COMPARATIVE ANALYSIS FOR ESTIMATING DEVELOPMENT EFFORT OF SOFTWARE PROJECTS USING MODA, ANFIS AND COCOMO

[1]Yogesh Kumar, [2]Rahul Rishi
[1]Research Scholar, UIET, MDU Rohtak

Abstract : The consistent and precise effort prediction in software package engineering is a current challenge. There are various methods and technique available of precise effort prediction. In this paper a reengineering model is proposed based on MODA to measure software development effort. The proposed model is then compared with ANFIS and COCOMO models. From the experimental results, it is concluded that the proposed MODA model has low Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Mean Magnitude of Relative Error (MMRE).

Index Terms **– MODA, ANFIS, COCOMO, Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Mean Magnitude of Relative Error (MMRE).**

1. Introduction

Software project supervisor typically estimate the software package development effort. If the exertion is calculable value and time-span are often simply calculated as a result of value is that the product of operating hours and remuneration, time-span depends on what percentage hours someone works. Precise Effort is critical within the early stages of a software package life cycle so as to befittingly arrange, monitor and management of the allotted resources for software package project development activities. Hence, preciseness in estimating the specified software package development effort plays a vital issue on the success of a software package development or maintenance project.
The consistent and precise effort prediction in software package engineering is a current challenge. There are various methods and technique available of precise effort prediction.

2. PROPOSED MULTIOBJECTIVE DRAGONFLY ALGORITHM (MODA) FOR OPTIMAL FEATURE SELECTION

The way of multiobjective optimization is phenomenally particular from mono objective optimization. Hence to solve multiobjective troubles, Pareto optimality based non-ruled arranging is utilized to distinguish arrangements of the ensuing innovation. So any abatement in time many-sided quality of the non-commanded arranging set of principles may even enhance the general execution of the MODA algorithm[7].

The principle thought of the DA starts from static and dynamic swarming behaviours. These two swarming behaviours are fundamentally the same as the consequent two basic periods of advancement the utilization of meta-heuristics: exploration and exploitation. The rule objective of any swarm is survival; thus, the greater part of the people of a swarm should be pulled in nearer to dinners sources and occupied toward outward enemies. Considering those two practices, there are five principle components inside the position updating of individuals in swarms.

These five parameters[7] consist of control cohesion, alignment, separation, appeal (closer to meals resources), and distraction (in the direction of outward enemies) of individuals in the swarm. The separation of the i^{th} dragonfly, S_i shape its neighbours is computed as Eq.1

$$S_i = -\sum_{j=1}^{N'} \left(X' - X'_j \right) \tag{1}$$

Where X' is the location of the current individual, X'_j is the location the j^{th} neighbouring individual, and N' is the amount of neighbouring individuals. Alignment is designed as shown in Eq.2

$$A_i = \frac{\sum_{j=1}^{N'} V'_j}{N'} \tag{2}$$

Where V_j' displays the velocity of the j^{th} neighbouring individual. The cohesion is intended as shown in Eq.3

$$C_i = \frac{\sum_{j=1}^{N'} X'_j}{N'} - X' \quad (3)$$

Attraction to a food source is intended as exposed in Eq.4

$$F_i = Food - X' \quad (4)$$

Where Food displays the position of the food source. Distraction towards an outwards an enemy is intended as exposed in Eq.5

$$E_i = Enemy + X' \quad (5)$$

Where Enemy displays the location of the enemy. The behaviour of dragonflies is expected to be the grouping of these five corrective patterns in this section.

Input:

$p_m, m = 1,2,.....M.Multiobjeative\ problem with\ Mobjectives$

$S_D = searchspace of\ D decisionvar iables$

$ITR = Maximum Number of\ Iterations$

Output:

New Population

Begin Procedure

for i=1 to N

Initialize each solution randomly with in respective search space range

Apply non-dominated sorting according to their pareto front number.

Apply control cohesion, alignment, separation, attraction (towards food sources), and distraction formulas

Evaluate Each Solution fitness

Apply selection and first keep good rank solution, in case of tie solution choose solution with higher crowding distance.

