A Project Report on

Optimizing Software Effort Estimation Models Using Enhancement of Firefly Algorithm

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology

In

Computer Science and Engineering

Submitted by

G. Kavya
(21H55A0506)

K. Suresh
(21H55A0511)

Under the esteemed guidance of

Dr. V. Venkataiah

(**Associate Professor**)

Department of Computer Science and Engineering

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC Autonomous)

*Approved by AICTE *Affiliated to JNTUH *NAAC Accredited with A$^+$ Grade

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2020- 2024

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Major Project report entitled **"Optimizing Software Effort Estimation Models Using Enhancement of Firefly Algorithm "** being submitted by G. Kavya(21H55A0506), K. Suresh(21H55A0511) in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out his/her under my guidance and supervision.

The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

Dr. V. Venkataiah	**Dr. Siva Skandha Sanagala**	**EXTERNAL EXAMINER**
Associate Professor &	**Associate Professor and HOD**	
Additional Controller of Examination	**Dept. of CSE**	
Dept. of CSE		

ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express our heartfelt gratitude to all the people who helped in making this project a grand success.

We are grateful to **Dr . V. Venkataiah ,** Associate Professor & Additional Controller of Examination, Department of Computer Science and Engineering for his valuable technical suggestions and guidance during the execution of this project work.

We would like to thank**, Dr. Siva Skandha Sanagala,** Head of the Department of Computer Science and Engineering, CMR College of Engineering and Technology, who is the major driving forces to complete our project work successfully.

We are very grateful to **Dr. Ghanta Devadasu**, Dean-Academics, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Major Dr. V A Narayana,** Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the **Teaching & Non- teaching** staff of Department of Computer Science and Engineering for their co-operation

We express our sincere thanks to **Shri. Ch. Gopal Reddy**, Secretary& Correspondent, CMR Group of Institutions, and **Shri Ch Abhinav Reddy**, CEO, CMR Group of Institutions for their continuous care and support.

Finally, we extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly or indirectly in completion of this project work.

G. Kavya	21H55A0506
K. Suresh	21H55A0511

TABLE OF CONTENTS

List of Figures

List of Tables

List of Graphs

ABSTRACT

Software development effort estimation is considered a fundamental task for software development life cycle as well as for managing project cost, time and quality. Therefore, accurate estimation is a substantial factor in projects success and reducing the risks. In recent years, software effort estimation has received a considerable amount of attention from researchers and became a challenge for software industry. In this project, enhancement of Firefly Algorithm is proposed as a metaheuristic optimization method for optimizing the parameters of three COCOMO-based models. These models include the basic COCOMO model and other two models proposed in the literature as extensions of the basic COCOMO model. The developed estimation models are evaluated using different evaluation metrics are Root Mean Square Error (RMSE) .

CHAPTER 1
INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 Introduction

Software time and effort development of estimation models are fire topic of the study over the three decades for the engineering of software community with systematic approach to the development, maintenance and retirement of software[1]. Computer program taken a toll estimation alludes to the expectations of the probable sum of effort, time, and the number of employees required to construct a program. The essential work of the venture advancement is to guarantee that the venture is completed with the objective: "high quality of computer program must be created with a moo taken a toll that's inside time and budget"[2]. The duties of venture improvement are arranging, organizing, staffing, coordinating, and controlling the exercises. Belittling computer program costs can have an inimical effect on the quality of conveyed program and hence a company's trade reputation and competitiveness. The primary vital component of computer program extends administration is compelling preparing of the improvement of the computer program which determines the assets required to total the extend effectively. The assets incorporate the number and ability level of the individuals, and the sum of computing resources[3]. The fetched of a program venture is directly relative to the individuals / assets required for the project[5]. Overestimation of computer program fetched, on other hand, can result in missed chances to use reserves in other referential ventures. The need for solid and accurate taken a toll expectations in program building is a continuous challenge[3]. Good estimation can improves performances of the company particularly in overseeing the venture plan, human asset assignment, fetched estimation, etc. Those points of interest can decrease the disappointment possibility or extend delay. The Fetched for a extend may be a function of numerous parameters. Measure may be an essential fetched figure in most models and can be measured utilizing code snippets (or) thousands of conveyed KDLOC or work focuses. The no. of models are advanced to set up the connection between Measure and Exertion for Computer program Exertion Estimation[6].Algorithmic models are derived from the statistical analysis of historical project data, for example, Constructive Cost Model (COCOMO) and Software Life Cycle Management (SLIM)No algorithmic techniques include Price-to-Win, Parkinson, expert judgment, and machine learning approaches.

Computer program taken a toll estimation strategies can broadly classified as algorithmic and non-algorithmic models. The algorithms-based models come from the factual examination of historical venture data, for illustration, Helpful Fetched Show (COCOMO) and Program Life Cycle Administration (SLIM)No algorithmic methods incorporate Price-to-Win, Parkinson, master judgment, machine learning approaches[7]. Machine learning is utilized to group together set of methods that embody a few of the characteristics of human intellect, for the case, foggy frameworks, similarity, regression trees, run the show acceptance neural systems, and Developmental algorithms[3]. Among the machine learning approaches, foggy frameworks and neural systems, and Developmental calculations are regarded as have a place to the delicate computing group[11]. The algorithmic as well as the non-algorithmic (based on master judgment) taken a toll estimation models, be that as it may, are not without error. A few exertion estimation model have been develops and advanced over time superior forecast precision moreover, in this way for better improvement quality. The Such model range will be from complex calculations , statistical evaluation of the project's specifications to improve machine learning approaches[10]. Heuristic optimization is plan are relies on few attempt to find the best solutions. Heuristic optimizers are have used in the software cost estimation, such as to application of the genetic programming an model optimizations. Another case is the portion that the PSO is taken as an optimizer using heuristics. Besides, the crossover methods include the combination of heuristic algorithms just like the utilization of Hereditary Algorithm , Ant Colony[12]. In spite of an outsized number tests on finding foremost satisfactory forecast show, there's not clear prove of an extremely precise and effective strategy. At same time, it is significant to create a forecasting strategy that's fewer intricate and very extra valuable. It could become more productive on construct a demonstrate with a least number of the variables, hopefully finding foremost vital and the common factors bland project development efforts[10].By using the advantages of CS and HS, this paper proposes a hybrid model to estimate the optimal coefficients of software effort and software time development based on cuckoo search and harmony search (CSHS) algorithm by using NASA 93 dataset.

1.2. Problem Statement

Effort estimation in software development is a critical challenge faced by the software engineering community. Reliable estimation is fundamental for project scheduling, resource allocation, cost estimation, and minimizing the risk of project failures or delays. Despite its importance, a significant number of software projects experience effort or schedule overruns, leading to project failures. Research surveys indicate that inaccurate estimation models are a primary cause of these overruns. The complexity of software projects often leads to vagueness in the early stages, making accurate estimation difficult. Furthermore, each project possesses unique characteristics, making it even harder to estimate the required effort for completion. Existing efforts to address this challenge suggest the development of adaptable estimation models capable of accommodating a wide range of project types. However, the uncertain and variable nature of software projects, coupled with small and often incomplete datasets, intensify the complexity of prediction tasks.

1.3. Research Objective

The primary objective of this study is to explore the effectiveness of the Firefly Algorithm as a predictor for generic software effort estimation models. The research aims to validate the suitability of the Firefly Algorithm in enhancing the accuracy of software effort predictions while addressing the challenges posed by the vague and diverse nature of software projects. The study endeavours to demonstrate substantial performance improvements over existing methods, establishing the Firefly Algorithm as a robust tool for efficient effort estimation. Furthermore, the research focuses on the feasibility of employing machine learning approaches with a minimal set of input variables and dataset instances, emphasizing simplicity and efficiency in the prediction process. Through rigorous experimentation and analysis, this project aims to contribute valuable insights to the field of software engineering by advancing the development of reliable, adaptable, and streamlined software effort estimation techniques.

1.4. Project Scope and Limitations

The project, titled "Effort Estimation Using Firefly Algorithm," ambitiously aims to redefine software effort estimation methodologies. Focusing on the adaptability of the Firefly Algorithm, the scope encompasses validating its effectiveness across various project types. The study will rigorously evaluate the algorithm's predictive capabilities, emphasizing its utility in generating precise estimations with fewer variables. By optimizing the prediction process, the project aims to streamline project activities, enhance resource allocation, and minimize the risk of overruns. The research also seeks to establish a framework for future software effort estimation endeavors, emphasizing simplicity and real-world applicability. Through extensive experimentation and analysis, the project strives to contribute significantly to the field of software engineering, fostering more reliable project planning and decision-making processes.

- **Limited Dataset:** The project may face limitations due to the availability of a limited dataset, potentially restricting the algorithm's adaptability to a wide range of software projects.
- **Algorithm Complexity:** The complexity of the Firefly Algorithm might pose challenges for implementation, requiring a deep understanding of its intricacies and potential difficulties in fine-tuning.
- **Dependency on Initial Parameters:** The performance of the algorithm could be sensitive to the selection of initial parameters, making it essential to choose suitable values for accurate predictions.
- **Interpretability:** The results generated by the Firefly Algorithm might be difficult to interpret, hindering the understanding of how specific estimations are derived.
- **Algorithm Convergence:** Ensuring the algorithm converges efficiently for various project types and sizes might be a challenge, impacting the reliability of the estimation process.

CHAPTER 2
BACKGROUND WORK

CHAPTER 2
BACKGROUND WORK

2.1. A Baseline Model for Software Effort Estimation

2.1.1 Introduction

Software effort estimation (SEE) remains a paramount challenge in the domain of software engineering, with the objective of creating robust, accurate predictive cost models dating back to seminal works by Albrecht, Gaffney, and Boehm in the early 1980s[2]. The intricate nature of software development, marked by diverse projects, inherent uncertainties, and intricate interactions of variables, has rendered the task of accurate estimation exceptionally daunting [Albrecht and Gaffney 1983; Boehm 1981; Nelson 1966][1]. The search for a universal estimation method is further complicated by the dynamic, nonstationary characteristics of the software development environment [Collopy 2007; Menzies et al. 2006].

Over the years, researchers have explored myriad methodologies, from traditional linear models to sophisticated machine learning techniques, in pursuit of accurate SEE. The application of machine learning methods was anticipated to offer solutions to the complexities; however, the results have been highly variable and often difficult to interpret [Myrtveit and Stensrud 2012; Kitchenham and Mendes 2009][3]. The choice of models, influenced by dataset peculiarities and transformation methods, has led to conflicting conclusions [Dejaeger et al. 2012; Mair et al. 2000].

This project focuses on establishing a reliable baseline for SEE by revisiting Multiple Linear Regression (MLR), a method that had previously fallen out of favor in light of more complex machine learning approaches[4]. We contend that with meticulous data-driven transformations, MLR can be revitalized as a potent tool for SEE, especially when addressing issues such as non-normality and categorical data handling. While machine learning models have shown promise, their success is heavily contingent on data suitability, often leading to divergent conclusions.

2.1.2. Merits & Demerits

Merits:

- Assess the ease with which the model can be replicated. Provide clear instructions and code snippets for implementing the baseline model. Consider metrics like ease of understanding the implementation and availability of resources.
- Conduct a comparative analysis with other SEE models, showcasing how the baseline model serves as a standard benchmark. Compare its performance metrics with those of other models to demonstrate its utility as a reference point.
- Measure the percentage improvement in prediction accuracy achieved by the baseline linear model compared to existing methods. This can be calculated using metrics such as Mean Absolute Error (MAE) or Root Mean Square Error (RMSE).
- Evaluate the simplicity of the baseline model by comparing the number of input variables and computational complexity with other existing models. A lower number of variables and computational steps often indicate efficiency.

Demerits

- Linear models, including the baseline, might struggle with capturing highly nonlinear relationships in complex software projects. It may not be suitable for projects where variables interact in intricate and nonlinear ways.
- The accuracy of the baseline model heavily relies on the quality and representativeness of the training data. If the dataset used for training is biased, incomplete, or unrepresentative of real-world scenarios, the model's estimations could be inaccurate.
- Linear models are generally not adept at handling unstructured data such as text, images, or audio. If your software effort estimation relies heavily on unstructured data, the baseline model might not be the best choice.
- Linear regression models can be sensitive to outliers in the dataset. Outliers can significantly impact the coefficients of the model, leading to skewed predictions. Robust techniques may be required to handle outliers effectively.

