# Optimizing Effort Estimation Model Using Bat Algorithm

**A Dissertation submitted in the partial fulfillment for the award of**

**MASTER OF TECHNOLOGY**

**IN**

**SOFTWARE ENGINEERING**

***by***


## Neha Gupta

**Roll no. 2k12/SWE/16**


**Under the Esteemed Guidance of**

## Dr. Kapil Sharma



**Department of Computer Engineering**

**Delhi Technological University**

**New Delhi-110042**


**2013-2014**

# ABSTRACT

Software development is most difficult as compared to other types of projects as there is uncertainty in the customer requirements, the process of development is complex, and the final product is intangible in nature. As per the IBM report, "31%of the project gets cancelled before they are completed, 53% overrun their cost estimates by an average of 189% and for every 100 projects, and there are 94 restarts". In order to increase the likelihood of success of a software project, the project managers must do project planning well and for that they need proper effort estimation of the project.

Software effort estimation is amongst the most important tasks in software project management as many decisions like cost estimation, deadline of submitting of project and timely planning a project are dependent on it. Many Algorithmic models are used for effort estimation like COCOMO, Function Points, Use case points etc. Most widely used software cost model or effort model is the Constructive Cost Model (COCOMO).

This thesis introduces a new calibrated Intermediate COCOMO model (for all types of system i.e. organic, semi-detached and embedded) with Bat Algorithm, which is newest Algorithm amongst the category of Meta Heuristic and population based Algorithms. For estimation we have used NASA 63 dataset and Results show that Bat Algorithm gives better results in terms of MMRE (Mean Magnitude of Relative Error) for projects as compared to COCOMO Model



**Keywords:**

**Effort Estimation, Bat Algorithm, COCOMO Model**

# CERTIFICATE

Date: _____

This is to certify that the thesis entitled **Optimizing Effort Estimation Model using Bat Algorithm** submitted by **NEHA GUPTA (Roll Number: 2K12/SWE/16),** in partial fulfillment of the requirements for the award of degree of Master of Technology in Software Engineering, is an authentic research work carried out by her under my guidance. The content embodied in this thesis has not been submitted by her earlier to any institution or organization for any degree or diploma to the best of my knowledge and belief.

**Project Guide**

**Dr. Kapil Sharma**
**Associate Professor**
**Department of Computer Engineering**
**Delhi Technological University, Delhi-110042**

# ACKNOWLEDGEMENT

With due regards, I hereby take this opportunity to acknowledge a lot of people who have supported me with their words and deeds in completion of my research work as part of this course of Master of Technology in Software Engineering.

To start with I would like to thank the almighty for being with me in each and every step of my life. Next, I thank my parents and family for their encouragement and persistent support.

I would like to express my deepest sense of gratitude and indebtedness to my guide and motivator, **Dr. Kapil Sharma**, Associate Professor, Department of Computer Engineering, Delhi Technological University for his valuable guidance and support in all the phases from conceptualization to final completion of the project.

I wish to convey my sincere gratitude to **Prof. O.P. Verma,** Head of Department, and all the faculties and Ph.D. Scholars of Computer Engineering Department, Delhi Technological University who have enlightened me during my thesis.

I humbly extend my grateful appreciation to my friends whose moral support made this project possible.

Last but not the least; I would like to thank all the people directly and indirectly involved in successfully completion of this project.

**NEHA GUPTA**

**Roll No. 2K12/SWE/16**

# DECLARATION

I hereby declare that the thesis entitled "**Optimizing Effort Estimation Model using Bat Algorithm"** which is being submitted to the **Delhi Technological University**, in partial fulfillment of the requirements for the award of degree of **Master of Technology in Software Engineering** is an authentic research work carried out by me. The material contained in this thesis has not been submitted to any university or institution for the award of any degree.

**NEHA GUPTA**
**Master of Technology**
**(Software Engineering)**
**College Roll No. 2K12/SWE/16**
**Department of Computer Engineering**
**Delhi Technological University,**
**Delhi.**

# TABLE OF CONTENT

## LIST OF TABLES

## LIST OF FIGURES

# ABBREVIATIONS

| | |
|---|---|
| NASA | National Aeronautic and Space Administration |
| COCOMO | Constructive Cost Model |
| KLOC | Kilo Lines of Code |
| EAF | Effort Adjustment Factor |
| MRE | Mean Relative Error |
| MMRE | Mean Magnitude of Relative Error |
| MAE | Mean of Absolute Error |
| RMSRE | Root Mean of square relative Error |
| a_best | Best value of coefficient a in all iterations and among all bats |
| b_best | Best value of coefficient b in all iterations and among all bats |

# Chapter One: **INTRODUCTION**

## 1.1 EFFORT ESTIMATION

Effort estimation of software means, we want to know the amount of effort to be put in the development of software. It is usually measured by the number of person-hours that were spent in developing the software from specification until delivery. The prediction of the effort to be consumed in a software project is the most sought after variable in the process of project management as its determination in the early stages of a software project drives the planning of remaining activities.

