An Effort Estimation Approach for Agile Software Development using Fireworks Algorithm Optimized Neural Network

Article · August 2016

CITATIONS
8

READS
617

2 authors:

Tung Khuat
University of Technology Sydney
45 PUBLICATIONS **185** CITATIONS
SEE PROFILE

Le Thi My Hanh
Danang University of Technology
39 PUBLICATIONS **194** CITATIONS
SEE PROFILE

An Effort Estimation Approach for Agile Software Development using Fireworks Algorithm Optimized Neural Network

Thanh Tung Khuat, My Hanh Le

***Abstract*—Software effort estimation is one of the most critical steps in the software development process. The success or failure of projects relies greatly on the accuracy of effort estimation and schedule results. Agile software development process has become prevalent in the industry and replacing the conventional approaches of software development. Nevertheless, the question of accurate estimation of effort for this novel method has still been a challenging problem with regard to researchers and practitioners. This study aims to propose a novel method to ameliorate the accuracy of agile software effort prediction process using Artificial Neural Network (ANN) optimized by Fireworks Algorithm (FWA). The performance of the proposed approach is compared to the various types of neural networks and the regression model. In addition, the role of Fireworks Algorithm in optimizing the weights and biases of the ANN is also compared with other optimization algorithms.**

***Index Terms*— Software Effort Estimation, Agile Software Development, User Story, Artificial Neural Network, Fireworks Algorithm, Levenberq-Marquardt.**

I. Introduction

Software effort estimation has a critical role to play in the process of software development. Both underestimation and overestimation of effort have a negative influence on the success of projects. Therefore, it is expected to find out a method to estimate the software effort precisely. A variety of estimation techniques using data collected from past projects combined to mathematical formula to predict the project cost introduced such as COCOMO II [1], SLIM [2], PRICE-S [3]. Estimation techniques are distributed into regression-based models, expert-based methods, learning-oriented approaches, and Bayesian methods. The diversity of novel software development methodologies has resulted in the limitation of traditional approaches.

In recent years, the appearance of agile software development process has met the progress of the new software engineering methodology. The use of agile methods enables organizations to respond the volatility in the software development life cycle. The application of an estimation method in the agile process is a difficult task in which we need to anticipate the size and complexity of the products to be constructed in order to specify what to do next [4]. To meet this requirement, user stories of the product need to be collected and analyzed the complexity to determine story points for components of the project. Another factor which may impact the agile software development is team velocity. That is the total number of story points that a team is able to implement in a sprint. This study uses story points and velocity to train the ANN, then the model is used to give the estimated time to complete a novel project.

The efficiency of ANN largely depends on their architecture, their training algorithm, and the selection of features utilized in training. The process of network learning optimization is to find out the weights configuration associated with the minimum output error. Many algorithms applied to training process including Ant Colony Optimization [5], Simulated Annealing and Genetic Algorithms [6], Tabu search [7]. As for Multi-layer Perceptron (MLP) neural networks, the Back-propagation (BP) algorithm and Levenberq-Marquardt (LM) are widely used for the training process [8]. The researchers prefer to use LM among the conventional approaches because of its speed of convergence and performance. However, this algorithm is easy to be stuck at the local optimum. To cope with this issue, some global optimization algorithms were employed to tune weights of MLP aiming to avoid the local optimum such as evolutionary algorithm [9], artificial bee colony (ABC) algorithm [10], ant colony optimization [5], and hybrid particle swarm optimization and gravitational search algorithm [11]. In this paper, Fireworks algorithm and LM are utilized as new training methods for the Feedforward Neural Network (FNN) in order to figure out the effectiveness of these algorithms in reducing the problems of trapping in the local minimum and the slow convergence rate of LM algorithm.

FWA proposed by Tan and Zhu [12] is a population-based algorithm inspired by the explosion process of fireworks. The FWA has a strong capability of seeking global optimal result and LM algorithm has a strong ability to find the local optimal result. Therefore, this study proposes a combination method of FWA and LM for training the FNN aiming to minimize the output error of the FNN in order to give the accurate estimation result of the effort for the agile software

Thanh Tung Khuat is now with the Information Technology Faculty, University of Science and Technology - The University of Danang, Vietnam (e-mail: thanhtung09t2@gmail.com).

My Hanh Le is now with the Information Technology Faculty, University of Science and Technology - The University of Danang, Vietnam (e-mail: ltmhanh@dut.udn.vn).

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

development process.

The rest of this paper is organized as follows. Section II represents the previous methods for estimating the effort of agile software development process. Section III shows a model used to predict effort and introduces proposed steps to apply neural network models for the agile software effort estimation. The combination of FWA with LM applied to neural network training is presented in section IV. Section V is the experiment and the analysis of the performance of proposed model. Threats to validity of the study are discussed in section VI. Finally, section VII gives the conclusion of this paper.