2.1.3 Implementation

This section will give two examples using the baseline implementation ATLM. The intention is to show that ATLM satisfies the properties stated in Section 2. In terms of training and testing a model, there are a number of different approaches commonly used in SEE. Menzies et al. [2006][3] argue that a small test set size should be used because “that is how they will be assessed in practice.” Some studies have used a test set size of 10, including Wittig and Finnie [1997], Mair et al. [2000, 2006], probably because this was used in an early work on effort prediction [Adrangi and Harrison 1987][8]. Other studies have used a test size of 1 (i.e., leave-one-out cross-validation), including Samson et al. [1997], Shepperd and Schofield [1997], Kitchenham et al. [2002], and, in particular, Myrtveit et al. [2005] who describe this as the “real-world situation.” Recently there have been arguments presented for leave-one-out as the most appropriate validation method [Kocaguneli and Menzies 2013][9], although this recommendation has not yet influenced the field. Since there is no agreed standard method for assessing the quality of a model in SEE, the baseline implementation supports leave-one-out cross-validation, n-way cross-validation, and a sampling of any size for repeated training/test set measurements. Since we cannot anticipate the total range of error measurements of training/test regimes, the baseline model R code offers a simple interface taking a single training and test set. All runs of ATLM return the predicted and actual values as a table that can be exported for further analysis if required[7].

Ex 1: Randomized Training/Test Sets

The paper by Minku and Yao [2013] examined the concept of ensemble methods for software effort estimation. Their paper used a number of datasets and measured performance by randomly selecting, without replacement, 10 examples for testing with the remaining data used for training, as suggested by Menzies et al. [2006][5]. This was repeated 30 times to produce an overall estimate of accuracy. The error measurements MMRE, LSD, and PRED(25) were used to assess model quality. We calculate and report RE$*$ for ATLM but, as we do not have the residuals for the ensemble method or ANN methods, we cannot calculate RE$*$ for these models. To enable comparison we therefore also report LSD, MMRE, and PRED(25)[8].

ATLM was used to predict effort on three of the datasets analyzed by Minku and Yao [2013][6], that is, the Cocomo81,Desharnais, and Org All datasets, to demonstrate that the proposed baseline model produced predictions of comparable quality to those of the complex methods proposed in this article (Table II). The Cocomo81 dataset [Menzies et al. 2012][9] used 16 numerical features: 15 cost drivers and lines of code; in addition, the categorical development type (Mode) parameter was used as input. Following the default approach of "R" and that of Minku and Yao [2013], this categorical feature was handled using the approach of dummy variables [James et al. 2013]. In addition, the Language Type (Development Type) variable, coded as an integer value 1, 2, or 3 representing the language types Cobol, Advanced Cobol, and 4GL, was converted to a factor so that it was handled appropriately by ATLM[4]. It is worth noting that the Non-Adjusted Points variable was included in the model even though it is a linear combination of the two variables Transactions and Entities. Although this is not good practice when constructing a linear model, this redundant feature was included so that a direct comparison using the same data and variables could be made. The model predicted actual effort in person hours[5].

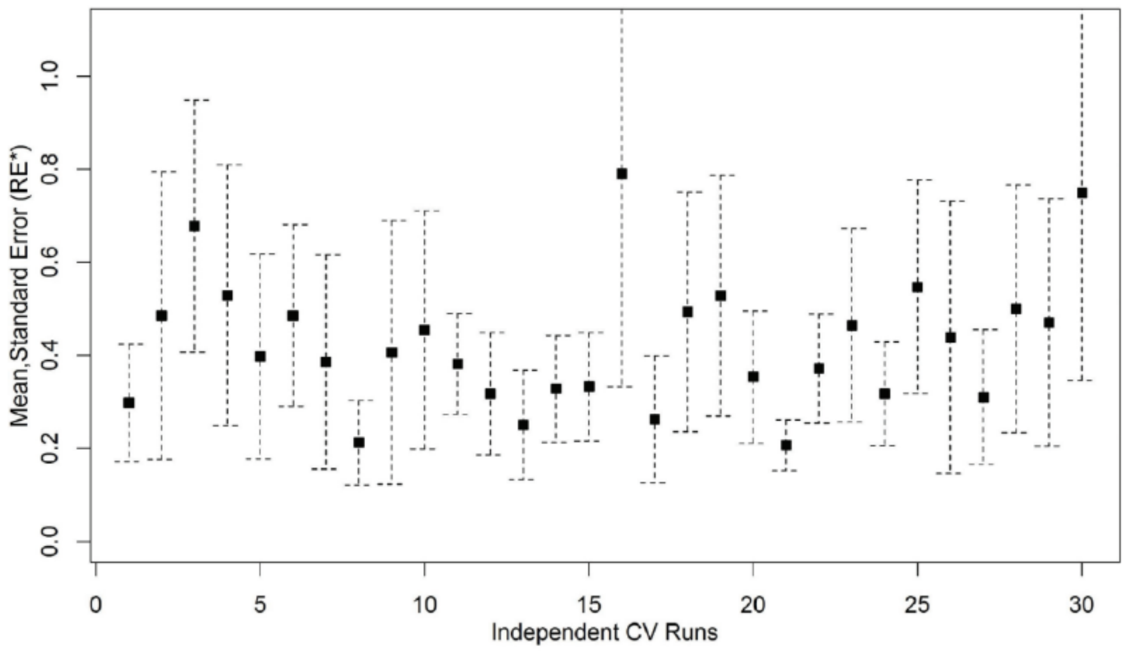


Fig :1 - Variation of RE* between runs of tenfold cross-validation using ALTM

2.2. A Hybrid Model for Estimating Software Project Effort from Use Case Points

2.2.1. Introduction

In the dynamic realm of software engineering, the accurate estimation of software project effort is pivotal for effective project management. Traditional methods such as Use Case Points (UCP) have long served as metrics, yet persisting challenges arise from the arbitrary nature of these numbers and the intricate conversion of UCP size into tangible effort metrics[4]. Recognizing these complexities, this project introduces a revolutionary approach: the Hybrid Model. By merging established techniques from clustering and classification, specifically bisecting k-medoids clustering and support vector machine, with the precision of Radial Basis Neural Network (RBFNN), our model delves deep into the intricate relationship between UCP, productivity, and environmental factors[6].

What sets our model apart is its ability to not only learn from historical data but also dynamically adapt to the ever-changing software landscape. This adaptability offers a flexible and adjustable productivity prediction mechanism, breaking free from the constraints of static estimation models[8]. In essence, our research seeks to redefine the paradigm of software effort estimation. Our goal is not merely accuracy but a fundamental shift in how this critical task is approached. We strive for estimations that are not just precise but also adaptable, providing invaluable insights during the pivotal stages of project feasibility and inception[2].

Through the integration of innovative methodologies and advanced algorithms, this project signifies a significant leap toward more reliable software project effort estimation practices[10]. By embracing the power of hybrid models, we aim to empower project managers and software engineers with a tool that not only understands the nuances of their projects but also evolves with them, ensuring a more strategic, data-driven, and ultimately successful approach to software development endeavors[8].

2.2.2. Merits & Demerits

Merits

- **Improved Accuracy:** Enhanced precision in software effort estimation through intricate relationship modeling.
- **Adaptability:** Dynamic adjustments tailored to project-specific attributes, ensuring flexible estimations.
- **Dynamic Productivity Learning:** Continuous refinement of predictions based on historical data and project complexities.
- **Incorporation of Environmental Factors:** Comprehensive insights by considering team workload and project dynamics.
- **Advanced Algorithm Integration:** Utilization of cutting-edge algorithms for superior predictive capabilities.
- **Managerial Insight:** Empowering decision-making with informed resource allocation and project planning.

Demerits

- **Complex Implementation:** Advanced algorithms may require complex implementation and specialized knowledge.
- **Dependency on Historical Data:** Accuracy heavily relies on the availability and quality of historical data, limiting applicability.
- **Sensitivity to Initial Parameters:** Model performance may be affected by the sensitivity of initial setup parameters.
- **Difficulty in Interpreting Results:** Complex algorithms might produce results that are challenging to interpret and validate.
- **Resource Intensiveness:** High computational resources needed for clustering and neural network training may pose challenges.
- **Limited Generalization:** Model's adaptability might be restricted when applied to vastly different project domains.

2.2.3 Implementation

In this section we discuss the implementation of the proposed effort estimation model. This paper mainly focuses on two important issues related to effort estimation based upon UCP[3]. The first issue is related to predicting productivity from eight environmental factors ($E1$, $E2$, … , $E8$), and the second issue is to estimate effort from the predicted productivity and UCP. Therefore, the proposed prediction model is a hybrid model that consists of two stages to cope with the abovementioned issues[6]. As mentioned in Section 2, there is a strong relationship between environmental factors and project productivity because each factor has a significant impact on the project productivity. For example, the first factor, which is "Familiar with Rational Unified Process (RUP)," evaluates the familiarity with RUP, so higher experience with RUP leads to better productivity and less effort. Another example is the "Lead Analyst capability" factor that reflects the capability and knowledge of the developer, so better capability means better productivity[8]. Therefore, we believe that all environmental factors could be good indicators of project productivity. Next we will discuss the construction of the proposed model during training and testing stages[3].

2.2.3.A. Training Phase

During the training phase we construct two prediction models: one for productivity and one for effort. In the next subsections we will discuss the implementation of both models.

2.2.3.B. Productivity Prediction

The eight environmental factors (*E1*, *E2*, …, *E8*) are used to classify and predict productivity. Since each environmental factor exhibits the ordinal scale (i.e., each factor can take a distinct value between 0 and 5), we decided to build a classification model[9]. But first we need to convert productivity variable from a numerical scale to the nominal scale, which represents levels of productivity[10]. To accomplish that we applied a clustering technique on the productivity to create coherent and finer labels. The unsupervised learning method called k-medoids has been used for that goal. The k-medoids is a centroid-based clustering method that divides data into C distinct groups[3]. The major distinction between k-medoids and conventional k-means is the choice of cluster canters.

In k-means method the center is calculated based on the average of instances within a class, but in k-medoid the center is identified based on the most centrally located instance in the cluster[8]. The popularity of making use of k-medoids clustering comes from its ability to use arbitrary dissimilarity or distance functions, which also makes it an appealing choice of a clustering method for software effort data as software effort datasets often exhibit very dissimilar characteristics.

2.2.3.C. Effort Prediction

The effort estimation model is constructed using RBFNN. The inputs for this model are UCP size measure and productivity of training projects. It is important to note here that the productivity variable is the actual numerical values, not the labels that are used in the previous step[6]. The RBFNN is a feed-forward network which consists of three layers: input, hidden, and output.The neurons in the hidden layer include a non-linear activation function(mainly a Gaussian function), whereas the output layer includes linear function[10].

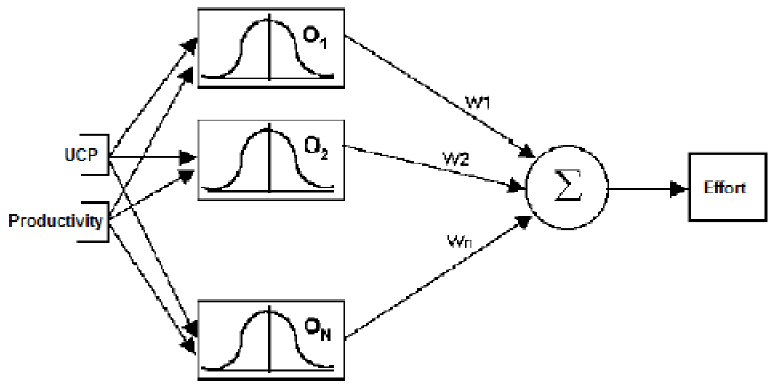


Fig 2 Radial Basis Function Neural Network

2.3. Software Effort Estimation using Machine Learning Technique

2.3.1. Introduction

In the ever-evolving landscape of software development, accurate effort estimation emerges as a linchpin for project success[8]. Yet, the traditional methods once relied upon are now challenged by the swift currents of changing work dynamics and technological progress. This project embarks on a transformative journey into the heart of software engineering: Software Effort Estimation. Employing cutting-edge Machine Learning techniques, notably Support Vector Machines (SVM), K-Nearest Neighbor (KNN), and Decision Trees (DT), this study pioneers an innovative approach that promises to revolutionize the field[6].

