Scientific Research and Essays Vol. 6(30), pp. 6382-6382, 9 December, 2011
Available online at http://www.academicjournals.org/SRE
DOI: 10.5897/SRE11.1736
ISSN 1992-2248 ©2011 Academic Journals

Full Length Research Paper

A hybrid method for increasing the accuracy of software development effort estimation

Vahid Khatibi B.[1], Dayang N. A. Jawawi[1]*, Siti Zaiton Mohd Hashim[1] and Elham Khatibi[2]

[1]Department of Software Engineering, Faculty of Computer Science and Information Systems Universiti Teknologi Malaysia (UTM), Skudai 81300, Johor Bahru, Malaysia.
[2]Department of Computer Engineering, Bardsir Branch, Islamic Azad University, Kerman, Iran.

Accepted 18 October, 2011

Since software development environments, methods and tools are changing rapidly, the importance of accurate estimations in software projects is increasing significantly. Inaccurate estimations can lead to unpleasant results in the software projects so that many projects are failed at the early stages of the project. During the recent years, numerous estimation methods have been proposed that most of which are based on statistical techniques. Among all existing methods, simplicity of analogy based method makes it so common in this field. Analogy methods usually present accurate estimations but if the level of non normality in the software project datasets is high or type of most project features is categorical, these methods are confronted with inaccurate estimation problem. In this paper, genetic algorithm has been used under a new framework to improve the performance of analogy methods. A large dataset has been employed to evaluate the performance of the proposed method and the results have been compared with the other estimation methods. The results showed that the proposed method outperformed the other methods considerably.

Key words: Effort estimation, analogy method, feature weighting, genetic algorithm.

INTRODUCTION

Since the role of software in today's business market is undeniable, accurate estimation of software effort is very important. Planning, developing, constructing and all aspects of the software projects are affected by accurate estimations. During the recent decades, many methods for software effort estimation have been presented. Selecting a method as the best seems impossible because the performance of each method depends on the various factors such as available information, used technique, project features and so on, but the main aim of all methods is presenting the results which are more accurate. Since at the early stages of the project, all the earned information is incomplete, the predictions may be inaccurate and this problem is seen in software projects rather than the other project types. The first idea for software effort estimation returns to 1950 by presenting the manual rule of thumb (Jones, 2007). By increasing the number of software projects and need of user society to earn high quality software, some models based on the linear equations were presented as the software effort techniques in 1965 (Boehm and Valerdi, 2008). As the pioneers of software estimation methods we can consider the name of Larry Putnam, Barry Bohem and Joe Aron (Jones 2007). Afterward in 1973, the IBM researchers presented the first automated tool, interactive productivity and quality (IPQ) (Jones, 2007). Barry Boehm proposed

*Corresponding author. E-mail: dayang@utm.my.

a new method based on computing some of the software project factors by means of several mathematical equations called COCOMO (Boehm, 1981). In addition, Boehm explained several algorithms in his book "Software Engineering Economics" (Boehm, 1981) that still are used by researchers. Other models such as Putnam Lifecycle Management (SLIM) (Putnam, 1987) and software evaluation and estimation of resources – software estimating model (SEER - SEM) continued the principals of COCOMO (Boehm and Valerdi, 2008). Introducing the function point (FP) as a metric for software size estimation by Albrecht and Gaffney (1983) was the other important event in that decade. Analogy based method was proposed in 1997 (Shepperd and Schofield, 1997) and its usage was increased significantly because it follows the human manners to solve the problems. Although this method usually present acceptable results but there are some constraints. For example, the software project indicators may demonstrate unnecessary or unreal specifications of the projects and particularly in some projects the amount of effort is surprising. In addition, achieving to detailed information in many projects is impossible and we have to predict the amount of effort with limited features.

Several studies have tried to improve the performance of analogy methods and overcome the mentioned limitations by using mathematical and statistical methods (Keung and Kitchenham, 2007; Jingzhou and Guenther, 2008; Keung, 2008; Jianfeng et al., 2009; Tosun et al., 2009). Since software projects are usually complicated and relations between features are hard to understand, soft computing techniques are widely used to improve the performance of analogy methods (Chiu, 2007; Li et al., 2009; Pahariya et al., 2009; Oliveira et al., 2010). In current study we are going to use genetic algorithm and hybrid optimization functions to improve the performance of analogy method. This paper is organized as follows: Subsequently, it includes principals of analogy methods, after which it describes the genetic algorithm, there after the performance metrics are presented; then, the proposed method is presented and numerical results are described; finally, conclusion and future works are presented.

ANALOGY BASED ESTIMATION (ABE)

ABE method was produced by Shepperd in 1997 as a substitute for algorithmic methods (Shepperd and Schofield, 1997). Structure of this method is based on comparison of new software project with some same historical projects to do the estimation. Usually, ABE includes four parts:

i) Historical dataset.
ii) Similarity function.
iii) Solution function.
iv) The associated retrieval rules.

Each part can be described as follows:

i) Gathering the previous projects data and producing the historical dataset.
ii) Choosing new proper features of the project such as (FP) and line of code (LOC).
iii) Retrieving the previous projects and calculating the similarities between the target project and the previous projects. Usually the weighted Euclidean distance and the weighted Manhattan distance are used at this stage.
iv) Estimating the effort of the target project.