II. Related Work

Keaveney and Conboy [13] figured out the applicability of conventional estimation techniques to agile development approaches by focusing on four case studies of using agile of different organizations. The authors used the main estimation techniques being expert knowledge and analogy to past projects. The obtained results revealed that the estimation inaccuracy using the proposed method was a less frequent occurrence compared to the use of traditional approaches. Coelho and Basu [14] gave an overview of the various size estimation techniques based on story points for the agile software development process. The authors showed the steps followed in the story point based approach and highlighted the area which needs to be studied further. Abrahamsson and Koskela [15] described the way to collect metrics to gauge the productivity, quality and schedule estimation, cost and effort estimation for an agile software development project using extreme programming. The authors provided evidence that agile approaches are efficient and appropriate for a variety of situations and environments. Hussain *et al.* [16] presented an approach to approximate COSMIC functional size from informally written textual requirements demonstrating its applicability in popular agile processes. Popli and Chauhan [17] introduced a model for effort and cost estimation in the agile software development by using the regression analysis. Hamouda [18] proposed a process and methodology that assure relativity in software sizing while using agile story points on the level of the CMMI organizations. Oliveira [19] provided a comparative research on Support Vector Regression (SVR), Radial Basis Function Neural Networks (RBFNs) and the linear regression for the estimation of software development effort. The experiment was conducted on NASA project data sets and the experimental results showed that SVR performed better than the RBFN and the linear regression analysis. Satapathy *et al.* [16] estimated the effort of the agile software using story point approach in which the total number of story points and project velocity were employed to predict the effort involved in developing an agile software product. The obtained results were optimized by utilizing four various support vector regression kernel methods. The authors concluded that the radial basis function kernel-based support vector regression technique outperformed other three kernel methods.

Zia *et al.* [21] developed an effort estimation model for agile software projects. Authors introduced a method to compute the team velocity and story points from the user stories and features of the product, and then the regression analysis method was employed to give the completion time of the project. The proposed model was validated using the empirical data collected from 21 software projects. The obtained results showed that the model had good estimation accuracy in terms of the Mean Magnitude of Relative Error. Panda et al. [22] attempted to enhance the prediction accuracy of the agile software effort estimation process proposed by Zia. To solve this problem, various types of neural networks including General Regression Neural Network (GRNN), Probabilistic Neural Network (PNN), Group Method of Data Handling (GMDH) Polynomial Neural Network and Cascade-Correlation Neural Network were used and compared. Our study also aims at ameliorating the accuracy of estimation model of Zia by introducing an ANN optimized by the combination of Fireworks and LM algorithms.

III. Effort Estimation Approach for the Agile Software Development Process

A. An effort estimation model for agile projects

In [21], Zia et al. proposed a model to estimate the effort of agile software projects. This model uses story points and team velocity to predict the effort for a project.

1) Computing the story point of an agile project

Story points are a number of user stories associated with their complexity completed in a unit time. In order to compute story points of an agile project, we first determine the story sizes and the complexity of each story size. Story size is an estimate of the relative scale of the work in terms of actual development effort. Each story size is assigned a value from 1 to 5 based on their scales. Value 1 indicates that the story is very small which needs tiny effort level with only a few hours of work. Value 2 shows that it is expected to finish the user story in a day or two of work meanwhile value 3 presents that we need from two to five days of work to complete the user story. Value 4 is given to the story having a very large size and requiring more than a week of work to accomplish as well as we need to take into account breaking it down into a set of smaller stories. Value 5 represents an extremely large story and it is really hard to estimate time accurately. After specifying the scale of the story, we have to consider its complexity. The complexity is also measured by five values assigned to the user story according to its nature. Value 1 states that the story requires basic programming skills to complete, and their technical and business requirements are very clear with no ambiguity. Value 5 shows that the story is extremely complex with many dependencies on other stories, systems or subsystems, and it needs a skill set or experience that is important, but absent in the team along with the extensive research and significant refactoring. The details of the user story complexity are clearly described in [21].

The total story points for N user stories of a project is computed as Eq. (1).

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

$$SP = \sum_{i=1}^{N} C_i \cdot S_i \quad (1)$$

2) Determining agile velocity

The initial agile velocity of a team is simply how many units of effort that this team is able to complete in a typical sprint. It can be also defined as how many story points that a team can handle in one sprint, and it is determined as follows:

$$V_i = \textit{Units of Effort Completed / Sprint Time}$$

In reality, the velocity of a project is not only being simply measured by units of effort and sprint time but it also is influenced by two other factors including friction and variable or dynamic forces.

The friction forces are constants that drag on productivity and reduce the project velocity. They consist of team composition, process, environmental factors, and team dynamics. Their effects are long term, but they are easy to tackle. Table I shows four friction factors with a range of values and these values have been tuned following their risk severity [21].

TABLE I. FRICTION FACTORS

Fiction Factor	Stable	Volatile	Highly Volatile	Very HighlyVolatile
Team composition	1	0.98	0.95	0.91
Process	1	0.98	0.94	0.89
Environmental Factors	1	0.99	0.98	0.96
Team dynamics	1	0.98	0.91	0.85

The value of friction (FR) can be computed as the product of all four fraction factors (FF) shown in Eq. (2).

$$FR = \prod_{i=1}^{4} FF_i \quad (2)$$

The variable or dynamic forces decelerate the project or the performance of team members and bring about the project velocity to be irregular. These forces are usually unpredictable and unexpected. They include team changes, new tools requiring learning, vendor defects, responsibilities outside of the project of team members, personal issues, stakeholders, unclear requirements, changing requirements, and relocation. Table II describes variable or dynamic force factors and the values associated with them on the basis of same analogy as for size.