End for

End Procedure

Fig 1: Multiobjective Dragonfly Algorithm

At first we need to configuration design targets in view of software testing parameters, for example, core necessities, Risks and intricacy, Team quality, Historical information, Resources accessibility. Additionally, re-engineering is allowed in created configuration design pattern model based on the type of object oriented software projects. Based on the definition of configuration design destinations, a MODA is utilized to locate the best exchange off between the plan design goals. The proposed multi-objective bio- inspired algorithm yields programmed modification with ideal example based re-engineering model for Software effort estimation. At last, we lead Software estimation on one protest arranged software project in view of the proposed ideal example based re-designing model utilizing the assistance of Extreme learning machine (ELM).

2. ADAPTIVE NEURO FUZZY INFERENCE SYSTEM (ANFIS):

Adaptive neuro fuzzy inference system (ANFIS) is a mix of two delicate figuring strategies for ANN and fuzzy logic. Fuzzy logic can change the subjective parts of human learning and bits of knowledge into the procedure of exact quantitative investigation. In any case, it doesn't have a characterized strategy that can be utilized as a guide during the time spent change and human idea into administer base FIS, and it additionally sets aside a significant long opportunity to alter the membership function (MFs). Dissimilar to ANN, it has a higher ability in the learning procedure to adjust to its condition. Along these lines, the ANN can be utilized to consequently change the MFs and diminish the rate of blunders in the assurance of standards in fuzzy logic. This segment will portray in points of interest of the engineering of FISs, ANFIS and adaptability of system, and algorithm of hybrid learning.

ANFIS is the Combination of the ANN and the FIS as shown in Fig. 2. ANN engineering is utilized and the fuzzy logic rules are utilized for the ANFIS.

2.1 ANFIS Architecture:

Neural network is good in learning and highly interpretable but fuzzy logic is good in handling imprecision [1]. The advantage of both the neural network and fuzzy logic can be combined by using the neuro fuzzy model. The neuro-fuzzy approach has added the advantage of reduced training time, not only due to its smaller dimensions but also because the network can be initialized with parameters relating to the problem domain.

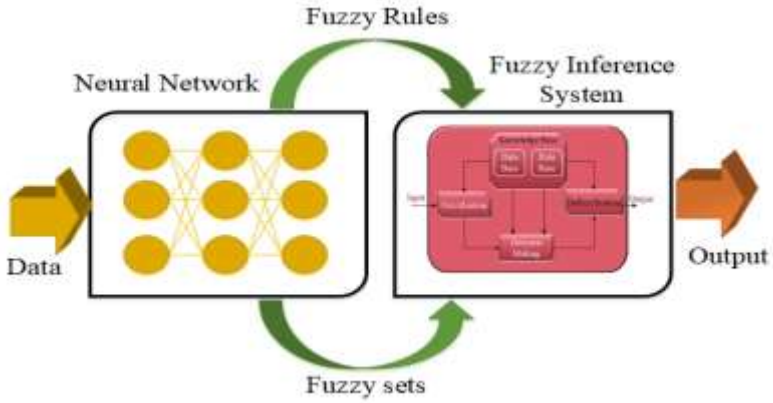

Fig 2: Combined Architecture of ANN & FIS

A specific approach in neuro-fuzzy [2] development is the adaptive neuro-fuzzy inference system (ANFIS), which has shown significant results in modelling nonlinear functions. ANFIS uses a feed forward network to search for fuzzy decision rules that perform well on a given task. Using a given input-output data set, ANFIS creates a FIS whose membership function parameters are adjusted using a back propagation algorithm alone or a combination of a back propagation algorithm with a least squares method. This allows the fuzzy systems to learn from the data being modelled.

ANFIS (Adaptive Neuro-Fuzzy Inference system) used in this paper is of sugeno type Fuzzy inference system. It applies the combination of Least square method and the back propagation gradient descent method for training FIS membership function parameters to emulate the given training dataset[1].