Similarity function

ABE uses a similarity function which compares the features of two projects. There are two popular similarity functions, Euclidean similarity (ES) and Manhattan similarity (MS) (Shepperd and Schofield, 1997). Equation 1 shows the Euclidean similarity function:

$$\mathrm{Sim}(p,p\prime)=1/\left[\sqrt{\sum_{i=1}^{n} w_i \mathrm{Dis}(f_i,f\prime_i)}+\delta\right] \quad \delta=0.0001$$

$$\mathrm{Dis}(f_i,f_i')=\begin{cases}(f_i-f_i')^2, & \text{if } f_i \text{ and } f_i' \text{ are numeric or ordinal}\\ 1 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i=f_i'\\ 0 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i\neq f_i'\end{cases} \quad (1)$$

Where, p and p' are the projects, w_i is the weight assigned to each feature and varies between 0 and 1. Fi and f_i' display the ith feature of each project and n demonstrates the number of features. δ is used for obtaining the none zero results.

The MS formula is very similar to the ES but it computes the absolute difference between the features. Equation 2 shows the Manhattan similarity function.

$$\mathrm{Sim}(p,p\prime)=1/\left[\sum_{i=1}^{n} w_i \mathrm{Dis}(f_i,f_i')+\delta\right] \quad \delta=0.0001$$

$$\mathrm{Dis}(f_i,f_i')=\begin{cases}|f_i-f_i'| & \text{if } f_i \text{ and } f_i' \text{ are numeric or ordinal}\\ 1 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i=f_i'\\ 0 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i\neq f_i'\end{cases} \quad (2)$$

Solution functions

After choosing the K most similar projects, it is possible to

estimate the effort and cost of the new project according to the selected features. The common solution functions are: the closest analogy as most similar project (Walkerden and Jeffery, 1999), the average of most similar projects (Shepperd and Schofield, 1997), the median of most similar projects (Angelis and Stamelos, 2000) and the inverse distance weighted mean (Kadoda et al., 2000). The mean describes the average of the effort of K most similar projects, where K > 1. The median describes the median of the effort of K most similar projects, where K > 2. The inverse distance weighted mean adjusts the portion of each project in estimation by using Equation 3.

$$\widehat{C}_p = \sum_{k=1}^{K} \frac{\mathrm{Sim}(p, p_k)}{\sum_{i=1}^{n} \mathrm{Sim}(p, p_k)} C_{p_k} \quad (3)$$

Where p shows the new project, p_k illustrates the kth most similar project, C_{pk} is the effort value of the kth most similar project p_k, Sim (p, p_k) is the similarity between projects p_k and p and K is the total number of most similar projects.

GENETIC ALGORITHM

Genetic algorithm is a search based algorithm which follows the concept of natural evolution. Optimization problems are the main domain of genetic algorithm usage. An initial solution is determined as a genome or a chromosome. A population including several solutions (chromosomes) is constructed and it is treated as the first generation. Each solution (chromosome) is given a fitness value based on its merit and next generation is produced by using some operators called selection, mutation and crossover. Some irrelevant and unsuitable solutions are omitted during the generation production. The main operators of genetic algorithm are described as follows:

Selection operator

This operator selects the best solutions to go to the next generation. The amount of fitness value is the most important factor considered by selection operator.

Crossover operator

Crossover makes the genetic algorithm different as compared to the other optimization methods. The idea behind this operator is that by combining two parents (chromosomes) we can obtain two new children which are better than their parents. Some random interchanges are performed on two parents and two new children are produced. Several types of crossover are used in genetic algorithm based on user definitions.

Mutation operator

This operator is used to hold the diversity of the population and is comparable with biological mutation. In addition, by using mutation, the problem of local minimum will be solved because chromosomes will be sufficiently different in each population. Mutation operator changes some bits in a solution and produces new solution which may be better than the first one. The overall genetic algorithm based on previous concepts can be simply described as follows:

i) Randomly generate a population.
ii) Compute the fitness of each individual in population.
iii) Repeat.

1) Select parents from population.
2) Performing the ‘crossover’ on parents to generate next population.
3) Performing the ‘mutation’ on parents to generate next population.
4) Compute the fitness of each individual in new population.

iv) Until the best individuals are collected.

PERFORMANCE METRICS

Performance of estimation methods is evaluated by several metrics including RE (relative error), MRE (magnitude of relative error) and MMRE (mean magnitude of relative error) which are computed as follows (Shepperd and Schofield, 1997):

RE = (estimated – actual)/(actual) (4)

MRE = |Estimated – Actual |/(Actual) (5)

MMRE = ∑MRE/N (6)

The other parameter used in evaluation of performance is PRED (percentage of the prediction) which is determined as:

$$PRED(X) = A/N \quad (7)$$

Where, A is the number of projects with MRE less than or equal to X and N is the number of considered projects. Usually, the acceptable level of X in software cost estimation methods is 0.25 and the various methods are compared based on this level. Decreasing of MMRE and increasing of PRED is the main aim of all estimation techniques regarding the software cost.