Dynamic Force (DF) then is computed as the product of all nine variable factors (VF) as shown in Eq. (3).

$$DF = \prod_{i=1}^{9} VF_i \quad (3)$$

Deceleration of an agile project is the product of friction and dynamic forces impacting the velocity as Eq. (4).

$$D = FR \cdot DF \quad (4)$$

The final velocity of a project under the influence of friction and dynamic forces is computed as Eq. (5).

$$V = (V_i)^D \quad (5)$$

From the team velocity and story points of a project, Zia *et al.* [21] used the regression method to predict the duration needed to complete the project.

TABLE II. DYNAMIC FORCE FACTORS

Variable Factor	Normal	High	Very High	Extra High
Expected team changes	1	0.98	0.95	0.91
Introduction of new tools	1	0.99	0.97	0.96
Vendor's defects	1	0.98	0.94	0.90
Team member's responsibility outside the project	1	0.99	0.98	0.98
Personal issues	1	0.99	0.99	0.98
Expected delay in Stakeholder response	1	0.99	0.98	0.96
Expected ambiguity in details	1	0.98	0.97	0.95
Expected changes in environment	1	0.99	0.98	0.97
Expected relocation	1	0.99	0.99	0.98

B. Proposed steps for determining the predicted effort using Artificial Neural Network

This study proposes to employ the ANN optimized by the combination of Fireworks algorithm and LM for the training process of ANN. The inputs of the neural network models are a total number of story points and project final velocity and an output is the effort such as the completion time of the project. In our work, in order to enhance the accuracy of the estimation effort, FWA and LM algorithms are used to optimize the weights and biases of the ANN. The steps to estimate the effort of an agile software project are shown as follows:

Step 1: Collecting the total number of story points, the project final velocity, and the actual effort. In this paper, these data are taken from Zia's work [21].

Step 2: Normalizing the data of story points, project velocity, and actual effort values within the range of [0, 1]. Let X is the data set, x is an item of the data set then the normalized value x' of x can be computed as Eq. (6).

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)} \quad (6)$$

where max(X) and min(X) are the minimum and maximum values of the data set X respectively.

Step 3: Splitting the data set into training and testing sets. In this study, the first fifteen projects are used for training and the others for testing.

Step 4: Training ANN: the FWA and LM are used together to optimize the weights and biases of the ANN.

Step 5: Testing and evaluating the performance of the proposed model using criteria shown in section V.

After the neural network implementation is completely done, the obtained results are compared to the other types of ANN as well as assessing the role of FWA in optimizing the weights and biases of the ANN.

IV. TRAINING FEED-FORWARD ARTIFICIAL NEURAL NETWORK

A. Feed-forward Artificial Neural Network

A feed-forward neural network (FFNN) is an artificial neural network in which connections between the units do not

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

form a cycle. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes, and to the output nodes without cycles or loops. Each node in the FFNN receives a signal from the nodes in the previous layer, and each of those signals is multiplied by a specific weight value. The weighted inputs are then summed and passed through an activation function which scales the output to a fixed range of values. The output of the node is broadcast to all of the nodes in the next layer. The output value of a node is computed by using Eq. (7).

$$o_i = f(\sum_{j=1}^{n} w_{ij} \cdot x_j + b_i) \quad (7)$$

where o_i is the output of the i^{th} node, x_j is the input of the j^{th} node, w_{ij} is the connection weight between the current node and input x_j, b_i is the bias of the i^{th} node, and f is an activation function. The activation function is often a nonlinear function such as a Bipolar Sigmoid, a logarithmic sigmoid, a Gaussian function. This study uses the sigmoid function as the transfer function for hidden and output layer neurons.

$$f = \frac{1}{1 + e^{-x}} \quad (8)$$

To enhance the accuracy of estimated results, the FWA is used to optimize the network weights. The main idea is to convert the weight matrices of the ANN into individuals of population-based optimization algorithms. This study chooses the mean squared error as a network error function shown in Eq. (9) and the objective of the FWA is to minimize this function.

$$E\big(w(t)\big) = \frac{1}{T} \cdot \sum_{i=1}^{T} \sum_{k=1}^{K} (P_k - A_k)^2 \quad (9)$$

where E(w(t)) is the error at the t^{th} iteration, $w(t)$ is the vector of weights of the connections at the t^{th} iteration, P_k and A_k are predicted and actual values of the effort of the k^{th} output node. K is the number of output nodes and T is the number of patterns in the training dataset.

B. Proposed Architecture of FNN

This study employs a multilayer perceptron neural network to estimate the completion time for an agile project. A MLP is a feed-forward artificial neural network model that maps sets of input data onto a set of appropriate outputs. A MLP includes multiple layers of nodes in a directed graph, with each layer fully connected to the next one. In general, the architecture of the MLP might have many hidden layers and each hidden layer can consist of many neurons. However, many studies and experimental results also indicate that one hidden layer is sufficient for most of the forecasting problems as [23], [24], and [25]. Therefore, this work utilizes the architecture of the MLP with one hidden layer.