At its core, this research introduces a groundbreaking dataset meticulously tailored to the post-COVID hybrid work environment and the dynamic demands of contemporary businesses[7]. This dataset serves as the bedrock for our exploration into predicting software efforts with unprecedented accuracy. Notably, this project stands as a trailblazer, marking the inaugural instance of employing such a specialized dataset for software development predictions[2]. The fusion of advanced Machine Learning algorithms with this unique dataset unveils a new era in software effort estimation.

In the upcoming sections, we delve into the intricacies of existing effort estimation methodologies, providing a comprehensive understanding of the challenges faced by the industry[5]. We detail our meticulously crafted research methodology, designed to harness the full potential of Support Vector Machines, K-Nearest Neighbor, and Decision Trees. Through rigorous evaluation, we demonstrate the superior efficacy of these Machine Learning models, showcasing their ability to accurately predict future software efforts in the contemporary landscape[3]. This project does not merely seek to refine existing practices; it aims to redefine the very essence of how software effort estimation is approached. By bridging the gap between traditional methods and the exigencies of the modern software industry, this pioneering effort heralds a new era where precision, adaptability, and innovation harmoniously converge, promising a brighter, more efficient future for software development endeavors[10].

2.3.2. Merits & Demerits

Merits

- **Innovative Dataset:** Tailored to modern work dynamics, capturing remote work nuances and changing client demands.
- **Machine Learning Precision:** Utilizes advanced algorithms for accurate and data-driven software effort predictions.
- **Pioneering Approach:** First-of-its-kind utilization of innovative dataset, setting new standards in effort estimation.
- **Adaptability:** Techniques accommodate industry evolution, reflecting changes in technology and market demands.
- **Real-World Application:** Provides valuable insights for project planning, resource allocation, and risk management.
- **Enhanced Project Management:** Equips managers with tools for efficient project execution and risk mitigation.

Demerits

- **Limited Historical Data:** Dependency on a new dataset might lack extensive historical context, affecting prediction accuracy.
- **Algorithm Sensitivity:** Performance could be influenced by hyperparameters and algorithm sensitivity, demanding careful tuning.
- **Resource Intensive:** Utilization of complex Machine Learning models may require substantial computational resources.
- **Data Quality Concerns:** Inadequate or biased data might compromise the reliability of predictions, leading to inaccuracies.
- **Interpretability Challenges:** Advanced algorithms might produce results difficult to interpret, posing challenges for non-experts.
- **Dependency on External Factors:** Accuracy might be influenced by external variables, such as market fluctuations, beyond the project's control.

2.3.3. Implementation

There has been extensive study into the use of machine learning (ML) based prediction methods for software development effort estimation to improve predictions[4]. The goal of this machine learning method is to minimize the loss function while simultaneously optimizing the support vector boundaries by transferring non-linear separable patterns in the input into higher feature space[8]. The methodology of software effort estimation. Three common machine-learning techniques are described below.

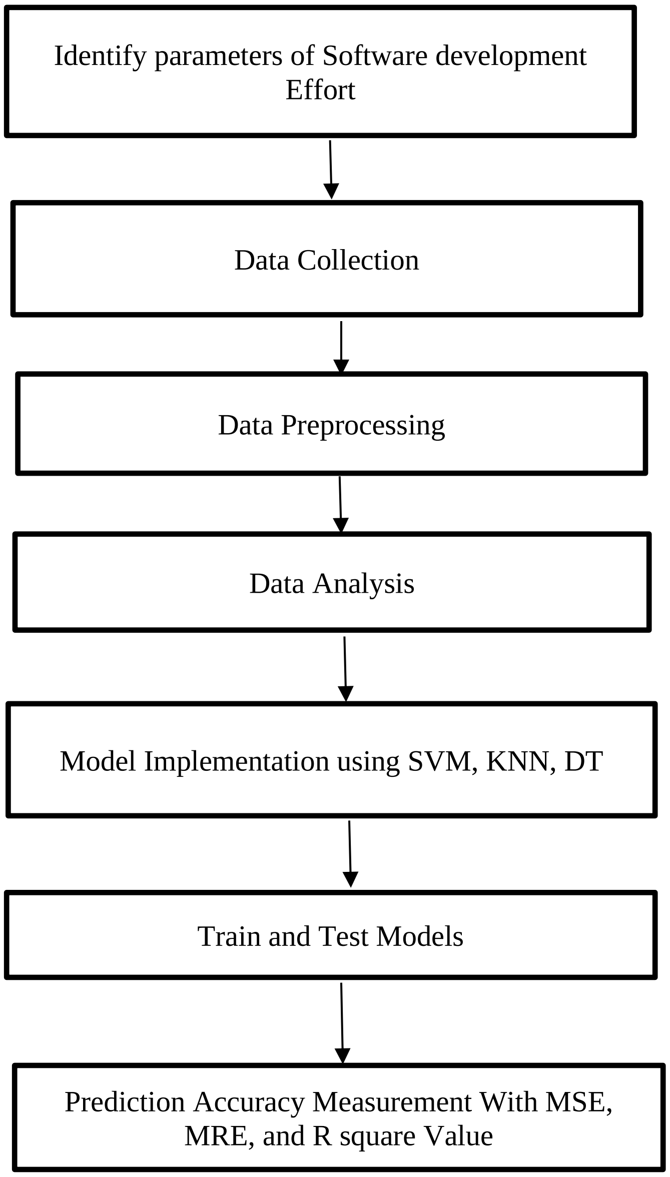

Identify parameters of Software development Effort
Data Collection
Data Preprocessing
Data Analysis
Model Implementation using SVM, KNN, DT
Train and Test Models
Prediction Accuracy Measurement With MSE, MRE, and R square Value

Fig 3 Flow chart of Methodology of SEE

2.3.3.A. Support Vector Regression (SVR)

A support vector machine (SVM) handles classification and regression issues. In machine learning, a classifier is a model built to make inferences about a class from additional features. The term “classification” refers to the act of labelling an unlabelled record with a unique value An SVM variant is an SVR[6]. The regression issues are transformed into classification issues. A linear decision surface (known as a hyperplane) divides two segments of vectors in the SVM training process. The margin between the LDS and the vectors that are nearest to it is maximum in this separation. They are known as support vectors. This LDS has the best generalization of all speculative hyperplanes[8]. Using this ideal hyperplane, classification or regression models’ predictive ability is improved.

$$f(x) = \omega^T\phi(x) + b \qquad \text{--(1)}$$

where x is the input vector, $\phi(x)$ is the feature mapping function, ω is the weight vector, and b is the bias term.

2.3.3.B. K-Nearest Neighbour Regression (KNN)

KNN, one of the most straightforward estimation techniques, was chosen for this study because of its perceived resemblance to human-based expert opinion. Technically, KNN does not train a model but rather uses Euclidean distance to calculate distances between locations[3]. The algorithm makes a prediction about the class to which a point belongs by gathering the nearest samples. Regression involves taking the average of the closest samples to a location to determine its value[8]. The effort of the target project is then estimated by averaging the efforts of the k projects that are the closest analogs.

$$\text{Euclidean} = \sqrt{\sum}(x_i-y_i)^2 \qquad \text{--(2)}$$

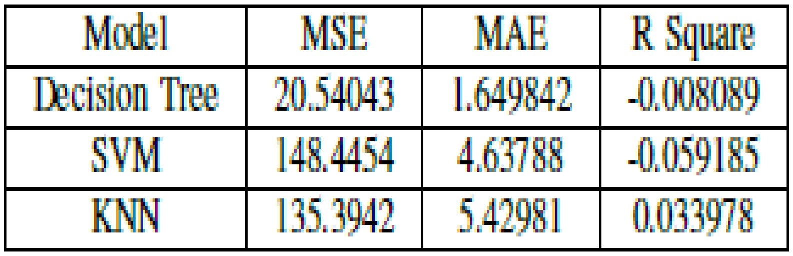
Where k is the user-defined constant, i is the number of instances x and y are the vectors of each instance

2.3.3.C. Decision Tree (DT)

To obtain insightful information that will help it achieve its objectives, data mining uses DT. DT is an intelligent model that looks like a binary tree with the root at the top that has been turned on its side[7]. The decision tree model is used to turn the data into a tree structure to help with the machine learning challenge[2]. DT serves as an example of forecasting a dependent variable using a set of predictor variables. The decision-making process in this model is analogous to other models, which facilitates understanding[4]. Because a picture is worth a thousand words, this approach makes it simple for anyone to comprehend the essence of a complex problem by simply looking at its schematic[9]. The method used by DT is comparable to how people make decisions. DT does have certain disadvantages, though. Contrasted with other machine learning models, the accuracy of the dataset predictions is lower. Because DTs provide a collection of if-then-else rules, they are easier to grasp and analyze when compared to other machine learning techniques like neural networks and Bayesian networks[1].

$$F(x) = \sum C_j I\,(x \in R_j) \qquad \text{--(3)}$$

where x is the input vector, f(x) is the predicted output label or value, J is the number of leaf nodes in the tree, R_j



Model	MSE	MAE	R Square
Decision Tree	20.54043	1.649842	-0.008089
SVM	148.4454	4.63788	-0.059185
KNN	135.3942	5.42981	0.033978

Table 1 Comparison of all models Effectiveness in terms of MSE, MAE, and R Square

CHAPTER 3
PROPOSED SYSTEM

CHAPTER 3
PROPOSED SYSTEM

3.1 Regression Methods

The goal of regression methods is to build a function f(x) that adequately maps a set of independent variables (X1, X2..., Xn) into a dependent variable Y. In our case, we aim to build regression models using a training dataset to use subsequently to predict the total effort for the development of software projects in man-months.

3.2 FIREFLY ALGORITHM:

Firefly Algorithm (FA) is a multimodal optimization algorithm, which belongs to the nature-inspired field, is inspired from the behaviour of fireflies or lightning bugs. FA was first introduced by Xin-She at Cambridge University in 2007. The Firefly algorithm offers a unique approach, optimizing the parameters of a model that relates various project factors to effort. This introduction sets the stage for exploring how the algorithm navigates the solution space, iteratively refining its estimates. In this pursuit, we delve into the key steps of defining variables, creating fitness functions, initializing fireflies, and optimizing parameters.

FA is empirically proven to tackle problems more naturally and has the potential to over-perform other metaheuristic algorithms. FA relies on three basic rules, the first implies that all fireflies are attracted to each other with disregard to gender. The second rule states that attractiveness is correlated with brightness or light emission such that bright flies attract less bright ones, and for absence of brighter flies the movement becomes random. The last main rule implies that the landscape of the objective function determines or affects the light emission of the fly, such that brightness is proportional to the objective function.

3.2.1. Pseudo code of FA

```
Begin
    1. Initialisation max iteration, $\alpha$ $\beta_0$, $\gamma$
    2. Generate initial population
    3. Define the Objective function f(x),
    4. Determine Intensity(I) at cost (x) of each individual determined by $f(x_i)$
    5. While (t < Iter max)
                For i= 1 to n
                    For j = 1 to n
                        If ($I_j > I_i$)
                            Move firefly i towards j in k dimension
                        End if
                      Evaluate new solutions and update light intensity
                    End for j
                  End for i
              Rank the fireflies and find the current best
          End while
    6. Post process result and visualization
End procedure
```