Consider a first order Takagi-Sugeno fuzzy model with a two input (x,y), one output system having two membership functions for each input. A first-order Sugeno fuzzy model has two rules:

• Rule1: If x is A1 and y is B1, then f1 = p1x + q1y + r1

• Rule2: If x is A2 and y is B2, then f2 = p2x + q2y + r2

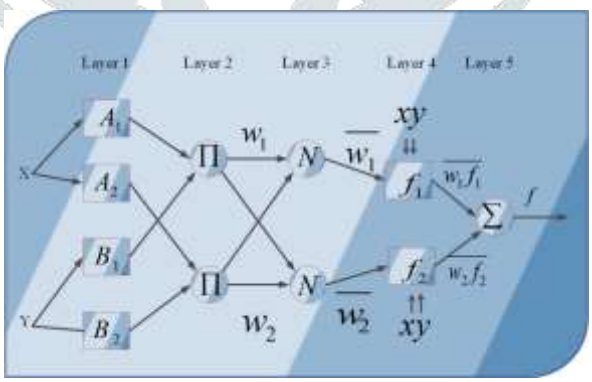

Fig.3 Architecture of ANFIS

The functioning of ANFIS is a five-layered feed-forward neural structure, and the functionality of the nodes in these layers can be summarized as:

Layer 1: Every hub I in this layer is a square hub with a hub work

$$o_i^1 = \mu_{A_i}(a)$$

Where a is the contribution to hub i, Ai is the semantic mark related with this hub work. In other words, Oi is the participation capacity of Ai and it indicates how much the given x fulfils the quantifier Ai. Superscript 1 of Oi demonstrates this is the yield of first layer. Parameters in this layer are alluded to as the start parameters [3].

Layer 2: Every hub in this layer is circle hub marked II, which duplicates the approaching signs and sends the item out. Every hub yield speaks to the terminating quality (or weight) of a run the show.

Layer 3: Every hub in this layer is a circle hub marked N. The I-th hub figures the proportion of the I-th rules terminating quality to the whole of all rules‟ terminating qualities.

Layer 4: Every hub I in this layer is a square hub with a hub work,

$$o_i^{\,4} = \overline{w_i}\left(p_i x + q_i y + r_i\right)$$

Layer 5: It is a circle hub that totals every approaching sign.

Obviously the ANFIS has two arrangement of customizable parameters, to be specific the introduce and ensuing parameters. Amid the learning procedure, the preface parameters in the layer 1 and the resulting parameters in the layer 4 are tuned until the coveted reaction of the FIS is accomplished. The half breed learning calculation, which joins the slightest square technique and the back-propagation calculation, is utilized to quickly prepare and adjust the FIS. At the point when the commence parameter estimations of the Membership Functions are settled, the yield of the ANFIS is spoken to as a direct mix of the subsequent parameters [4].

The Least Square Method is utilized to decide ideally the estimations of the subsequent parameters. At the point when the preface parameters are not settled, the hunt space ends up bigger and the joining of preparing turns out to be slower. The cross breed learning calculation can be utilized to take care of this issue. This calculation has a two-advance process. To begin with, while holding the introduce parameters settled, the useful signs are proliferated forward to layer 4, where the resulting parameters are recognized by the Least Square Method. At that point, the ensuing parameters are held settled while the error signals, the subordinate of the mistake measure regarding every hub yield, are spread from the yield end to the info end. The introduce parameters gets refreshed by utilizing Back-propagation calculation.

3. CONSTRUCTIVE COST MODEL (COCOMO):

It is a method used in very wide and it gives the results usually are accurate. It was first introduced by Barry W. Boehm in 1981.Dataset used for analysis and validation of the model can be got from historic projects of NASA. One set of dataset consists of 63 projects and other consists of 93 datasets. The datasets is of COCOMO II format. In our experiment 93 datasets are used for training and 63 data is used for testing. The Dataset need for training as well as testing is available in www.promisedata.org/?p=6 and in www.promisedata.org/?p=35 . The dataset available is of COCOMO 81 format which is to be converted to COCOMO II by following the COCOMO II Model definition manual [5] and Rosetta stone [6] COCOMO 81 is converted to COCOMO II. COCOMO 81 is the earlier version developed by Barry Boehm in 1981 and COCOMO II is the next model developed by Barry Boehm in year 2000. Some of the attributes like TURN are used only in COCOMO 81 and some new attributes like RUSE, DOCU, PCON, SITE are introduced in COCOMO II.