It's been used as input to project plans, iteration plans, budgets, investment analyses, pricing processes, bid proposals and deciding the execution boundaries of the project (Molokken, 2007). It's a critical activity for planning and monitoring software project development and for delivering the product on time and within budget.(Q. Alam)

**Why is proper effort estimation important?**

- Effort estimation is essential for many people and different departments in an organization as it is needed at various points of a project lifecycle.
- Presales teams need effort estimation in order to know the cost price of custom software. Without effort estimation pricing is impossible and the price you will give will probably bind you for the whole project, so it is important to have a good estimation from the beginning.
- Project managers need it in order to allocate resources and timely plan a project.
- In order to plan a project and inform the project owners about deadlines for submitting the project.

- It also shows if you have the resources to finish the project within customer or project owner predefined time limits, based on your available man power.

## 1.2 MOTIVATION

Software projects development can be considered to be the most uncertain and complex when compared to other types of engineering projects. The 2009 Standish Group Chaos report (The 10 laws of chaos, 2009) showed that only 32% of such projects succeeded and were delivered on time, with the required features and functions within budget: 44% did not meet these three requirements, and 24% failed, they were cancelled prior to completion. Based on the results of several investigations of software development projects, the main areas responsible for project failure were found to be as follows: project goal setting, improper project scheduling, , ambiguous customer requirements, unmanaged risks, improper project execution, project staffing (availability and capabilities), stakeholder politics, and commercial pressures (Five reason why software projects fail, 2002).

Reason for all these failures can be weak project planning and management. For both we need proper effort estimation. When we get the requirements from customers, we need to tell the customer about pricing and time required for its completion. This's required as input to project plans, iteration plans, budgets, investment analyses, resource allocation scheme etc. Thus we can say that proper effort estimation is important from the starting point of project and till the end of it.

But over the past few years, software development effort is found to be one of the worst estimated attributes. Scientific studies show the poor state of software effort estimation. A recent review (M.jorgensen, 2003) reports that 70-80% of software development projects overrun their estimates and that average overruns are about 30-40%.Significant over or underestimation can be very expensive for company as

Overestimation results in wasting of resources, whereas underestimation results in schedule/budget overruns and thus quality compromise. (F. Ferrucci, 2010)

The problem of accurate effort estimation is still open and the project manager is confronted at the beginning of the project with the same question that what effort is required for project (Dolado, 2009)

For support of project managers in a software development, several models have been developed to calculate the required Effort. The most significant effort estimation models that have been used in software development projects are:

- The Constructive Cost Model (COCOMO) (Boehm., 1981)
- The System Evaluation and Estimation of Resource Software Evaluation Model (SEER-SEM) (Segundo, 2001)
- Putnam Model (Putnam, 1978)
- Function Point Model (Albrecht, 1979)

COCOMO is still the best approach for some software projects. If you're using a fairly traditional approach, and using a 3GL (third generation language), such as C and development tools and processes haven't changed much then COCOMO will give you good results. Companies generally use two to three methods for effort estimation and one of them is generally COCOMO. (Facts about COCOMO And Costar, 2012) Thus we can say that COCOMO model is the most widely used estimation model for software project.

Further to solve the problem of accurate effort estimation many optimization algorithms such as particle swarm optimization, genetic algorithm, firefly algorithm, cuckoo search algorithm, Bat Algorithm etc. have been incorporated with these models to further improve them. (Brajesh Kumar Singh, 2013), (F, 2006) (Reddy, 2010) (Sheta, 2007) (P.R Srivastava, 2014)

In this thesis, we propose a new calibrated Intermediate COCOMO model (for all types of system i.e. Organic, semi-detached and embedded) with Bat Algorithm, where

we have calculated new values for coefficients a and b. For calculation we have used the dataset of NASA 63 projects. Results show that optimized COCOMO model with Bat Algorithm give more better results in terms of MMRE of all projects.

## 1.3 GOALS OF THE THESIS

1) To develop New calibrated COCOMO model with Bat Algorithm.
2) To do in deep study of Bat Algorithm and map BAT Algorithm to COCOMO Model.
3) To develop calibrated model for all organic, semidetached and embedded systems in Intermediate model of COCOMO.
4) Compare the result of original Intermediate COCOMO Model with newly calibrated model in terms of MMRE (Mean Magnitude of Relative Error).

