genetic algorithms making use of the identical operators such as the crossover, mutation and selection. The vital difference in configuring the superior solutions relies on the fact that the genetic algorithms are invariably dependent on the crossover operation while DE is basically based on the mutation function.

The major task invariably depends on the divergences of arbitrarily sampled couples of solutions in the population. This new technique employs the mutation function as a search mechanism and the selection function to manage the search toward the potential zones in the search space. Further, it utilizes a non-uniform crossover which is capable of taking the child vector parameters from one parent more frequently than in the case from others. The following Figure 1 gives the framework for using Differential Evolution Algorithm with Analogy for improving software effort estimation

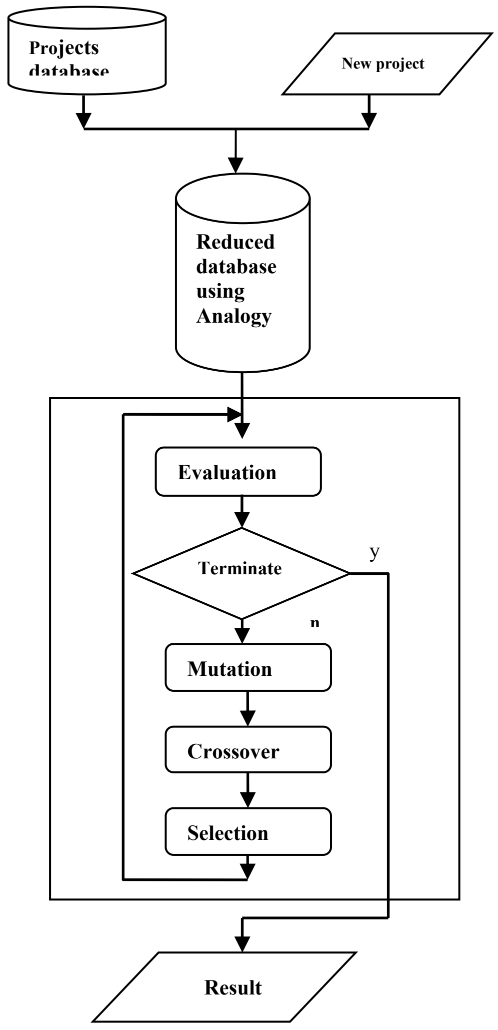

Figure 1: Differential Evolution with Analogy for SEE

Here, we propose a new model called DEAPS (Differential Evolution in Analogy for Project Selection). The estimation is done in two stages. In the 1st stage, there is a reduction of historical database to a set of most similar projects using Similarity Measure. In the 2nd stage, DE is applied to retrieve the most relevant project from which the effort required for a new project is estimated. The main advantages of DE are its ability to provide multiple solutions. It can be easily applied to real problems despite noisy and multidimensional space. It is simple but has effective mutation process that ensures search diversity.

3. RESULTS

The DEAPS Model is tested with the most popular datasets. The input is the project parameters from the Albrecht dataset, Desharnais dataset and COCOMO dataset whose values are slightly changed. It is found that by using the DE Algorithm with the concept of Analogy method the most relevant project is retrieved by which the accuracy of effort needed is analyzed easily. The metrics that are commonly used for evaluating a software estimation model are given below which is followed by the results.

3.1 Performance Evaluation Metrics

Metrics are used for the validation of Effort Estimation Models. The most commonly used metrics are given below:

3.1.1 MRE (Magnitude of Relative Error): Relative Error is the difference between the actual and estimated value. MRE is the absolute value of the relative error.

$$MRE = \frac{|A - E|}{A}$$

A is the Actual Effort and E is the Estimated Effort

3.1.2 MMRE (Mean Magnitude of Relative Error):

MMRE is the percentage of the MRE over an entire dataset

$$MMRE = \sum_{i=1}^{i=n} \frac{|A_i - E_i|}{A_i} * \frac{100}{n}$$

Ai is the Actual Effort and Ei is the Estimated Effort of the ith project, n is the number of projects.