THE PROPOSED METHOD

Here, we are going to propose a new method for increasing the accuracy of software cost estimations by combining analogy method and genetic algorithm. As software projects are naturally complicated and ambiguous, the analogy method cannot present precise estimations to stand alone. High level of non normality and high number of outliers in software project datasets lead to inaccurate estimations. Outlier refers to a project in which there is no significant relation between the amount of effort and features which describe the project. Existing high number of outliers in a dataset leads to increase in the degree of non normality. In addition, high number of categorical features decreases the accuracy of analogy methods. Therefore, in this study the genetic algorithm has been used as a complement method for improving the performance of analogy method. More details about proposed method are presented subsequently.

Methodology

As can be seen in Equation 1, several weights are used as an adjustment to determine the similarity of projects. We can use the genetic algorithm to optimize the amount of MMRE by varying the weight of each feature. Therefore, finding the best weights is our main goal. Achieving to the best possible optimization needs to adjust the genetic algorithm parameters accurately. Our methodology comprises of two studies. Firstly, settings of genetic algorithm are presented and afterward training and testing stage are described. Selected dataset is presented thereafter.

Genetic algorithm settings

Here, all settings regarding the genetic algorithm are described. An intensive search has been performed to find the best parameters for high accurate estimation. In current study, 250 individuals are considered in first population; because in selected dataset each project has 26 features which 25 out of them are used in genetic algorithm. Final feature is effort which should be estimated. As our main goal is minimizing the MMRE and increasing the accurate of estimations, the fitness function is defined based on MMRE because this metric can clarify the real amount of error regarding the estimation method. PRED is not a suitable metric to use as fitness function because the amount of PRED is determined based on the specified limit and the process of specifying the mentioned limit is performed subjectively. Minimizing of MMRE can ensure achieving to proper estimation model based on analogy method. Gaussian is used as mutation function and the amount of 'scale' and 'shrink' are adjusted to 1. Scattered type of crossover functions is selected to produce the next generation. Number of generations is 100 and the amount of 'function tolerance' is determined by 0.000001.

Training stage

All data is divided into three sets called D_1, D_2 and D_3 which are equaled (approximately equal). D_1 and D_2 are considered as training sets and D_3 is used as test set. In training stage, genetic algorithm tries to minimize the amount of MMRE for set D_2. Indeed, analogy method is performed on two sets: D_1 is used as basic dataset (prediction of projects of D_2 by using projects of D_1). Genetic algorithm adjusts the weights so that the MMRE is minimized. For more optimizing the MMRE, a hybrid optimization function is used after obtaining the results from genetic algorithm. Indeed the output of genetic algorithm is treated as input of this function. The mentioned function attempts to find a constrained minimum of a scalar function of several variables starting at an initial estimate. This is generally referred to as constrained nonlinear optimization or nonlinear programming. Several trial and error processes showed that hybrid function could decrease the MMRE by more optimizing the weights. In this study, Fmincon (has been implemented in Matlab software) has been used as hybrid function (letcher and Powell, 1963; Goldfarb, 1970).

Testing stage

After finding the most suitable weights, test stage is started. In this stage all projects of D_3 are treated as test set and analogy method is used to estimate the projects effort. The most important point is that the obtained weights from previous section are applied to the analogy method for estimating in this stage. Figure 1 shows the framework of proposed method.

Dataset

For the purpose of evaluating the proposed method, Maxwell dataset (Maxwell, 2002) is used because this dataset is relatively new and comprises of 62 software projects (enough large). Each project in this dataset is described by 26 features and this high number of features is useful to show the performance of proposed method. All features excluding features 1, 24, 25, 26 are categorical and the mentioned features are numerical. High number of categorical features usually decreases the accuracy of analogy method. Therefore, we selected this dataset with 22 categorical features to appear as the real improvement of analogy method by using proposed method. Table 1 gives some statistical information about Maxwell dataset.

NUMERICAL RESULTS

At first, all 62 projects are divided into three sets D_1, D_2 and D_3 randomly with 22, 20 and 20 projects. Training stage is started by two first sets to find the best weights. As mentioned previously, 25 features are applied to the genetic algorithm to find the best values for weights. After

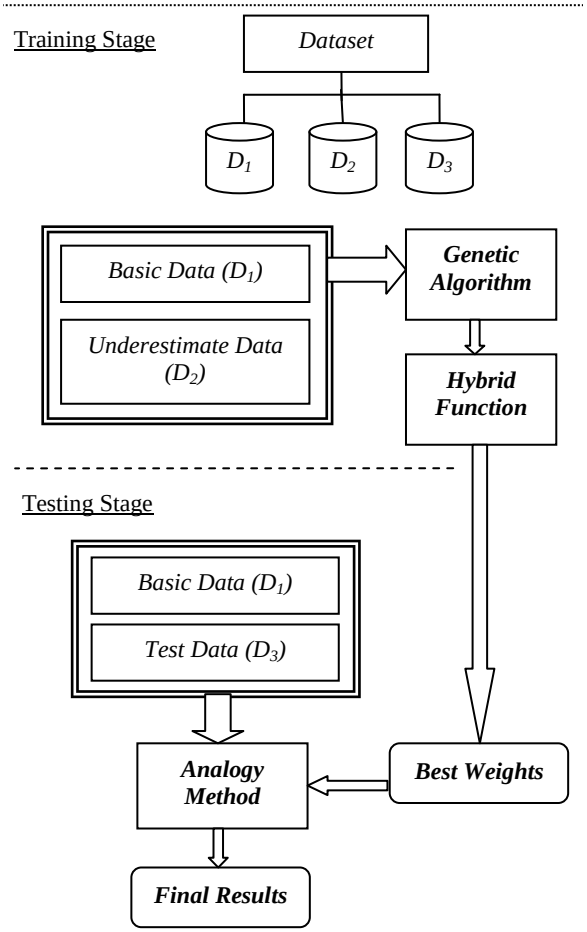

Figure 1. Proposed framework.