3.3 Design

3.3.1 Flow chart of Firefly Algorithm

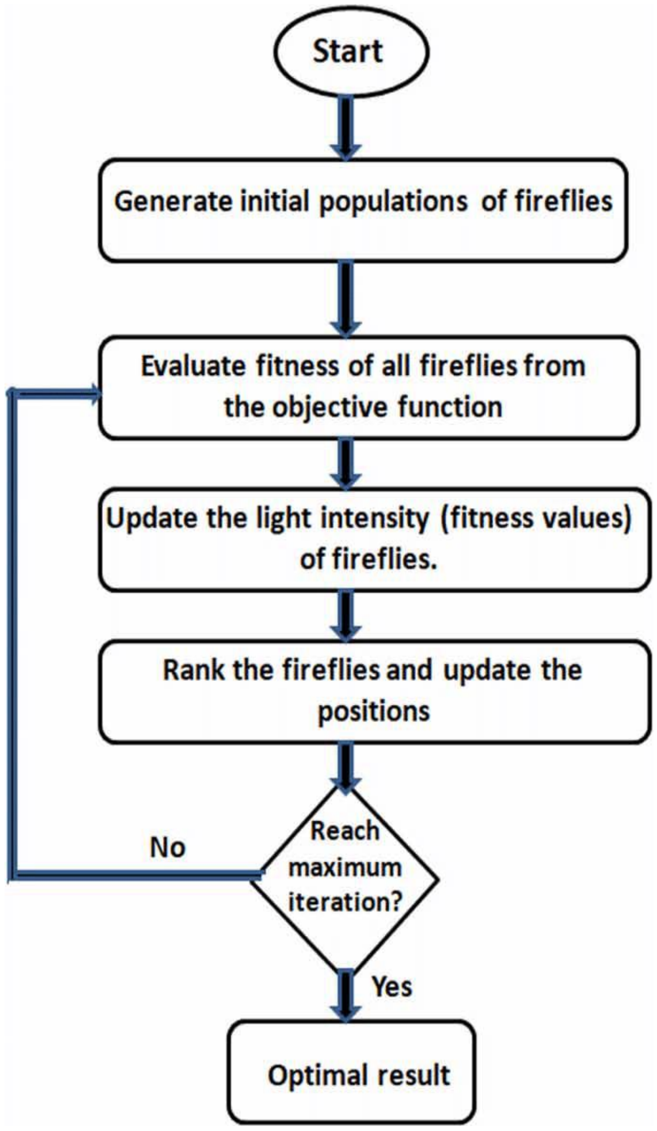

Start
Generate initial populations of fireflies
Evaluate fitness of all fireflies from the objective function
Update the light intensity (fitness values) of fireflies.
Rank the fireflies and update the positions
Reach maximum iteration?
No
Yes
Optimal result

Fig : 4 Flow chart of Firefly Algorithm

3.3.2 Architecture of Proposed System

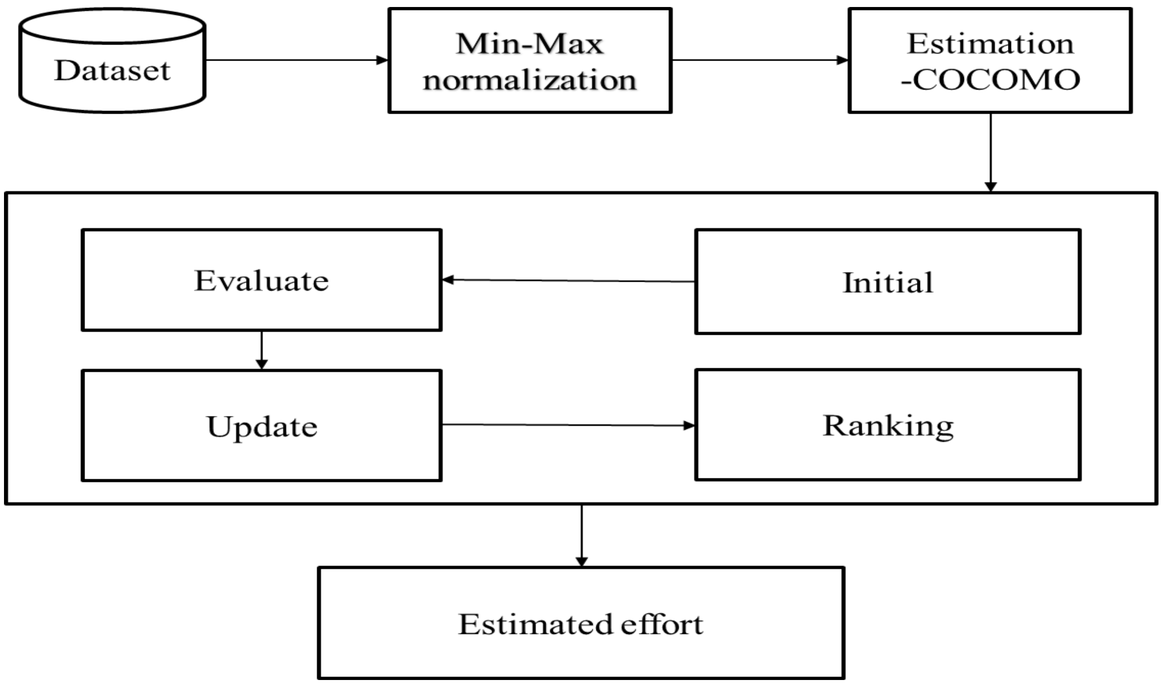


Fig:5 Architecture of Proposed System

Data set (COCOMO81): In this Paper we are taking dataset from the literature review (i.e COCOMO81 dataset).

3.3.2.1 Preprocessing:

MIN-MAX Normalization: In this process the Preprocessing of the dataset to Normalize the values in between 0 and 1. This can improve the performance of the Firefly Algorithm by making the data more uniform.

$$X_{new} = X - X_{min} / X_{max} - X_{min} \quad \text{--(4)}$$

3.3.2.2 Estimation Model:

Cocomo model: The Constructive Cost Model (COCOMO), It primarily relies on the size of the software, measured in Delivered Lines of Code (KLOC).

1. Basic COCOMO: This is the simplest model, suitable for organic mode projects with experienced staff and well-defined requirements. It uses the following formula:

*Effort = a * (KLOC^b) --(5)

3.3.3.3 Optimization Loop:

- Firefly Algorithm: Each firefly iteratively adjusts its position in the search space based on its brightness and the attractiveness of other fireflies. Brighter fireflies (those representing more accurate estimations based on the evaluation criteria) attract other fireflies, moving the population towards better solutions.
- Initialize Firefly Algorithm: Initializing firefly algorithm, which adjusts its position in the search apace based on brightness and attractiness of the firefly.
- Evaluate: The fitness of each firefly (how well its solution matches the actual effort values) is calculated using a fitness function.
- Update: The steps involve the representing of the weights of the every fireflies after calculating the fitness function.
- Ranking: Ranking is the process arranging the fireflies with the least values to get the optimal out of all the firefly weight.
- Optimal Result: The firefly with the best fitness value, representing the most accurate effort estimation for the new software project, is selected as the optimal result.

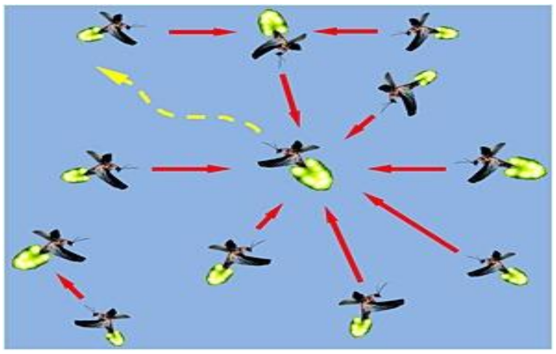
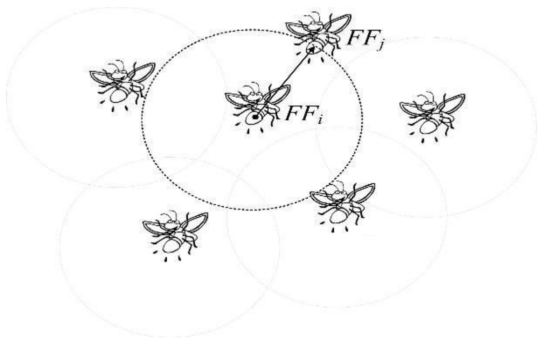



Fig:6 Examples of Fireflies

Firefly Algorithm (FA) is a multimodal optimization algorithm, which belongs to the nature-inspired field, is inspired from the behaviour of fireflies or lightning bugs. FA was first introduced by Xin-She at Cambridge University in 2007. FA is empirically proven to tackle problems more naturally and has the potential to over-perform other metaheuristic algorithms. FA relies on three basic rules, the first implies that all fireflies are attracted to each other with disregard to gender. The second rule states that attractiveness is correlated with brightness or light emission such that bright flies attract less bright ones, and for absence of brighter flies the movement becomes random. The last main rule implies that the landscape of the objective function determines or affects the light emission of the fly, such that brightness is proportional to the objective function.

The attractiveness among the flies in FA has two main issues that are; the modelling of attractiveness and the various light intensities. For a specific firefly at location X brightness I is formulated as I (X) α f(X) . While attractiveness β is proportional to the flies and is related to the distance R_{ij} , between fireflies i and j. the inverse square of light intensity I (r) in which I_0 represents the light intensity at the source.

$$I(r) = I_0^{-\gamma r2} \qquad \text{--(6)}$$

Assuming an absorption coefficient of the environment γ, intensity is represented in which I_0 is the original intensity.

$$I(r) = I_0 / 1 + \gamma\, r^2 \qquad \text{--(7)}$$

Generally, the Euclidean distance is illustrated, which represents the distance between a firefly at location X_i and another at location X_j. In which $X_{i\,k}$, is the kth component of the spatial coordinate X_i .

$$R_{ij} = \sqrt{}(X_1 - Y_1)^2 + (X_2 - Y_2)^2 \qquad \text{--(8)}$$

A firefly i attracted to a brighter one j as where attraction is represented by $\beta_0 e^{-\gamma r2} (X_j – X_i)$, and α (rand – ½) represents the randomness according to the randomization parameter α.

$$X_{new} = Xi + \beta_0 e^{-\gamma r2}(X_j – X_i) + \alpha\delta \text{ (rand} – ½) \qquad \text{--(9)}$$

Furthermore, variations of attractiveness are determined by γ which on its turn affects the behaviour and convergence speed of FA.

3.4. Implementation of code:

```
“import numpy as np
import random as rand
import math
import time
import pandas as pd
class FireflyAlgorithm:”
```

- ➢ import numpy as np: Imports the NumPy library and assigns it the alias np. NumPy is a powerful library for numerical computations in Python, especially when dealing with arrays and matrices.

- ➢ import random as rand: Imports the random module from Python's standard library and assigns it the alias rand. This module provides functions for generating random numbers.

- ➢ import math: Imports the math module from Python's standard library. This module provides mathematical functions and constants.

- ➢ import time: Imports the time module from Python's standard library. This module provides functions for working with time-related tasks.

- ➢ import pandas as pd: Imports the Pandas library and assigns it the alias pd. Pandas is a popular library for data manipulation and analysis in Python.

➢ The FireflyAlgorithm class is defined with the following attributes and methods:

```
"def __init__(self, D, Lb, Ub, n, alpha, beta0, gamma, theta,
iter_max,size,ae,func):
        self.D = D
        self.Lb = Lb
        self.Ub = Ub
        self.n = n
        self.alpha = alpha
        self.beta0 = beta0
        self.gamma = gamma
        self.theta = theta
        self.iter_max = iter_max
        self.func = func
        self.populationArray = np.zeros((n, D))
        self.functionArray = np.zeros(n)
        self.tmpArray = np.zeros(D)
        self.size=size
        self.ae=ae"
```

➢ Attributes:

- D: Dimensionality of the problem (number of decision variables).
- Lb: Lower bounds for the decision variables.
- Ub: Upper bounds for the decision variables.
- n: Number of fireflies (population size).
- alpha: Light absorption coefficient.
- beta0: Attraction coefficient at the initial distance.
- gamma: Light absorption coefficient (rate).
- theta: Randomness parameter.
- iter_max: Maximum number of iterations.
- func: The objective function to be optimized.
- populationArray: Array to store the positions of fireflies in the population.
- functionArray: Array to store the objective function values of fireflies.
- tmpArray: Temporary array for updating firefly positions.
- size: Size of the dataset.

- ae: Actual effort values from the dataset.

```
“def init_FA(self):
        for i in range(self.n):
            for j in range(self.D):
                self.populationArray[i][j] = rand.uniform(self.Lb, self.Ub)
            self.functionArray[i] = self.func(self.populationArray[i,:],
self.D,self.size,self.ae)

    def update(self, i, j):
        scale = self.Ub - self.Lb
        r = 0
        for k in range(self.D):
            r += (self.populationArray[i][k] - self.populationArray[j][k])**2
        beta = self.beta0*math.exp(-self.gamma*r)
        for k in range(self.D):
            steps = (self.alpha*self.theta)*(rand.random() - 0.5)*scale
            self.tmpArray[k] = self.populationArray[i][k] +
beta*(self.populationArray[j][k] - self.populationArray[i][k]) + steps
        if(self.func(self.tmpArray,self.D,self.size,self.ae) < self.functionArray[i]):
            for k in range(self.D):
                self.populationArray[i][k] = self.tmpArray[k]
            self.functionArray[i] = self.func(self.tmpArray, self.D,self.size,self.ae)

    def doRun(self):
        start = time.time()
        self.init_FA()
        for gen in range(self.iter_max):
            print("Generation ", gen+1)
            for i in range(self.n):
                for j in range(self.n):
                    if(self.functionArray[i] > self.functionArray[j] and i != j):
                        self.update(i,j)
            print(self.populationArray)
            print(self.functionArray)
        end = time.time()
        #print(" : %f " % (end - start))
        return self.functionArray.min()”
```

1. init_FA(self): This method initializes the firefly population and evaluates their objective function values. Here's what it does:
   - It iterates over each firefly in the population (self.n).
   - For each firefly, it iterates over each dimension (self.D) of the decision space.
   - For each dimension, it assigns a random value within the specified lower bound (self.Lb) and upper bound (self.Ub) using rand.uniform.
   - After initializing the position of each firefly, it evaluates the objective function value for that firefly using the provided objective function (self.func) and stores it in the self.functionArray.