## 1.4 THESIS ORGANIZATION:

**This thesis is structured as follows:**

**Chapter 2 (Literature Review)**

**Part A:** Discussion about software Effort Estimation and models used for it (especially COCOMO Model) and measures for accuracy assessment.

**Part B:** Discussion about optimization and various algorithms available for it. In algorithms we mainly discuss about Bat Algorithm which we have used in this thesis work.

**Part C:** Discussion about similar type of COCOMO optimization works which have been already done by various Authors.

**Chapter 3 (Optimized COCOMO Model with Bat Algorithm)**

In this chapter we introduce our proposed model and show how Bat Algorithm is used for optimizing coefficients (a and b) in the Intermediate COCOMO Model (with all types of system i.e. Organic, Semi-detached and embedded).

**Chapter 4 (Result and discussion)**

In this chapter we discuss about Result of the proposed model with available dataset NASA 63.

**Chapter 5 (Conclusions and Future Work)**

In this chapter we summarize the conclusions reached based on the research activity and describes the direction of future work in the area of software effort estimation.

# Chapter Two: **LITERATURE REVIEW**

## 2.1 PART A (SOFTWARE EFFORT ESTIMATION):

Software development effort estimation is the process of predicting the effort required to develop or maintain software based on incomplete, uncertain and noisy input. Effort estimates may be used as input to project plans, iteration plans, budgets, and investment analyses, pricing processes and bidding rounds etc.

**Who should do effort estimation and who is interested in it?** Usually Project Managers are responsible for effort estimation. Depending on the chosen effort estimation method, they can estimate alone or with expert advice from developers, designers and testers.

Apart from managers, project owners and sales people need most of the effort estimation. Most of the times, your effort estimation may be challenged by sales or management teams. There exists a bridge between sales people and developer's team regarding efforts as Sales people want low cost whereas developers and designers know the actual time and resources required for development. Thus when giving estimates, they will take the worst case scenario.

### 2.1.1 ESTIMATION APPROACHES

Estimation Approaches can be categorized as (Shepperd, 2007):

- **Expert estimation:** It's based on judgment process by the experts. The quantification step, i.e., the step where the estimate is produced is based on previous knowledge of experts.
- **Formal estimation model:** The quantification step is based on mechanical processes, e.g., the use of a formula derived from historical data.
- **Combination-based estimation:** The quantification step is based on a judgmental and mechanical combination of estimates from different sources.

In Table 2.1 classifications of estimation approaches within each category is illustrated:

**Table 2.1: Categorization of Estimation Approaches**

| Estimation approach | Category | Examples |
|---|---|---|
| Analogy-based estimation | Formal estimation model | ANGEL |
| WBS-based (bottom up) estimation | Expert estimation | company specific activity templates |
| Parametric models | Formal estimation model | COCOMO, SLIM, SEER-SEM |
| Size-based models | Formal estimation model | Function Point Analysis, Use Case Analysis |
| Group estimation | Expert estimation | Planning poker, Wideband Delphi |
| Mechanical combination | Combination-based estimation | Average of an analogy-based and a Work breakdown structure-based effort estimate |
| Judgmental combination | Combination-based estimation | Expert judgment based on estimates from a parametric model and group estimation |

#### 2.1.1.1 **EXPERT ESTIMATION**

**Expert Judgment Method:** In this technique we consult with software cost estimation expert or a group of the experts to use their experience and understanding of the proposed project to arrive at an estimate of its cost. A group consensus technique or Delphi technique is the best way to be used.

**The estimation steps used in this method:**

- Coordinators present each expert with a specification and an estimation form.
- Coordinator calls a group meeting in which the experts discuss estimation issues with each other and coordinator
- Experts fill out forms anonymously
- Coordinator prepares and distributes a summary of the estimation on an iteration form.
- Coordinator calls a group meeting, and experts discuss their points and estimates which varied widely.
- Experts fill out forms, again anonymously, and steps 4 and 6 are iterated for as many rounds as appropriate.

**The advantages of this method are:**

- The experts can tell about differences between past project experience and requirements of the new proposed project.
- The experts can tell about impacts caused by new technologies, architectures, applications and languages involved in the future project and can also factor in exceptional personnel characteristics and interactions, etc.

**The disadvantages include:**

- This method cannot be quantified. It is hard to document the factors used by the experts or experts-group.

### 2.1.1.2 FORMAL ESTIMATION MODELS

#### 2.1.1.2.1 Estimating by Analogy

In this comparisons are made in between the proposed project and previously complete similar project where the project development information id known. The proposed project is estimated by extrapolating actual data from the completed project. This method can be used either at system-level or at the component-level. Estimating by analogy is relatively straightforward and in some respects, it is a systematic form of expert judgment since experts often search for analogous situations so as to inform their opinion.

