International Journal of Information Technology Project Management
Volume 11 • Issue 2 • April-June 2020

The Role of Neural Networks and Metaheuristics in Agile Software Development Effort Estimation

Anupama Kaushik, Maharaja Surajmal Institute of Technology, Delhi, India; Indira Gandhi Delhi Technical University for Women, Delhi, India

Devendra Kumar Tayal, Indira Gandhi Delhi Technical University for Women, Delhi, India

Kalpana Yadav, Indira Gandhi Delhi Technical University for Women, Delhi, India

ABSTRACT

In any software development, accurate estimation of resources is one of the crucial tasks that leads to a successful project development. A lot of work has been done in estimation of effort in traditional software development. But, work on estimation of effort for agile software development is very scant. This paper provides an effort estimation technique for agile software development using artificial neural networks (ANN) and a metaheuristic technique. The artificial neural networks used are radial basis function neural network (RBFN) and functional link artificial neural network (FLANN). The metaheuristic technique used is whale optimization algorithm (WOA), which is a nature-inspired metaheuristic technique. The proposed techniques FLANN-WOA and RBFN-WOA are evaluated on three agile datasets, and it is found that these neural network models performed extremely well with the metaheuristic technique used. This is further empirically validated using non-parametric statistical tests.

KEYWORDS

Friedman Test, Functional Link Artificial Neural Network, Non-Parametric Tests, Radial Basis Function Neural Network, Statistical Tests, Whale Optimization Algorithm, Wilcoxon Matched Pair Test

INTRODUCTION

In software development firms, two development approaches are present, the traditional software development approach and agile software development approach. In traditional software development approach requirements are well understood and there are predefined stages of development. This type of development is driven by process and tool. The requirements once decided is difficult to change and the customer's involvement is limited in this development. Here, the iterations are longer and the working software is not quickly available.

In agile software development approach customers can do modifications until late in project's life. They are people and collaboration driven. So, there is a continuous involvement of customers'. This development approach is more user friendly and follows incremental and iterative development. The iterations are shorter here and working software is available quickly. Now-a-days, software development firms are moving towards adopting agile methodologies (Dingsøyr, Nerur, Balijepally, & BredeMoe, 2012; Papadopoulos, 2015).

DOI: 10.4018/IJITPM.2020040104

Copyright © 2020, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

The success of a software project mainly depends upon the accuracy of estimation of its resources like effort, schedule etc. There are many effort estimation studies for traditional software development present in literature (Nguyen, Boehm &LiGuo, 2019; Venkataiah, Mohanty, Pahariya & Nagaratna, 2017; Kaushik, Verma, Singh & Chabbra, 2017; Kaushik, Tayal, Yadav & Kaur, 2016). These studies are based on algorithmic and non-algorithmic approach. The COCOMO model (Boehm, 1994) commonly used for effort estimation in traditional software development uses algorithmic approach. The non-algorithmic approach uses various soft computing techniques like fuzzy logic, neural network, genetic algorithms etc.

In agile software development approach, not much of work has been done in estimation of resources for the projects but a lot of work is going on for developing agile methodologies (Curiel, Jacobo, Alfaro, Zepeda & Delgado, 2018; Tolfo, Wazlawick, Ferreira & Forcellini, 2018; Perkusich, Gorgônio, Almeida, & Perkusich, 2017).This work is dedicated towards estimation of effort for agile projects using story point approach which finds the effort of a project in terms of story points. In the past, few researchers have applied various machine learning techniques for effort estimation using story point approach (Satapathy, Panda & Rath, 2014; Panda, Satapathy & Rath, 2015; Satapathy & Rath, 2017).

The current work integrates artificial neural networks (ANN) with a metaheuristic technique for effort estimation of projects following agile methodologies. The ANN used are RBFN and FLANN and, the metaheuristic technique used is whale optimization algorithm (WOA).

The ANN models incorporated have no relationship with each other and they are evaluated independently. These models are used as they have their own advantages. The major advantages of FLANN are: it has less computational complexity, faster convergence and handles the non-linear data (Mishra & Dehuri, 2007) ; and the major advantages of RBFN are: its easy design, good generalization, strong tolerance to input noise and has faster online learning ability (Yu, Xie, Paszczyñski & Wilamowski, 2011). These models are also chosen as no earlier study exists based on these models for agile environment.

Now-a-days metaheuristic techniques have come up. These are the optimization techniques which mimics the biological or physical phenomenon to solve various engineering problems. They can even find the solutions for the problems with very less and incomplete information. Many new metaheuristic algorithms are developed and many researchers (Kaushik, Tayal, Yadav, & Kaur, 2016; Kaushik, Verma, Singh, & Chhabra, 2017; Benala & Mall, 2018) have used these techniques in estimations for traditional software development environment, but according to the best of our knowledge these techniques have not been explored for resource estimations in agile software development. So the current work is an attempt to introduce metaheuristic techniques in effort estimation for agile software development environment. Here, whale optimization algorithm (WOA) is the metaheuristic approach used with ANN due to its striking features which are at par than other optimization methods (Mirjalili & Lewis, 2016). The study also examines the effort estimation accuracy of the selected ANN models by incorporating WOA.

The paper consists of literature review section which discusses the work done by the researchers concerning software effort estimation in agile environment, followed by the section on background concepts used in the current work. After that, innovation in the paper is discussed in research contribution section followed by simulation and results section. Finally, conclusion of the study is presented.

LITERATURE SURVEY

Few of the effort estimation studies related to agile software development are discussed below.

Lang, Conboy and Keaveney (2011) discussed four case studies to explain the cost estimation process, causes of inaccurate estimates and steps to improve the process in agile software development. They also discussed, how agile handles the classical problems which affects the cost estimation in

comparison to traditional information system development. Their study recommended three points for agile projects and they are: estimation models are not necessarily required, documentation of past project data and experience, and fixed price budgets.

Coelho and Basu (2012) discussed effort estimation in agile software development in detail using story points and directed further scope of improvement. They also provided various size estimation techniques used traditionally, discussed user stories prioritization and delivery date estimations. They concluded that some hidden factors were also responsible for the delay in software project deployment.

Ziauddin, Tipu and Zia (2012) provided a software effort estimation model based upon user stories. This model included all the characteristics of agile software development. The model used various equations and demonstrated its effectiveness using data collected from 21 software projects. They provided guidelines to measure user story size and complexity in the scale of 1-5. They also discussed various factors affecting the projects velocity and provided a way to handle uncertainty in calculation of completion time.

Choudhari and Suman (2012) proposed an effort estimation model for software maintenance which was validated using various types of maintenance projects. This was based on story points to calculate the volume of maintenance and value adjustment factors. The model was developed in order to help the project managers to estimate the software maintenance for agile and extreme programming environment.

Hussain, Kosseim and Ormandjieva (2013) used COSMIC standard to approximate the functional size in early effort estimation in agile processes. Their methodology used supervised text mining approach where COSMIC function size is calculated from the textual requirements and it was also able to recognize the striking features of the functional processes in order to determine their size.

Popli and Chauhan (2014) provided an estimation method for projects using agile methodologies. The algorithm efficiently proposed effort, cost and duration for small and medium sized agile projects. They also discussed already existing methods of agile estimation along with their limitations. Their estimation model was based on story point approach and a case study was discussed with the proposed model.

Satapathy, Panda and Rath (2014) used Support Vector Regression (SVR) for effort estimation method using story point approach. They optimized the result of story point approach using SVR kernel methods for better prediction accuracy. The kernel methods used by them were linear, polynomial, Radial Basis Function (RBF) and Sigmoid kernels. They concluded that RBF kernel based SVR was at par than the rest of the three kernel methods for effort estimation.