2. update(self, i, j): This method updates the position of a firefly based on the attractiveness of another firefly. Here's what it does:
   - It calculates the distance r between fireflies i and j in the decision space.
   - It calculates the attractiveness beta based on the distance and the attraction coefficients self.beta0 and self.gamma.
   - It iterates over each dimension of the decision space and updates the position of firefly i using the formula of the Firefly Algorithm, which involves attraction towards firefly j and random movement.
   - If the new position results in a lower objective function value than the previous one, it updates the position and the corresponding objective function value in self.populationArray and self.functionArray, respectively.

3. doRun(self): This method executes the Firefly Algorithm optimization process. Here's what it does:
   - It starts a timer using time.time() to measure the execution time.
   - It initializes the firefly population and evaluates their objective function values using the init_FA() method.
   - It iterates over a specified number of generations (self.iter_max).
   - Within each generation, it iterates over each firefly (i) in the population and compares its objective function value with other fireflies (j).

- If the objective function value of firefly i is greater than that of firefly j and they are different fireflies (i != j), it updates the position of firefly i using the update() method.
- It prints the population and objective function values for each generation (you might want to remove or comment out these print statements for large problems to avoid cluttering the output).
- After completing the specified number of generations, it stops the timer and calculates the execution time.
- It returns the minimum objective function value found in the population (self.functionArray.min()).

```
“def CostEstimation(x,D,size,ae):
    #print(x)
    mre=[]
    pe=[]
    rsme=0
    for i in range(0,len(size)):
       pe.append(x[0]*(size[i])**x[1])
       mre.append((ae[i]-pe[i])**2)
    y=sum(mre)
    rsme=math.sqrt(y/len(size))
    return rsme”
```

The Cost Estimation function takes model parameters (x), dataset sizes (size), and actual effort values (ae). It computes the predicted effort for each dataset size using the model equation specified by x, calculates the squared errors between predicted and actual effort values, and then returns the Root Mean Squared Error (RMSE) as the output. This function is likely used as the objective function in optimization tasks to minimize the difference between predicted and actual effort values.

```
“file_path =
"C:/Users/ketha/Desktop/Projects/major/code/dataset/albrecht2.csv"
df=pd.read_csv(file_path)
size=df["size"]
ae=df["Effort"]
#FireflyAlgorithm(D, Lb, Ub, n, alpha, beta0, gamma, theta, iter_max, func)
FA = FireflyAlgorithm(2, -5, 5, 63, 0.1, 1.0, 0.01, 0.97,
10,size,ae,CostEstimation)
ans = FA.doRun()
print("Minimal",ans)”
```

file_path = "C:/Users/ketha/Desktop/Projects/major/code/dataset/albrecht2.csv": Specifies the file path where the dataset is located.

df=pd.read_csv(file_path): Reads the dataset from the CSV file into a Pandas DataFrame named df.

size=df["size"] and ae=df["Effort"]: Extracts the "size" column and "Effort" column from the DataFrame df and assigns them to variables size and ae, respectively. These columns likely represent the dataset sizes and actual effort values.

FA = FireflyAlgorithm(2, -5, 5, 63, 0.1, 1.0, 0.01, 0.97, 10,size,ae,CostEstimation): Creates an instance of the FireflyAlgorithm class with specific parameters:

ans = FA.doRun(): Executes the Firefly Algorithm optimization process using the doRun method of the FA object and stores the minimal objective function value in the variable ans.

print("Minimal",ans): Prints the minimal objective function value obtained from the optimization process.

CHAPTER 4
RESULTS AND DISCUSSION

CHAPTER 4

RESULTS AND DISCUSSION

This research considers the COCOMO 81 projects' effort data set to produce comparable results. The data set is challenging due to the small number of instances and limited variables analyzed. However, for the objectives of this research, the data set is considered adequate. The COCOMO 81 data set is split into two parts: a training set comprising 13 instances, which accounts for about 60% of the data, and a testing set with 63 instances, making up about 30% of the total projects.

The three main variables considered in this research are Project Size in thousand Lines of Code (KLOC), Methodology (ME), and Actual Effort (AE). The training data set includes instances 1 to 13, while instances 14 to 63 are reserved for testing the model. Table 1 displays the actual values of the training and testing data sets.

KLOC	Measured Effort	Size(MIN_MAX)	Effort(MIN_MAX)
113	2040	0.096706	0.178522
293	1600	0.253497	0.139906
132	243	0.113256	0.020809
60	240	0.050539	0.020546
16	33	0.012212	0.002378
4	43	0.00176	0.003256
6.9	8	0.004286	0.000184
22	1075	0.017439	0.093829
30	423	0.024407	0.036661
29	321	0.023536	0.027655
32	218	0.026149	0.018615
37	201	0.030505	0.017123
25	79	0.020052	0.006416
3	60	0.000888	0.004748

3.9	61	0.001672	0.004836
6.1	40	0.003589	0.002993
3.6	9	0.001411	0.000272
320	11400	0.277016	1
1150	6600	1	0.578729
299	6400	0.232592	0.561173
252	2455	0.217784	0.219446
118	724	0.101061	0.063024
77	539	0.065347	0.046787
90	453	0.076671	0.03924
38	523	0.031376	0.045383
48	387	0.040086	0.033447
9.4	88	0.006463	0.007205
13	98	0.009599	0.008083
2.14	7.3	0.000139	0.000123
1.98	5.9	0	0
62	1063	0.052281	0.092776
390	702	0.337991	0.061093
42	605	0.03486	0.05258
23	230	0.01831	0.019668
12	82	0.008728	0.066789
15	55	0.011341	0.004309
60	47	0.050539	0.003607
15	12	0.011341	0.000535
6.2	8	0.003676	0.000184
3	8	0.000888	0.000184
5.3	6	0.002892	8.78E-06
45.5	45	0.037909	0.003432

28.6	83	0.023188	0.006767
30.6	87	0.02493	0.007117
35	106	0.028763	0.008785
73	126	0.061863	0.010541
23	36	0.01831	0.002642
464	1272	0.402443	0.111119
91	156	0.077542	0.013173
24	176	0.019181	0.014929
10	122	0.006959	0.010189
8.2	41	0.005418	0.003081
5.3	14	0.002892	0.000711
4.4	20	0.002108	0.001237
6.3	18	0.003763	0.001062
27	958	0.021734	0.083561
17	237	0.013083	0.020282
25	130	0.020052	0.010892
23	70	0.01831	0.005629
6.7	57	0.004111	0.004485
28	50	0.022665	0.00387
9.1	38	0.006202	0.002817
10	15	0.006986	0.000799

Table 2 – COCOMO 81 Dataset

Firefly algorithm parameters settings:

Parameters	**values**
Maximum iterations	1000
Numbers of Fireflies	63
Alpha	0.1
Betamin	1.0
Gamma	0.01
Theta	0.97

Table 3 – Parameters of Firefly Algorithm

The evaluation results for training and testing for Firefly optimizing the basic COCOMO model and the Model I by means of all evaluation metrics. .

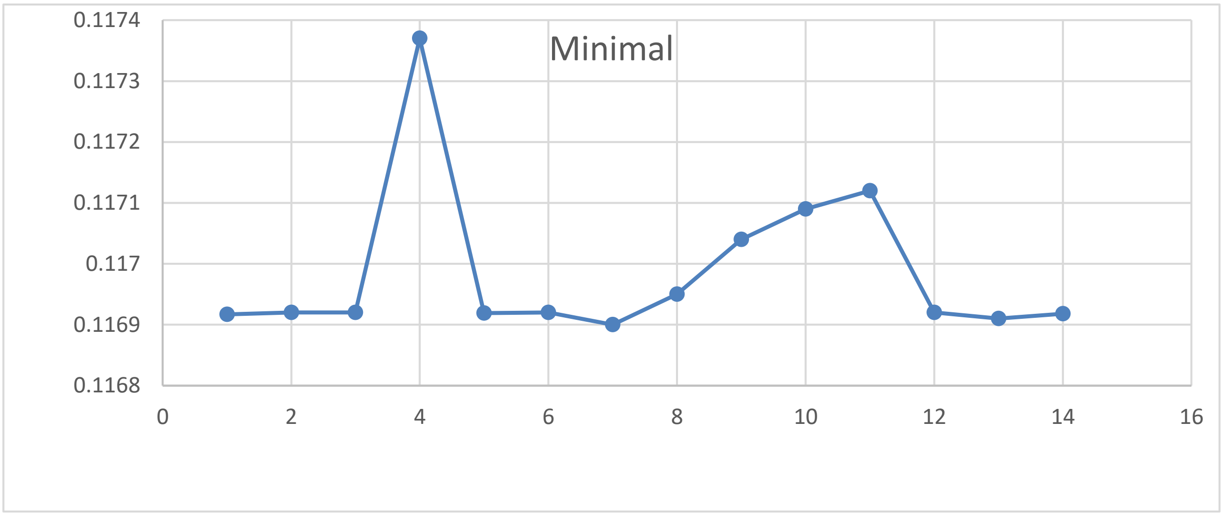
In summary, FA as a metaheuristic optimization algorithm over-performs of higher estimation accuracy for the software effort COCOMO based models. The minimal value of this is 0.1167917

4.1 Performance Measure:

Root Mean Square Error

Root mean square error or root mean square deviation is one of the most commonly used measures for evaluating the quality of predictions. It shows how far predictions fall from measured true values using Euclidean distance. To compute RMSE, calculate the residual (difference between prediction and truth) for each data point, compute the norm of residual for each data point, compute the mean of residuals and take the square root of that mean. RMSE is commonly used in supervised learning applications, as RMSE uses and needs true measurements at each predicted data point.

RSME = $\sqrt{(\sum((y_i - \hat{y}_i)^2)/n)}$ --(10)



Graph 1: Minimal value of RMSE

CHAPTER 5
CONCLUSION

CHAPTER 5
CONCLUSION

The Firefly algorithm shows potential in software effort estimation due to its ability to effectively optimize parameters and handle complex, nonlinear relationships within datasets. Its iterative nature allows for continuous improvement and adaptation, potentially leading to more accurate estimations over time. However, further research and validation are necessary to fully assess its effectiveness and compare it with existing estimation methods**.**Conducting comparative studies with other optimization algorithms to assess the effectiveness and efficiency of the enhanced Firefly Algorithm. Exploring the scalability of the proposed approach to handle larger and more complex software projects. Investigating the adaptability of the enhanced Firefly Algorithm to different domains and types of software development projects. Integrating additional factors or variables into the estimation model to improve accuracy and reliability. Conducting sensitivity analysis to understand the impact of parameter settings on the performance of the enhanced algorithm.

REFERENCES

1. Zareei, M. and Hassan-Pour, H.A., 2015. A multi-objective resource-constrained optimization of time-cost trade-off problems in scheduling project. *Iranian Journal of Management Studies*, *8*(4).
2. Ranichandra, S., 2020. Optimizing non-orthogonal space distance using ACO in software cost estimation. *Mukt Shabd J*, *9*(4), pp.1592-1604
3. Ghatasheh, N., Faris, H., Aljarah, I. and Al-Sayyed, R.M., 2019. Optimizing software effort estimation models using firefly algorithm. *arXiv preprint arXiv:1903.02079*.
4. Ahadi, M. and Jafarian, A., 2016. A new hybrid for software cost estimation using particle swarm optimization and differential evolution algorithms. *Informatics Engineering, an International Journal (IEIJ)*, *4*(1).
5. Dizaji, Z.A. and Khalilpour, K., 2014. PARTICLE SWARM OPTIMIZATION AND CHAOS THEORY BASED APPROACH FOR SOFTWARE COST ESTIMATION. *International Journal of Academic Research*, *6*(3).
6. Braga, P.L., Oliveira, A.L. and Meira, S.R., 2007, September. Software effort estimation using machine learning techniques with robust confidence intervals. In *7th international conference on hybrid intelligent systems (HIS 2007)* (pp. 352-357). IEEE.
7. Dizaji, Z.A., Ahmadi, R., Gholizadeh, H. and Gharehchopogh, F.S., 2014. A bee colony optimization algorithm approach for software cost estimation. *International Journal of Computer Applications*, *104*(12).
8. Puspaningrum, A. and Sarno, R., 2017. A hybrid cuckoo optimization and harmony search algorithm for software cost estimation. Procedia Computer Science, 124, pp.461-469.
9. Wu, D., Li, J. and Bao, C., 2018. Case-based reasoning with optimized weight derived by particle swarm optimization for software effort estimation. Soft Computing, 22, pp.5299-5310.
10. Li, Y.F., Xie, M. and Goh, T.N., 2009. An analysis of feature weighting and project selection for analogy-based software cost estimating. *Journal of systems and software*, *82*(2), pp.241-252.