**The steps used in estimation by analogy are:**

- Characterizing the proposed project.
- Selecting the most similar completed projects whose characteristics have been stored in the historical data base.
- Deriving the estimate for the proposed project from the most similar completed projects by analogy.

**The main advantages of this method are:**

- The estimation is based on actual project characteristic data.
- The estimator's past knowledge and experience can be used which is not easy to be quantified.
- The differences between the completed and the proposed project can be identified and impacts estimated.

#### 2.1.1.2.2 Top-Down Estimating Method

Top-down estimating method is also called Macro Model. In top-down estimating method, first an overall cost estimation for the project is derived from the global properties of the software project, and then the project is partitioned into various low-level components. This method has wide scope in early cost estimation when only

global properties are known. It is very useful in the early phase of the software development as there is no detailed information available.

**The advantages of this method are:**

- Its main focus is on system-level activities such as integration, documentation, configuration management, etc., many of which may be ignored in other estimating methods and it does not miss the cost of system-level functions of the project.
- It requires minimal project detail, and it is usually faster, easier to implement.

**The disadvantages are:**

- It often does not identify difficult low-level problems that are likely to escalate costs and sometimes low-level components are overlooked by it.
- No detailed basis for justifying decisions or estimates is provided.

The leading method using this approach is Putnam model.

**2.1.1.2.2.1 Putnam model:**

Another popular software cost model is the Putnam model. The form of this model is:

**Technical constant C= size * B $^{1/3}$ * T $^{4/3}$**

**Total Person Months B=1/T $^{4}$ *(size/C) $^{3}$**

**T= Required Development Time in years**

Size is estimated in LOC Where: C is a parameter dependent on the development environment and it is determined on the basis of historical data of the past projects.

Rating: C=2,000 (poor), C=8000 (good) C=12,000 (excellent)

#### 2.1.1.2.3 Function Point Analysis

The function point measurement method was developed by Allan Albrecht at IBM. (Albrecht, 1979) The Function Point Analysis is another method of quantifying the size and complexity of a software system in terms of the functionalities that user has demanded. Albrecht believes function points offer several significant advantages over SLOC counts of size measurement.

**There are two steps in counting function points:**

- **Counting the user functions:** The raw function counts are arrived at by considering a linear combination of five basic software components: external inputs, external outputs, external inquiries, logic internal files, and external interfaces, each at one of three complexity levels: simple, average or complex. Then the sum of these numbers, weighted according to the complexity level is calculated, and is known as the number of function counts (FC).
- **Adjusting for environmental processing complexity:** The final function points is arrived at by multiplying FC by an adjustment factor which is determined by considering 14 aspects of processing complexity. This adjustment factor allows the FC to be modified by at most 35% or -35%.

The collection of function point data has two primary motivations. One is the desire by managers to monitor levels of productivity. Another use of it is in the estimation of software development cost.

#### 2.1.1.2.4 **Bottom-up Estimating Method**

Using bottom-up estimating method, the cost of each software components is estimated and then combines the results to arrive at an estimated cost of overall project. It aims at constructing the estimate of a system from the knowledge accumulated about the small software components and their interactions.

**The advantages:**

- It allows the software group to handle an estimate in an almost traditional fashion.
- It is more stable because the estimation errors in the various components have a chance to balance out.

**The disadvantages:**

- It may overlook many of the system-level costs (integration, configuration management, quality assurance, etc.) associated with software development.
- It may be inaccurate because the complete information may not available in the early phase.

**The leading method using this approach is COCOMO model.**

### 2.1.1.3 **COCOMO MODEL**

Constructive Cost Model (COCOMO) is an algorithmic model developed by Barry W. Boehm. The model uses a basic regression formula with parameters that are derived from historical project data and current as well as future project characteristics. It drew on a study of 63 projects at TRW Aerospace where Boehm was Director of Software Research and Technology. The study examined projects ranging in size from 2,000 to 100,000 lines of code, and programming languages ranging from assembly to PL/I. These projects were based on the waterfall model of software development which was the prevalent software development process in 1981.(http://en.wikipedia.org/wiki/COCOMO)

The basic COCOMO model has a very simple form:

$$\text{MAN-MONTHS} = K1 * (\text{Thousands of Delivered Source Instructions})^{K2}$$

Where K1 and K2 are two parameters dependent on the application and development environment.