Panda, Satapathy and Rath (2015) used different types of Neural Networks to increase the accuracy estimation of agile projects using story point approach. The parameters used by them in estimation were story points, velocity and actual effort. They used dataset of 21 projects to validate their approach and used Mean Square Error (MSE), squared correlation coefficient (R^2), Mean Magnitude of Relative Error (MMRE) and Prediction Accuracy (PRED) as the evaluation criteria.

Garg and Gupta (2015) proposed a cost estimation method where they identified the key attributes that have maximum impact on development cost using principal component analysis. In order to satisfy the criteria imposed by agile manifesto they further used constraint solving approach. Their model also worked in absence of historical data and expert opinion. The model provided lower MMRE value in comparison to planning poker and mapped the agile manifesto.

Tanveer (2016) provided a hybrid methodology with tool support for effort estimation which handled change impact analysis, expert judgement and software visualization. They proposed that agile projects were usually estimated using expert judgement which was not accurate and reliable. So a new technique was proposed by them to improve the estimation process in agile environment.

Raslan, Darwish and Hefny (2015) proposed an effort estimation methodology using fuzzy logic. They used story point approach and trapezoidal membership functions to represent the input parameters which consisted of Story Points (SP), Implementation Level Factor (ILF), FRiction factors

(FR), and Dynamic Forces (DF). They designed the proposed fuzzy inference system in MATLAB and calculated the effort.

Dragicevic, Celar and Turic (2017) proposed an effort prediction model for agile projects using bayesian network. They validated their technique on the agile project data of a single company. The model was further assessed using various statistical parameters. The model could be used with any agile methodology in the initial planning phase. It basically provided task effort estimation based on various parameters like working hours, requirements complexity, developer's skills etc.

Salmanoglu, Hacaloglu, and Demirors (2017) evaluated three case studies on agile projects to compare effort estimation using COSMIC and story points. They found that effort estimation models developed using COSMIC size performed better on all the three case studies. They also concluded that COSMIC provided better productivity for the data which was less dispersed than story points. They also shared their datasets for future research.

Satapathy and Rath (2017) improved the prediction accuracy of story point approach using various machine learning techniques and compared the proposed techniques with the existing techniques in the literature. The machine learning techniques used were decision tree (DT), stochastic gradient boosting (SGB) and Random forest (RF). They developed this estimation model considering scrum projects. The technique SGB gave the best results than the rest of the two techniques on the examined dataset.

Tanveer, Vollmer and Braun (2018) proposed an estimation technique for agile development teams. They designed a hybrid method where the impact of change is analysed and incorporated for effort estimation. They also proposed an estimation technique using boosted trees and discussed their methodology using a case study on a German software company. Their method was more effective and accurate than the expert based methods.

Usman, Britto, Damm and Borstler (2018) identified and analysed the effort estimation process in large scale distributed agile projects. They also identified various factors effecting the effort of such projects. They devised a two stage effort estimation and restimation process to improve the accuracy of estimation in such projects. They concluded that effort overruns in large scale distributed agile projects could be limited by considering various factors like requirements size, its priorities, maturity and team distribution of agile projects.

Martínez, Noriega, Ramírez, Licea and Jiménez (2018) proposed a Bayesian network model to handle the complexity and importance associated with the user stories used for estimation of scrum projects. Their model could replace the traditional planning poker used in estimations and could be used by an inexperienced or a new developer. They used estimations provided by students and professionals in order to validate the model. They found professionals' estimation more correlated to the proposed model than the students' estimation as the proposed model included various factors considered in real world application.

Mensah, Keung, Bosu, and Bennin (2018) designed a duplex output model for software effort estimation. The first output here was software effort and the second output was the classification of effort in order to identify the level of effort. The study was motivated by conclusion instability problem faced by effort estimation models. They did comparison of six different regression-based techniques which included the state-of-the-art baseline model (ATLM) and ElasticNet regression to solve conclusion instability. They found ElasticNet regression providing superior accuracy than the rest of the techniques.

Bilgayian, Das and Mishra (2019) solved effort estimation problem for agile projects by using back propagation neural networks and Elman neural networks. They validated the model on Zia dataset using standard evaluation criteria used in software effort estimation. They simulated their model using MATLAB and found feed forward back propagation neural networks performed better than Elman neural networks and cascade correlation network present in the literature.

Tanveer, Vollmer, Braun and Ali (2019) proposed a hybrid model based on gradient boosted trees (GBT) which estimates the effort including the impact of changes on the existing system. They

evaluated their model in a German software company. The results showed that their model provided more accurate estimates than expert based and model based techniques.

The given literature review reveals that there are many effort estimation studies present but the studies using metaheuristic techniques are still unavailable. All the above studies have their strengths and weaknesses. As the plethora of new techniques are introduced every year, there is always a scope of improvement which exists.

BACKGROUND CONCEPTS USED

In this section the background concepts used are reviewed.

Story Point Approach

In agile environment story point is the most commonly used unit of measure followed for effort estimation. In story point approach the user story is the term for requirements (Cohn, 2005). The effort of a particular user story is determined by its size and complexity and based upon that a story point value is assigned (Ziauddin, Tipu, & Zia, 2012). The commonly used assignment criteria for story points are t-shirt sizing, Fibonacci sequence or simply small vs. needs-to-be-split. Story point is the amount of effort completed in a unit time. For agile projects the unit of Effort is Story Point (SP) (Ziauddin, Tipu, & Zia, 2012). Agile velocity is the amount of work done by a project team in a single sprint. It is calculated based upon the effort and sprint time. Sprint time is the time period during which a particular work is completed by the project team and the work is available for review.

In all the effort estimation studies based upon story point approach and using machine learning techniques and neural networks, the input parameters are story points and velocity, and the output is the effort of a project (Popli & Chauhan, 2014; Satapathy, Panda & Rath, 2014; Panda, Satapathy & Rath, 2015; Satapathy &Rath, 2017; Bilgayian, Das & Mishra, 2019).

Functional Link Artificial Neural Networks (FLANN)

FLANN consists of three layers i.e. first, middle and the last layer. As the name goes by, FLANN uses various functions like Power Series polynomials, Boubaker polynomials, Fibnocci polynomials, Chebyshev polynomials and Legendre polynomials etc. to expand the input pattern supplied to it. The input data is supplied to the first layer, which navigates it to middle layer where the polynomial function is applied which converts the n-dimensional input data to K dimensions where $n < K$. These data values are represented in the matrix form and are combined linearly after multiplying with the weight matrix. This provides a scalar quantity which forms the required output. The current work uses Chebyshev polynomial function which is defined as:

Chebyshev polynomials: $F_0\left(z\right) = 1$

$$F_1\left(z\right) = z$$

$$F_2\left(z\right) = 2z^2 - 1 \tag{1}$$

$$F_3\left(z\right) = 4z^3 - 3z \; F_4\left(z\right) = 8z^3 - 8z^2 + 1$$

These polynomials are further generated as:

$$F_n\left(z\right) = 2zF_{n-1}\left(z\right) - F_{n-2}\left(z\right), \; n \geq 2 \tag{2}$$

Chebyshev polynomials were also used by early researchers in software cost estimations and are considered as a basis function in this field (Benala, Korada, Mall, & Dehuri, 2013).

