Dragonfly Estimator: A Hybrid Software Projects' Efforts Estimation Model using Artificial Neural Network and Dragonfly Algorithm

Qais M. Yousef*
Computer Engineering department,
ATIT Academy, Amman, Jordan;

Yasmeen A. Alshaer
Project Management department,
ATIT Academy, Amman, Jordan;

Noor K. Alhammad
Computer Engineering department,
ATIT Academy, Amman, Jordan;

Abstract
The estimation of software development efforts has become a crucial activity in software project management. Due to this significance, a few models have been proposed so far to build a connection between the required efforts to be employed, and the software size, time schedule, budget and similar requirements. However, various holes and slips can still be noticed in software effort's estimation processes due to the lack of enough data available in the initial stage of project's lifecycle. In order to improve the accuracy of time estimation in the software industry, this work used NASA projects dataset to train and validate the proposed model, which is based on Feedforward Artificial Neural Network. Moreover, Dragonfly Algorithm was used to provide optimal training, which in consequence offered more enhanced and accurate software estimation model. Randomly selected project datasets were used to test the proposed model, which resulted in clear enhanced results in comparison to similar estimation models. Different performance criteria were used to validate and accept the hypothesis suggested by this paper that the proposed model could be used in predicting the efforts required for various types of software projects.

Keywords
effort estimation, software projects, software development, swarm intelligence, artificial intelligence

1. Introduction

Nowadays, there is an increasing trend of using effort estimation information, especially in the business sector. Recently, a huge development in the business sector has been noticed, which require accurate effort estimation during the initial stages of project's lifecycle in order to ensure better results, failure avoidance and bid balance between the developer and the customer, which is known as size estimation paradox [1]. Software development cost and effort estimation are important tasks for software project management [2]. Prediction of software development effort is the critical and important task for the effective management of any complex and large software industry [3]. Therefore, effort estimation gains an increasing importance all over the world.

Moreover, estimation for any industry is designed to save competitive managing balance between the quality and the cost of software [3]. Another important aspect of project management exists in avoiding project abortion and/or restarts. This may sometimes relate to effort estimation, but may also be due to organizational restructuring, changes in the market, customer orders, or related reasons [4].

The accuracy and reliability of the prediction mechanisms are also important. The improvement in accuracy of estimation is a great challenge for software engineering and computer science in general [5]. Overestimation of effort can misguide the management team for excess use of manpower, and delay the project inappropriately, so that the cost of the project may rise to unacceptable high values. While underestimation of project effort puts extreme pressure on project's team and makes it difficult for them to get a quality outcome in a stipulated time, which ultimately results in an unsuccessful project [6].

Correct estimation of software effort is extremely difficult. There is a number of models have been proposed to construct a relationship between software size and effort, for instance, COnstructive COst MOdel (COCOMO), which was published in 1981 [7], by analyzing 63 software project data. Moreover, SLIM [8][9], Walston-Felix [10], Bailey-Basili [11], Doty [12], to name but a few, are software-effort estimation models, may not be able to provide accurate effort estimation, due to the shortage of enough project data in the initial stage of the project. Usually, less information is available before starting the project then the information increases while working in the project. Most of the times, more accurate effort estimation, can be obtained, after deep analysis of a large amount of project data [13]. The need for accurate effort estimation in the software industry is still a challenge. This estimation includes effort, cost, size and time.

Reviewing the literature, most of the proposed solutions can be categorized into non-algorithmic and algorithmic

Manuscript received September 5, 2017
Manuscript revised September 20, 2017

techniques [14]. The non-algorithmic techniques are learning-based while the algorithmic ones are described to be model-based. Although that the learning-based approaches consume more time for estimation and may needs more complicated processes compared to the algorithmic techniques, but on the other side, it has wide flexibility to deal with the new circumstances using various methods, which improve their adaptability to deal with the changes with a lower number of bugs and holes. In addition, it enables the experts to adjust these methods on the opposite of the algorithmic techniques where the models cannot be human interfered. However, using various and wider categories of parameters, the estimation can be achieved accurately at early stages [14].

Evolutionary neural networks, fuzzy and optimization algorithms are examples of non-algorithmic techniques. They are considered as a type of soft computing, which gained increasing interest from 1960 due to its human reasoning characteristics [15]. The importance of neural networks can be derived from its suitability for data generated from large projects [16]. As illustrated in Figure I, big projects are not easy to be estimated, although the needed precision is precise [14]. Due to this importance of such techniques, this work proposes a solution to the issue of software effort estimation based on Feedforward Artificial Neural Network (ANN) [17] models and Dragonfly Algorithm (DA) [18] as a training function. The proposed model evaluated using NASA93 [19] dataset and showed clearly enhanced results in comparison with previous works.

The main purpose of this work is to improve the accuracy of the software effort estimation to mitigate the aforementioned problems that are resulted from the inaccurate overestimation or underestimation. The rest of this paper goes as the following; Section 2 reviews some related works in order to reach contributed design. Section 3 summarizes the methodology that was followed during the work. While section 4 illustrates the proposed method, the experiment and the obtained results are analyzed in section 5.

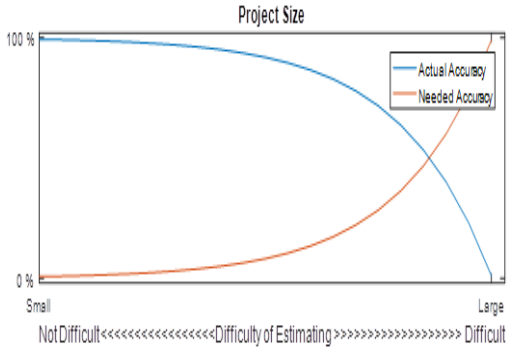

Fig. I Estimation Accuracy [14]

2. Literature review

The aforementioned problems of effort estimation for software projects have been addressed since the 1960s [20][21]. Since then, many researchers have been focused their efforts to innovate new models to solve those problems. These innovated models can be categorized into three categories; expert, formal and combined-based estimation models [22].

Expert-based estimation is the steps where the estimate is coming as a result of judgmental processes. Playing poker model, which is also called Scrum poker, is one of the expert estimation models. This model is a consensus-based, gamified technique for speculation. It is mostly used to estimate effort and required size in software development. James Grenning initially developed this method in 2002, but it was popularized by Mike Cohn in the book Agile Estimating and Planning [23].