GIT HUB LINK

https://github.com/Suresh1728/Optimizing-Software-Effort-Estimation-Models

Code:

```
import numpy as np
import random as rand
import math
import time
import pandas as pd
class FireflyAlgorithm:

    def __init__(self, D, Lb, Ub, n, alpha, beta0, gamma, theta, iter_max,size,ae,func):
        self.D = D
        self.Lb = Lb
        self.Ub = Ub
        self.n = n
        self.alpha = alpha
        self.beta0 = beta0
        self.gamma = gamma
        self.theta = theta
        self.iter_max = iter_max
        self.func = func
        self.populationArray = np.zeros((n, D))
        self.functionArray = np.zeros(n)
        self.tmpArray = np.zeros(D)
        self.size=size
        self.ae=ae

    def init_FA(self):
        for i in range(self.n):
            for j in range(self.D):
                self.populationArray[i][j] = rand.uniform(self.Lb, self.Ub)
            self.functionArray[i] = self.func(self.populationArray[i,:], self.D,self.size,self.ae)
    def update(self, i, j):
        scale = self.Ub - self.Lb
        r = 0
        for k in range(self.D):
            r += (self.populationArray[i][k] - self.populationArray[j][k])**2
        beta = self.beta0*math.exp(-self.gamma*r)
        for k in range(self.D):
            steps = (self.alpha*self.theta)*(rand.random() - 0.5)*scale
            self.tmpArray[k] = self.populationArray[i][k] + beta*(self.populationArray[j][k] -
self.populationArray[i][k]) + steps
        if(self.func(self.tmpArray,self.D,self.size,self.ae) < self.functionArray[i]):
            for k in range(self.D):
                self.populationArray[i][k] = self.tmpArray[k]
            self.functionArray[i] = self.func(self.tmpArray, self.D,self.size,self.ae)

    def doRun(self):
        start = time.time()
        self.init_FA()
        for gen in range(self.iter_max):
            print("Generation ", gen+1)
            for i in range(self.n):
                for j in range(self.n):
                    if(self.functionArray[i] > self.functionArray[j] and i != j):
                        self.update(i,j)
            print(self.populationArray)
            print(self.functionArray)
        end = time.time()
        #print(" : %f " % (end - start))
        return self.functionArray.min()
```

```
def CostEstimation(x,D,size,ae):
    #print(x)
    mre=[]
    pe=[]
    rsme=0
    for i in range(0,len(size)):
        pe.append(x[0]*(size[i])**x[1])
        mre.append((ae[i]-pe[i])**2)
    y=sum(mre)
    rsme=math.sqrt(y/len(size))
    return rsme

file_path = "C:/Users/ketha/Desktop/Projects/major/code/dataset/albrecht2.csv"
df=pd.read_csv(file_path)
size=df["size"]
ae=df["Effort"]
#FireflyAlgorithm(D, Lb, Ub, n, alpha, beta0, gamma, theta, iter_max, func)
FA = FireflyAlgorithm(2, -5, 5, 63, 0.1, 1.0, 0.01, 0.97, 10,size,ae,CostEstimation)
ans = FA.doRun()
print("Minimal",ans)
```

Volume: 12 Issue: III Month of publication: March 2024
DOI:

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue III Mar 2024- Available at www.ijraset.com

Optimizing Software Effort Estimation Models Using Enhancement of Firefly Algorithm

B. Prabhanjali[1], G. Kavya[2], K. Suresh[3], V. Venkataiah[4]
[1, 2, 3]UG Students, [4]Associate Professor, Dept. of Computer Science & Engineering, College Of Engineering & Technology, Hyderabad, Telangana

Abstract: Estimating the amount of work required in order to create software development is regarded as essential to the software development life cycle and to the control of project costs, schedules, and quality. As a result, precise estimation plays a critical role in project success and risk mitigation. Software effort estimation has drawn a lot of interest from scholars recently and presented a problem to the software business. Three COCOMO-based models' parameters are to be optimized using a metaheuristic method called improvement of Firefly Algorithm, which is proposed in this research. The basic COCOMO model and the other two models that have suggested in literature as expansions of basic COCOMO model are among these models. Root Mean Square Error (RMSE) is one of the evaluation measures used to assess the created estimate models.
Index Terms: Firefly Algorithm, Metaheuristic Optimization, Effort Estimation, and Software Quality.

I. INTRODUCTION

Software time and effort development of estimation models are fire topic of the study over the three decades for the engineering of software community with systematic approach to the development, maintenance and retirement of software[1]. Computer program taken a toll estimation alludes to the expectations of the probable sum of effort, time, and the number of employees required to construct a program. The essential work of the venture advancement is to guarantee that the venture is completed with the objective: "high quality of computer program must be created with a moo taken a toll that's inside time and budget"[2]. The duties of venture improvement are arranging, organizing, staffing, coordinating, and controlling the exercises. Belittling computer program costs can have an inimical effect on the quality of conveyed program and hence a company's trade reputation and competitiveness. The primary vital component of computer program extends administration is compelling preparing of the improvement of the computer program which determines the assets required to total the extend effectively. The assets incorporate the number and ability level of the individuals, and the sum of computing resources[3]. The fetched of a program venture is directly relative to the individuals / assets required for the project[5]. Overestimation of computer program fetched, on other hand, can result in missed chances to use reserves in other referential ventures. The need for solid and accurate taken a toll expectations in program building is a continuous challenge[3]. Good estimation can improves performances of the company particularly in overseeing the venture plan, human asset assignment, fetched estimation, etc. Those points of interest can decrease the disappointment possibility or extend delay. The Fetched for a extend may be a function of numerous parameters. Measure may be an essential fetched figure in most models and can be measured utilizing code snippets (or) thousands of conveyed KDLOC or work focuses. The no. of models are advanced to set up the connection between Measure and Exertion for Computer program Exertion Estimation[6]. Computer program taken a toll estimation strategies can broadly classified as algorithmic and non-algorithmic models. The algorithms-based models come from the factual examination of historical venture data, for illustration, Helpful Fetched Show (COCOMO) and Program Life Cycle Administration (SLIM)No algorithmic methods incorporate Price-to-Win, Parkinson, master judgment, machine learning approaches[7]. Machine learning is utilized to group together set of methods that embody a few of the characteristics of human intellect, for the case, foggy frameworks, similarity, regression trees, run the show acceptance neural systems, and Developmental algorithms[3]. Among the machine learning approaches, foggy frameworks and neural systems, and Developmental calculations are regarded as have a place to the delicate computing group[11]. The algorithmic as well as the non-algorithmic (based on master judgment) taken a toll estimation models, be that as it may, are not without error. A few exertion estimation model have been develops and advanced over time superior forecast precision moreover, in this way for better improvement quality. The Such model range will be from complex calculations , statistical evaluation of the project's specifications to improve machine learning approaches[10]. Heuristic optimization is plan are relies on few attempt to find the best solutions. Heuristic optimizers are have used in the software cost estimation, such as to application of the genetic programming an model optimizations.

Another case is the portion that the PSO is taken as an optimizer using heuristics. Besides, the crossover methods include the combination of heuristic algorithms just like the utilization of Hereditary Algorithm , Ant Colony[12]. In spite of an outsized number tests on finding foremost satisfactory forecast show, there's not clear prove of an extremely precise and effective strategy. At same time, it is significant to create a forecasting strategy that's fewer intricate and very extra valuable. For occasion, in few forecasting models, an outsized number of factors that are utilized to build the show don't reflect or make strides the precision of prediction show. In this way, gathering additional or disconnected factors are laborious and lacking in centrality. It could become more productive on construct a demonstrate with a least number of the variables, hopefully finding foremost vital and the common factors bland project development efforts[10].

II. BACKGROUND WORK

This here section provides an overview of some recent research which talks about the field of effort estimation for those software projects. This is actually a very active area in research and there have been lot of papers that were published recently. In this specific section, we are going to highlight and sort of review some of very important studies that kind of applied some machine learning techniques in order estimate the effort that goes to the software projects. The dynamic area that involves effort estimation for software projects, well, let's just say that recent research have actually contributed significantly to much further development of methods, especially through the uses of some ML techniques. Miltveit et al. [1], Ganesan et al. [2], and Bhattacharjee et al. [3] sort of made a pretty significant contribution by exploring this case-based approach and then proposing this model that is actually based on those expert cases. The use ML optimization algorithms has sort of become a focus on the purpose of improving that accurate of the software effort estimation, with some techniques such as Cuckoo Search and PSO, Bat Algorithm, and Firefly Algorithm gaining some traction. [3].

Shin and Goel [6] actually questioned the common use of linear regression in empirical methods and then found actually a radial basis function (RBF) neural I suggested in using the network. His proven COCOMO II model, devised by Boehm in 1981, is actually still very influential in estimating effort and development time take into accounts parameter such as effort multipliers, scaling factors, and software size [8]. A.F.Sheta [7] kind of presents this innovative model that sort of combines genetic algorithms (GA) with his COCOMO and demonstrates improved accuracy in the cost estimation. Social behavior-inspired particle swarm optimization algorithms has been sort of proven successful in variety of optimization problems and have actually demonstrated The capability for rapidly converge [9]. Additionally, continuous training of Artificial Intelligence (AI) models on data positions them as a sort of superior alternative to algorithm models inside the software cost estimation, allowing optimization of critical elements. It can sort of minimize project manpower [13]. Selecting metrics for the sizing of those software projects, including: B. Lines of code, function points and cosmic function points (CFSUs) sort of playing an important role. Cosmic FFP is characterized by use of the functional size units as a practical and kind of proven solution for size estimation and quality improvement [10]. Kara Giannopoulos et al. [11] proposed five wrapper character selection method using regression algorithms, including forward selection (FS), backward selection (BS), best first forward selection (BFFS), and best first backward selection (BFBS). By comparing, we kind of contribute to the field. Genetic Search Selection (GS) whose performance was analyzed on 12 of his UCI datasets. In domain of software cost estimation, AI techniques display superior accuracy compared to algorithmic models and iteration optimized factors to reduce the money and effort associated with software projects. As the having complexity of optimization problems increase, metaheuristic algorithms are used of particular note are genetic algorithms & Ant colony optimization (ACO) that use population-based search and evaluation. It helps to really approach the problem with an iteratively until an optimized solution is obtained with high precision!

This comprehensive background work is focusing on different approaches and innovations in software money estimation, integrating varied methods and techniques of ML, optimization algorithms, AI models, and [3] used the hybrid approach to address in-context parameter selection and models optimization. It estimating software struggle and effort. This study is focused on use genetic algorithms (GA) to optimize support vector regression (SVR) models. The authors specifically have investigated and in the impact of GA on character selection and parameter optimization, compared with other approaches, he demonstrated its effectiveness in improving on performance of SVR models. This result is highlighting the applicability of GA to improve accuracy of man power estimation models. Furthermore, a general that framework for estimate the software man power was introduced in [3].

This innovative framework aimed is to emulate human thought processes by incorporating fuzzy rule models. The generated models leveraged the expertise to ensure interoperability and applicability to different been problem areas such as risk analysis and software quality prediction.

III. EMPLOYED TECHNIQUE

A. *Regression Methods*

"The objective of regression methods is effectively maps a group of independent variables (X1, X2..., Xn) to the dependent variables Y; in our context, our sole goal is to construct regression models using a training dataset! Using these models will; result in predicting the total effort is required for the process of creating software projects that is measured in man-months! Extra points should be the considered for the achievement of our ultimate spelling bee success."