Radial Basis Function Network (RBFN)

RBFN is a three layer neural network consisting of the first, middle and the last layer (Idri, Zahi, Mendes, & Zakrani, 2007; Kaushik, Soni, & Soni, 2013). The input supplied to the network is first clustered and its output is provided to the middle layer neurons where the Gaussian radial basis function as given by (3) is applied.

$$f\left(y\right) = e^{\left(-\frac{\left\|y-v_i\right\|^2}{\sigma_i^2}\right)} \tag{3}$$

Here, y is the input, v_i and σ_i are the center and the wideness of the ith neuron of the middle layer respectively. ||.|| denotes the Euclidean distance. Intutionistic Fuzzy C-Means (IFCM) clustering algorithm (Bezdek, 1981; Kaur, Soni, & Gossain, 2012; Chaira, 2011; Atanassov, 1983) is used to determine v_i and σ_i is calculated using p-nearest neighbour heuristic (Moody, & Darken, 1989).

In this study, the weights of ANNs are obtained using whale optimization algorithm and updated using delta rule. The delta rule is a learning rule for updating the weights of the neurons using gradient descent learning which minimizes the error between the target value and the estimated value.

Whale Optimization Algorithm (WOA)

Mirjalili and Lewis (2016) framed this algorithm on hump back whales for optimization which modelled the spiral bubble-net feeding behaviour of these whales. The basic phases are:

Encircling Prey

Hump back whales encircles their prey after recognizing them. Here, the prey which is targeted, is assumed as the current best candidate solution and a best explore agent is defined. This is shown by:

$$\vec{E} = \left|\overrightarrow{D.K_p}\left(l\right) - 2\vec{K}\left(l\right)\right| \tag{4}$$

$$\overrightarrow{K}\left(l+1\right) = \overrightarrow{K_p}\left(l\right) - \vec{B}\,.\,\vec{E} \tag{5}$$

Where, $\vec{B}$ and $\vec{D}$ are the coefficient vectors, l is the current iteration, K_p is the position vector of the predator, $\vec{K}$ is the position vector of the whale. $\vec{B}$ and $\vec{D}$ are calculated as follows:

$$\vec{B} = 2\vec{b_1} \cdot \vec{s_1} \cdot \vec{b_1} \tag{6}$$

$$\vec{D} = 2.\vec{s_2} \tag{7}$$

Where, $\vec{b_1}$ is linearly reduced from 2 to 0 over the course of iterations and $\vec{s_1}$, $\vec{s_2}$ are the random vectors in [0, 1].

Bubble Net Attacking Method

In this method bubbles are formed by the whales to attack their prey either along a circle or 9 shaped path. It is mathematically modelled using two approaches:

Shrinking Encircling Mechanism

It is obtained by decreasing $\vec{b}_1$ in (6). It also affects the range of $\vec{B}$ which is in [-1, 1]. It finds the new location of an explore agent.

Spiral Updating Position

It finds the distance between the whale and prey and mimics the spiral motion of whales as shown in (8):

$$\overrightarrow{K}\left(l+1\right)=\overrightarrow{E'}.e^{ol}.\cos\left(2\pi l\right)+\overrightarrow{K_p}\left(l\right) \tag{8}$$

Where, $\overrightarrow{E'}=\left|\overrightarrow{K_p}\left(l\right)-\vec{K}\left(l\right)\right|$ and gives the distance of the *i*th whale to the predator, *o* is constant that represents the shape of the logarithmic spiral, *l* is a random number in [-1, 1].

Search for Prey

The $\vec{B}$ vector is varied between [-1, 1], to allow these hump back whales to randomly search for their predator according to position of each other. WOA performs global search as the location of the explore agent is upgraded according to randomly chosen explore agent which is given as:

$$\vec{E}=\left|\vec{D}.\overrightarrow{K_{rand}}-\vec{K}\right| \tag{9}$$

$$\overrightarrow{K}\left(t+1\right)=\overrightarrow{K_{rand}}\ -\vec{B}.\vec{E} \tag{10}$$

Where, $\overrightarrow{K_{rand}}$ is a random position vector (a random whale) which is chosen from the current population.

This algorithm was tested on 29 mathematical optimization functions by the authors Mirjalili and Lewis (2016) in order to test its efficiency and it gave the best results for all the problems. The function used in algorithm for the current work is given in (11) (Mirjalili & Lewis, 2016):

$$F5\left(y\right)=\sum_{i=1}^{n-1}\left[100\left(y_{i+1}-y_i^2\right)^2+\left(y_i-1\right)^2\right] \tag{11}$$

This function is used as a cost function in WOA to create the optimized weights of range [0, 1]. There are various reasons for choosing WOA over other metaheuristic algorithms. WOA exhibits high exploration, exploitation, local optima avoidance and convergence speed. The whales randomly move around each other initially and their positions get updated. This represents the high exploration behaviour of WOA as given in (10).In later iterations they exhibit high exploitation and convergence. As these two phenomenon are separately performed by whales, this leads to high local optima avoidance (Mirjalili & Lewis, 2016).

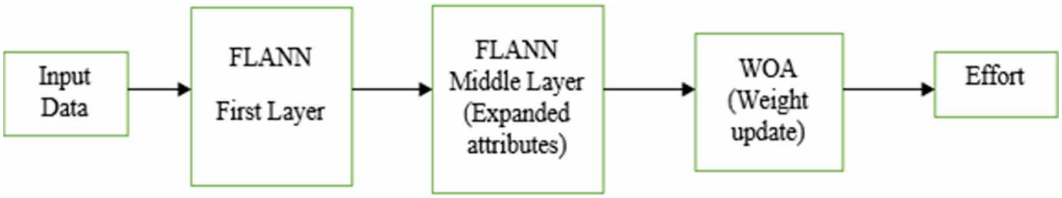
RESEARCH CONTRIBUTION

Effort estimation is an important issue faced by the project managers. A good estimation model will always help the managers to estimate accurately which will lead the project to success. The work presented in this paper contributes to effort estimation of agile projects following the story point approach. The novelty here is the integration of a metaheuristic technique and neural networks. There are limited studies present in the literature for effort estimation of agile projects based on machine learning and neural networks (Popli & Chauhan, 2014; Satapathy, Panda & Rath, 2014; Panda, Satapathy & Rath, 2015; Satapathy &Rath, 2017; Bilgayian, Das & Mishra, 2019) and none of them have used FLANN and RBFN for agile effort estimation, also no study has incorporated any metaheuristic technique in their estimation model. The models proposed in the current work are FLANN-WOA and RBFN-WOA described subsequently.

FLANN-WOA

In this framework, first the input data to be trained is supplied as an input to the first layer of the neural network. The data is expanded by FLANN using Chebyshev polynomials as shown in (1) and (2). After applying Chebyshev expansion, the non-linear outputs become the nodes for the middle layer. The estimated effort is then calculated by multiplying these nodes with the weight vector calculated using WOA. The block diagram is represented in Figure 1.

Figure 1. FLANN-WOA block diagram



The procedure of FLANN-WOA is given in Figure 2.

RBFN-WOA

The input neurons of the first layer of RBFN receives the input data and, the gaussian radial basis function as given by (3) is applied to form the middle layer. The nodes of the middle layer is multiplied by the weight values obtained using WOA and the weighted sum is obtained to get the required output. This is demonstrated in Figure 3.

The procedure for RBFN –WOA is given below in Figure 4.

Both the above models are described by taking examples from the datasets in the next section.

SIMULATION AND RESULTS

The proposed approaches FLANN-WOA and RBFN-WOA are evaluated on three datasets. The input datasets are Zia dataset, Company Dataset -1 (CD1) and Company Dataset-2 (CD2).The first dataset

Figure 2. FLANN-WOA

FLANN-WOA
Step 1: Preprocess the input data.
Step 2: Split the input data into training and testing data.
Step 3: For all the training and testing data do
{
Expand the input data using Chebyshev polynomial expansion.
While (the stopping condition not met)
{
For each expanded input pattern
{
- Call WOA () and assign the best position between the second layer and the last layer as the weight.
- Calculate the weighted sum and generate the estimated output at the last layer.
- Generate the difference between the actual output and the calculated output.
- Upgrade the weights between the second and the last layer.