finding the best weights, test stage is started, D_1 is treated as the basic set and D_3 as test set. In this stage analogy method is used for forecasting the development effort for projects in D_3 by using the obtained weights from training stage. To earn more reliable results, this process is repeated three times (round) so that each time, three sets D_1, D_2 and D_3 are produced randomly. Train and test stages are applied and the results of each round are recorded. In the following study, more details about the executing of the proposed method are presented. Figure 2 shows the trend of minimizing the MMRE regarding round one. 52 generations are seen in the mentioned figure. Regarding each generation, the best MMRE and mean MMRE have been computed. According to Figure 2, the best MMRE in round one is close to 0.47 and the best mean MMRE among all generations is related to generation 31. In addition, the amount of MMRE has been decreased significantly during the generations 1 to 25; afterward there is no high decrease regarding MMRE. Figure 3 shows the trend of minimizing the MMRE regarding round two. As seen in Figure 3, genetic algorithm has reduced the amount of MMRE from 0.79 to less than 0.65 for sets D_1 and D_2 as basic and test sets respectively. The best amount of MMRE is a little more than 0.64 obtained on generation 54. The best mean MMRE is obtained on generation 31. Most significant decreasing of MMRE has happened during generations 1 to 15; afterward there is no significant decrease regarding MMRE. Figure 4 shows the trend of minimizing the MMRE regarding round three. The best value of MMRE is obtained on generation 55 and the best mean MMRE is obtained also on generation 55. As seen in Figure 4, 'genetic algorithm' has reduced the amount of MMRE from above 0.59 to less than 0.53 for sets D_1 and D_2 as basic and test sets respectively.

Process of deceasing the MMRE has been continued

Table 1. Maxwell dataset details.

Feature	Description	Mean	Std Dev	Min	Max
Time	Time	5.58	2.13	1	9
App	Application type	2.35	0.99	1	5
Har	Hardware platform	2.61	1	1	5
Dba	Database	1.03	0.44	0	4
Ifc	User interface	1.94	0.25	1	2
Source	Where developed	1.87	0.34	1	2
Telon use	Telon use	2.55	1.02	1	4
Nlan	Number of different development languages used	0.24	0.43	0	1
T01	Customer participation	3.05	1	1	5
T02	Development environment adequacy	3.05	0.71	1	5
T03	Staff availability	3.03	0.89	2	5
T04	Standards use	3.19	0.70	2	5
T05	Methods use	3.05	0.71	1	5
T06	Tools use	2.90	0.69	1	4
T07	Software's logical complexity	3.24	0.90	1	5
T08	Requirements volatility	3.81	0.96	2	5
T09	Quality requirements	4.06	0.74	2	5
T10	Efficiency requirements	3.61	0.89	2	5
T11	Installation requirements	3.42	0.98	2	5
T12	Staff analysis skills	3.82	0.69	2	5
T13	Staff application knowledge	3.06	0.96	1	5
T14	Staff tool skills	3.26	1.01	1	5
T15	Staff team skills	3.34	0.75	1	5
Duration	Duration	17.21	10.65	4	54
Size	Application size	673.31	784.08	48	3,643
Effort	Effort	8,223.21	10,499.90	583	63,694

during all generations. Table 2 shows the overall results after completing all the three rounds. Each round in Table 2 has been divided into two sections; showed results of the first section (analogy) have been computed by applying analogy method without weighting (w = 1 in Equation 1) on D_1 and D_3 as basic and test sets respectively. Showed results of second section (proposed) have been computed by applying analogy method with weighting (best weights from train stage) on D_1 and D_3 as basic and test sets respectively. As can be seen in Table 2, the performance of proposed method regarding all three rounds based on MMRE and PRED (0.25) is better than analogy method. This means that genetic algorithm can improve the accuracy level of predictions computed by analogy method. Figure 5 depicts the percentage of improvement in terms of MMRE and PRED (0.25) obtained from each round.

According to Figure 5, the most improvement is related to the first round with about 50% for MMRE and 36% for PRED (0.25). Also the least improvement is seen in round two. Since performance metrics are computed based on the mean of results, we can say that the percentage of improvement regarding the proposed method is significant. An important point about choosing the test sets is that if the test set is selected from various types of projects in dataset, the performance of proposed method will be better. This means that there is the significant relation between the diversity of training set and the level of estimation accuracy in proposed method.