Numerous related works [6] [24] [25] show that the expert estimation is the dominant method used for software development effort estimation. The first estimation of software effort in the 1960s relied on expert judgment [26]. Different variations are then proposed, for instance, the estimate in Delphi expert is the mean value of different independent estimates formulated by the developers. The main drawback of expert-based estimation is the absence of the primary objective of accuracy. The field expert is the estimator, which means additional risks. Hence, [26] concluded that in some situations expert estimates are more likely to be accurate, where; similarly, models are more accurate due to situational and human biases. Expert-based estimation can be found as group approaches such as Wideband Delphi and Planning poker or Work breakdown structure (WBS-based approaches) [27].

On the other hand, formal estimation models are based on mechanical processes, and the formula is derived from historical data. The most common models based on this estimation are COCOMO [28], SLIM [29] [30] and SEER-SEM [31] model.

In COCOMO (Constructive Cost Model), which is a procedural software cost estimation model developed by Barry W. Boehm [28], the regression formula is fitted using historical projects data (63 projects for COCOMOII and 163 projects for COCOMO II) to derive the model parameters. COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. The first level, Basic COCOMO is suitable for quick, early and rough order of magnitude estimates of software costs, but its accuracy is limited due to its lack of factors for differences in project attributes (Cost Drivers). Intermediate COCOMO considers these cost drivers

and details them for the influence of individual project phases.

Regression techniques can result with linear models [32][33], nonlinear approaches such as neural networks, tree-based models such as CART [34][35] and case-based reasoning strategies (CBR) [36]. In a non-statistical comparison, compares artificial intelligence models: ANN and CBR with Ordinary Least Squares regression (OLS), and found that ANN and CBR outperformed OLS.

However, formal estimation models can be: 1. parametric, such as COCOMO [28], SLIM [29] [30] and SEER-SEM [31], 2. Analogy-based [37] such as weighted micro function points, and 3. Size-based approaches such as Software Size Unit, Use Case Analysis, Function Point Analysis and Story points-based estimation.

Finally, Combined-based estimation is produced based on a judgmental and mechanical combination of estimates from different sources. Combined-based estimation category shows that combination of estimates from independent sources, preferably applying different approaches, will on average improve the estimation accuracy [38] [39].

Biologically inspired models also have been targeted with many types of research; one of them is Artificial Neural Network (ANN), which is usually combined with other models. Neuro-Fuzzy with SEER-SEM model work [40] which combine both neuro-fuzzy models described in the patent US-7328202-B2 (Huang, Ho, Ren, and Capretz 2008) and SEER-SEM Model shows that how biologically inspired models can be combined with another model to produce unique characteristics and performance improvements.

Combined-based estimation can be 1. A mechanical combination such as the combination of the average of WBS-based approaches and analogy-based ones, and 2. Judgment combination such as the expert judgment based on estimates from a group estimation and algorithmic model.

The best approach to be used depends on the requirements of the project and the strengths and drawbacks of the approaches, which means that no standard estimation approach can be used for all projects and development environments. At this point, the following proposed approach deploys the Feedforward ANN together with DA in order to provide more accurate effort estimation based on roughly presented inputs and outputs.

3. methodology used

The methodology that is followed during the pursuit of this work can be summarized as follow:

a) Problem Definition

Based on the analysis of software effort estimation issues, the problems that face the software effort estimation were analyzed. We find that the major problem is the changeable circumstances of the development environments, which results with uncertain estimation with respect to the needed cost, time, requirements and resources of the software project.

The drawbacks of the cost overestimating will result with undesirable extravagance, which can cause the contract and jobs to be lost. Underestimating the cost will cause the project budget to exceed what is assigned by the project development entity and the quality to be lower than expected if it is completed within its assigned deadline.

Achieving a real and accurate estimation of the software effort is the target of this work to avoid uncertain estimation drawbacks. The motivation behind this work is to hunt the best solution using swarms behavior of dragonfly algorithm during the training method in order to reach a stable neural network behavior later on.

b) Method Selection and Design

Choosing the most appropriate software effort-estimation technique was the core of this work. In this phase, all of the estimation models and methods are analyzed with respect to their strengths such as flexibility, availability and adaptability and their deficiencies such as complexity. At the end of this phase, it is found that the artificial neural networks trained by dragonfly algorithm will improve the currently available estimations.

c) Dataset Selection

Choosing the appropriate dataset to be used for our model verification is a key step of our work. The dataset must be carefully selected [41]. In our work, the NASA93 [19] dataset was selected due to its public availability and its good documentation. This dataset will be normalized first before it is used in the ANN model.

d) Model Verification

After method implementation, the achieved estimations are compared to COCOMO [7], BPNN [42] and RBNN [43] models based on Mean Square Error (MSE), Mean Magnitude of Relative Error (MMRE) and Mean Absolute Error (MAE) performance metrics.

4. Background

a) Software Effort Estimation

Historically, software effort estimation has been addressed in literature since at least 1960s [44] [45]. Software cost estimation is required to develop or maintain software

according to uncertain, noisy and incomplete inputs. It includes the determination of one or more of 1. Effort (usually measured in person-hours or months), 2. Project duration (measured in calendar time) and 3. Cost (measured in dollars).

Accurate software effort estimation is a demand for both customers and developers. Its importance stems from its use in proposals requests, contract transactions, time scheduling, control and monitoring [46]. The drawbacks of the cost overestimating will result with undesirable extravagance, which can cause the contract and jobs to be lost. Underestimating the cost will cause the project budget to exceed what is assigned by the project development entity and the quality to be lower than expected if it is completed within its assigned deadline.

Although that the effort estimation models have an objectivity, repeatability, the presence of supporting sensitivity analysis, and the ability to calibrate to previous experience strengths [7], the required accuracy is becoming increasingly important as the development environment is changeable from time to time. The deadline for any project can be affected by the implementation due to this instability of circumstances, which leads to inaccurate estimation due to the outdated inputs.