}
}
}

is taken from Zia (Ziauddin, Tipu, & Zia, 2012). This dataset was a collection of 21 agile software projects taken from 6 software houses. The features of the dataset used in the study is given in Table 1.

Here, the first dimension is P.No showing the project number, the second is Effort in developing the project, the third one shows the velocity values for different projects and the fourth gives the actual time in completing a project.

As the agile studies lack in availability of public datasets, the second and third dataset as given in Table 2 and Table 3, is shared by a company in Delhi NCR on special request. The information provided by the datasets is on two different projects only. The Company Dataset -1 (CD1) consists of a project with 23 issues and the Company Dataset-2 (CD2) points to a project with 25 issues. Both the datasets are mapped according to the requirement of the existing study by calculating the estimated effort of various issues using the proposed methodologies.

In Table 2 and 3, the first dimension is the Issue Number of various issues, the second gives the Effort values for these issues, the third and fourth dimension provides the time and story point values for these issues.

Figure 3. RBFN-WOA block diagram

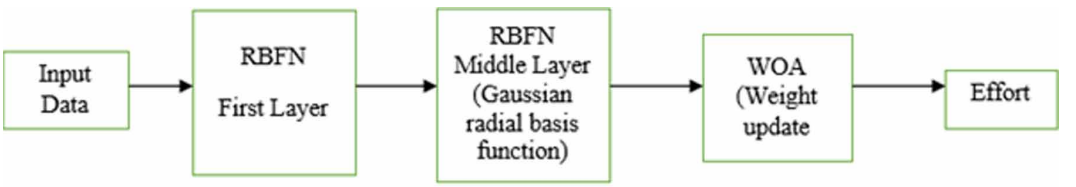


Figure 4. RBFN-WOA

RBFN-WOA
Step 1: Preprocess the input data.
Step 2: Split the input data into training and testing data.
Step 3: For all the training and testing data do
{
Feed the input data to the first layer.
While(the stopping criteria not met)
{
- Generate the middle layer by injecting the Gaussian function on input neurons.
- Call WOA () and assign the best position between the second layer and the last layer as the weight.
- Produce the estimated output at the last layer by calculating the weighted sum.
- Generate the difference between the actual output and the calculated output.
- Upgrade the weights between the second and the last layer.

}
}

Table 1. Zia dataset

P.No	Effort	V	Actual Time *(months)*
1	156	2.7	63
2	202	2.5	92
3	173	3.3	56
4	331	3.8	86
5	124	4.2	32
6	339	3.6	91
7	97	3.4	35
8	257	3	93
9	84	2.4	36
10	211	3.2	62
11	131	3.2	45
12	112	2.9	37
13	101	2.9	32
14	74	2.9	30
15	62	2.9	21
16	289	2.8	112
17	113	2.8	39
18	141	2.8	52
19	213	2.8	80
20	137	2.7	56
21	91	2.7	35

Table 2. Company dataset-1

Issue No.	Effort	Time *(minutes)*	Story Point
1	5600	93600	9
2	4879	15000	8
3	7200	15000	7
4	7654	14000	9
5	5674	22200	6
6	8900	23400	8
7	57600	25800	8
8	8432	18000	7
9	28800	136800	8
10	7896	10800	8
11	9879	25200	5
12	3789	28800	8
13	9756	9000	9
14	8763	14400	8
15	3567	12600	12
16	9875	25200	8
17	8945	21600	8
18	5678	10800	8
19	5699	12780	9
20	4321	11900	8
21	8754	8600	8
22	7542	9300	7
23	9050	7800	8

In order to test the validity of the proposed estimation models leave-one-out (LOO) sampling method, which is a type of K-fold Cross-Validation is used. Here K equals to N, the number of data points in the dataset. This sampling method is employed due to its advantages over N-Way sampling (Kocaguneli & Menzies, 2013).

Overfitting is another issue faced by the estimation models. It is important to recognize and manage this issue otherwise our models will not be able to generalize well and make predictions on the unseen data. An important technique to handle it is the resampling technique which is, LOO in our case.

Before running the procedures of FLANN-WOA and RBFN-WOA, the three datasets are pre-processed. The following three steps are used for data pre-processing:

Step 1: Check whether the data is normally distributed or not. It is found that for the given datasets, the data values were not normally distributed, so logarithmic transformation is done on the data values in order to make it normally distributed.
Step 2: Normalize the input data between [0 1] to discard the scaling effects on various dimensions.
Step 3: Partition the dataset into training and testing datasets.

Table 3. Company Dataset-2

Issue No.	Effort	Time *(minutes)*	Story Point
1	8700	99800	9
2	6487	76000	8
3	7872	25000	9
4	9876	19000	9
5	6675	56200	13
6	3489	23500	9
7	7876	15800	9
8	7832	28000	12
9	52880	128800	10
10	3487	20600	9
11	8987	15200	7
12	7785	12800	8
13	7785	5000	6
14	5769	24200	12
15	4356	22600	11
16	6879	15200	8
17	8843	25600	8
18	8671	20800	8
19	8856	12880	8
20	2352	12600	9
21	7651	6800	9
22	3542	6300	7
23	4550	8800	7
24	6879	4800	6
25	8543	6500	6

Now, both the procedures, FLANN-WOA and RBFN-WOA are called individually on the pre-processed data as discussed in the previous section. The results are recorded and evaluated using the commonly used evaluation criteria used in software cost estimations (Foss, Stensrud, Kitchenham, & Myrtveit, 2003; Stensrud, Foss, Kitchenham, & Myrtveit, 2002) . They are Magnitude of Relative Error (MRE), Mean Magnitude of Relative Error (MMRE), Prediction (t) and Median Magnitude of Relative Error (MdMRE) which are given below:

$$MRE = \frac{\left| Actual\,Data - Estimated\,Data \right|}{Actual\,Data} \tag{12}$$

MMRE is the average of all the MRE values.

$$MMRE = \frac{1}{N}\sum_{X=1}^{N} MRE \tag{13}$$

$$MdMRE = Median\left(MRE\right) \tag{14}$$

$$Pred\left(t\right) = \frac{k}{n} \tag{15}$$

Where, n′ shows the total number of projects and k′ is the number of projects with MRE less than or equal to 0.25 (Rao et al., 2009).

All the above evaluation criteria are the accuracy measurers. They show how close a measurement is to an existing value that has already been known.

Table 4 lists the initialization of parameters done for WOA for both FLANN and RBFN.

Let us consider the dataset CD1 to explain FLANN-WOA. In this dataset for Issue No. 2, the normalized input values for story points and velocity are 0.4285 and 0.3252 respectively. This data is mapped to the input layer of FLANN which expands it using Chebyshev polynomial expansion given in (1). The WOA algorithm is then called with the parameters given in Table 4 which provided the best value as 0.3387 in this case. This value is now assigned as a weight between the hidden layer and the output layer. The weighted sum is now calculated using the weight and hidden layer values to produce the estimated output. It is then tuned by reducing the error as given in Step 3 of Figure 2 to calculate the final estimated output which is 0.02399, whereas the normalized actual effort is 0.02428. So, the estimated effort is very near to the actual effort.