COCOMO consists of a hierarchy of three increasingly detailed and accurate forms illustrated in Figure 2.1:

- Basic Model
- Intermediate Model
- Detailed Model

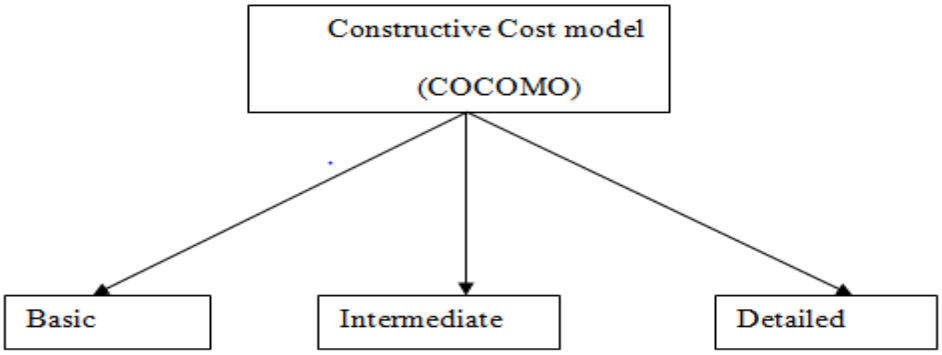


**Figure 2.1: COCOMO Model**

**All models of COCOMO are applied to three classes of software projects:**

- **Organic projects** - "small" teams with "good" experience working with "less than rigid" requirements
- **Semi-detached projects** - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements
- **Embedded projects** - developed within a set of "tight" constraints. It is also combination of organic and semi-detached projects.(hardware, software, operational)

In Table 2.2 comparison of modes (organic, semi-detached and embedded) in terms of size, nature of project and development environment is illustrated.

**Table 2.2: The comparison of three classes of software projects**

| **Mode** | **Project Size** | **Nature Of Project** | **Innovation** | **Deadline of Project** | **Development Environment** |
|---|---|---|---|---|---|
| Organic | Typically2 to 50 KLOC | Small size project, experienced developers in the familiar environment For example : pay roll , inventory projects etc. | Little | Not tight | Familiar and in house. |
| Semidet ached | Typically 50 to300 KLOC | Medium size project, Medium size team, Average previous experience on similar project. For example: Utility systems like compilers , database systems , editors etc. | Medium | Medium | Medium |
| Embedd ed | Typically over 300 KLOC | Large projects, Real time systems complex interfaces. Very little previous experience For example: ATM'S Air Traffic Control etc. | Significant | Tight | Complex Hardware/ customer interfaces required. |

**A) Basic Model**

Basic COCOMO model takes the form

$E = a (KLOC)^{b}$

$D = c (KLOC)^{d}$

Where E is effort applied in Person-Months, and D is the development time in months. The coefficients a, b, c and d are given in table 2.3.

**Table 2.3: Basic COCOMO coefficients**

| Software Project | a | B | C | d |
|---|---|---|---|---|
| Organic | 2.4 | 1.05 | 2.5 | 0.38 |
| Semidetached | 3.0 | 1.12 | 2.5 | 0.35 |
| Embedded | 3.6 | 1.20 | 2.5 | 0.32 |

**B) Intermediate COCOMO:**

It computes software development effort as function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel and project attributes. This extension considers a set of four "cost drivers", each with a number of subsidiary attributes:-

1) **Product attributes**
   - Required software reliability
   - Size of application database
   - Complexity of the product
2) **Hardware attributes**
   - Run-time performance constraints
   - Memory constraints

- Volatility of the virtual machine environment
- Required turnabout time

**3) Personnel attributes**

- Analyst capability
- Software engineering capability
- Applications experience
- Virtual machine experience
- Programming language experience

**4) Project attributes**

- Use of software tools
- Application of software engineering methods
- Required development schedule

Each of the 15 attributes receives a rating on a six-point scale that ranges from "very low" to "extra high" (in importance or value). An effort multiplier from the table below applies to the rating. The product of all effort multipliers results in an effort adjustment factor (EAF) which ranges from 0.9 to 1.4.

**The Intermediate COCOMO formula now takes the form:**

$E=a\,(KLOC)^{b} * EAF$

$D=c\,(E)^{d}$

Where E is the effort in person-months, KLoC is the estimated number of thousands of delivered lines of code for the project, and EAF is adjustment factor calculated according to Table 2.4