Different performance criterion metrics are used to evaluate the accuracy of estimation [47]. These metrics include:

1. Mean Magnitude Relative Error (MMRE) [48]: it is appropriate for comparisons using datasets but it depends on the scale [49]. A model, which gives lower MMRE, is better than the model that has higher MMRE [50].
2. Mean Absolute Relative Error (MARE): The lower MARE is better [51] [52]. Moreover, each of the MMRE or MARE can be used on the dataset even under a complex trial and error heuristic evaluation [53].
3. Mean Absolute Error (MAE): MAE shows the difference between the mean value and the recorded value, which clarifies the variance of the used algorithm. However, for this criterion, the lower the better.
4. Variance Absolute Relative Error (VARE): The models that provide lower VARE are better [54].
5. Balance Relative Error (BRE): decreasing the value of BRE enhances the efficiency of the model.
6. Prediction (n): it is appropriate for underestimation cases. It is defined as the percentage of projects that has an absolute relative error less than n. A model which provides higher Pred (n) is more efficient than the ones with lower Pred (n) values [55].
7. Variance Accounted For (VAF): the higher the value of VAF the more satisfactory the model is.
8. Median Magnitude of Relative Error (MdMRE): MdMRE [54] exhibits a similar pattern to MMRE but it is more likely to select the true model especially in the underestimation cases since it is less sensitive to extreme outliers.
9. Root Mean Square Error (RMSE): It is appropriate when errors are measured using the same units. The criterion of good quality using this metric depends on the variables measurement criterion and the degree of forecasting precision [56], so there is no standard criterion of quality.

b) Artificial Neural Networks

Artificial Neural Network (ANN) is an approach of artificial intelligent utilized to generate mathematical models that mimic the structure of humans neural system [57]. In ANN, the model consists of layers; input, hidden and output. Each layer consists of a number of neurons. Each one of these neurons is a logistic function that takes inputs and produces outputs according to the function implemented in the neuron.

The number of neurons in the input layer equals the number of features utilized in the experiment. For example, the number of input nodes in the input layer in our work is 24 (as shown in Table VI). The number of nodes in the hidden layer is an optimization issue, but in our model, this parameter is coming by experiment. Different empirical values have been used in order to choose the best value that achieves results that are more accurate. In our model, it contains 1 hidden layer with 30 nodes. Finally, the number of nodes in the output layer depends on the number of required output values. In our model, there is 2 output neuron, which represents the required code length and efforts to be able to implement the requested software project successfully.

The mathematical model of the ANN consists of a hypothesis function $h\theta$, which is a summation of functions of the other layers. In this work, the following mathematical models demonstrate the utilized functions. Table I shows variable descriptions.

$$g(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_n x_n \quad (1)$$

$$h(\theta)_i^j = \frac{1}{1 + e^{-g(x)}} \quad (2)$$

$$fi = \sum_{i=1}^{m} h(\theta)_i \quad (3)$$

$$f_{final} = \sum fi \quad (4)$$

The input to the artificial neurons is numeric values (vector x) which are perceived with particular independent weights (vector θ). To optimize variables or weights θ, DA algorithm has been utilized. In the next sections, we will describe how this algorithm was adopted to find the optimal values of these weights.

c) Dragonfly Algorithm

Dragonfly Algorithm (DA) is a relatively new intelligent swarm optimization algorithm proposed in 2015 [18]. The algorithm has been proposed based on dynamic and static swarming behaviors. DA consists mainly of two phases; exploration and exploitation. In exploration phase, dragonflies search for swarm or areas of interest. Subsequently, they create sub statistical swarms. These statistical swarms are used in exploration process to find the optimal solution of the problem.

As illustrated in Figure II, dragonflies have five different flying behaviors. First, separation in which they fly away from each other to explore a vast area to generate sub swarms. Second, alignment in which they fly parallel to each other in army way to reach a globally optimal solution. Third, cohesion in which they meet at a certain point. The last two behaviors are an attraction to food and distraction from the enemy in which they try to search near better solutions and stop searching in far swarms.

The optimization problem must be predefined carefully to utilize DA for optimization. In this work, our optimization problem is to find the global minimum value of cost function of ANN algorithm, which is defined in Equ.5 and its variables' description are summarized in Table I.

$$Cost = \frac{-1}{m} * \left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log_k\left(h_\theta(x^i)\right) + \left(1 - y_k^{(i)}\right)\log_k\left(1 - h_\theta(x^i)\right)\right] + \frac{\beta}{2m}\sum\sum\sum \theta_{ik}^{l} \quad (5)$$

The optimization goal is to find values of θ to minimize the cost function. In other words, we seek to find the global minimum value of this function, which in turn results in obtaining the most optimal values of weights for the used ANN model.

DA starts to search for the global optimal minimum value of this function. However, to reduce searching or exploration time, a minimum value and a maximum value of the searching area should be given. Figure II shows the behavior of DA particles in looking for the optimal solutions.

Subsequently, a number of dragonflies start to search the area and divide it into sub-areas for potential solutions. These potential solutions are implemented in the cost function. If the output of the function is higher than last round, these solutions

are marked enemies and dragonflies will run away from them. If the output is smaller, the dragonflies will be attracted to this area to find better solutions. This process, which is summarized in algorithm I, continues for a number of iterations when most of the dragonflies cohesion in the same point.

Feedforward ANN [58] can be trained using DA to approximate the needed effort accurately. In other words, the ANN flows in one direction starting from the inputs towards the outputs, which is the estimated effort, Personnel, Product, Platform and Project factors in our case. The optimal output values are chosen, after training the ANN by the DA as is illustrated in following sections.

Table 1: description of variables

M	Number of training data
h_θ	The hypothesis function
$y_k^{(i)}$	The real output of sample i in the training data
θ	Optimization variables
β	Scaling variable

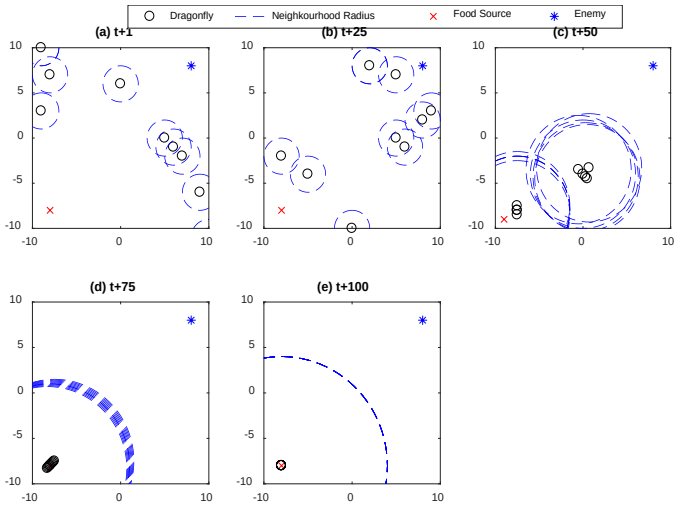

Fig. II Dragonfly Behaviours [18]

d) NASA93 Database

In order to be able to develop an estimation model using DA training algorithm and Feedforward ANN method and evaluate it successfully, NASA93 dataset [19] was used, which was obtained from different project centers, collected by Jairus Hihn [59]. It is constituted of 93 flight or ground projects, and 24 cost drives to contribute to productivity, and then to cost. These cost drives consider Kitchenham analysis [60], which includes the subjective assessment of product, hardware, personnel and project attributes, and can take the values as in Table II. The classified cost drives are illustrated in Table I in Annex A.