Another difficult task when choosing good parameters for the ANN is the number of hidden neurons. Setting a suitable architecture of the ANN for a specific problem is an essential issue because the network topology directly influences its computational complexity and generalization ability. Too much hidden neurons will drive the ANN to the over-fitting in which the ANN performs well on training data and poorly on data it has not seen. We conducted the experiments with the number of hidden neurons varying from 4 to 60. It can be seen that the value of hidden neurons which is more than 40 will give a low error rate with regard to the training data set, but the testing error is quite high meanwhile if the number of hidden neurons is 27 then the training error is very low and the testing error is lowest. In our study, we therefore choose 27 neurons for the hidden layer.

As for the input layer, there are two neurons being the total number of story points and the project final velocity. The output of the ANN has only one neuron being the completion time of the project.

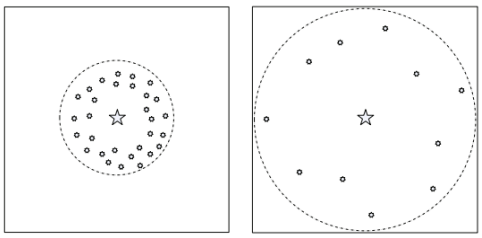
C. Fireworks Algorithm

Fireworks algorithm inspired by observing fireworks explosion is proposed by Tan and Zhu [12] for global optimization of complex functions. In this algorithm, two kinds of search processes are employed as well as generating mechanisms for keeping the diversity of sparks. When a firework explodes, a shower of sparks will be created around the firework. The explosion process of a firework might be considered as a local search around a specific point. In order to seek a point x_j such that $f(x_j) = y$, 'fireworks' are constantly set off in potential space until one 'spark' target or reaching a point that is relatively close to the point x_j.

In this paper, a location presents a possible set of optimized weights and biases for the ANN. In the FWA, for each generation of the explosion, n locations are chosen. After the explosion, the locations of sparks are obtained and assessed. The algorithm stops when the optimal location is found. Otherwise, n other locations are selected from the current sparks and fireworks for the next generation of the explosion. It can be seen that the success of the FWA relies greatly on a good design of the explosion process and an appropriate approach for choosing locations.

1) Design of Fireworks explosion

Through inspecting fireworks display, two specific kinds of the fireworks explosion behavior are found. A good firework explosion creates numerous sparks which centralize the explosion center. On the other hand, a bad firework explosion generates a few sparks which scatter around the explosion center. Two these behaviors are shown in Fig. 1.



(a) Good explosion (b) Bad explosion

Fig. 1. Two kinds of fireworks explosion

From the viewpoint of a search algorithm, a good firework means that the firework might be near to the optimal location. Therefore, it is proper to use more sparks to search the local area around the firework. In contrast, a bad firework shows

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

that the firework can be far from the optimal location. In this case, the search radius should be larger. In the FWA, a good firework generates more sparks and the explosion amplitude is smaller when compared to the bad one.

Number of Sparks: Suppose that the FWA is designed for finding the optimal solution of a general optimization problem as follows:

$$Minimize\ f(x) \in R, x_{min} \le x \le x_{max} \tag{10}$$

where $x = [x_1, x_2, \dots, x_D]$ is a location in the potential space, $f(x)$ is an objective function and in this paper *f*(x) = E(w(t)), x_{min} and x_{max} denote the bounds of the potential space, D is the dimensionality of the location x. Let *HN* be the number of hidden neurons, the value of D can be computed by using Eq. (11) whose details are shown in Table III.

$$D = IW\{1,1\} + b\{1,1\} + OW\{2,1\} + b\{2,1\} \tag{11}$$

TABLE III. PARAMETERS FOR SPECIFYING THE SIZE OF INDIVIDUALS

Value	Symbol	Description
2·*HN*	IW{1, 1}	Weights of the connections from the input layer to the hidden layer
HN	b{1, 1}	Biases of neurons in the hidden layer
HN·1	OW{2, 1}	Weights of the connections between the output layer and the hidden layer
1	b{2, 1}	Biases of output neurons

As mentioned above, the number of hidden neurons in this study is 27, so *HN* = 27.

Next, the number of sparks provided by each firework x_i is determined as Eq. (12).

$$s_i = t \times \frac{y_{max} - f(x_i) + \delta}{\sum_{i=1}^{n}\left(y_{max} - f(x_i)\right) + \delta} \tag{12}$$

where t is a parameter controlling the total number of sparks created by *n* fireworks, $y_{max} = max\left(f(x_i)\right)\ (i = 1, 2, \dots, n)$ is the maximum or worst value of the objective function among *n* fireworks, and δ denotes the smallest constant in the computer, is employed to avoid zero-division-error.

In order to avoid overwhelming impacts of gorgeous fireworks, bounds are defined for s_i, which is described in Eq. (13).

$$\hat{s}_i = \begin{cases} round(a \times t) & if\ s_i < a \cdot t \\ round(b \times t) & if\ s_i > b \cdot t, \quad a < b < 1 \\ round(s_i) & otherwise \end{cases} \tag{13}$$

where a and b are const parameters.

Amplitude of Explosion: In contrast to the design of sparks number, the amplitude of a good firework explosion is smaller than that of a bad one. The amplitude of the explosion for each firework is specified as Eq. (14).

$$A_i = \hat{A} \cdot \frac{f(x_i) - y_{min} + \delta}{\sum_{i=1}^{n}(f(x_i) - y_{min}) + \delta} \tag{14}$$

where $\hat{A}$ is the maximum explosion amplitude, and $y_{min} = min\left(f(x_i)\right)\ (i = 1, 2, \dots, n)$ is the minimum or best value of the objective function among the *n* fireworks.