Table.1 Scale factors and their range

Scale Factor		Range
Precedentedness	PREC	0.00-6.20
Development Flexibility	FLEX	0.00-5.07
Architecture/Risk Resolution	RESL	0.00-7.07
Team Cohesion	TEAM	0.00-5.48
Process Maturity	PMAT	0.00-7.80

Table.2 Effort Multipliers and their Range

Effort Multipliers		Range
Required software Reliability	RELY	0.82-1.26
Data base size	DATA	0.90-1.28
Product Complexity	CPLX	0.73-1.74
Developed for Reusability	RUSE	0.95-1.24
Documentation Match to Life- Cycle needs	DOCU	0.81-1.23
Execution Time Constraints	TIME	1.00-1.63
Main storage Constraint	STOR	1.00-1.46
Platform Volatility	PVOL	0.87-1.30
Analyst capability	ACAP	1.42-0.71
Programmer capability	PCAP	1.34-0.76
Personal continuity	PCON	1.29-0.81
Applications Experience	APEX	1.22-0.81
Platform Experience	PLEX	1.19-0.85
Language and Tool Experience	LTEX	1.20-0.84
Use of software tool	TOOL	1.17-0.78
Multisite Development	SITE	1.22-0.80
Required Development Schedule	SCED	1.43-1.00

Every input as Effort multiplier has been tuned by following the COCOMO II model definition manual [5]. The scale factor and Effort multiplier and their range is given in Table 1 and 2.

4. EXPERIMENTATION & RESULTS

The results are compared between MODA, ANFIS and COCOMO. Data available with open source (https://sites.google.com/site/whuxyjfmq/home/see_osp_dataset) public dataset is used for experiment. Out of 10 cross company projects, 6 projects data is used for training the MODA and 4 project data is used for testing.

COCOMO has three models however the barest essential is Post Architecture Model. This model is used when all the designing of the wander headway has been picked. It uses the degree of undertaking (in regards to Kilo source line of codes (KSLOC)) and furthermore 22 cost drivers, which joins five scale segments and 17 effort multipliers, for the estimation of effort. The yield of COCOMO, to the extent Effort (singular/month), is figured as :

$$E = A.(KSLOC)^{B}.\prod_{j=1}^{17} Effort\,Multiplier(j)$$

Where $B = 1.01 + 0.01 \times \sum_{j=1}^{5} scale\ factor(j)$

A is called as multiplicative perpetual.

Our proposed MODA algorithm follows Parkinson's Law, a numerical model is communicated for processing STE for each unsupervised task before the setting up of PBRE demonstrate. Here, unsupervised endeavour alludes back to the dataset without STE .

$$STE = \left(\sum_{i=1}^{t} DDX \sum_{j=1}^{t} DP\right) \times TP\%$$

Where t is total projects, DD the extension is time frame in months; DP is the change work force and TP is the trying extent.

The data about the Lines of code and capacity purposes of each cross organization venture is accessible in Table 3. The result of prediction co-efficient is depicted in Table 4.

Table 3: Lines of code and function points of each cross company project

Cross Company Name	Line of code	Function Points	Actual Effort
C12	14000	218	38
C13	12000	187	32
C14	09000	140	24
C15	06000	93	15
C23	08000	125	21
C24	05000	78	13
C25	10000	156	26
C34	15000	234	41
C53	11000	171	29
C45	13000	203	35

Table 4: Predicted Effort for COCOMO, ANFIS and MODA

Cross Company Name	COCOMO	ANFIS	MODA
C12	33.54	35.43	37.83
C13	27.89	30.87	32.33
C14	18.32	20.98	23.756
C15	12.14	14.26	15.2772
C23	17.16	19.86	21.05
C24	8.76	10.14	12.57
C25	22.76	24.57	26.6
C34	35.12	37.45	40.71
C53	24.45	26.56	29.34
C45	30.12	32.45	35.1212

The mean outright blunder measures of how far the evaluations are from genuine esteems. It could be connected to any two sets of numbers, where one set is "actual" and the other is a gauge forecast.

$$MAE = 1/n \sum_{i=1}^{n} |P_i - A_i|$$

The RMSE (Root Mean Square Error) is defined as,

$$RMSE = \left(\sqrt{1/n \sum_{i=1}^{n} (P_i - A_i)^2} \right)$$

Where P_i = Predicted rate
A_i =Actual rate
n = Total number of data points.