B. *Firefly Algorithm*

The Firefly Algorithm (FA) is a multimodal optimization algorithm that is inspired by the activity of fireflies and falls under the category of nature-inspired algorithms. At the University of Cambridge, Xin-She first introduced FA in 2007. It has been demonstrated through empirical evidence that FA approaches the problem more organically and could outperform other metaheuristic algorithms. FA is predicated on his three core principles. According to the first, fireflies of all genders are drawn to one another. According to the second rule, attraction is connected with luminescence or brightness, therefore brighter flies will draw in fewer brighter flies, and movement will be random in the absence of brighter flies. According to the final main rule, brightness varies with the goal function because the objective's landscape

```
Pseudo code of FA
Begin
1) Initialisation max iteration, $\alpha$ $\beta_0$, $\gamma$
2) Generate initial population
3) Define the Objective function f(x),
4) Determine Intensity(I) at cost (x) of each individual determined by $f(x_i)$
5) While (t < Iter max)
 For i= 1 to n
     For j = 1 to n
        If ($I_j > I_i$)
           Moves firefly i towards j in k dimension
        End if
  Evaluate new solutions and update light intensity                    End for j
               End for i
             Rank the fireflies and finds current best
            End while
6) Post process result and visualization
End procedure
```

C. *Root Mean Square Error (RMSE)*

Root mean square error, often confused on root mean squared deviation, are not really the same thing! They differ in terms of how they calculate prediction quality. RMSE or RMSD (let's keep it fancy) is super widely used for evaluating predictions - it's like the ruler of measures! Like, it tells us how far predictions be sliding away from actual true values using some magical Euclidean distance trick.

$$\text{RSME} = \sqrt{(\sum((y_i - \hat{y}_i)^2)/n)} \quad \text{--(1)}$$

IV. PROPOSED SYSTEM

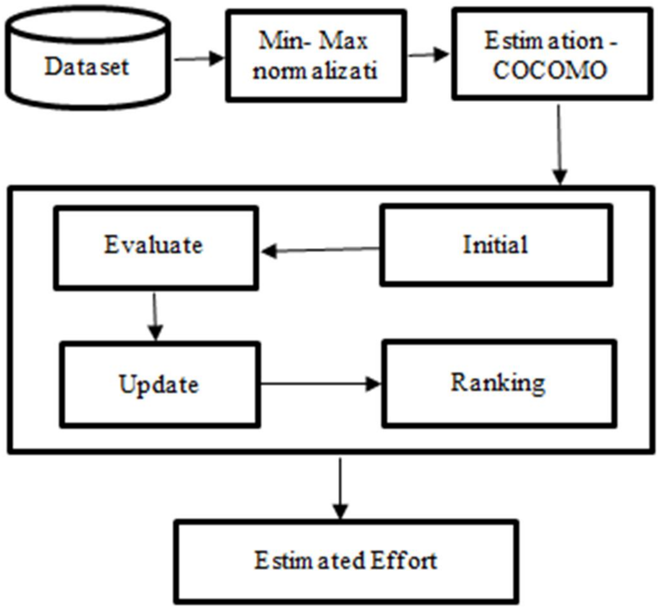


Fig 1 : Architecture of Firefly

Data set (COCOMO81): In this Paper we are taking dataset from the literature review (i.e COCOMO81 dadaset).

1) Preprocessing:

MIN-MAX Normalization: In this process the Preprocessing of the dataset to Normalize the values in between 0 and 1. This can improves performance of Firefly Algorithm by making the data more uniform.

$X_{new} = X – X_{min} / X_{max} - X_{min}$ --(2)

Estimation Model:

Cocomo model: The Constructive Cost Model (COCOMO), It primarily relies on the measurements of the software, measured in Delivered KLOC.

Basic COCOMO: This is the simplest model, suitable for organic mode projects with experienced staff and well-defined requirements. It uses the following formula:

*Effort = a * (KLOC^b) --(3)

2) Optimization Loop:

Firefly Algorithm: Each firefly iteratively adjusts its position in search space based on its brightness and the attractiveness of other fireflies. Brighter fireflies (those representing more accurate estimations based on evaluation criteria) attract other fireflies, moving the population towards better solutions.

Initialize Firefly Algorithm: Initializing firefly algorithm, which adjusts its position in the search apace based on brightness and attractiness of the firefly.

Evaluate: The fitness of each firefly (how well its solution matches the actual effort values) is calculated using a fitness function.

Update: The steps involve the representing of the weights of the every fireflies after calculating the fitness function.

Ranking: Ranking is the process arranging the fireflies with the least values to get optimal out of all firefly weight.

Optimal Result: The firefly with the best fitness value, representing the accurate effort estimation for new software project, is selected as the optimal result.

There were two main problems with the fly attraction in FA: attraction modeling and light intensity differences. For particular firefly at location X, the brighter I is formulated as I (X) a f(X). On the other hand of the attractive force ß is proportional to the fly and related by the distance Rij in between fireflies i & j. Inverse square of the light intensity I (r). Here, I0 represent the light intensity as the light source.

$I(r) = I0{-}\gamma r2$ --(4)

Considering that the environment has as been absorption coefficient γ, intensity is perfectly represented in which I0 is the questionable original intensity.

$I(r) = I0 / 1 + \gamma\ r2$ --(5)

Typically, the distance among a certain firefly at one location (Xi) & another an enigmatic location is representes by the Euclidean distance. The perplexing kth component by the spatial coordinate Xi is represented by X j, in which Xi k

$Rij = \sqrt{(X1 - Y1)2 + (X2 - Y2)2}$ --(6)

A firefly drawn a brighter one, j, it is represented by the attraction as β0e-γr^2 (Xj – xi), and the randomness it is represented by α (rand – ½), which is determined by the enigmatic randomization parameter α.

$Xnew = Xi + \beta 0e\text{-}\gamma r\text{^}2(Xj - Xi) + \alpha\delta\ (rand - ½)$ --(7)

Moreover, FA's silly movements and convergence speed are influenced by γ, which also explains variations in ugly attractiveness.

V. EXPERIMENTS AND RESULTS

This research says the COCOMO 81 project effective data set to maybe produce almost comparable results. The data set kind a challenges due to small no. of the instances & limited variables analyzed, almost. However, for the objective of this remarkably unique research, by the data set kind a sort considered kind a maybe adequate, possibly. The COCOMO 81 data set is, like, split in to two parts: a training set comprising 13 instances, which accounts for like about 60% of the data, and a testing set with 63 instances, making kind up about 30% of the total projects, I guess.
They are three main variable kind a sort considered in this research are Project Size will be in KLOC, Methodology and Actual Effort . The training data set includes instances 1 to 13, and instances 14 to 63 .

Table 1 – COCOMO 81 Dataset

KLOC	Measured Effort	Size(MIN_MAX)	Effort(MIN_MAX)
113	2040	0.096706	0.178522
293	1600	0.253497	0.139906
132	243	0.113256	0.020809
60	240	0.050539	0.020546
16	33	0.012212	0.002378
4	43	0.00176	0.003256
6.9	8	0.004286	0.000184
22	1075	0.017439	0.093829
30	423	0.024407	0.036661
29	321	0.023536	0.027655
32	218	0.026149	0.018615
37	201	0.030505	0.017123

25	79	0.020052	0.006416
3	60	0.000888	0.004748
3.9	61	0.001672	0.004836
6.1	40	0.003589	0.002993
3.6	9	0.001411	0.000272
320	11400	0.277016	1
1150	6600	1	0.578729
299	6400	0.232592	0.561173
252	2455	0.217784	0.219446
118	724	0.101061	0.063024
77	539	0.065347	0.046787
90	453	0.076671	0.03924
38	523	0.031376	0.045383
48	387	0.040086	0.033447
9.4	88	0.006463	0.007205
13	98	0.009599	0.008083
2.14	7.3	0.000139	0.000123
1.98	5.9	0	0
62	1063	0.052281	0.092776
390	702	0.337991	0.061093
42	605	0.03486	0.05258
23	230	0.01831	0.019668
12	82	0.008728	0.066789
15	55	0.011341	0.004309
60	47	0.050539	0.003607
15	12	0.011341	0.000535
6.2	8	0.003676	0.000184
3	8	0.000888	0.000184
5.3	6	0.002892	8.78E-06
45.5	45	0.037909	0.003432
28.6	83	0.023188	0.006767
30.6	87	0.02493	0.007117
35	106	0.028763	0.008785
73	126	0.061863	0.010541
23	36	0.01831	0.002642
464	1272	0.402443	0.111119
91	156	0.077542	0.013173
24	176	0.019181	0.014929
10	122	0.006959	0.010189
8.2	41	0.005418	0.003081
5.3	14	0.002892	0.000711
4.4	20	0.002108	0.001237
6.3	18	0.003763	0.001062
27	958	0.021734	0.083561

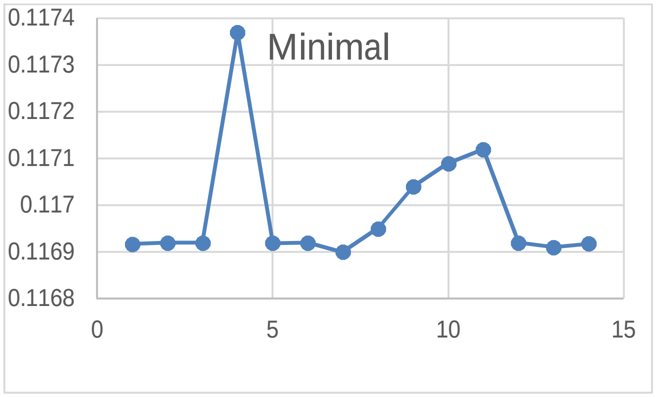
17	237	0.013083	0.020282
25	130	0.020052	0.010892
23	70	0.01831	0.005629
6.7	57	0.004111	0.004485
28	50	0.022665	0.00387
9.1	38	0.006202	0.002817
10	15	0.006986	0.000799

Firefly algorithm parameters settings:

Table 2 – Parameters of Firefly Algorithm

Parameters	values
Maximum iterations	1000
Numbers of Fireflies	63
Alpha	0.1
Betamin	1.0
Gamma	0.01
Theta	0.97

To scaling dada by using min-max normalization which send as input to proposed algorithm. Setting the parameters of Firefly algorithm for tuning the COCOMO81 model parameters for accurate SCE. Thorough The experiment was conducted with will be get the best estimated effort. To evaluate the performance of proposed technique by using RMSE.
FA, a metaheuristic optimization technique, outperforms COCOMO81-based software effort models in terms of estimation accuracy. The minimal value of this is 0.1167917