3.1.3 Pred(q):

The prediction level pred(q), is the percentage of prediction that falls within a specified percentage (q%) of the actual value.

$$pred(q) = \frac{p}{n}$$

p is the number of projects whose MRE value is less than or equal to q. The commonly used metric is pred(0.25) is the percentage of predictions that is less than 25% of the actual value

$$pred(.25) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|A_i - E_i|}{A_i}\right) <= 0.25$$

The results of the DEAPS model with the various datasets are given in the following Table 1, 2 and 3.

Table1:Results of DEAPS Model on Albrecht dataset

S. No	Project Id	Estimated Effort	Actual Effort	MRE	Pred (0.25)
1	1	5152	7124.3	0.2768	0
2	4	3829	3154.8	0.2137	1
3	5	2149	1915.4	0.1220	1
4	6	2821	4181.8	0.3254	0
5	7	2569	2977.1	0.1371	1
6	15	4977	3536.1	0.4075	0
7	17	3192	3360.4	0.0501	1
8	19	4494	5799	0.2250	1
9	20	840	850.1	0.0119	1
10	23	5775	5697.1	0.0137	1
11	27	3542	4326.9	0.1814	1
12	28	4277	7035	0.3920	0
13	32	710	484.9	0.4642	0
14	33	2429	3043.5	0.2019	1
15	36	9135	6357.4	0.4369	0
16	39	847	640.9	0.3216	0
17	40	8050	7477	0.0766	1
18	43	2174	2129.7	0.0208	1
19	45	6699	5479.6	0.2225	1
20	47	4004	3028.3	0.3222	0
21	52	3136	1708.1	0.8360	0
22	54	2583	2122.3	0.2171	1
23	56	8232	6743.6	0.2207	1
24	57	3276	2290.9	0.4300	0
25	58	2723	1774.8	0.5343	0
26	61	2926	3101	0.0564	1
27	64	1603	2050.5	0.2182	1
28	68	1267	930.8	0.3612	0
29	70	1155	829.8	0.3919	0
30	71	546	689.8	0.2085	1

Table2:Results of DEAPS Model on Desharnais Dataset

Project Id	Estimated Effort	Actual Effort	MRE	Pred(0.25)
1	100	102.4	0.0234	1
2	94	105.2	0.1065	1
3	25	11.1	1.2523	0
4	15	21.1	0.2891	0
5	35	28.8	0.2153	1
6	6	10	0.4000	0
7	10	8	0.2500	1
8	1	4.9	0.7959	1
9	20	12.9	0.5504	0
10	20	19	0.0526	1
11	10	10.8	0.0741	1
12	10	8	0.2500	1
13	9	7.5	0.2000	1
14	10	12	0.1667	1
15	1	0.5	1.0000	0
16	12	15.8	0.2405	1
17	15	18.3	0.1803	1
18	6	8.9	0.3258	0
19	30	38.1	0.2126	1
20	51	38.1	0.3386	0
21	5	3.6	0.3889	0
22	11	11.8	0.0678	1
23	1	0.5	1.0000	0
24	2	6.1	0.6721	0

Table 3 : Results of DEAPS Model on COCOMO Dataset

S. No	Project Id	Estimated Effort	Actual Effort	MRE	Pred (0.25)
1	1	671.271	2040	0.6709	0
2	16	27.0555	40	0.3236	0
3	17	10.6371	9	0.1819	1
4	22	236.4989	724	0.6733	0
5	28	61.878	98	0.3686	0
6	29	5.5143	7.3	0.2446	1
7	30	4.7255	5.9	0.1991	1
8	32	586.189	702	0.1650	1
9	35	55.3676	82	0.3248	0
10	37	58.118	47	0.2366	1
11	38	9.5728	12	0.2023	1
12	40	4.1316	8	0.4836	0
13	45	54.2911	106	0.4878	0
14	46	112.7067	126	0.1055	1
15	48	777.5163	1272	0.3887	0
16	49	55.3096	156	0.6455	0
17	59	62.9629	70	0.1005	1
18	60	23.3272	57	0.5908	0
19	62	16.7788	32	0.4757	0
20	63	7.3076	15	0.5128	0

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4. DISCUSSION

The consolidated results of the DEAPS Model using the parameters from the Albrecht dataset, Desharnais dataset and COCOMO dataset are given in the Table 4 below. It can be seen that the prediction value is more than 50% while using Albrecht and Desharnais dataset. DE has a very effective mutation process which improves the ability of exploration. So we got promising results which indicate that the use of this model could significantly improve the efficiency of Analogy based Software Effort Estimation.