Evaluation of proposed method

Here, we are going to evaluate the proposed method against the other estimation methods. Several common methods like ANN (artificial neural network) (Mair et al., 2000), classification and regression trees (CART) (Stensrud, 2001), stepwise regression (SWR) (Mendes et

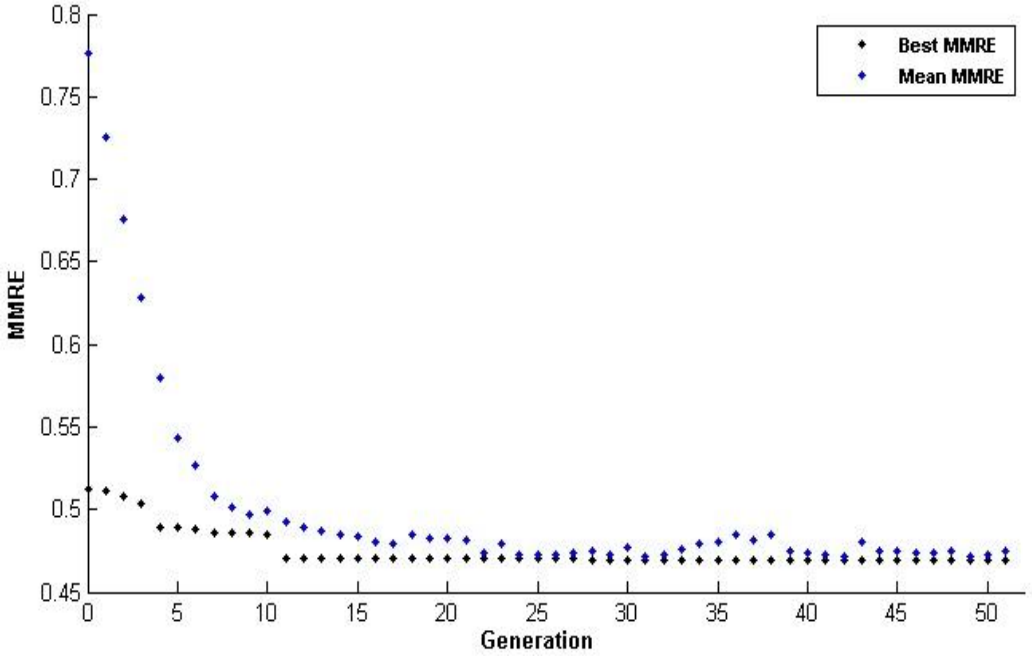

Figure 2. Performance of Genetic Algorithm in round 1.

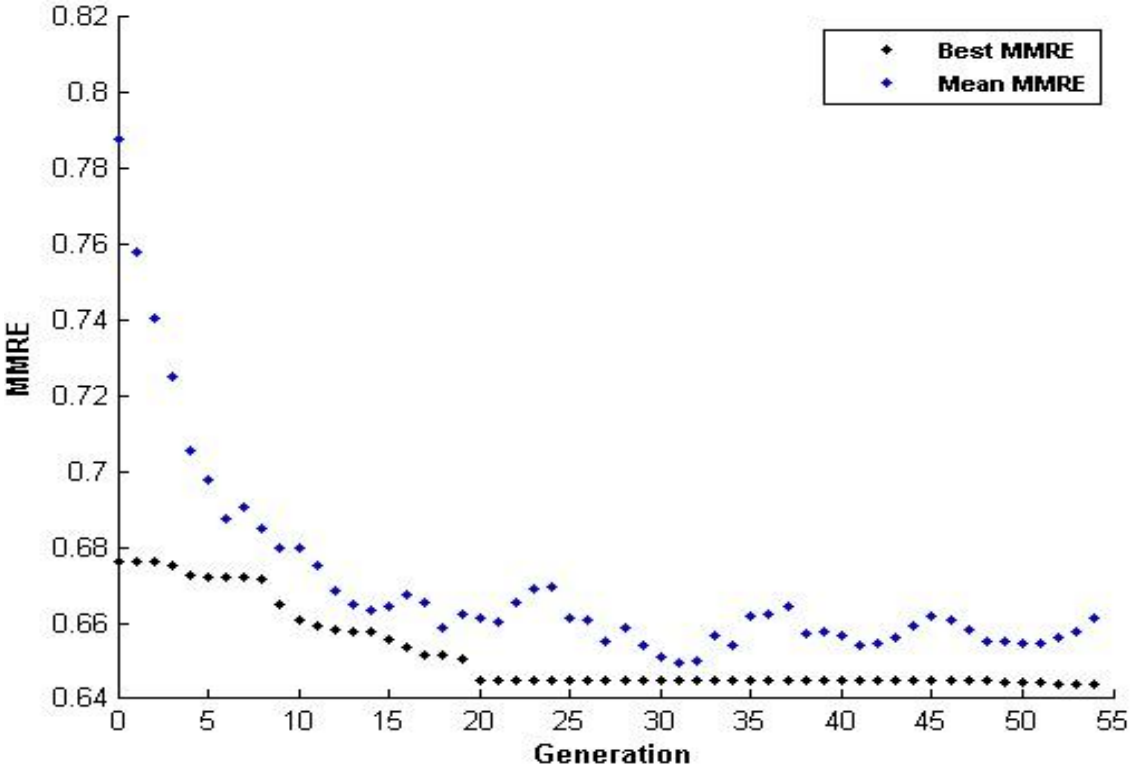

Figure 3. Performance of 'genetic algorithm' in round 2.