For assessing the distinctive programming exertion estimation demonstrate, the most generally acknowledged assessment criteria are the mean magnitude of relative error (MMRE). The magnitude of relative error (MRE) is characterized as takes after:

$$MRE_i = \frac{absoulte(Actual\,Effort_i - Predicted\,Effort_i)}{Actual\,Effort_i}$$

The MRE esteem would be figured for every perception whose exertion is anticipated. The summation of MRE over different perceptions (N) can be accomplished through the Mean MRE (MMRE) as takes after:

$$MMRE = \frac{1}{N}\sum_i^N MRE_i$$

Table 5: Accuracy rate of Effort Estimation of 10 cross company selected software project from open source Projects Data

Cross company Name.	COCOMO	ANFIS	MODA
C12	87.67	93.14	95.87
C13	86.34	92.89	96.34
C14	87.45	93.45	95.23
C15	87.13	93.76	96.31
C23	86.43	92.78	95.87
C24	86.67	93.67	96.32
C25	87.54	92.64	95.78
C34	87.34	93.17	96.65
C53	536.657	16.500	95.56
C45	546.700	955.250	96.43

Table 6: Comparison of errors with different Estimation Models

Error Analysis	COCOMO	ANFIS	MODA
Mean Absolute Error (MAE)	3.45	2.21	1.32
Root Mean Square Error (RMSE)	3.56	2.45	1.14
Mean Magnitude of Relative Error (MMRE)	3.78	2.89	1.34

The table 5 describes the accuracy rate of three methods. The proposed algorithm give better performance than other algorithms. The table 6 describes the error comparison of three methods. The proposed algorithm gives better performance than other existing algorithms.

5. CONCLUSIONS

This paper presents evaluation of our proposed MODA with ANFIS in amalgamation with COCOMO II. The modification in GUI enabled estimator to enter cost driver values on continuous scale. The result of comparison indicated that Mean Absolute Error, Root Mean Square Error and Mean Magnitude of Relative Error are reduced in our proposed MODA algorithm compared to other models.

REFERENCES :

[1]. Xishi Huang, Danny Ho, Jing Ren, Luiz F. Capretz, "Improving the COCMO model using the Neuro fuzzy approach", Journal of applied soft computing, pp 29- 40, 2007. approach", Journal of applied soft computing, pp 29- 40, 2007.
[2]. Anish Mittal, Kamal Prakash, Harish Mittal, " Software Cost estimation using fuzzy logic", ACM SIGSOFT software Engineering Notes, Vol.35, Nov 2010.
[3]. Zia, Z., Abdur Rashid, and Khair uz Zaman, "Software cost estimation for component-based fourth-generation-language software applications", IET software, vol.5, no. 1, pp.103-110, 2011.
[4] Nagar, Chetan, and Anurag Dixit, "Software Efforts and Cost Estimation with a Systematic Approach 1, 2011.
[5] Barry Boehm, COCOMO II: Model Definition Manuel. Version 2.1, Center for Software Engineering, USC, 2000.
[6] Donald J. Reifer, Barry W. Boehm and Sunitha chulani, "The Rosetta stone: Making COCOMO 81 Estimates work with COCOMO II", CROSSTALK The Journal of Defense Software Engineering, pp 11 – 15, Feb.1999.
[7] Seyedali Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problemsDragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems" in Neural Computing and applications, May-2016,Vol. 27, Issue 4, pp. 1053-1073.
[8]. Yogesh Kumar, Neeraj Varshney " Comparative analysis of software size estimation techniques in project management", in International journal for research in applied science & engineering technology, Vol. 5, Issue VIII, Aug-2017. Pg 1470-1477.
[9]. Mani, Yogesh Kumar " Software development life cycle models", in International Journal of Emerging Technologies and Innovative Research, Vol 5, Issue 2, Feb-2018. Pg 107-113.
[10]. Tannu, Yogesh Kumar, "Comparative Analysis of Different Software Cost Estimation Methods", International Journal of Computer Science and Mobile Computing, Volume 3, Issue 6, 04 July 2014, pg.547-557.
[11]. Yogesh Kumar "A Review on Effort Estimation Techniques used in Software Projects", in International Journal of Computer Science & management Studies, Volume 14, Issue 3, March 2014. Pg. 25-31.
[12]. Yogesh Kumar, Rahul Rishi "A Review of Different Approaches/Methodologies used for Programming Exertion Estimation", in International Journal of Emerging Technologies and Innovative Research, Vol 5, Issue 6, June-2018. Pg 573-579. (ISSN : 2349-5162).