Algorithm 1: Dragonfly Algorithm (DA)

1. ***Initialize Particles X_i (i = 1, 2, ..., n)***
2. ***Initialize Step vectors ΔX_i (i = 1, 2, ..., n)***
3. *while* ***not (stopping criteria)***
4. ***Measure the objectives for all particles***
5. ***Update solutions***
6. ***Update coefficients (a, c, e, f, s, w)***
7. ***Calculate the five behaviors***
8. ***Update neighbor list***
9. *if* ***a particle has a minimum of one neighbor particle***
10. ***Update velocity vector***
11. ***Update position vector***
12. *else*
13. ***Greedy update position***
14. *end if*
15. ***Validate the new position***
16. *end while*

Alg. I. Dragonfly Algorithm (DA)

Moreover, the projects used (forg), their category (cat2), in addition to the projects, which were developed in centers (center), and the mode of software development (mode), are also considered and shown in Table I in Annex A. tool attribute enables selection the records about different development languages such as FORTRAN and year attribute enables selection records related to the development year, while project name means Project Name, From 8-22 attributes enables different COCOMOI designations such as BUS for business applications and SYS for system software. MODE attribute describes the development mode. It can take one of the following the values shown in Table II. While attributes equivphyskloc and act_effort are real numbers describe the required code length and effort respectively.

It is worthy to say that COCOMOI is the first version of COCOMO model. It follows the waterfall process using the aforementioned imperative programming languages [61], which is evaluated using COCOMO dataset, which is adopted here for comparison purposes. In addition to its lineage. NASA93 is adopted in this work due to its wide use and evaluation in a wide range of organizations. Also, it has an available well documentation, which is supported by public domain and commercial tools.

Table 2: cost drive values [60]

Super Low	SL
Extra Low	EL
Very Low	VL
Low	L
Normal	N
High	H
Very High	VH
Extra High	XH
Super High	SH

5. proposed approach

In this work, Artificial Intelligent (AI) methods have been utilized to generate a prediction model to predict and estimate the required time to produce a new software or programming project. Artificial neural networks and the case-based reasoning [62] are potential approaches to artificial intelligence. While the case-based reasoning depends on exploiting the solutions of old problems that are similar to the current problem, the artificial neural networks can estimate accurately in case of distorted inputs or when the variables have complex relationships. To build the proposed model, supervised Feedforward Artificial Neural Network (ANN) model has been utilized. In supervised learning, we should have a preconceived idea about the input and output data. In other words, the features and the predicted effort values should be presented. Subsequently, they are used to train the mathematical model [63].

To train this proposed mathematical model NASA93 dataset has been utilized. The dataset consists of 93 samples. Each sample has 24 cost drives (Features) and the estimated output values, as shown in table I in Annex A. These features have been normalized as the first step of the proposed approach. Equ.6 has been used in normalizing this data. Subsequently, these features have been fed into the ANN model as illustrated in Tables III and IV

$$\frac{x_i - \mu}{Max(x)} \quad (6)$$

In order to train the ANN, several training algorithms were tested in order to have the lowest Mean Square Error (MSE), which results in the best overall system performance. FigureIII shows that DA has the lowest MSE in training the ANN for NASA93 dataset compared to Back Propagation Neural Network (BPNN) [42] which showed the heights MSE value of 0.04. Recurrent Neural Network (RNN) [64] with lower MSE value, but still not enough to be used to train the ANN. Artificial Bee Colony (ABC) [65] reached a good result with 0.005 MSE, and Genetic Algorithm (GA) [43], which was higher in MSE with 0.021. However, the DA reached the best MSE result with 1.1×10^{-8} which is a promising value, which indicates that most optimal weights and bias values for the ANN can be reached.

While BPNN calculates the error at the output and distributes it back through the network layers, RNN uses loops for signal flow, which means the signal can flow backward or shift for a number of steps. On the other hand, ABC adopted swarm intelligence and is motivated by the intelligent behavior of honey bees, while GA adopted the Darwinian notion of natural selection. GAs search all possible solutions of the problem under study which are computed using their fitness. The best solutions will remain with the next generations and produce "offspring" which are variations of their parents.

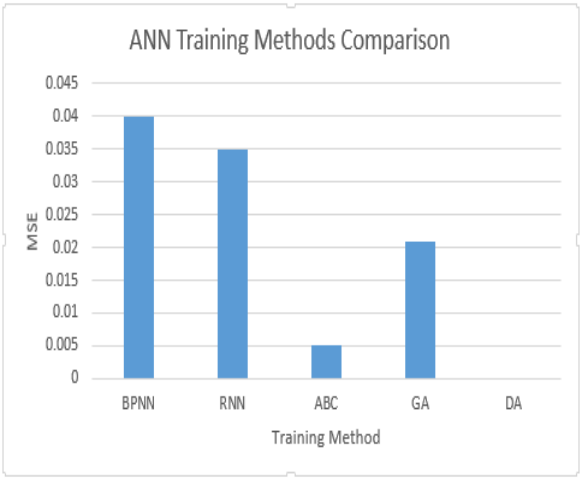

Fig. III ANN Training Methods Comparison

e) Model Design

After obtaining the above results, ANN was trained with DA using the specifications shown in Table V and VI, which are used for the selection process. Figure IV (below), summarizes the steps followed in the proposed method.