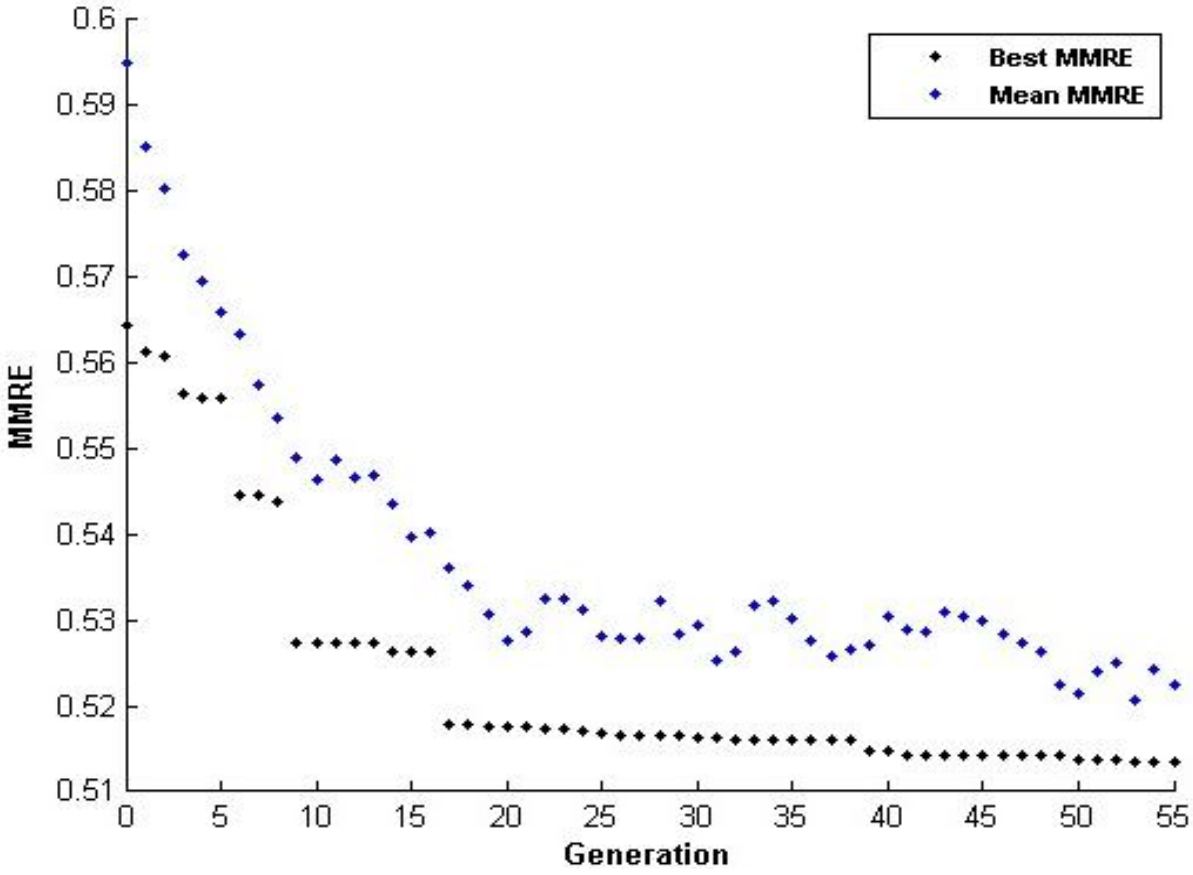

Figure 4. Performance of Genetic Algorithm in round 3.

Table 2. Overall results on three rounds.

	Method	MMRE	PRED (0.25)
Round 1	Analogy	0.82	0.3
	Proposed	0.52	0.45
Round 2	Analogy	0.88	0.2
	Proposed	0.74	0.25
Round 3	Analogy	0.64	0.35
	Proposed	0.50	0.45
Average	Analogy	0.78	0.28
	Proposed	0.59	0.38

al., 2003) are considered for comparison. In addition, several adjusting methods which have been proposed to improve the analogy methods are participated in the comparison. The adjusting methods include adjusting

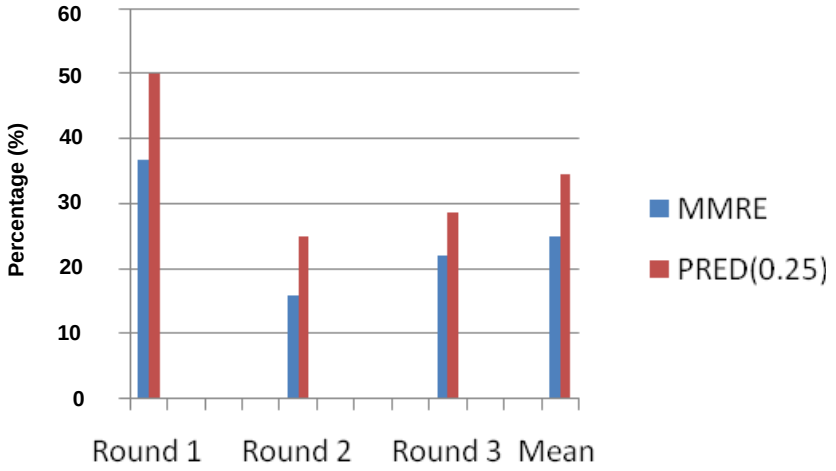

Figure 5. Percentage of improvement in each round.