Generating Sparks: In the explosion process, sparks might meet the influences of explosion from random z dimensions. In the FWA, the number of the affected directions is randomly obtained from Eq. (15).

$$z = round\left(D \times rand(0,1)\right) \tag{15}$$

where D is the dimensionality of the location x, and $rand(0,1)$ is a random number distributed uniform in the range of [0, 1].

The location of a spark of the firework x_i is obtained by using Algorithm 1.

Algorithm 1. Find out the location of a spark

Initialize the location of the spark: $\tilde{x}_j = x_i$
$z = round\left(D \times rand(0,1)\right)$
Randomly choose z dimensions of x_j
Compute the displacement: $d = A_i \times rand(-1,1)$;
for each dimension $\tilde{x}_k^j \in$ {pre-selected z dimensions of $\tilde{x}_j$} **do**
  $\tilde{x}_k^j = \tilde{x}_k^j + d$;
  if $\tilde{x}_k^j < x_k^{min}$ or $\tilde{x}_k^j > x_k^{max}$ **then**
    map $\tilde{x}_k^j$ to the potential space: $\tilde{x}_k^j = x_k^{min} + \left|\tilde{x}_k^j\right| \%\ (x_k^{max} - x_k^{min})$;
  end if
end for

To maintain the diversity of sparks, there is another way of generating sparks called Gaussian explosion, which is presented in Algorithm 2. A function $Gaussian(1,1)$, which is a Gaussian distribution with mean 1 and standard deviation 1, is utilized to define the coefficient of the explosion. $\widehat{m}$ sparks of this type are created in each explosion generation.

Algorithm 2. Find out the location of a specific spark

Initialize the location of the spark: $\hat{x}_j = x_i$;
$z = round(D \times rand(0,1))$;
Randomly choose z dimensions of $\hat{x}_j$;
Compute the coefficient of Gaussian explosion: $g = Gaussian(1,1)$;
for each dimension $\hat{x}_k^j \in$ {pre-selected z dimensions of $\hat{x}_j$} **do**
  $\hat{x}_k^j = \hat{x}_k^j \cdot g$;
  if $\hat{x}_k^j < x_k^{min}$ or $\hat{x}_k^j > x_k^{max}$ **then**
    map $\hat{x}_k^j$ to the potential space: $\hat{x}_k^j = x_k^{min} + \left|\hat{x}_k^j\right| \%\ (x_k^{max} - x_k^{min})$;
  end if
end for

2) Selection of locations

At the beginning of each explosion generation, *n* locations will be selected for the fireworks explosion. In the FWA, the current best location x^* in the current generation is always retained for the next explosion generation. After that, $n-1$ locations are chosen based on their distance to other locations in order to maintain the diversity of sparks. The general distance between a location x_i and other locations are defined as Eq. (16).

$$R(x_i) = \sum_{j \in C} d\left(x_i, x_j\right) = \sum_{j \in C} \left\| x_i - x_j \right\| \tag{16}$$

where C is the set of all current locations of both fireworks and sparks.

Then the selection probability of a location x_i is defined as Eq. (17).

$$p(x_i) = \frac{R(x_i)}{\sum_{j \in C} R(x_j)} \tag{17}$$

When assessing the distance, any distance measure might be used involving Euclidean distance, Manhattan distance, Angle-based distance, *etc*. In this paper, Euclidean distance is employed.

Algorithm 3 shows the overview of the FWA. During each

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

explosion generation, two kinds of sparks are generated respectively as presented in Algorithm 1 and Algorithm 2. In the first kind, the explosion amplitude and the number of sparks rely on the quality of the corresponding firework. In contrast, the second type is produced using a Gaussian explosion process, which carries out seeking in a local Gaussian space around a firework.

Algorithm 3. Framework of the FWA

Randomly choose *n* locations for fireworks;
while stopping criteria is not met **do**
  Set off *n* fireworks respectively at the *n* locations:
  for each firework x_i **do**
    Compute the number of sparks that the firework produces: $\hat{s}_i$ by using Eq. (12) and Eq. (13);
    Find out locations of $\hat{s}_i$ sparks of the firework x_i using Algorithm 1;
  end for
  for $k = 1 \rightarrow \hat{m}$ **do**
    Randomly choose a firework x_j;
    Create a specific spark for the firework by using Algorithm 2;
  end for
  Choose the best location and keep it for the next explosion generation;
  Randomly select $n - 1$ locations from the two types of sparks and the current fireworks according to the probability given in Eq. (17);
end while

D. Training ANN using Fireworks Algorithm and Levenberq-Marquardt

The FWA has a strong capability of finding the global optimized result and the LM algorithm [26] has a strong ability to seek the local optimized solution. This paper proposes the combination of FWA with LM to train the ANN. The key idea of this method is that the FWA is employed at the beginning stage of searching for the optimum. Then, the training process is continued with the LM algorithm. In the first stage, the FWA finishes its training process; LM algorithm then begins training with the optimal weights of FWA algorithm with 1000 epochs more. The LM algorithm interpolates between the Newton method and gradient descent approach where it approximates the error of the network with a second order expression.