Graph 1 : Minimal value of RMSE

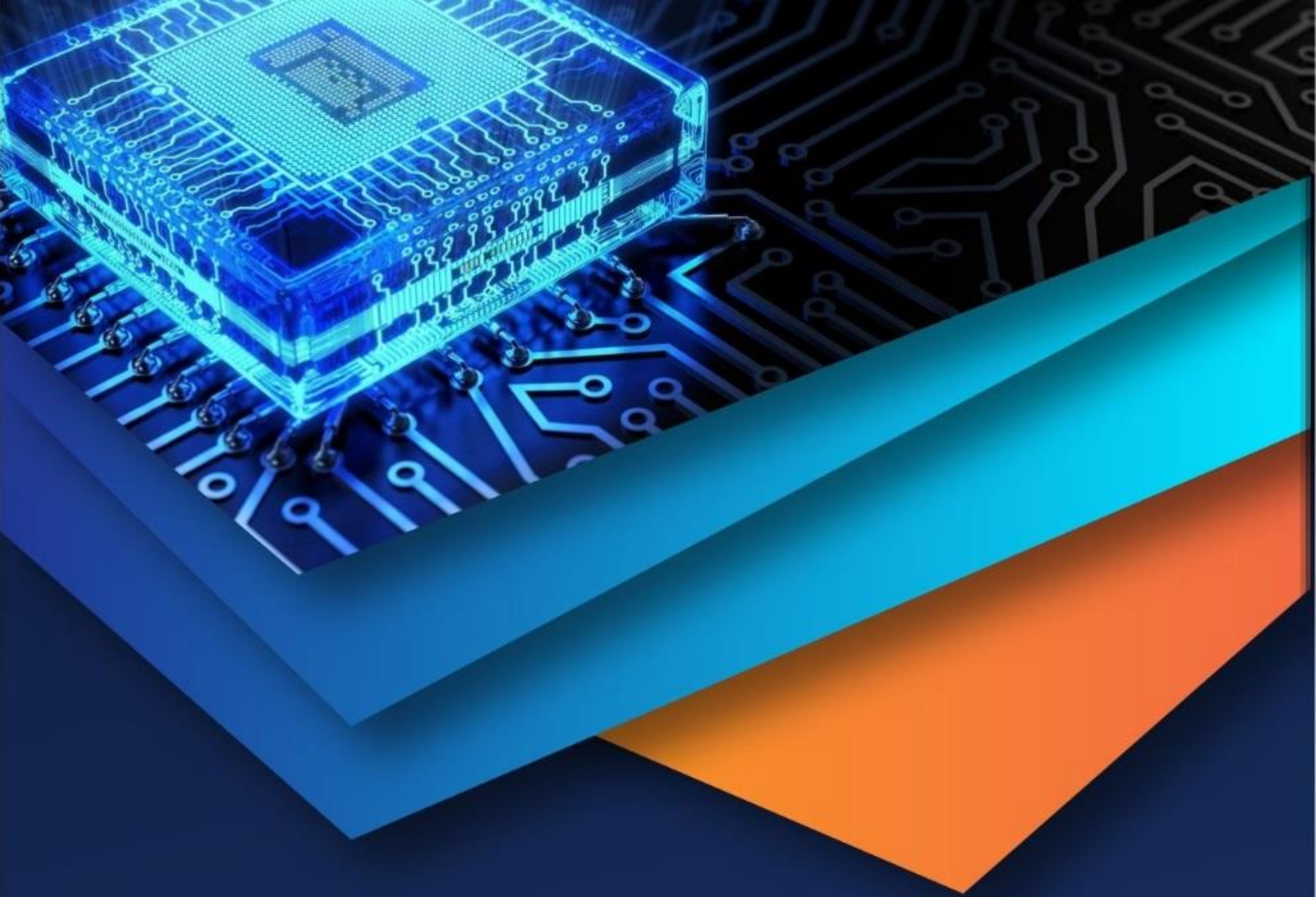
VI. CONCLUSION

The Firefly calculation seems potentially in program exertion estimation because of its capacity to optimize parameters and managing complex, non-linear connections within datasets. Its iterative nature permits for continuously advancements and adjustment, possibly leading to more accurate estimations over time. However, encouraging investigate and approvals are essential to fully evaluate its adequacy and comparing it with the existing estimation strategies.

REFERENCES

[1] Zareei, M. and Hassan-Pour, H.A., 2015. A multi- objective resource-constrained optimization of time-cost trade-off problems in scheduling project. Iranian Journal of Management Studies, 8(4).
[2] Ranichandra, S., 2020. Optimizing non-orthogonal space distance using ACO in software cost estimation. Mukt Shabd J, 9(4), pp.1592-1604
[3] Ghatasheh, N., Faris, H., Aljarah, 2019. Optimizing software cost estimation models using firefly algorithm. arXiv preprint arXiv:1903.02079.

[4] Ahadi, M. and Jafarian, A., 2016. A new hybrid for software effort estimation using particle swarm optimization and differential evolution algorithms. Informatics Engineering, an International Journal (IEIJ), 4(1).
[5] Dizaji, Z.A. and Khalilpour, K., 2014. APPARATUS BASED ON CHAOS THEORY AND PARTICLE SWARM OPTIMIZATION FOR SOFTWARE COST ESTIMATION. International Journal of Academic Research, 6(3).
[6] Braga, P.L., Oliveira, A.L. and Meira, S.R., 2007, September. estimating software effort with solid confidence intervals with machine learning algorithms. During the 7th International Conference on Hybrid Intelligent Systems (HIS 2007) (pp. 352-357). IEEE.
[7] Dizaji, Z.A., Ahmadi, R., Gholizadeh, H. and Gharehchopogh, F.S., 2014. A software cost estimating method using a bee colony optimization algorithm. International Journal of Computer Applications, 104(12).
[8] Puspaningrum, A. and Sarno, R., 2017 A software cost estimating technique combining harmony search and cuckoo optimization. Procedia Computer Science, 124, pp.461-469.
[9] Wu, D., Li, J. and Bao, C., 2018. Software effort estimation using case-based reasoning and optimum weight obtained from particle swarm optimization. Soft Computing,22, pp.5299-5310.
[10] Li, Y.F., Xie, M. and Goh, T.N., 2009. An analysis of feature weighting and project selection for analogy-based software cost estimating. Journal of systems and software, 82(2), pp.241-252.
[11] Harou, J.J., Pulido-Velazquez, M., Rosenberg, D.E., Medellín-Azuara, J., Lund, J.R. and Howitt, R.E., 2009. Hydro-economic models: Concepts, design, applications, and future prospects. Journal of Hydrology, 375(3-4), pp.627-643.
[12] Adamu, A., Abdullahi, M., Junaidu, S.B. and Hassan, I.H., 2021. A hybrid particle swarm optimization method for feature selection that combines the crow search technique. Machine Learning with Applications, 6, p.100108.



ISSN No. : 2321-9653

iJRASET

International Journal for Research in Applied Science & Engineering Technology

IJRASET is indexed with Crossref for DOI-DOI : 10.22214

Website : www.ijraset.com, E-mail : ijraset@gmail.com

ISRA Journal Impact Factor: **7.429**

45.98 INDEX COPERNICUS

THOMSON REUTERS
Researcher ID: N-9681-2016

10.22214/IJRASET

TOGETHER WE REACH THE GOAL
SJIF 7.429

Certificate

It is here by certified that the paper ID : IJRASET59309, entitled

Optimizing Software Effort Estimation Models Using Enhancement of Firefly Algorithm

by

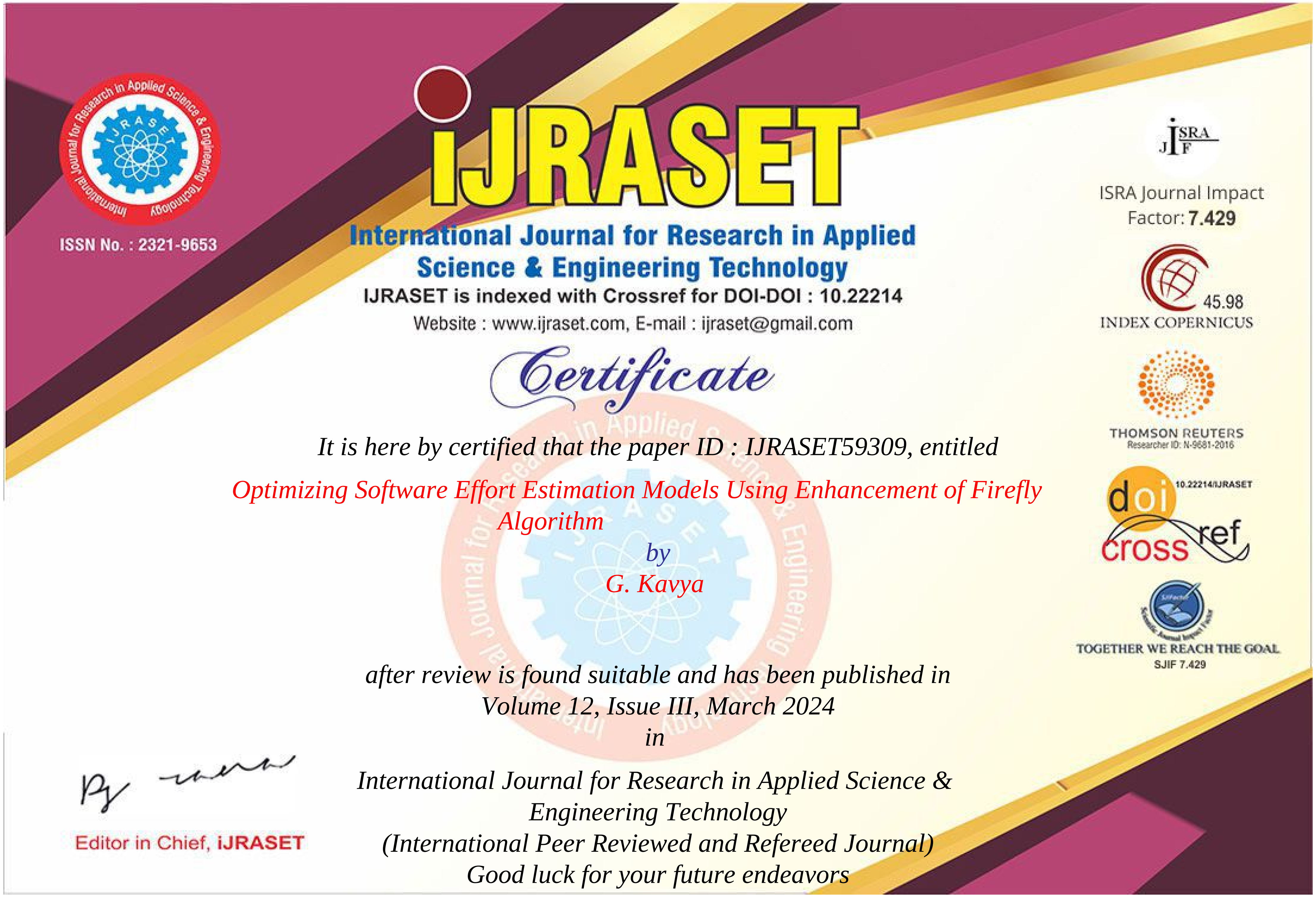
B. Prabhanjali

after review is found suitable and has been published in
Volume 12, Issue III, March 2024
in

International Journal for Research in Applied Science & Engineering Technology
(International Peer Reviewed and Refereed Journal)
Good luck for your future endeavors

[illegible]

Editor in Chief, iJRASET



ISSN No. : 2321-9653

iJRASET

International Journal for Research in Applied Science & Engineering Technology

IJRASET is indexed with Crossref for DOI-DOI : 10.22214

Website : www.ijraset.com, E-mail : ijraset@gmail.com

ISRA Journal Impact Factor: **7.429**

45.98 INDEX COPERNICUS

THOMSON REUTERS
Researcher ID: N-9681-2016

10.22214/IJRASET
doi crossref

TOGETHER WE REACH THE GOAL
SJIF 7.429

Certificate

It is here by certified that the paper ID : IJRASET59309, entitled

Optimizing Software Effort Estimation Models Using Enhancement of Firefly Algorithm

by

G. Kavya

after review is found suitable and has been published in
Volume 12, Issue III, March 2024
in

International Journal for Research in Applied Science & Engineering Technology
(International Peer Reviewed and Refereed Journal)
Good luck for your future endeavors

[illegible]

Editor in Chief, iJRASET

ISSN No. : 2321-9653

iJRASET

International Journal for Research in Applied Science & Engineering Technology

IJRASET is indexed with Crossref for DOI-DOI : 10.22214

Website : www.ijraset.com, E-mail : ijraset@gmail.com

ISRA Journal Impact Factor: **7.429**

45.98 INDEX COPERNICUS

THOMSON REUTERS Researcher ID: N-9681-2016

10.22214/IJRASET

TOGETHER WE REACH THE GOAL SJIF 7.429

Certificate

It is here by certified that the paper ID : IJRASET59309, entitled

Optimizing Software Effort Estimation Models Using Enhancement of Firefly Algorithm

by

K. Suresh

after review is found suitable and has been published in
Volume 12, Issue III, March 2024
in

International Journal for Research in Applied Science & Engineering Technology
(International Peer Reviewed and Refereed Journal)
Good luck for your future endeavors

[illegible]

Editor in Chief, iJRASET

ISSN No. : 2321-9653

iJRASET

International Journal for Research in Applied Science & Engineering Technology

IJRASET is indexed with Crossref for DOI-DOI : 10.22214

Website : www.ijraset.com, E-mail : ijraset@gmail.com

ISRA Journal Impact Factor: **7.429**

45.98 INDEX COPERNICUS

THOMSON REUTERS Researcher ID: N-9681-2016

10.22214/IJRASET

TOGETHER WE REACH THE GOAL SJIF 7.429

Certificate

It is here by certified that the paper ID : IJRASET59309, entitled

Optimizing Software Effort Estimation Models Using Enhancement of Firefly Algorithm

by

V. Venkataiah

after review is found suitable and has been published in Volume 12, Issue III, March 2024

in

International Journal for Research in Applied Science & Engineering Technology

(International Peer Reviewed and Refereed Journal)

Good luck for your future endeavors

[illegible]

Editor in Chief, iJRASET