**Table 2.4: Value of Effort Adjustment Factors (EAF)**

| **Cost Drivers** | **Ratings** | | | | | |
|---|---|---|---|---|---|---|
| | Very Low | Low | Nominal | High | Very High | Extra High |
| **Product attributes** | | | | | | |
| Required software reliability | 0.75 | 0.88 | 1.00 | 1.15 | 1.40 | ------ |
| Size of application database | ------ | 0.94 | 1.00 | 1.08 | 1.16 | ------ |
| Complexity of the product | 0.70 | 0.85 | 1.00 | 1.15 | 1.30 | 1.65 |
| **Hardware attributes** | | | | | | |
| Run-time performance constraints | ------ | ------ | 1.00 | 1.11 | 1.30 | 1.66 |
| Memory constraints | ------ | ------ | 1.00 | 1.06 | 1.21 | 1.56 |
| Volatility of the virtual machine environment | ------ | 0.87 | 1.00 | 1.15 | 1.30 | ------ |
| Required turnabout time | ------ | 0.87 | 1.00 | 1.07 | 1.15 | ----- |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Personnel attributes** | | | | | | |
| Analyst capability | 1.46 | 1.19 | 1.00 | 0.86 | 0.71 | ------ |
| Applications experience | 1.29 | 1.13 | 1.00 | 0.91 | 0.82 | ------ |
| Software engineer capability | 1.42 | 1.17 | 1.00 | 0.86 | 0.70 | ------ |
| Virtual machine experience | 1.21 | 1.10 | 1.00 | 0.90 | ------ | ------ |
| Programming language experience | 1.14 | 1.07 | 1.00 | 0.95 | ------ | ------ |
| **Project attributes** | | | | | | |
| Application of software engineering methods | 1.24 | 1.10 | 1.00 | 0.91 | 0.82 | ----- |
| Use of software tools | 1.24 | 1.10 | 1.00 | 0.91 | 0.83 | ----- |

The coefficient (a and b) are given in Table 2.5.

**Table 2.5: Intermediate COCOMO coefficients**

| PROJECT | A | b | c | d |
|---|---|---|---|---|
| Organic | 3.2 | 1.05 | 2.5 | 0.38 |
| Semidetached | 3.0 | 1.12 | 2.5 | 0.35 |
| Embedded | 2.8 | 1.20 | 2.5 | 0.32 |

C) **Detailed COCOMO**

Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process.

The detailed model uses different effort multipliers for each cost driver attribute. These Phase Sensitive effort multipliers are each to determine the amount of effort required to complete each phase. In detailed COCOMO, the whole software is divided in different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort. In detailed COCOMO, the effort is calculated as function of program size and a set of cost drivers given according to each phase of software life cycle.

### 2.1.2 MEASURES FOR ACCURACY ASSESMENT

Accuracy is defined as the measure of how close a result is to its correct value. There are two ways to compare a result and its correct value: their difference and their ratio.(Gao, 1997) Let n be the number of projects in a data set, $act_i$ be the actual effort of $i^{th}$ project (i= 1, 2, 3...n) and $est_i$ be the corresponding estimated value.

The difference measure of estimation accuracy is based on the difference between estimated value and actual value

$$\mathbf{Est_i - Act_i} \qquad \mathbf{For\ (i = 1, 2, 3, \ldots n)} \qquad \mathbf{(1)}$$

The ratio measure of accuracy is based on the ratio of estimated value to actual value

$$\frac{\mathbf{Estimated_i}}{\mathbf{Actual_i}} \qquad \mathbf{For\ (i = 1, 2, 3, \ldots n)} \qquad \mathbf{(2)}$$

In evaluating the accuracy of software cost estimation models, both difference and ratio measures have been used. These are discussed below.

**A) Difference Measures of Accuracy**

**(1) Mean of Absolute Errors (MAE)**

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|act_i - est_i| \qquad \text{For } (i = 1, 2, 3, .. n) \qquad \mathbf{(3)}$$

**(2) Root Mean of Squares of Error (RMSE)**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(act_i - est_i)^2} \qquad \text{For (i=1, 2, 3...n)} \qquad \mathbf{(4)}$$

**B) Ratio Measures of Accuracy**

**(1) Magnitude of Relative Errors (MRE)**

$$\mathbf{MRE_i} = \frac{|\mathbf{ACTUAL_i - ESTIMATED_i}|}{\mathbf{ACTUAL_i}} \qquad \text{For (i=1, 2, 3...n)} \qquad \mathbf{(5)}$$

**(2) Mean of Magnitude of Relative Errors (MMRE)**

$$\mathbf{MMRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|act_i - est_i|}{act_i} \qquad \text{For (i=1, 2, 3...n)} \qquad \mathbf{(6)}$$

**(3)Root Mean of Square relative Errors (RMSRE)**

$$\mathbf{RMSRE} = \sqrt{\frac{\mathbf{1}}{\mathbf{n}}\sum_{\mathbf{i=1}}^{\mathbf{n}}\left(\frac{\mathbf{act_i - est_i}}{\mathbf{act_i}}\right)^{\mathbf{2}}} \qquad \text{For (i=1, 2, 3...n)} \qquad \mathbf{(7)}$$

**MMRE is the most widely used measure in the literature and we have also used it for estimations**

## PART B (OPTIMIZATION)

Optimization or mathematical programming is the selection of a best element from some set of available alternatives with regard to some criteria.("The Nature of Mathematical Programming)

An optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within a defined domain or a set of constraints and computing the value of the function.