In order to explain RBFN-WOA, let us consider the Zia dataset given in Table 1. In this dataset for P. No. 4, the normalized inputs provided to RBFN-WOA are 0.984 and 0.778. These values are provided as input to the first layer of RBFN. The Gaussian Radial Basis Function given in (3) is then applied to form the middle layer. Now the WOA algorithm is called which provided the best value as 0.2768. This value is used as a weight between the middle layer and the final layer. The estimated output is obtained by finding the weighted sum of middle layer inputs and the weight. This value is fine-tuned as given in Step 3 of Figure 4 to calculate the final estimated effort which is 0.950 and the normalized actual effort is 0.971. Here also, there is not much difference between the estimated and the actual output.

In this manner all the datasets are processed using FLANN-WOA and RBFN-WOA techniques to obtain the estimated effort. They are further assessed using evaluation criteria and the results are recorded in Table 5, 6 and 7.

The output after applying the proposed approaches are demonstrated in Table 5, 6 and 7 respectively. There is a comparison of FLANN to FLANN-WOA and RBFN to RBFN-WOA in order to judge whether the performance of FLANN and RBFN improves upon integration with WOA, as the current work discusses the role of neural networks and metaheuristics in agile software development effort estimation. From the results it is found that the performance of FLANN-WOA and RBFN-WOA are at par than FLANN and RBFN.

In order to have more clarity and representation of facts the performance of FLANN, FLANN-WOA, RBFN and RBFN-WOA is depicted graphically in Figure 5, 6 and 7 for the three datasets respectively. Satapathy and Rath (2017) provided results on Zia dataset for effort estimation using story points on agile software development. The results are from the techniques Decision Tree (DT),

Table 4. Initialization of parameters for WOA

Parameters Initialization
SearchAgents_no 30
Max_iteration 500
o 1
l [-1 1]
q [0 1]
$\overrightarrow{s_1}$ [0 1]
$\overrightarrow{s_2}$ [0 1]

Table 5. Results on Zia Dataset

Approaches Used	MMRE		MdMRE		PRED(0.25)	
	Train Data	Test Data	Train Data	Test Data	Train Data	Test Data
FLANN	.619	.672	.49	.510	.33	.34
FLANN-WOA	.180	.193	.167	.179	.869	.859
RBFN	.548	.512	.243	.232	.571	.587
RBFN-WOA	.196	.201	.172	.174	.876	.882

Table 6. Results on Company Dataset 1

Approaches used	MMRE		MdMRE		PRED(0.25)	
	Train Data	Test Data	Train Data	Test Data	Train Data	Test Data
FLANN	.612	.623	.449	.412	.394	.412
FLANN-WOA	.183	.192	.109	.172	.890	.874
RBFN	.721	.730	.77	.78	.26	.29
RBFN-WOA	.153	.185	.128	.143	.869	.871

Table 7. Results on Company Dataset 2

Approaches used	MMRE		MdMRE		PRED(0.25)	
	Train Data	Test Data	Train Data	Test Data	Train Data	Test Data
FLANN	.425	.487	.2	.294	.642	.631
FLANN-WOA	.123	.138	.17	.185	.912	.893
RBFN	.283	.311	.224	.247	.566	.581
RBFN-WOA	.151	.160	.191	.182	.882	.870

Figure 5. Performance of FLANN, FLANN-WOA, RBFN and RBFN-WOA on ZIA dataset

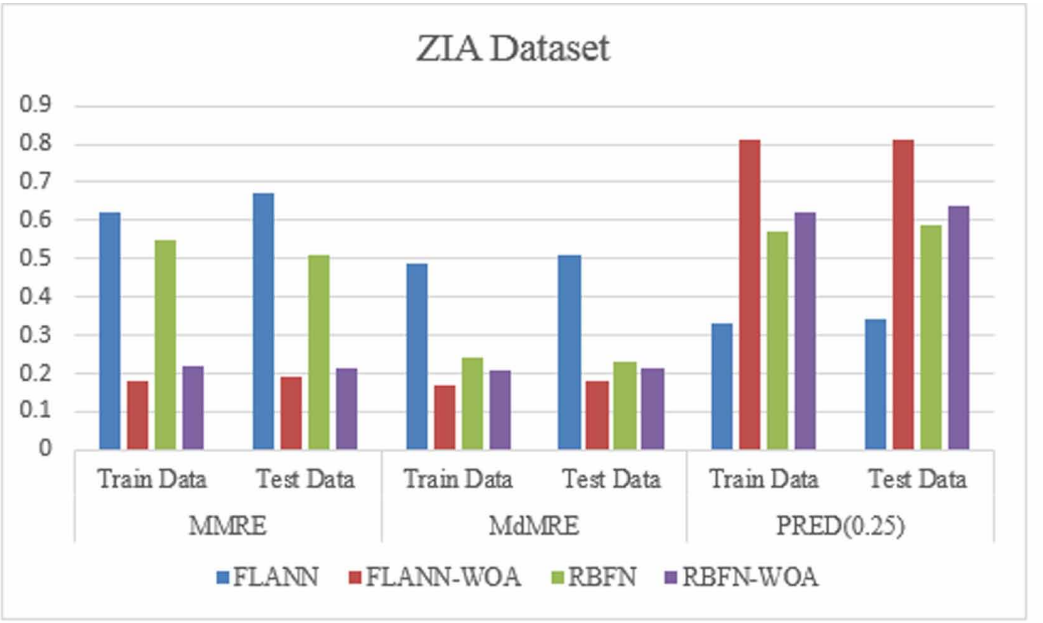


Figure 6. Performance of FLANN, FLANN-WOA, RBFN and RBFN-WOA on Company Dataset-1