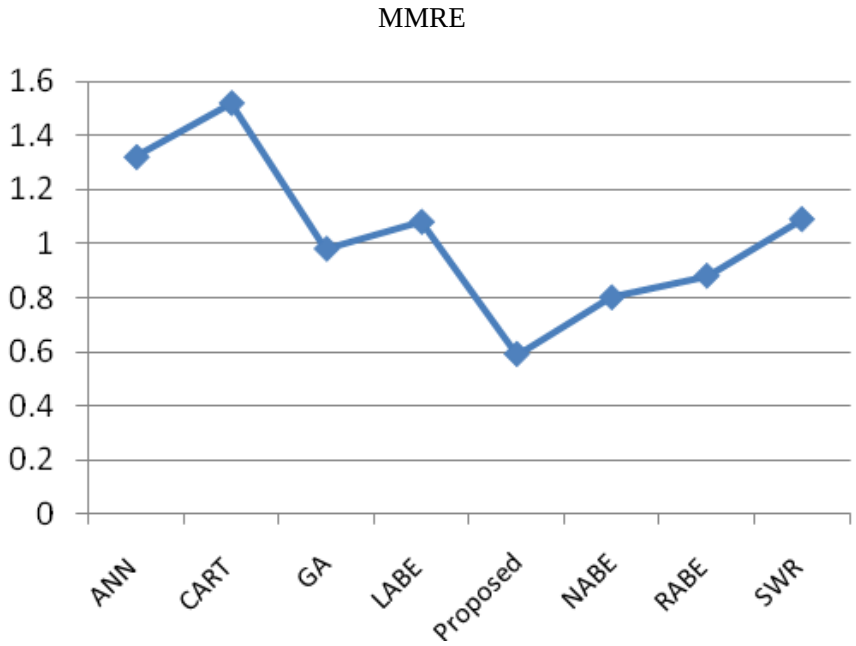

Figure 6. Evaluation of proposed method versus other methods based on MMRE.

with genetic algorithm (GA) (Chiu, 2007), linear adjusting the ABE (LABE) (Walkerden and Jeffery, 1999), regression adjusting the ABE (RABE) (Jorgensen et al., 2003) and non linear adjusting the ABE (NABE) (Li et al., 2009). All mentioned methods have been implemented in Li et al., 2009) on Maxwell dataset. The results of the proposed method should be compared with the other methods based on average amount of MMRE and PRED (0.25) in three rounds (Table 2). The following figures shows the performance of proposed method compared with the other estimation methods based on MMRE and PRED (0.25) to show the obtained improvement. Figure 6 shows the comparison of all methods based on MMRE. As can be seen in Figure 6, proposed method presents the lowest MMRE among all methods; NABE and RABE are located in next places. The worst amount of MMRE is

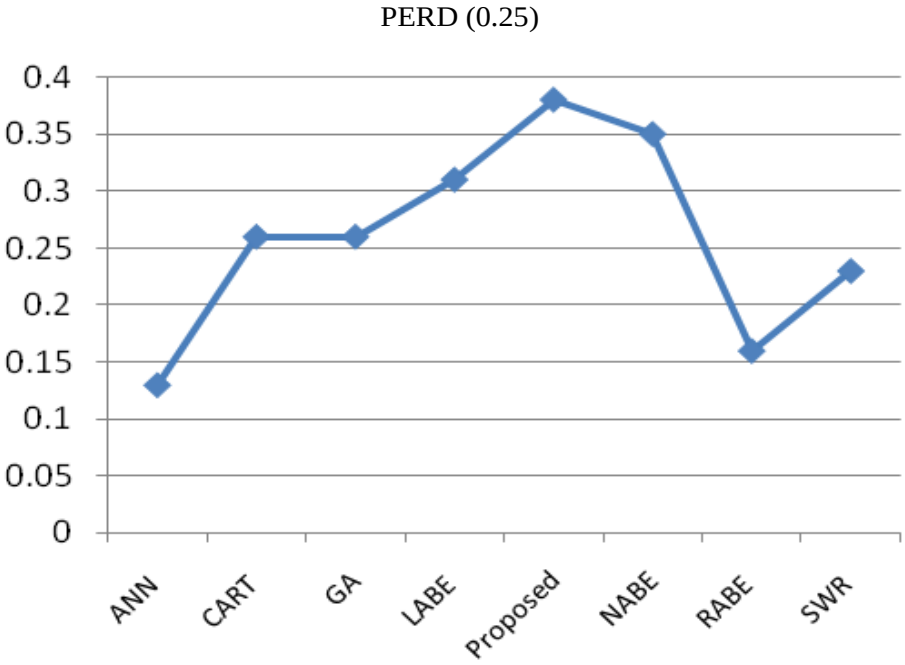

Figure 7. Evaluation of proposed method versus other methods based on PRED (0.25).

related to CART method with 1.4 and above error. Figure 7 depicts the comparison of all mentioned methods based on PRED (0.25). According to Figure 7, the proposed method presents the highest amount of PRED (0.25) among other methods; NABE and LABE are located in next places. The least amount of PRED is related to ANN method.