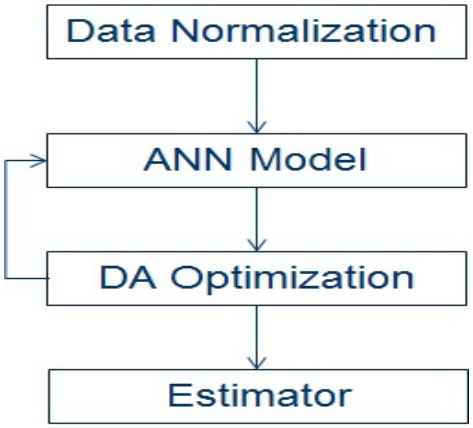

Fig. IV Steps of The Proposed Method

This model has two phases; the training phase and the testing phase. During the training phase, the selected training dataset will be initially normalized using Equ 6, in order to be able to be used with the ANN model. The normalized data then enters the ANN model, which is trained by the DA. Then the ANN can estimate the required efforts to complete the requested project successfully. The stopping criterion for this phase is tied to the MSE value of the ANN, in comparing the output values with the real targets. This criterion depends on the cost function value reached by the DA, which is required to be as close to zero

as possible. The entire model is illustrated in figure V. Moreover, the convergence curve for the used DA is shown in figure VI (below).

Then comes the testing phase, which is the operation phase, where the proposed model can be used in real projects. In this phase, the randomly selected testing dataset is normalized also and then entered to the ANN, which in turn can provide accurate and quick estimation for the required efforts.

Table 3: ANN Specifications

Hidden Layers	1
Nodes in hidden layer	30
Number of training data	93 vectors
Epoch	5500
Training Dataset	70% of 93 vectors
Validation Dataset	15% of 93 vectors
Testing Dataset	15% of 93 vectors

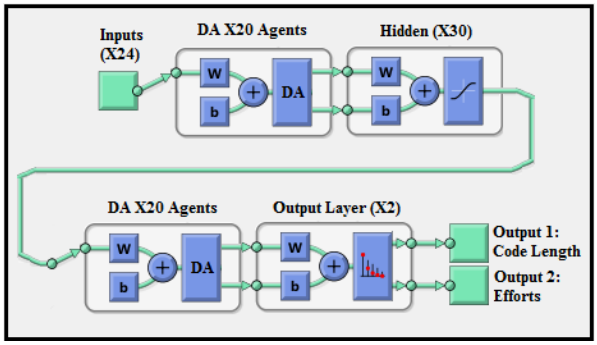

Fig V ANN-DA Model

Table 4: ANN-DA configuration parameters

Number of Nodes in Layer 1	24
Number of hidden Layers	1
Number of Nodes in hidden Layer	30
Number of Nodes in output layer	2
Iteration for DA	500
Number of Agents	20
Max DA value	500
Min DA value	-500

6. evaluation experiments

To measure the performance of the proposed model, three different models have been implemented and compared with our work. The first model is the COCOMOII model [7]. The second is the ANN model with backpropagation training process (BPNN). The last one is Radial Basis ANN (RBNN) model, which has been proposed in [43]. MATLAB has been used to implement these algorithms.

To compare these algorithms with the proposed model, three performance metrics have been evaluated. In the next sub-section, these metrics will be demonstrated. Subsequently, the obtained results will be discussed later in this section.

a) Performance Metrics

Three software effort estimation performance metrics are utilized in this work, and they are described in the following:

1. Mean Square Error (MSE): MSE is the output of the final value the cost function gets in the training process. It can be defined in Equ.7. The MSE should be as low as possible in order to consider the model.

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - h_i)^2 \quad (7)$$

Where, y_i is the recorded output in the dataset, h_i is the estimated output obtained from the model and m is the number of training elements '93'.

2. Mean Magnitude of Relative Error (MMRE): MMRE measures the Balance Relative Error (BRE) for a group of data. BRE is defined in Equ.8 and MMRE is defined in Equ.9. A model, which gives lower MMRE, is better than the model that has higher MMRE

$$BRE = \frac{|Real - Predicted|}{\min(Real, Predicted)} \quad (8)$$

$$MMRE\ (\%) = \frac{1}{m}\sum_{i=1}^{m} BRE_i * 100 \quad (9)$$

3. Mean Absolute Error (MAE)
MAE is defined as the differences between the mean value and the recorded value. Equ.10 shows the MAE. The lower MARE is better

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|h_m - h_i| \quad (10)$$

b) Experimental Results

After implementing and training the proposed model, randomly selected project dataset, representing 15% of the NASA93 Dataset were used to evaluate the proposed model and compare its results with the other models, using

the same testing dataset. However, important results were obtained, which evaluated the proposed model.

Figure VII shows MSE of these algorithms. It can be observed from the figure that COCOMO recorded the highest error value. Since this algorithm has no optimization methods this result was expected. Moreover, we can observe that RBNN obtained results better than BPNN since RBNN attempts to memorize the training examples with only 44 training data; RBNN can memorize most of them. Finally, we can observe how DA attempted to minimize the cost function to around zero (1×10^{-4}).

Figure VIII shows MMRE result. As mentioned MMRE value is calculated utilizing BRE. We can observe from the figure that BRE values of RBNN and BPNN are massive. Even if the differences between the predicted value and the real value are small, the division with small value will amplify it. We can observe that COCOMO MMRE result is better than BPNN and RBNN. However, DANN obtained the smallest value.

Figure IX shows MAE of these algorithms. MAE depends on the mean value of all obtained outputs without any relation to the training values. If the data have vast variance, MAE will increase. We can observe that COCOMO has the highest variance of all algorithms and DANN obtained the lowest results. This means that the variance is small in the predicted outputs of DANN algorithm.

After testing the proposed model using randomly selected testing dataset, the Receiver Operating Characteristic (ROC) curve was measured, after calculating the confusion-matrix for the obtained results, which was not shown in this paper, as it is summarized by the ROC curve in Figure X. However, this curve shows the trade-off between specificity and sensitivity, which looks represents more accurate results as it is close from the sensitivity access as well as the top border of the ROC space. This result validates the hypothesis of this work as the proposed model, which is constituted of Feedforward ANN trained by DA showed more accurate results and contribute to the field under study.