Table 4 : Consolidated results of DEAPS Model

S. No	Datasets	MMRE	Pred (0.25)	MdMRE
1	Albrecht	0.37	0.54	0.25
2	Desharnais	0.23	0.53	0.24
3	COCOMO	0.37	0.40	0.22

The comparison of results are shown in the following Figure 2

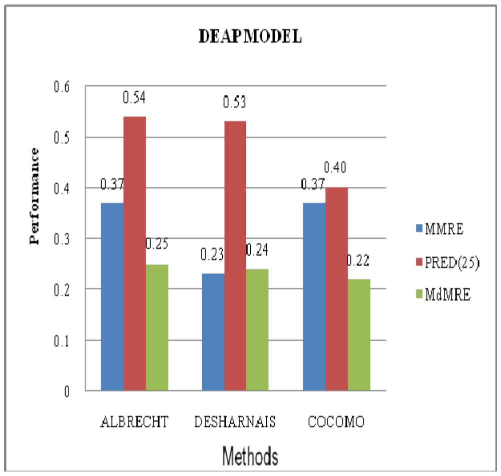

Figure 2 : Comparison with other datasets

The success of this model can be evaluated only by comparing with the already existing models in the area of software estimation. Hence the test results are also compared with the previous research results in the research paper of Y.F. Li et al (2009) [31]. The comparison is given in Table 5 below:

Table 5 : Comparison of result with previous Models

S.No	Methods	MMRE	PRED (0.25)	MdMRE
1	FWABE	0.42	0.25	0.46
2	PSABE	0.39	0.38	0.45
3	ANN	0.49	0.25	0.51
4	DEAPS (Alb)	0.37	0.54	0.25
5	DEAPS (Desh)	0.23	0.53	0.24
6	DEAPS (COC)	0.37	0.40	0.22

The diagrammatic representation of the comparison of DEAPS model with other existing models is shown in Figure 3 below:

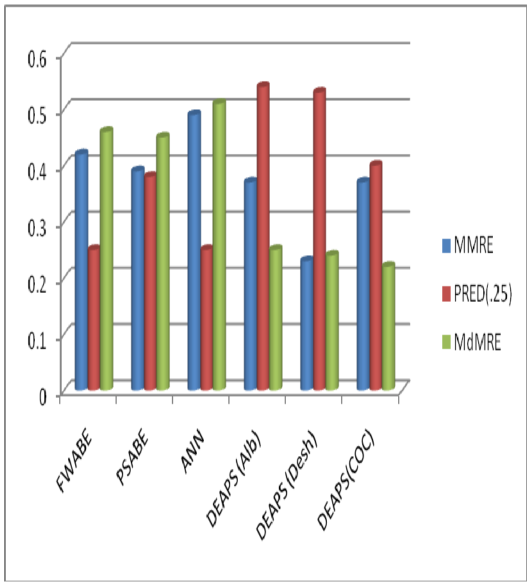

Figure 3 : Comparison with other Models

The results show that the proposed Model DEAPS for the estimation of software by the selection of relevant project has the best performance among all methods. The metrics MMRE and MdMRE are lesser than the other methods and also the probability of a project having MRE<=0.25, pred(0.25) is also very high when compared with other models

The advantages of Analogy based effort estimation are:

- The results from Analogy based methods are better than from formal based methods because it is very similar to reasoning of human problem solving method
- Analogy based estimation can deal with poorly understood domain that are difficult to model
- It can be applied in the very early phases of a software project planning and can be later improved when more details are available
- It has the potential to mitigate the effect of the outliers in a historical data set as it do not rely on calibrating a single model to suit all the projects