Conclusion

One of the most important goals of the project managers is to estimate their own projects accurately because inaccurate estimations can lead to project fail easily. Since usually there are previous experiences regarding the software projects, the analogy based methods could be an ideal choice for estimating. This method has been widely used in recent decade but if the level of non normality in software projects is more than usual and number of categorical features is high, some complement techniques are necessary for avoiding from inaccurate estimations. In this paper, genetic algorithm was combined by analogy method to present the more accurate results. By performing an intensive trial and error, the best structure for genetic algorithm was found and a hybrid function was added for more optimizing. For the purpose of achieving more reliable results we repeated our proposed framework three times with random sets in training and testing stages. In addition, evaluation of the proposed method was done by using a large dataset with high number of projects and features. The results showed that the proposed method can improve the accuracy of estimations in analogy method significantly. In addition, comparison of proposed method with other methods showed that it outperforms the other methods based on evaluation metrics. As future works we are going to add other soft computing techniques to analogy methods to present more accurate and reliable estimations.

ACKNOWLEDGEMENTS

Special thanks to the Universiti Teknologi Malaysia for financing and funding this research through Research University Grant and also to our Embedded & Real-Time Software Engineering Laboratory (EReTSEL) members for their continuous support.

REFERENCES

Albrecht AJ, Gaffney JA (1983). Software function, source lines of

codes, and development effort prediction: a software science validation. IEEE Trans Software Eng. SE., 9(6): 639-648.
Angelis L, Stamelos I (2000). A Simulation Tool for Efficient Analogy Based Cost Estimation. Empirical Software Eng., 5(1): 35-68.
Boehm BW (1981). Software engineering economics. Englewood Cliffs, NJ: Prentice Hall.
Boehm BW, Valerdi R (2008). Achievements and Challenges in Cocomo-Based Software Resource Estimation. IEEE Softw., 25(5): 74-83.
Goldfarb D (1970). A Family of Variable Metric Updates Derived by Variational Means. Math. Comput., 24: 23-26.
Jianfeng W, Shixian L, Linyan T (2009). Improve Analogy-Based Software Effort Estimation Using Principal Components Analysis and Correlation Weighting. Asia-Pacific Software Engineering. APSEC '09.
Jingzhou L, Guenther R (2008). Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA+. Empirical Softw. Eng., 13(1): 63-96.
Jones C (2007). Estimating software costs: Bringing realism to estimating. New York, NY: McGraw-Hill.
Jorgensen M, Indahl U, Sjoberg D (2003). Software effort estimation by analogy and regression toward the mean. J Syst Softw 68: 253-262.
Kadoda G, Cartwright M, Chen L Shepperd M (2000). Experiences Using Case-Based Reasoning to Predict Software Project Effort Empirical Assessment and Evaluation in Software Engineering.
Keung JW (2008). Theoretical Maximum Prediction Accuracy for Analogy-Based Software Cost Estimation. Software Engineering Conference, 2008. APSEC '08. 15th Asia-Pacific.
Keung JW, Kitchenham B (2007). Optimising Project Feature Weights for Analogy-Based Software Cost Estimation using the Mantel Correlation. Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific.
letcher R, Powell MJD (1963). A Rapidly Convergent Descent Method for Minimization." Comput. J., 6: 163-168.
Li YF, Xie M, Goh TN (2009). A study of project selection and feature weighting for analogy based software cost estimation. J. Syst. Softw., 82(2): 241-252.
Li YF, Xie M, Goh TN (2009). A study of the non-linear adjustment for analogy based software cost estimation. Empir Software Eng 14: 603-643.
Mair C, Kadoda G, Lefley M, Phalp K, Schofield C, Shepperd M, Webster S (2000). An investigation of machine learning based prediction systems. J. Syst. Softw., 53(1): 23-29.
Maxwell K (2002). Applied statistics for software managers. Englewood Cliffs,NJ, Prentice-Hall.
Mendes E, Watson I, Triggs C, Mosley N, Counsell S (2003). A comparative study of cost estimation models for web hypermedia applications. Empir. Softw. Eng., 8: 163-196.
Chiu SJH (2007). The Adjusted Analogy-Based Software Effort Estimation Based on Similarity Distances. J. Syst. Softw., 80: 628-640.
Oliveira ALI, Braga PL, Lima RMF, Cornélio ML (2010). GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. Inf. Softw. Technol., 52(11): 1155-1166.
Pahariya JS, Ravi V, Carr M (2009). Software cost estimation using computational intelligence techniques. Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on.
Putnam LH (1987). A general empirical solution to the macrosoftware sizing and estimating problem. IEEE Trans. Software Eng., 4(4): 345-361.
Shepperd M, Schofield C (1997). Estimating Software Project Effort Using Analogies. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING 23(11): 736-743.
Stensrud E (2001). Alternative approaches to effort prediction of ERP projects. .Inf. Softw. Technol., 43(7): 413-423.
Tosun A, Turhan B, Bener AB (2009). Feature weighting heuristics for analogy-based effort estimation models. Expert Syst. Appl., 36(7): 10325-10333.
Walkerden F, Jeffery R (1999). An Empirical Study of Analogy-based Software Effort Estimation. Empirical Softw. Eng., 4(2): 135-158.