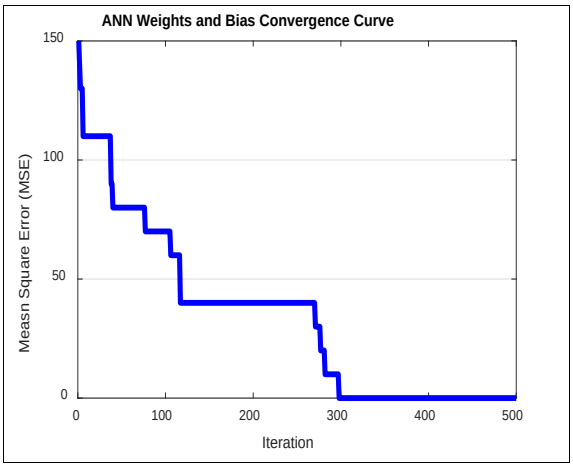

Fig. VI AD Convergence Curve

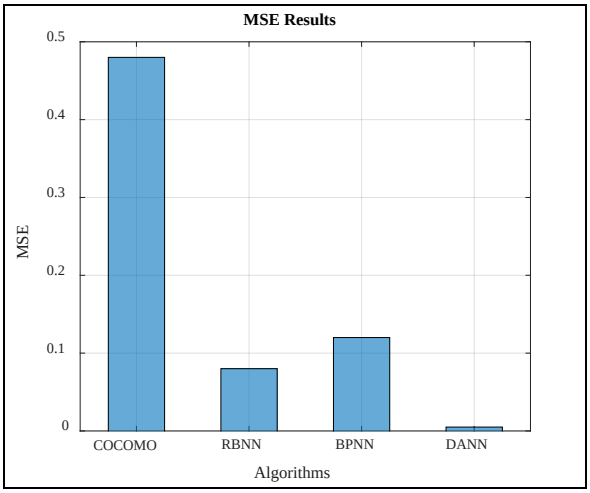

Fig. VII Mean Square Error Results

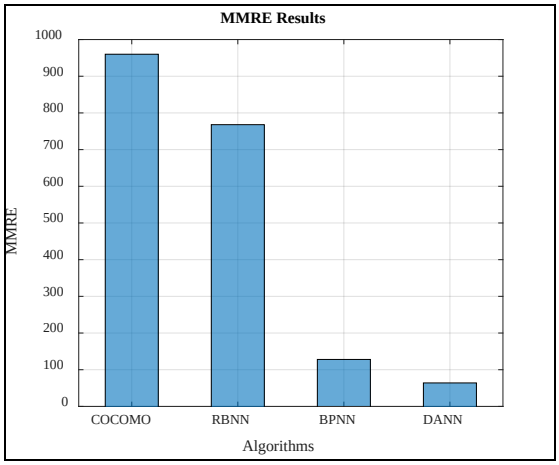

Fig. VIII Mean Magnitude of Relative Error Results

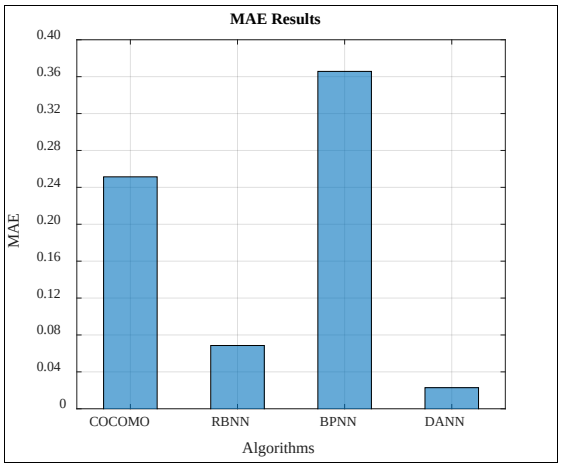

Fig. IX Mean Absolute Error Results

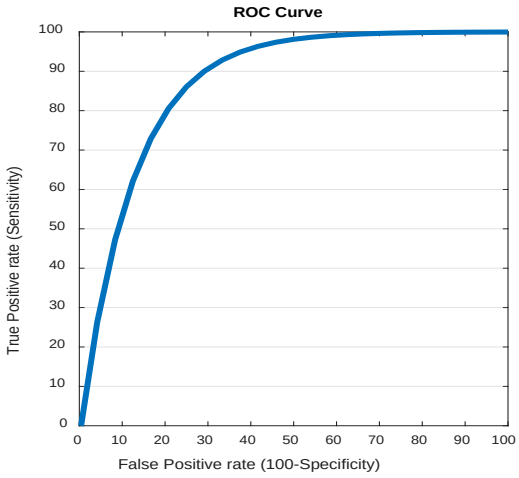

Fig. X Proposed Model ROC Curve

7. Conclusions

In this work, aiming to provide accurate software project efforts estimation, ANN was used with DA training algorithm. Various algorithms have been implemented and evaluated, such as; BPNN, RNN, ABC, GA, and DA, using NASA93 datasets. Software effort estimation, ANN, DA, and NASA93 are explained briefly. Moreover, the accuracy of the proposed approach, the used models, and their data fitness has been tested.

DA found to provide the best performance in terms of MSE, thus it was used as training algorithm for the ANN, which provided enhanced results for software effort estimation. By evaluating the algorithms using NASA93 dataset and calculating the MSE, RBNN, MMRE and MAE values, it can be concluded from the MSE calculations that COCOMO has the highest error values. Moreover, it can be concluded that RBNN outperforms BPNN since RBNN attempted to memorized most of training data. For the DA, it attempted to minimize the cost function to reach zero. In terms of MMRE, the BRE values of RBNN and BPNN are massive regardless of the difference between the predicted and real values. However, in this situation, COCOMO MMRE outperformed BPNN and RBNN. However, DANN showed the best value. On the other hand, COCOMO showed the highest variance of other algorithms, in comparison with DANN, which showed the lowest results.

From the obtained results of this study, it can be concluded that the proposed ANN-DA model provides more accurate software efforts' estimation, in comparison with other models. However, it is an important step to apply this proposed model on a different dataset that contains a larger number of projects, in order to study the stability of this model.

References

[1] Demirors, Onur, and Cigdem Gencel. "A comparison of size estimation techniques applied early in the life cycle." Lecture notes in computer science(2004): 184-194.

[2] Jones, Capers Capers. Software quality: Analysis and guidelines for success. Thomson Learning, 1997.

[3] Kan, Stephen H. Metrics, and models in software quality engineering. Addison-Wesley Longman Publishing Co., Inc., 2002.

[4] Moløkken, K., & Jørgensen, M. (2003, September). A review of software surveys on software effort estimation. In Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium on (pp. 223-230). IEEE.