**Optimization problems are of various types:**

- **Discrete optimization:**

In this the variables used in the optimization are restricted to assume only a finite or discrete set of values, such as the integers.

- **Continuous Optimization:**

In continuous optimization the variables used in the objective function can assume real values, e.g., values from intervals of the real line.

We can classify these algorithms into various types. Further we illustrate the classification in Figure 2.2:

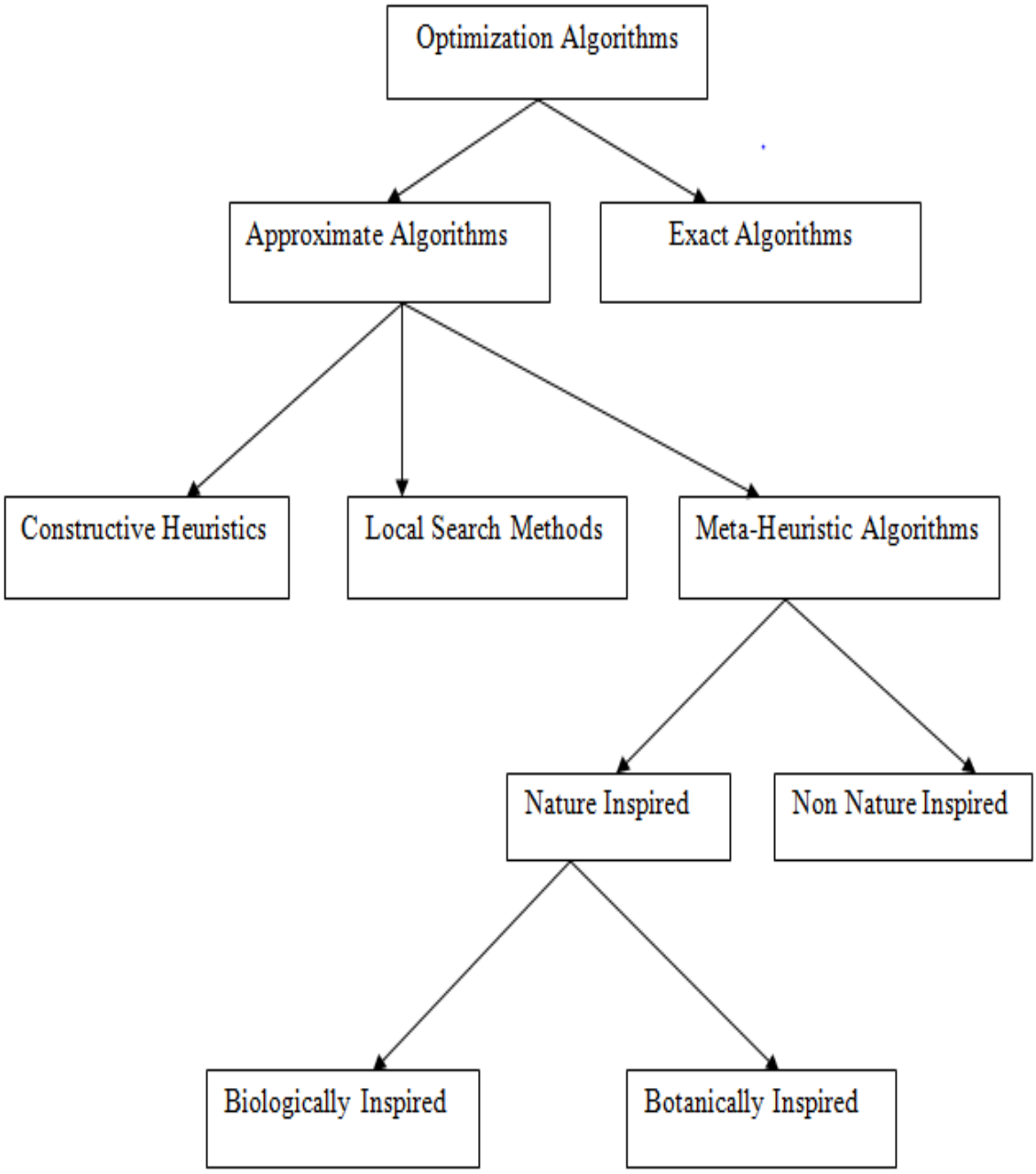


**Figure 2.2: Classification of Optimization Algorithms**

We are going to study Meta Heuristic Algorithms in detail as Bat Algorithm is one of the meta-heuristic and nature inspired Algorithm.