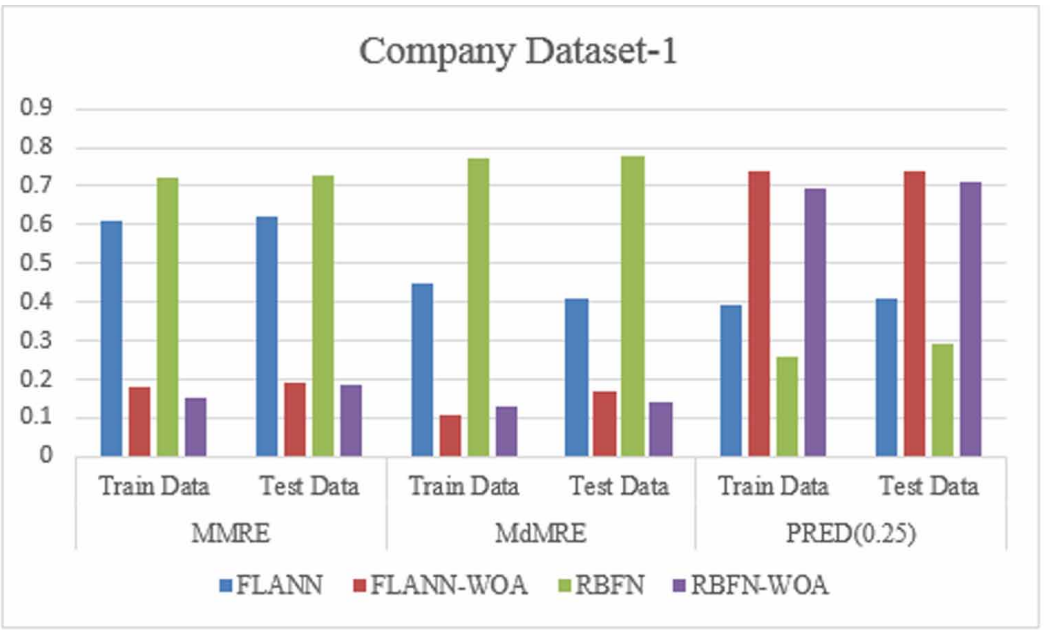
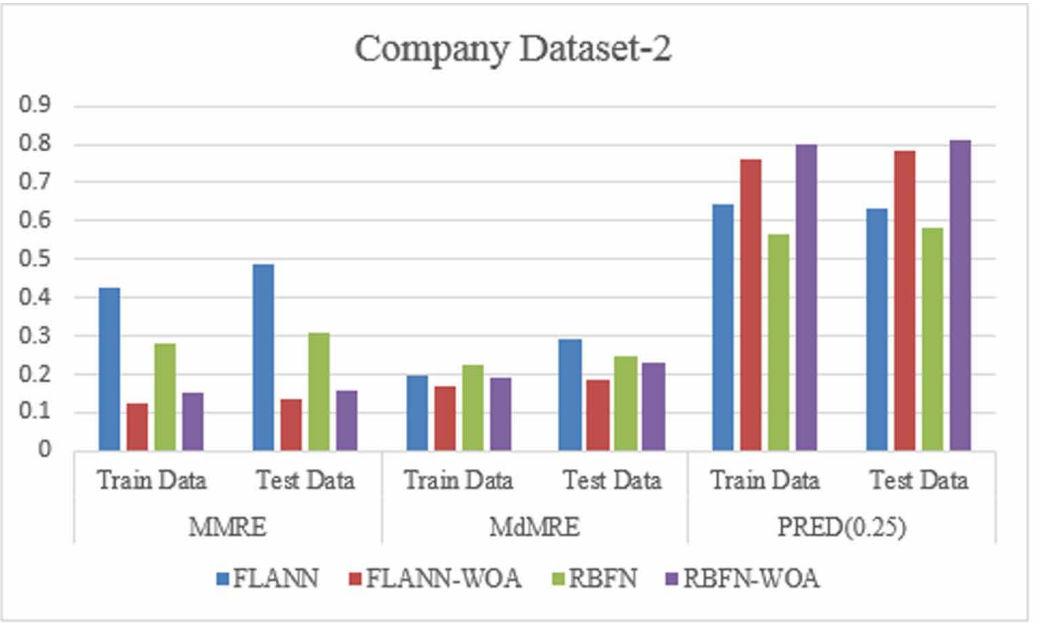


Figure 7. Performance of FLANN, FLANN-WOA, RBFN and RBFN-WOA on Company Dataset-2



Stochastic Gradient Boosting (SGB) and Random Forest (RF) which are compared with the results of the proposed techniques and depicted in Table 8.

From, the given data in Table 8 it can be seen that FLANN-WOA and RBFN-WOA gave better results than the techniques used by researchers, except for Stochastic Gradient Boosting (SGB) technique. The SGB technique provided better results than FLANN-WOA and RBFN-WOA for MMRE and MdMRE criterias.

So, it can be concluded that by integrating WOA with FLANN and RBFN has definitely enhanced the performance of FLANN and RBFN. This is also statistically validated.

Statistical validation is required to confirm the validity of any proposed approach. The statistical tools allow researchers to know whether the results evaluated with the proposed approaches can be accepted with confidence or rejected. These tools allow one to trust the new developed method. Therefore, it is necessary and valuable to compare FLANN to FLANN-WOA and RBFN to RBFN-WOA using these tests.

Statistical inferential tests are categorized into three kinds: parametric, semi-parametric and non-parametric. The parametric statistical tests relies on prior information. They make an assumption

Table 8. Comparison with earlier approaches on ZIA dataset

Techniques	MMRE	MdMRE	PRED (25)	PRED (50)	PRED (75)	PRED (100)
DT	.382	.289	38.095	71.428	80.952	90.476
SGB	.163	.115	85.714	95.238	95.238	95.238
RF	.251	.203	66.666	80.952	90.476	95.238
FLANN-WOA	.186	.173	86.4	95.238	100	100
RBFN-WOA	.198	.173	87.9	95.238	100	100

Table 9. Test statistics of Friedman test on all the datasets

Test Statistics	ZIAFLANN & ZIAFLANN-WOA	ZIARBFN & ZIARBFN-WOA	CD1FLANN & CD1FLANN-WOA	CD1RBFN & CD1RBFN-WOA	CD2FLANN & CD2FLANN-WOA	CD2RBFN & CD2RBFN-WOA
N	21	21	23	23	25	25
Chi-Square	5.000	7.200	8.909	4.545	6.000	10.667
DF	1.0	1.0	1.0	1.0	1.0	1.0
Asy. Sig.	.025	.007	.003	.033	.014	.001

about the population parameters and its probability distribution whereas the non-parametric tests doesn't assume anything about the population parameters and its distribution. The merits of both the parametric and non-parametric tests are present in semi-parametric tests.

The existing study uses IBM SPSS tool to statistically validate the models. The input to the tool is MRE values obtained after applying FLANN, FLANN-WOA, RBFN and RBFN-WOA on the datasets. Tests of normality is then performed on this data through SPSS and it is found that the data is not following a normal distribution. So, non-parametric tests are chosen to further validate the model. The first non-parametric test used is Friedman test. It is used for finding the differences between several related samples. Here, this test is used in order to initially compare whether the two techniques are different or not. The test statistics for Friedman test is defined as:

$$\chi_F^2 = \frac{12Q}{n(n+1)}\left[\sum_{j=1}^{n} R_j^2 - \frac{n(n+1)^2}{4}\right] \tag{16}$$

With, R_j , the average rank of algorithms j=1,2,….n over Q datasets.

Here, the null hypotheses and alternate hypotheses for both FLANN and RBFN are:

(a) *Null Hypotheses:* Performance of FLANN = FLANN-WOA
Alternate Hypotheses: Performance of FLANN ≠FLANN-WOA
(b) *Null Hypotheses:* Performance of RBFN = RBFN-WOA
Alternate Hypotheses: Performance of RBFN ≠ RBFN-WOA

Table 9 provides the output of Friedman test on all the three datasets.

Here, N is the total data in each dataset, Chi-Square is the value of test statistics calculated by the test, DF is the degree of freedom which is 1 as DF= n-1 . As there are two approaches to be tested on all the datasets, here n is 2 and DF is 2-1 = 1.The last row gives the Asymptotic Significant values calculated by the test. The $\chi 2$ (Chi-Square) value for DF= 1 and $\alpha = 0.05$ is 3.841. If $\chi 2$ (Chi-Square) in our test statistics is greater than 3.841 for asymptotic significant (p-value) less than 0.05 then the null hypothesis is rejected. From the Chi-Square and asymptotic significant values present in Table 9, the null hypotheses is rejected and alternate hypotheses is accepted.

To further confirm the performance of FLANN-WOA and RBFN-WOA, a second non-parametric test is applied which is Wilcoxon matched pair's test. It ranks the two models by calculating the positive and negative differences of ranks and is calculated as:

$$\min\left(\sum_{d_i>0} R(d_i) + \frac{1}{2}\sum_{d_i=0} R(d_i), \sum_{d_i<0} R(d_i) + \frac{1}{2}\sum_{d_i=0} R(d_i)\right) \quad (17)$$

Where, $R\left(\mathrm{d_i}\right)$ is the difference of ranks of performance between two models, ignoring signs. The null hypotheses and alternate hypotheses are:

Null Hypotheses: Median difference between pairs of observations is zero i.e. there is no difference in performance of FLANN and RBFN with and without WOA.

Alternate Hypotheses: Median difference between pairs of observations is not zero i.e. there is difference in performance of FLANN and RBFN with and without WOA.