An exhaustive survey of Software Effort Estimation and the various types of methodologies used has been done. The Algorithmic Models and Non-Algorithmic Models have been studied in detail. Analysis of the various datasets used in the models and the metrics used for the performance valuation were also done. The merits and demerits of each model have also been analyzed. The entire procedure has been done on well established statistical methodologies taking multiple comparisons into consideration. Analogy was seen as the most efficient method for Software Effort Estimation. The focal point of the current study is analysis of the computational intelligence techniques. The use of the Evolutionary Computation Model has been suggested recently for the estimation of the software projects. They have the advantage of handling large search space. The basic idea is the Darwin's theory of evolution according to which the genetic operations between chromosomes lead to the survival of the fittest among individuals. These methods are the extension of machine learning algorithms such as ANN.

The use of Evolutionary Computation Algorithms for Software Effort Estimation has also been studied in detail. Various popular Evolutionary computation Algorithms such as Genetic Algorithm, Genetic Programming, Differential Evolution Algorithm have been studied. Optimization through Genetic Algorithm and Genetic Programming takes a longer time to train and hence reduces the performance of the model. Use of Differential Evolution Algorithm has been seen in recent days to improve the exploration ability. Differential Evolution Algorithm is an Evolutionary Computation Algorithm that is being used in recent days for many real word problems and hence its application is extended to software effort estimation. Differential Evolution is similar to the Genetic Algorithm, but it differs in the sense that distance and direction information from the current population are used for guiding the search process.

Time complexity is a critical issue for all the population based search techniques like Genetic Algorithm, Genetic Programming, Differential Evolution Algorithm, etc. The average runtime of a standard DE Algorithm depends on the stopping criteria. There are three important parameters in DE, namely, NP (Size of the population), Cr (Cross over Rate) and F (Mutation Factor). There is also D vector which consists of candidate solutions of the problem in each generation. Let Max be the maximum number of generations. For implementation, we have taken 1000 as the maximum number of generation. In each generation of DE, a loop over D is conducted in which the fundamental operations such as Mutation and Cross over are performed. Hence, the runtime complexity of this algorithm is O(NP. D. Max)

Merits: The study of the application of Differential Algorithm with Analogy is the most promising methodology for Software Effort Estimation. Differential Evolution is a method that optimizes a problem, iteratively for improving a solution . DE performs better than any other contemporary algorithm and its offers of good optimization due to higher number of local optima and higher dimensionality has been established. The main advantages of DE are its ability to provide multiple solutions and that it can be easily applied to real problems despite noisy and multidimensional space. It is simple and has effective mutation process that ensures search diversity. Hence, we propose a new model called DEAP (Differential Evolution with Analogy for Project Estimation) which uses the Analogy concept with Differential Evolution Algorithm.

Limitations: A primary limitation in this study lies in the selection of the datasets used in this study. In this approach, the real datasets Albrecht dataset, Desharnais dataset, COCOMO 81 and

NASA dataset have been considered. This imposes some limitations when the model is used for other application domains. Different dataset characteristics may favor different prediction models with prediction performance differing from one dataset to another. In real life, the project consists of attributes that are not limited to the attributes of these data sets which need to be explored.

5. CONCLUSION

The software effort evaluation has surfaced as one of the vital functions in software project management. It is not always feasible to anticipate the precise estimates in the development of software. Hence the focus is on the open research problem of the creation of the best software effort estimation model. Although the term best is subjective, a project manager usually ranks an estimation model on the basis of pre-defined accuracy measure. The aim is to propose a model that has a framework that is better than the other contemporary models. In this regard, the vital constraints to be taken into account for the software effort evaluation encompass the size of the project, schedule and number of persons concerned. The intangible nature of the software makes the software effort estimation process unreliable. But improving the accuracy of the effort estimation models would facilitate more effective control of time and budgets during software development for the Project Managers and others involved in the project. Systematic illustration of techniques has been in most of the recent methods. But the best estimation method should be based on the conditions and status of the project in comparison with the previous projects.