The diagram of hybrid FWA-LM algorithm is shown in Fig. 2.

V. EXPERIMENTS

A. Evaluation criteria

The performance of the proposed approach is assessed by using criteria as below:

- The Mean Squared Error (MSE) is computed by the following equation:

$$MSE = \frac{1}{T} \cdot \sum_{i=1}^{T} (A_i - P_i)^2 \quad (18)$$

where A_i and P_i are actual and predicted effort values of the i^{th} test data respectively.

- The Mean Magnitude of Relative Error (MMRE) is the average percentage of the absolute values of the relative errors over an entire data set. Given *T* tests, the MMRE is calculated as Eq. (19).

$$MMRE = \frac{100}{T} \cdot \sum_{i=1}^{T} \frac{|P_i - A_i|}{A_i} \quad (19)$$

- *PRED(N)* indicates the average percentage of estimates that were within *N* percent of the actual values. Given *T* tests, then:

$$PRED(N) = \frac{100}{T} \cdot \sum_{i=1}^{T} \begin{cases} 1 & if \ \frac{|P_i - A_i|}{A_i} < \frac{N}{100} \\ 0 & otherwise \end{cases} \quad (20)$$

For example, PRED(25) = 80% means that 80% of the estimates are within 25 percent of the actual.

- The squared correlation coefficient (R^2), also known as the coefficient of determination is computed as Eq. (21)

$$R^2 = 1 - \frac{\sum_{i=1}^{T}(A_i - P_i)^2}{\sum_{i=1}^{T}(A_i - \overline{A})^2} \quad (21)$$

where $\overline{A}$ is the mean of actual effort values.

The higher the values of R^2 and PRED(*N*) are, the better the values of estimated results are. In contrast, the lower the values of MSE and MMRE are, the more accurate the values of estimated results are.

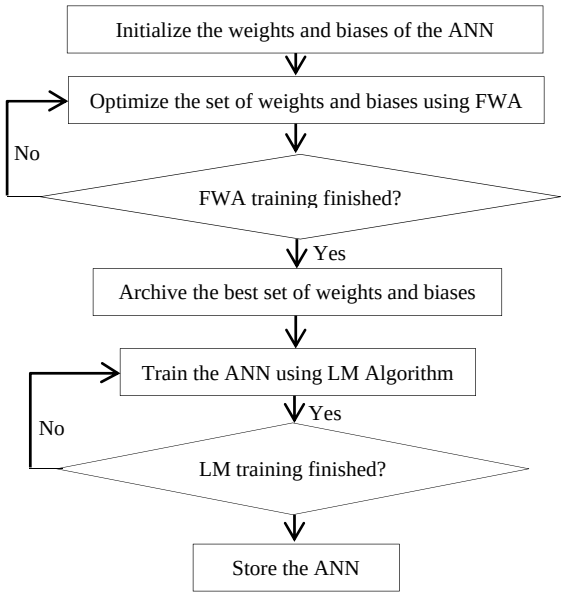


Fig. 2. The diagram of the hybrid FWA-LM Algorithm

B. Data sets

The proposed method is validated using the data set of twenty-one projects developed by six software companies as presented in Zia's work [21]. The data set is three-dimensional. The first dimension is the number story points required to accomplish the project, the second one shows the velocity of the project, and the third presents the actual effort required to complete that project. Zia *et al.* used this data for predicting effort using the linear regression analysis. In this study, the first fifteen projects were utilized to train the ANN and the others were used for testing the proposed approach.

C. Evaluating the performance of Fireworks Algorithm

In order to evaluate the effectiveness of the FWA, the

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

combination of LM with the other algorithms including directed artificial bee colony (DABC) algorithm [27], which is an improved version of artificial bee colony (ABC) algorithm, Teaching-Learning based optimization (TLBO) [27], Teaching-learning based artificial bee colony (TLBABC) [29] are compared to the proposed algorithm. LM algorithm without using optimization algorithms is also used to compare to the hybrid FWA-LM approach. In this case, the number of epochs for the training process using only LM is equal to the sum of the number of cycles of the optimization algorithm and the number of epochs of the LM algorithm. Table IV shows the average results of the experiment after 10 execution times.

TABLE IV. THE EXPERIMENTAL RESULTS OF ALGORITHMS

Effort	Velocity	Time	FWA-LM	TLBO-LM	DABC-LM	TLBABC-LM	LM
289	2.8	112	108.8	88.0	99.0	98.0	94.8
113	2.8	39	41.6	41.9	43.4	43.1	45.9
141	2.8	52	53.1	52.6	54.3	54.1	55.8
213	2.8	80	81.2	75.2	80.1	80.3	80.8
237	2.7	56	54.3	52.5	54.5	54.4	54.3
91	2.7	35	35.5	34.9	36.2	36.1	40.4
		MSE	**3.7983**	103.3450	32.8917	36.8467	65.0967
		MMRE (%)	**2.9339**	7.0923	5.5907	5.5710	9.9702
		PRED(7.19)	**100**	66.6667	66.6667	66.6667	33.3333
		R^2	**0.9946**	0.8530	0.9532	0.9476	0.9074

From Table IV, it can be seen that the hybrid FWA-LM algorithm gave the best results on all evaluation criteria compared to the TLBO-LM, DABC-LM, LBOABC-LM and LM algorithms. The value of PRED(7.19) is chosen according to the results of Zia's work [21]. It also finds that the testing errors using FWA-LM are relatively small. This experiment pointed out that the proposed method showed the effort estimation results quite accurate and the FWA is an effective algorithm to optimize the set of weights and biases of the ANN. It is also clear that if we only use the LM algorithm for the training process then the accuracy of estimated results is fairly low. This indicates the important role of optimization algorithms in enhancing the precision of the ANN.