[5] Kalichanin-Balich, I., & Lopez-Martin, C. (2010, May). Applying a feedforward neural network for predicting software development effort of short-scale projects. In Software Engineering Research, Management and Applications (SERA), 2010 Eighth ACIS International Conference on (pp. 269-275). IEEE.

[6] Molokken-Ostvold, K., & Jorgensen, M. (2005). A comparison of software project overruns-flexible versus sequential development models. Software Engineering, IEEE Transactions on, 31(9), 754-766.

[7] B.W. Boehm, Software Engineering Economics, Prentice-Hall, Englewood Cli4s, NJ, 1981.

[8] Putnam, Lawrence H. "A general empirical solution to the macro software sizing and estimating problem." IEEE transactions on Software Engineering 4 (1978): 345-361.

[9] Putnam. L.. and Fitzsimmons, A. Estimating software costs. Datamation 25, lo-12 (Sept.-Nov. 1979).

[10] Walston, Claude E., and Charles P. Felix. "A method of programming measurement and estimation." IBM Systems Journal 16.1 (1977): 54-73.

[11] Bailey, John W., and Victor R. Basili. "A meta-model for software development resource expenditures." Proceedings of the 5th international conference on Software engineering. IEEE Press, 1981.

[12] Herd, J. R., et al. "Software cost estimation study-study results." Doty Associates, Inc., Rockville, MD, Final Tech. Rep. RADC-TR-77-220 1 (1977): 40-54.
[13] Xia, W., Ho, D., & Capretz, L. F. (2015). A neuro-fuzzy model for function point calibration. arXiv preprint arXiv:1507.06934.
[14] Bardsiri, Amid Khatibi, and Seyyed Mohsen Hashemi. "Software effort estimation: A survey of well-known approaches." International Journal of Computer Science Engineering (IJCSE) 3.1 (2014): 46-50.
[15] Du, Wei Lin, Danny Ho, and Luiz Fernando Capretz. "Improving software effort estimation using neuro-fuzzy model with SEER-SEM." arXiv preprint arXiv:1507.06917 (2015).
[16] Barcelos Tronto, Iris Fabiana, José Demísio Simões da Silva, and Nilson Sant'Anna. "An investigation of artificial neural networks based prediction systems in software project management." Journal of Systems and Software81.3 (2008): 356-367.
[17] Dawson, C. W. (1996). A neural network approach to software project effort estimation. International conference on applications of artificial intelligence in engineering.
[18] Mirjalili, S. (2015). Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems.Neural Computing and Applications, 1-21.
[19] Murphy-Hill, T. M. (n.d.). Nasa93 Dataset. Retrieved September 18, 2017, from http://openscience.us/repo/effort/cocomo/nasa93.html
[20] Farr, L. (1964). Factors that affect the cost of computer programming. Santa Monica, CA: The Corporation.
[21] Nelson, E. A. (1967). Management handbook for the estimation of computer programming costs. Santa Monica, CA: System Development Corp.
[22] Jorgensen, M., & Shepperd, M. (2007). A Systematic Review of Software Development Cost Estimation Studies. IEEE Transactions on Software Engineering,33(1), 33-53. doi:10.1109/tse.2007.256943
[23] Cohn, M. (2006). Agile estimating and planning. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference.
[24] Moløkken-Østvold, Kjetil, et al. "A survey on software estimation in the Norwegian industry." Software Metrics, 2004. Proceedings. 10th International Symposium on. IEEE, 2004.
[25] Jørgensen, Magne. "A review of studies on expert estimation of software development effort." Journal of Systems and Software 70.1 (2004): 37-60.
[26] Kitchenham, Barbara, et al. "An empirical study of maintenance and development estimation accuracy." Journal of systems and software 64.1 (2002): 57-77.
[27] Tausworthe, Robert C. "The work breakdown structure in software project management." Journal of Systems and Software 1 (1979): 181-186.
[28] Selby, R. W. (2007). Software engineering: Barry W. Boehm's lifetime contributions to software development, management, and research. Hoboken, NJ: Wiley-Interscience.
[29] Putnam, L. H. (2013). Five core metrics: the intelligence behind successful software management.
[30] Putnam, L. (1978). A General Empirical Solution to the Macro Software Sizing and Estimating Problem. IEEE Transactions on Software Engineering,SE-4(4), 345-361. doi:10.1109/tse.1978.231521
[31] Applying SEER-SEM to Estimation Processes. (2006). Software Sizing, Estimation, and Risk Management, 397-498. doi:10.1201/9781420013122.ch11
[32] G. Finnie, G. Wittig, and J.-M. Desharnais, "A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models," J. Systems and Software, vol. 39, pp. 281-289, 1997.
[33] P. Sentas, L. Angelis, I. Stamelos, and G. Bleris, "Software Productivity and Effort Prediction with Ordinal Regression," Information and Software Technology, vol. 47, pp. 17-29, 2005.
[34] L. Briand, K.E. Emam, D. Surmann, and I. Wieczorek, "An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques," Proc. 21st Int'l Conf. Software Eng., pp. 313-323, May 1999.
[35] L. Briand, T. Langley, and I. Wieczorek, "A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques," Proc. 22nd Int'l Conf. Software Eng., pp. 377-386, June 2000.
[36] J. Li, G. Ruhe, A. Ak-Emran, and M. Richter, "A Flexible Method for Software Effort Estimation by Analogy," Empirical Software Eng., vol. 12, pp. 65-107, 2007.
[37] Idri, Ali, Fatima azzahra Amazal, and Alain Abran. "Analogy-based software development effort estimation: A systematic mapping and review." Information and Software Technology 58 (2015): 206-230.
[38] Tsunoda, M., Monden, A., Keung, J., & Matsumoto, K. (2012). Incorporating Expert Judgment into Regression Models of Software Effort Estimation. 2012 19th Asia-Pacific Software Engineering Conference. doi:10.1109/apsec.2012.58
[39] Blattberg, R. C., & Hoch, S. J. (1990). Database Models and Managerial Intuition: 50% Model 50% Manager. Management Science,36(8), 887-899. doi:10.1287/mnsc.36.8.887
[40] Du, W. L., Ho, D., & Capretz, L. F. (2015). Improving software effort estimation using neuro-fuzzy model with SEER-SEM. arXiv preprint arXiv:1507.06917.
[41] Briand, Lionel C., et al. "An assessment and comparison of common software cost estimation modeling techniques." Software Engineering, 1999. Proceedings of the 1999 International Conference on. IEEE, 1999.
[42] Buscema, M. (1998). Back propagation neural networks. Substance use & misuse, 33(2), 233-270.
[43] Whitley, D. (2014). An executable model of a simple genetic algorithm. Foundations of genetic algorithms, 2(1519), 45-62.
[44] Farr, Leonard, and Henry J. Zagorski. FACTORS THAT AFFECT THE COST OF COMPUTER PROGRAMMING. VOLUME II. A QUANTITATIVE ANALYSIS. SYSTEM DEVELOPMENT CORP SANTA MONICA CA, 1964.
[45] Nelson, Edward Axel. Management handbook for the estimation of computer programming costs. No. TM-3225/000/01. SYSTEM DEVELOPMENT CORP SANTA MONICA CA, 1967.
[46] Jensen, Randall W., L. H. Putnam, and William Roetzheim. "Software estimating models: three viewpoints." Software Engineering Technology 19.2 (2006): 23-29.