This paper gives a detailed study of using Evolutionary Computation Algorithms in the Software Effort Estimation models. Among the existing methods, Analogy Based Estimation (ABE) is the most flexible method for achieving better estimates during the initial stages of effort estimation. It inherits the formal expressions of case based reasoning and is a very popular method. However, it is criticized for its large computational usage. To alleviate this drawback, this study is devoted to improve the efficiency of the Analogy by using it with the recently popular Evolutionary Algorithms. The study eventually focused on using Differential Evolution Algorithm which is an Evolutionary computation Algorithm. The promising results show that the efficiency of Analogy method has improved by using it in combination with Differential Evolution Algorithm. This study makes significant contribution to the knowledge of Software Effort Estimation in the field of Software Engineering.

Differential Evolution Algorithm is used to select the most relevant project from set of historical projects that matches with the new project. The proposed method is implemented in JAVA platform. Based on the selection of relevant project, it is easier to estimate the effort required for the new project. The experimental result indicates that this model is better than existing methods. The metrics used are MMRE, MdMRE and pred(25%). As the search space is big, the Differential Evolution Algorithm is used which has been proved to be useful. Future work is to analyze the performance of the model with the combination of other Evolutionary Computation Algorithms.

REFRENCES:

[1] Linda M Laird. (2006). The Limitations of Estimation, IT Pro , pp. 40-45.
[2] M. Jørgensen and D.I.K. Sjøberg (2001). Impact of Effort Estimates on Software Project Work, Information and Software Technology, vol 43, no. 15, pp.939-948
[3] Ning Nan and Donald E.Harter (2009). Impact of Budget and Schedule Pressure on Software Development Cycle time and Effort, IEEE Trans. on Software Engineering., pp.624-637.
[4] Magne Jorgensen and Stein Grimstad (2011). The impact of irrelevant and misleading Info. on Software development Effort Estimates : A Randomized Controlled Field Experiment, IEEE Transactions On Software Engg., pp. 695-707.
[5] Tim Menzies, Andrew Butcher, David Cok, Lucas layman, Forrest Shull, Burak Turhan (2013). Local vs Global lessons for defect Prediction and Effort Estimation, IEEE Transactions on Software Engg., Vol.39, pp. 822-834.
[6] Ekram kocaguneli , Tim Menzies, Ayse Basar Bener and Jacky W Keung (2012). Exploiting the essential assumptions of

Analogy based Effort Estimation, IEEE Engg., pp. 425-437.

[7] Magne Jorgensen, Tanja M.Grusehke and R. Gupta (2009). The Impact of Lessons-Learned sessions on Effort Estimation and Uncertainity Assesments, IEEE Trans. on Software Engg., pp. 368-383.

[8] Magne Jorgensen and Martin Sheppard (2007). A Systematic review of Software Development Cost Estimation Studies, IEEE Trans. on Software Engg., pp. 33-53

[9] Karel Dejaeger, Wouter Verbeke, David Martens, Bart Baesens (2012). Data mining techniques for Software Effort Estimation: A Comparative study, IEEE Trans. on Software Engg., vol. 38, pp. 375-397.

[10] Tim Menzies, Andrew Butcher, David Cok, Lucas layman, Forrest Shull, Burak Turhan (2013). Local vs Global lessons for defect Prediction and Effort Estimation, IEEE Transactions on Software Engg., Vol.39, pp. 822-834

[11] Nikolos Mittas and Lefteris Angelis (2012). Ranking and clustering Software Cost Estimation Model through a multiple Comparison Algorithm, IEEE Transactions on Software Engg., pp. 537-551.

[12] Mark Harman, Afshin Mausouri (2010). Search based Software Engineering: Introduction to special issue of IEEE Trans. on Software Engg., IEEE Transactions on Software Engg., pp. 737-741.

[13] Ekrem Kocaguneli, Tim Menzies, Jacky Keung, David Cok, Ray Madachy (2013). Active Learning and Effort Estimation: finding the essential content of Software Effort Estimation data, IEEE Transactions on Software Engg., pp. 1039-1053.

[14] Magne Jorgensen (2005). Evidence based guidelines for assessment of Software development Cost Uncertainty, IEEE Transactions on Software Engg., pp. 942-954.