D. Comparison of proposed approach with the other types of Neural Networks

In [21], Zia *et al.* used the regression method to estimate the effort of agile projects and results were computed on twenty-first projects. This study uses fifteen projects for the training process and six other projects for testing. In order to ensure comparability, we only report the results of Zia's work on six projects of our testing set. In [22], Panda *et al.* used different types of neural networks such as GRNN, PNN, GMDH Polynomial Neural Network and Cascade-Correlation Neural Network. The performance of our proposed method will be compared with these studies. Table V presents the comparison of obtained results using different types of ANN.

TABLE V. COMPARISON OF RESULTS USING DIFFERENT TYPES OF ANN

Name of metric	R^2	MMRE	PRED(%)
General Regression Neural Networks	0.7125	0.3581	85.9182
Probabilistic Neural Networks	0.6614	1.5776	87.6561
GMDH Polynomial Neural Network	0.6259	0.1563	89.6689
Cascade Correlation Neural Network	0.9303	0.1486	94.7649
Zia's Regression	0.9661	0.6634	50
Our method	**0.9946**	**0.0293**	**100**

The obtained results indicated that our method outperformed all kinds of ANN in Panda's work and the regression method of Zia. The experimental results pointed out that our proposed approach is a promising orientation in improving the accuracy of ANN and giving the right effort estimation results for agile projects.

VI. THREATS TO VALIDITY

This paper proposed how to apply ANN trained by using Fireworks algorithm in cooperation with LM for estimating the effort of the agile software development project.

Threats to construct validity insist on the way that the effort estimation models are defined. In this paper, the proposed model assumes that the value of initial project velocity is given by taking from the past projects developed by the same team in similar working conditions. However, when a team is new, the company will not have any past record of it. In that case, no obvious assignment to initial project velocity can be allocated. To solve this issue, we can use the average velocity values of all the teams working in similar conditions with the same size of the project and assign them to initial project velocity and then these data are utilized to train the ANN.

In our study, threats to external validity are related to the generalization to the other types of dataset. In this work, records of twenty-one projects developed by six software houses from the work of Zia *et al.* [16] are used without information with regard to the type of projects taken for research. To increase the persuasiveness, data covering all categories of software developed by agile methods should be collected and experimented.

The threat to internal validity might be caused by how the empirical study was conducted. Because of a small amount of dataset, the testing data is quite small in size with only six projects being used for testing. Therefore, the optimal precision of the performance of proposed method might not be guaranteed. Moreover, the ANN training approaches in this paper using FWA have the initial population being usually randomly generated, so the experiments might deliver various results. To cope with this problem, we carried out each experiment 10 times, and we followed rigorous statistical

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

procedures to evaluate the obtained results. Ten runs were a rule-of-thumb limit proposed by Ali *et al.* [17].

VII. Conclusion

The story point method is one of the software effort estimation techniques that can be employed for agile software projects. In this study, the total number of story points and the project velocity are utilized to train the ANN to give the effort involved in developing an agile software product. The accuracy of estimation results is enhanced by using the FWA and LM to optimize weights and biases of the ANN. The experimental results indicated that the FWA is an effective algorithm to improve the accuracy of the ANN in comparison with the other algorithms such as DABC, TLBO, and TLBABC. The proposed ANN also outperformed the other types of ANN in other studies like the General Regression, Probabilistic, GMDH Polynomial, and Cascade Correlation neural networks. This study is also expanded by using other machine learning techniques such as support vector machine, random forest, stochastic gradient boosting based on the story point approach.

References

[1] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, R. Selby, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," *Annals of Software Engineering*, vol. 1, no. 1, pp. 57-94, 1995.

[2] L. H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IEEE Transaction on Software Engineering*, vol. SE-4, no. 4, pp. 345-361, 1978.

[3] F. R. Freiman, R. D. Park, "PRICE software model-Version 3, An overview," in *Proc. of IEEE-PINY Workshop on quantitative Software Models*, 1979, pp. 32-41.

[4] S. M. Satapathy, B. P. Acharya, S. K. Rath, "Class Point Approach for Software Effort Estimation Using Stochastic Gradient Boosting Technique," *ACM SIGSOFT Software Engineering Notes*, pp. 1-6, 2014.

[5] C. Blum, K. Socha, "Training feed-forward neural networks with ant colony optimization: an application to pattern classification," in *Proc. of the Fifth International Conference on Hybrid Intelligent Systems*, 2005, pp. 233-238.

[6] R. Sexton, R. Dorsey, and J. Johnson, "Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing," *European Journal of Operational Research*, vol. 114, pp. 589-601, 1999.