Table 10 gives the result of the test. Here, for all the instances the null hypothesis is not accepted as the asymptotic significant values are less than 0.05. It is also found that for all the cases the approach with WOA has lower mean rank than its counterpart.

So after applying statistical tests it is observed that FLANN and RBFN performed differently with and without WOA and the performance of FLANN-WOA and RBFN-WOA is higher than FLANN and RBFN.

Table 10. Test statistics of Wilcoxon Matched Pairs test on all the datasets

	Mean Rank	Asymptotic Significance
ZIAFLANN ZIAFLANN-WOA	12.13 5.60	0.004
ZIARBFN & ZIARBFN-WOA	11.25 7.50	0.005
CD1FLANN CD1FLANN-WOA	11.83 10	0.005
CD1RBFN CD1RBFN-WOA	14.13 4.50	0.001
CD2FLANN CD2FLANN-WOA	13.72 8.83	0.006
CD2RBFN CD2RBFN-WOA	7.25 13.55	0.001

CONCLUSION

Effort estimation forms an integral part of any software development. A good estimation model is always the need of the hour. The paper has proposed effort estimation models for the projects using agile methodology. There are few studies existing in the literature based on neural networks for effort estimation of agile projects but none of the study has explored the fusion of ANN and a metaheuristic algorithm. The proposed study integrates ANN models, FLANN and RBFN with a metaheuristic algorithm, Whale Optimization Algorithm (WOA). Both FLANN-WOA and RBFN-WOA are evaluated on three agile datasets, one is Zia dataset and the rest two datasets are gathered through a firm in Delhi NCR incorporating agile software development. The details of these datasets

are shared so that it can be used for future work. The study uses MMRE, MDMRE and PRED (0.25) as the evaluation parameters. The experimental results demonstrated that the ANN models provided excellent results by integrating with the metaheuristic technique used. This is further validated by using two statistical tests i.e. Friedman test and Wicoxon test. Though the techniques performed well but due to lack of availability of public datasets on agile software development methodology the proposed approaches are evaluated on limited datasets. So, the future work can include evaluating these techniques on more number of agile datasets.

REFERENCES

Atanassov, K. T. (1983). Intuitionistic fuzzy set. VII ITKR's Session, Sofia, 983. (Deposed in Central Science –Technology Library of Bulgaria Academy of Science, 1697/84)

Benala, T. R., Korada, C., Mall, R., & Dehuri, S. (2013). A particle swarm optimized functional link artificial neural networks (PSO-FLANN) in software cost estimation. In Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) Advances in Intelligent Systems and Computing (vol. 199, pp. 59-66). Academic Press.

Benala, T. R., & Mall, R. (2018). DABE: Differential evolution in analogy-based software development effort estimation. *Swarm and Evolutionary Computation*, *38*, 158–172. doi:10.1016/j.swevo.2017.07.009

Bezdek, J. C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithm*. New York: Plenum. doi:10.1007/978-1-4757-0450-1

Bilgayian, S., Mishra, S., & Das, M. (2019). Effort estimation in agile software development using experimental validation of neural network models. *International Journal of Information Technology*, *11*(3), 569–573. doi:10.1007/s41870-018-0131-2

Boehm, B. W. (1994). *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall.

Chaira, T. (2011). A novel intuitionistic fuzzy C means clustering algorithm and its application to medical images. *Applied Soft Computing*, *11*(2), 1711–1717. doi:10.1016/j.asoc.2010.05.005

Choudhari, J., & Suman, U. (2012). Story Points Based Effort Estimation Model for Software Maintenance. *Procedia Technology*, *4*, 761–765. doi:10.1016/j.protcy.2012.05.124

Coelho, E., & Basu, A. (2012). Effort estimation in agile software development using story points. *International Journal of Applied Information System*, *3*(7), 7–10. doi:10.5120/ijais12-450574

Cohn, M. (2005). *Agile Estimating and Planning*. Addison-Wesley.

Curiel, I.E.E., Jacobo, J.R., Alfaro, E.V., Zepeda J.A.F., & Delgado D.F. (2018). Analysis of the changes in communication and social interactions during the transformation of a traditional team into an agile team. *The Journal of Software Evolution and Process*. 10.1002/smr.1946

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, *85*(6), 1213–1221. doi:10.1016/j.jss.2012.02.033

Dragicevic, S., Celar, S., & Turic, M. (2017). Bayesian network model for task effort estimation in agile software development. *Journal of Systems and Software*, *127*, 109–119. doi:10.1016/j.jss.2017.01.027

Foss, T., Stensrud, E., Kitchenham, B., & Myrtveit, I. (2003). A simulation study of the model evaluation criterion MMRE. *IEEE Transactions on Software Engineering*, *29*(11), 985–995. doi:10.1109/TSE.2003.1245300

Garg, S., & Gupta, D. (2015). PCA based cost estimation model for agile software development projects. In *Proceedings of International Conference on Industrial Engineering and Operations Management* (pp. 1–7). doi:10.1109/IEOM.2015.7228109

Hussain, I., Kosseim, L., & Ormandjieva, O. (2013). Approximation of COSMIC functional size to support early effort estimation in Agile. *Data & Knowledge Engineering*, *85*, 2–14. doi:10.1016/j.datak.2012.06.005

Idri, A., Zahi, A., Mendes, E., & Zakrani, A. (2007). Software cost estimation models using radial basis function neural networks. In *Proceedings of International Conference on Software process and product measurements,* (vol. 4895, pp. 21–31). doi:10.1007/978-3-540-85553-8_2

Kaur, P., Soni, A. K., & Gossain, A. (2012). Novel intuitionistic fuzzy C means clustering for linearly and nonlinearly separable data. *WSEAS Transactions on Computers*, *11*(3), 65–76.

Kaushik, A., Soni, A. K., & Soni, R. (2013). Radial basis function network using intuitionistic fuzzy C means for software cost estimation. *International Journal of Computer Applications in Technology*, *47*(1), 86–95. doi:10.1504/IJCAT.2013.054305

Kaushik, A., Tayal, D. K., Yadav, K., & Kaur, A. (2016). Integrating firefly algorithm in artificial neural network models for accurate software cost predictions. *Journal of Software Evolution and Process*, *28*(8), 665–688. doi:10.1002/smr.1792

Kaushik, A., Verma, S., Singh, H. J., & Chhabra, G. (2017). Software Cost Optimization Integrating Fuzzy System and COA-Cuckoo Optimization Algorithm. *International Journal of System Assurance Engineering and Management*, *8*(S2), 1461–1471. doi:10.1007/s13198-017-0615-7

Kocaguneli, E., & Menzies, T. (2013). Software effort models should be assessed via leave-one-out validation. *Journal of Systems and Software*, *86*(7), 1879–1890. doi:10.1016/j.jss.2013.02.053

Lang, M., Conboy, K., & Keaveney, S. (2011). Cost Estimation in Agile Software Development Projects. In *Proceedings of International Conference on Information Systems Development* (pp. 1-12). Prato, Italy: Academic Press.