[15] Mudasir Manzoor Kirmani, Abdul Wahid (2015). Impact of Modification Made in Re-UCP on Software Effort Estimation, Journal of Software Engineering and Application, pp. 276-289.

[16] P.Subitsha, J.Kowski Rajan(2014) , Artificial Neural Network Models For Software Effort Estimation, International Journal of Technology Enhancements And Emerging Engineering Research, Vol 2, Issue 4, pp. 76-80.

[17] Sufyan Basri, Nazri Kama and Roslina Ibrahim (2015). A Novel Effort Estimation Approach for Requirement Changes during Software Development Phase, International Journal of Software Engineering and Applications Vol. 9, No. 1, pp. 237-252

[18] Tuan Khanh lee -Do, Kyung-A Yoon , Yeong – Seok Seo and Doo-Hwan Bae (2010), “Filtering of inconsistent Software Project data for Analogy based Effort Estimation”, Annual conference of IEEE on Computer Software and Application Conference, pp. 503-508.

[19] Hathaichanok Suwanjang and Nakornthip Prompoon (2012), “Framework for developing a Software Cost Estimation Model for Software modification based on a relational matrix of Project profile and Software Cost using an Analogy”, International journal of Computer and communication engineering, pp. 129-134.

[20] Tridas Mukhopadhyay, Stephen S Vicinanza, S. S. and Michael J Prietula (1992), “Examining the feasibility of a case-based reasoning model for software effort estimation”, MIS Research Center, University of Minnosota, Vol.16(2), pp. 155-171.

[21] Stamelos. I., Angelis. L., and Sakellaris. E. (2001), “BRACE: Bootstrap based Analogy Cost Estimation: Automated support for an enhanced effort prediction method”, Proceedings 12th European Software Control and Metrics Conference (ESCOM’2001), pp. 17-23.

[22] Myrtveit. I., and Stensrud. E. (1999), “A controlled experiment to assess the benefits of estimating with analogy and regression models”, IEEE Transactions on Software Engineering, Vol. 25(4), pp. 510-525

[23] Li. Y.F., Xie. M. and Goh, T.N. (2009), “A study of the non-linear adjustment for Analogy based Software Cost Estimation, Springer, pp. 603-642

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[24] Jianfeng Wen, Shixian Li and Linyan Tang (2009), "Improve analogy based software effort estimation using principal component analysis and correlation weighting", IEEE Transactions of Software Engineering, pp. 179 – 186

[25] Chao Jung Hsu , Nancy Urbina Rodas, Chin Yu Huang and Kuan- Li Peng (2010). A Study of improving the Accuracy of Software Effort Estimation using Linear Weighted Combination, Proc. of Annual IEEE Comp. Software Applications Conference, pp. 98-103.

[26] Shama Kousar Jabeen.A, B.Arthi. (2014) Software Effort Estimation for Size Proxy Metric Framework Modelling using Software Estimation Models and Neuro Fuzzy Logic Approach , International Journal of Soft Computing and Engineering (IJSCE) , Volume-4, Issue-2, pp 70-73.

[27] Mohita Sharma,, Neha Fotedar (2014). Software Effort Estimation with Data Mining Techniques- A Review, International journal of Engineering Sciences and Research Technology, pp.1646-1653,

[28] Y.F.LI, M.Xie, T.N.Goh (2010). A Study of Genetic Algorithm for Project Selection for Analogy based Software Cost Estimation, Proc. of IEEE IEEM, pp. 1256-1260.

[29] Colin J.Burgess and Martin Lefley (2000). Can Genetic Programming improve Software Effort Estimation? A Comparative Evaluation, Elsevier, pp. 863-873.

[30] Swagatam Das and Ponnuthurai Nagaratnam Suganthan (2011). Differential Evolution- A Survey of Art, IEEE Transactions on Evolutionary Computation, pp. 4-31.

[31] Y.F. Li , M. Xie, T.N. Goh (2009). A study of project selection and feature weighting for analogy based software cost estimation, The Journal of Systems and Software, pp 241-252