[7] R. Sexton, B. Alidaee, R. Dorsey, J. Johnson, "Global optimization for artificial neural networks: a tabu search application," *European Journal of Operational Research*, vol. 106, pp. 570-584, 1998.

[8] C. Ozturk, D. Karaboga, "Hybrid Artificial Bee Colony algorithm for neural network training," in *IEEE Congress of Evolutionary Computation*, 2011, pp. 84-88.

[9] T. Back, H. P. Schwefel, "An overview of evolutionary algorithms," *Evolutionary Computation*, vol. 1, no. 1, pp. 1-23, 1993.

[10] D. Karaboga, C. Ozturk, "Neural networks training by artificial bee colony algorithm on pattern classification," *Neural Nerwork World*, vol. 19, no. 3, pp. 279-292, 2009.

[11] S. Mirjalili, S. Z. M Hashim, H. M. Sardroudi, "Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 11125–11137, 2012.

[12] Y. Tan, Y. Zhu, "Fireworks Algorithm for Optimization," in *Proc. of the First International Conference Advances in Swarm Intelligence*, 2010, pp. 355-364.

[13] S. Keaveney, K. Conboy, "Cost estimation in agile development projects," in *Proceedings of the Fourteenth European Conference on Information Systems*, Göteborg, Sweden, 2006, pp. 183-197.

[14] E. Coelho, A. Basu, "Effort Estimation in Agile Software Development using Story Points," *International Journal of Applied Information Systems*, vol. 3, no. 7, pp. 7-10, 2012.

[15] P. Abrahamsson, J. Koskela, "Extreme Programming: A Survey of Empirical Data from a Controlled Case Study," in *Proceedings of International Symposium on Empirical Software Engineering*, 2004, pp. 73-82.

[16] I. Hussain, L. Kosseim, O. Ormandjieva, "Approximation of COSMIC functional size to support early effort estimation in Agile," *Data & Knowledge Engineering*, vol. 85, pp. 2-14, 2013.

[17] R. Popli, N. Chauhan, "Cost and effort estimation in agile software development," in *International Conference on Optimization, Reliabilty, and Information Technology*, 2014, pp. 57-61.

[18] A. E. D. Hamouda, "Using Agile Story Points as an Estimation Technique in CMMI Organizations," in *Agile Conference (AGILE)*, Kissimmee, Finland, 2014, pp. 16-23.

[19] A. L. Oliveira, "Estimation of software project effort with support vector regression," *Neurocomputing*, vol. 69, no. 13, pp. 1749–1753, 2006.

[20] S. M. Satapathy, A. Panda, S. K. Rath, "Story Point Approach based Agile Software Effort Estimation using Various SVR Kernel Methods," in *International Conference on Software Engineering and Knowledge Engineering*, Vancouver, Canada, 2014, pp. 304-307.

[21] Z. K. Zia, S. K. Tipu, S. K. Zia, "An Effort Estimation Model for Agile Software Development," *Advances in Computer Science and its Applications*, vol. 2, no. 1, pp. 314-324, 2012.

[22] A. Panda, S. M. Satapathy, S. K. Rath, "Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points," *Procedia Computer Science*, vol. 25, no. 2015, pp. 772-781, 2015.

[23] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303-314, 1989.

[24] A. Omidi, E. Nourani, M. Jalili, "Forecasting stock prices using financial data mining and Neural Network," in *Proc. of the 3rd International Conference on Computer Research and Development*, 2011, pp. 242-246.

[25] A. Adebiyi, C. Ayo, M. O. Adebiyi, S. Otokiti, "Stock Price Prediction using Neural Network with Hybridized Market Indicators," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 1, pp. 1-9, 2012.

[26] M. Hagan, M. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989 - 993, 1994.

[27] M. S. Kiran, O. Findik, "A directed artificial bee colony algorithm," *Applied Soft Computing*, vol. 26, pp. 454-462, 2015.

[28] R. V. Rao, V. Patel, D. P. Vakharia, "Teaching-Learning-Based Optimization: An optimization method for continuous non-linear large scale problems," *Information Sciences*, vol. 183, pp. 1-15, 2012.

[29] T. T. Khuat, M. H. Le, "Applying Teaching-Learning to Artificial Bee Colony for Parameter Optimization of Software Effort Estimation Model," *Journal of Engineering Science and Technology*, (Accepted), 2016.

[30] S. Ali, L. Briand, H. Hemmati, R. Panesar-Walawege, "A systematic review of the application and empirical investigation of search-based test case generation," *IEEE Transactions on Software Engineering*, vol. 36, no. 6, pp. 742-762, 2010.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Thanh Tung Khuat completed the B.S degree in Software Engineering from University of Science and Technology, Danang, Vietnam, in 2014. Currently, he is participating in the research team at DATIC Laboratory, University of Science and Technology, Danang. His research interests include software engineering, software testing, evolutionary computation, Intelligent Optimization Techniques and Applications in software engineering.

My Hanh Le is currently a lecturer of the Information Technology Faculty, University of Science and Technology, Danang, Vietnam. She gained M.Sc. degree in 2004 and Ph.D. degree in Computer Science from The University of Danang in 2016. Her research interests are about software testing, nature-inspired algorithms and more generally application of heuristic techniques to problems in software engineering.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500