Martínez, J. L., Noriega, A. R., Ramírez, R. J., Licea, G., & Jiménez, S. (2018). User stories complexity estimation using bayesian networks for inexperienced developers. *Journal of Cluster Computing*, *21*(1), 715–728. doi:10.1007/s10586-017-0996-z

Mensah, S., Keung, J., Bosu, M. F., & Bennin, K. E. (2018). Duplex output software effort estimation model with self-guided interpretation. *Information and Software Technology*, *94*, 1–13. doi:10.1016/j.infsof.2017.09.010

Menzies, T., Chen, Z., Hihn, J., & Lum, K. (2006). Selecting best practices for effort estimation. *IEEE Transactions on Software Engineering*, *32*(11), 883–895. doi:10.1109/TSE.2006.114

Mirjalili, S., & Lewis, A. (2016). The Whale Optimization Algorithm. *Advances in Engineering Software*, *95*, 51–67. doi:10.1016/j.advengsoft.2016.01.008

Mishra, B. B., & Dehuri, S. (2007). Functional Link Artificial Neural Network for classification task in Data Mining. *Journal of Computational Science*, *3*(12), 948–955. doi:10.3844/jcssp.2007.948.955

Moody, J., & Darken, C. J. (1989). Fast learning in networks of locally tuned processing units. *Neural Computation*, *1*(2), 281–294. doi:10.1162/neco.1989.1.2.281

Nguyen, V., Boehm, B., & Huang, L. G. (2019). Determining relevant training data for effort estimation using window based COCOMO calibration. *Journal of Systems and Software*, *147*, 124–146. doi:10.1016/j.jss.2018.10.019

Panda, A., Satapathy, S. M., & Rath, S. K. (2015). Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points. In *Proceedings of 3rd International Conference on Recent Trends in Computing* (*vol. 57*, pp.772 – 781). Procedia Computer Science. doi:10.1016/j.procs.2015.07.474

Papadopoulos, G. (2015). Moving from Traditional to Agile Software Development Methodologies Also on Large, Distributed Projects. *Procedia: Social and Behavioral Sciences*, *175*, 455–463. doi:10.1016/j.sbspro.2015.01.1223

Perkusich, M., Gorgônio, K. C., Almeida, H., & Perkusich, A. (2017). Assisting the continuous improvement of Scrum projects using metrics and Bayesian networks. *Journal of Software Evolution and Process*, *29*(6), e1835. doi:10.1002/smr.1835

Popli, R., & Chauhan, N. (2014). Cost and Effort estimation in agile software development. In *Proceedings of International conference on Reliability, Optimization and Information Technology* (pp. 57-61). Faridabad, India: Academic Press. doi:10.1109/ICROIT.2014.6798284

Rao, B. T., Dehuri, S., & Mall, R. (2012). Functional link artificial neural networks for software cost estimation. *International Journal of Applied Evolutionary Computation*, *3*(2), 62–82. doi:10.4018/jaec.2012040104

Rao, B. T., Sameet, B., Swathi, G. K., Gupta, K. V., Ravi Teja, Ch., & Sumana, S. (2009). A novel neural network approach for software cost estimation using functional link artificial neural network (FLANN). *International Journal of Computer Science and Network Society*, *9*(6), 126–131.

Raslan, A. T., Darwish, N. R., & Hefny, H. A. (2015). Towards a fuzzy based framework for effort estimation in agile software development. *International Journal of Computer Science and Information Security*, *13*(1), 37–45.

Salmanoglu, M., Hacaloglu, T., & Demirors, O. (2017). Efort Estimation for Agile Sofware Development: Comparative Case Studies Using COSMIC Functional Size Measurement and Story Points. In Proceedings of IWSM /Mensura'17 (pp.42-50). Gothenberg, Sweden: Academic Press.

Satapathy, S. M., Panda, A., & Rath, S. K. (2014). Story point approach based agile software effort estimation using various SVR kernel. In *Proceedings of 26th International Conference on Software Engineering and Knowledge Engineering* (pp. 304–307). Vancouver: Academic Press.

Satapathy, S. M., & Rath, S. K. (2017). Empirical assessment of machine learning models for agile software development effort estimation using story points. *Innovations in Systems and Software Engineering*, *13*(2-3), 191–200. doi:10.1007/s11334-017-0288-z

Stensrud, E., Foss, T., Kitchenham, B. A., & Myrtveit, I. (2002). An empirical validation of the relationship between the magnitude of relative error and project size. In *Proceedings of the IEEE 8th metrics symposium* (pp. 3–12). Ontario: Academic Press. doi:10.1109/METRIC.2002.1011320

Tanveer, B. (2016). Hybrid Effort Estimation of Changes in Agile Software Development. *Lecture Notes in Business Information Processing, 251, 316-320*. doi:10.1007/978-3-319-33515-5_33

Tanveer, B., Vollmer, A. M., & Braun, S. (2018). A hybrid methodology for effort estimation in Agile development: An industrial evaluation. In *Proceedings of International Conference on Software and System Process* (pp. 21-30). doi:10.1145/3202710.3203152

Tanveer, B., Vollmer, A. M., Braun, S., & Ali, N. B. (2019). An evaluation of effort estimation supported by change impact analysis in agile software development. *Journal of Software Evolution and Process.* .10.1002/smr.2165

Tolfo, C., Wazlawick, R.S., Ferreira, M.G.G., & Forcellini, F.A. (2018). Agile practices and the promotion of entrepreneurial skills in software development. *Journal of Software Evolution and Process*. .10.1002/smr.1945

Usman, M., Britto, R., Damm, L. O., & Borstler, J. (2018). Effort estimation in large scale software development: An industrial case study. *Journal of Information and Software Technology.*, *99*, 21–40. doi:10.1016/j.infsof.2018.02.009

Venkataiah, V., Mohanty, R., Pahariya, J. S., & Nagaratna, M. (2017). Application of ant colony optimization techniques to predict software cost estimation. In Computer Communication, Networking and Internet Security. Springer. doi:10.1007/978-981-10-3226-4_32

Yu, H., Xie, T., Paszczyñski, S., & Wilamowski, B. (2011). Advantages of Radial Basis Function Networks for Dynamic System Design. *IEEE Transactions on Industrial Electronics*, *58*(12), 5438–5450. doi:10.1109/TIE.2011.2164773

Ziauddin, T. S. K., & Zia, S. (2012). An effort estimation model for agile software development. *Advances in Computer Science and Its Applications*, *2*(1), 314-324.

Anupama Kaushik is working in Maharaja Surajmal Institute of Technology, New Delhi, India. She has 14 years of teaching experience and her research areas include Software Engineering and Machine Learning Techniques.

D. K. Tayal (PhD) is working as Professor in department of Computer Science & Engg. Earlier he was Head in Department of Computer Science & Engg. He has done M.Tech(Computer Engg.), Ph.D. (Computer Engg.) from Jawaharlal Nehru University. He has a teaching experience of more than 13 years. He did his research in the field of Intelligent Systems and has published approximately 30 research papers in International Journals & Conferences. He was earlier selected as a Research Engineer in C-DOT (Govt of India) and also a Class-I Gazetted Officer by UPSC. He has written numerous articles in newspapers and magazines as well. He is a member of International Advisory committee of International Journal of Computer Science, Hongkong and member of International Advisory Board of International Journal of Software Engg & Applications, Korea. Besides this he is referee on Editorial board of various International Journals including the infamous "IEEE Transactions on Fuzzy Systems" having the SCI Impact Factor 4.26. He has extensively delivered lectures in Conferences, Seminars and Workshops. He keeps on regularly conducting Conferences & Workshops in his field of interest, in his department and outside also. He is currently supervising Ph.D. in the field of Intelligent Systems, Data-Mining, DBMS and Text Mining & Natural Language Processing.

Kalpana Yadav has been an Assistant Professor at IGDTUW, Department of IT, from the last 8.6 years. She earned her PhD in software reliability from Jiwaji University, Gwalior and M.Tech in CSE from GJU, Hisar. She is life member of Computer Society of India (CSI) and Society for Reliability Engineering, quality and Operations Management (SREQOM). She has many national and international publications.