[47] Huang, Xishi, et al. "Improving the COCOMO model using a neuro-fuzzy approach." Applied Soft Computing 7.1 (2007): 29-40.
[48] Lee, Jooon-kil, and Ki-Tae Kwon. "Software cost estimation using SVR based on immune algorithm." Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, 2009. SNPD'09. 10th ACIS International Conference on. IEEE, 2009.
[49] Foss, Tron, et al. "A simulation study of the model evaluation criterion MMRE." IEEE Transactions on Software Engineering 29.11 (2003): 985-995.
[50] Azzeh, Mohammad, Daniel Neagu, and Peter I. Cowling. "Analogy-based software effort estimation using Fuzzy numbers." Journal of Systems and Software 84.2 (2011): 270-284.
[51] Reddy, P. V. G. D., et al. "Software effort estimation using radial basis and generalized regression neural networks." arXiv preprint arXiv:1005.4021 (2010).
[52] Prasad Reddy, P. V. G. D., K. R. Sudha, and P. Rama Sree. "Ramesh SNSVSC, Fuzzy Based Approach for Predicting Software Development Effort." International Journal of Software Engineering 1.1 (2010): 1-11.
[53] Keung, Jacky. "Software Development Cost Estimation using Analogy: A Review, Australian Software Engineering Conference." (2009).
[54] Mittal, Harish, and Pradeep Bhatia. "Optimization criteria for effort estimation using fuzzy technique." CLEI Electronic Journal 10.1 (2007): 1-11.
[55] Jodpimai, Pichai, Peraphon Sophatsathit, and Chidchanok Lursinsap. "Estimating software effort with minimum features using neural functional approximation." Computational Science and Its Applications (ICCSA), 2010 International Conference on. IEEE, 2010.
[56] Malathi, S., and S. Sridhar. "Analysis of size metrics and effort performance criterion in software cost estimation." Indian Journal of Computer Science and Engineering 3.1 (2012): 24-31.
[57] Graupe, Daniel. Principles of artificial neural networks. Vol. 7. World Scientific, 2013.
[58] Bebis, George, and Michael Georgiopoulos. "Feed-forward neural networks." IEEE Potentials 13.4 (1994): 27-31.
[59] Jairus Hihn, JPL, NASA, Manager SQIP Measurement & Benchmarking Element. Phone (818) 354-1248 (Jairus.M.Hihn@jpl.nasa.gov)
[60] B. Kitchenham. (1998). A Procedure for Analyzing Unbalanced Datasets, IEEE Transactions on Software Engineering, 24(4):278-301
[61] Pant, Bhushan, Hardwari Lal Mandoria, and Ashok Kumar. "Wavy Software Process Model: A Modern Approach." i-Manager's Journal on Software Engineering 9.1 (2014): 27.
[62] Mukhopadhyay, Tridas, Steven S. Vicinanza, and Michael J. Prietula. "Examining the feasibility of a case-based reasoning model for software effort estimation." MIS quarterly (1992): 155-171.
[63] Zhong, Shi, Taghi M. Khoshgoftaar, and Naeem Seliya. "Unsupervised Learning for Expert-Based Software Quality Estimation." HASE. 2004.
[64] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010, September). Recurrent neural network based language model. In INTERSPEECH (Vol. 2, p. 3).
[65] Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of global optimization, 39(3), 459-471.

Annex A

Table I: NASA93 Descriptive Attributes

Attribute Number	Notation	Description	Contents
1	recordnumber	Unique ID	real #
2	projectname	project name	de,erb,gal,X,hst,slp,spl,Y
3	cat2	Cagetory of application	avionics, application_ground, avionicsmonitoring, batchdataprocessing, communications, datacapture, launchprocessing, missionplanning, monitor_control, operatingsystem, realdataprocessing, science, simulation, utility
4	Forg	flight or ground system	f,g
5	Center	which nasa center?	1,2,3,4,5,6
6	year	year of development	real #
7	Mode	development mode	embedded,organic,semidetached
8	rely	cocomo attributes	vl,l,n,h,vh,xh

9	Data	cocomo attributes	vl,l,n,h,vh,xh
10	Cplx	cocomo attributes	vl,l,n,h,vh,xh
11	Time	cocomo attributes	vl,l,n,h,vh,xh
12	Stor	cocomo attributes	vl,l,n,h,vh,xh
13	Virt	cocomo attributes	vl,l,n,h,vh,xh
14	turn	cocomo attributes	vl,l,n,h,vh,xh
15	acap	cocomo attributes	vl,l,n,h,vh,xh
16	Aexp	cocomo attributes	vl,l,n,h,vh,xh
17	Pcap	cocomo attributes	vl,l,n,h,vh,xh
18	Vexp	cocomo attributes	vl,l,n,h,vh,xh
19	Lexp	cocomo attributes	vl,l,n,h,vh,xh
20	Modp	cocomo attributes	vl,l,n,h,vh,xh
21	Tool	cocomo attributes	vl,l,n,h,vh,xh
22	Sced	cocomo attributes	vl,l,n,h,vh,xh
23	equivphyskloc	equivalent physical 1000 lines of source code	real #
24	act_effort	development effort in months (one month =152 hours and includes development and management hours)	real #