### 2.1.3 META-HEURISTIC ALGORITHMS

Meta-heuristic algorithms are high level procedure which is designed to find or generate a low level procedure which may provide a sufficiently good solution to optimize a problem. We also need not to have complete information about the problem to get a solution. Thus we can conclude that "Meta-" means "beyond" or "higher level" and "heuristic" means "search" or "discover by trial or error".

Meta-heuristic algorithms usually consist of two major processes, i.e., solution exploration and solution exploitation. These two processes are iteratively performed to search for optimal or near-optimal solutions in reasonable computation time. The exploration process not only increases the diversity of solutions found, but also helps to overcome local optimal solutions to obtain better or optimal ones due to its randomization.(X.S., 2008)

To improve the quality of solutions obtained from the exploration process and to ensure that the solution converge to optimality, we use Exploitation. In some meta-heuristic algorithms, this exploitation process also helps to overcome local optimal solutions to search for better or optimal ones. The performance of meta-heuristic algorithms depends on the appropriate combination between these two processes (Exploration and Exploitation).

We have classified meta-heuristic algorithms into two major types, i.e**.,** nature-inspired algorithms and non-nature inspired algorithms. Nature has been evolving for millions of years and hence learning from the nature's success, we can design meta-heuristic algorithms (X.S., 2008). Nature-inspired algorithms can be further divided into

biologically inspired algorithms, botanically inspired algorithms. In Nature inspired algorithms we are going to focus mainly on biologically inspired algorithms.

2.1.3.1 **BIOLOGICALLY INSPIRED ALGORITHMS**

These algorithms are inspired by creatures in nature and they are further classified into three major groups: evolutionary algorithms, stigmergic optimization algorithms, and swarm-based optimization algorithms as illustrated in Figure2.3.

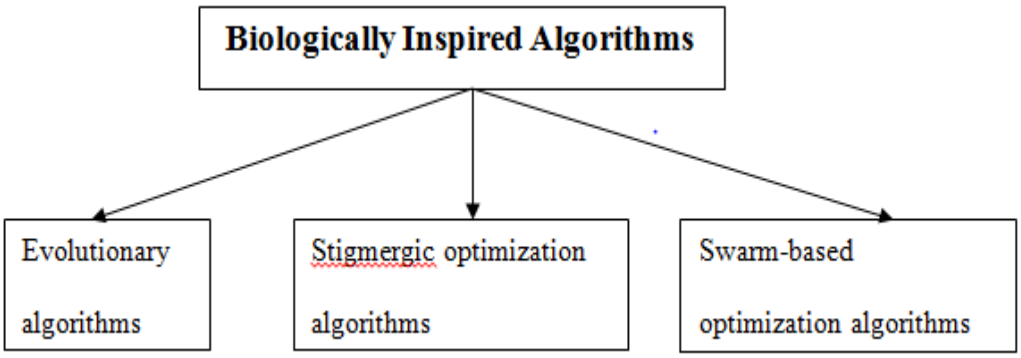


**Figure 2.3: Biologically Inspired Algorithms**

**1) Evolutionary Algorithms**

Evolutionary algorithms are based on the principles of natural evolution. Natural evolution is a complex process which operates on chromosomes, instead of organisms (Michalewicz, 1992). The chromosomes contain genetic information, called a gene, which is passed from one generation to next generation through reproduction. In reproduction, the most important operators are recombination and mutation. Organisms with good chromosomes have a higher chance to exist and develop in nature. According to Darwin's natural selection theory (Darwin, 1859), natural selection process selects best environment-adapted organisms.

For example: Genetic algorithm (GA)

**2) Stigmergic Optimization Algorithms**

According to (Abraham A., 2006) for Self-Organization insects often require interactions among themselves, such interactions can be direct or indirect. Direct interactions are the "obvious" interactions like food or liquid exchange, visual contact, chemical contact (the odour of nearby nest mates), etc. In Indirect interactions two individuals interact indirectly when one of them modifies the environment and the other responds to the new environment at a later time. Such an interaction is an example of stigmergy".

For example: Termite algorithm, Ant colony optimization and Bee colony optimization.

**3) Swarm-Based Optimization Algorithms**

Swarm-based optimization algorithms are inspired by the social behaviour of swarm-based animals or insects, such as a school of fish or a flock of birds, especially those in which the property of historical information exchange among individuals is magnified. These algorithms use many autonomous agents (particles) that act together in simple ways to produce seemingly complex behaviour.(Banks A., 2007)

For example: Particle swarm optimization, Firefly algorithm and Bat algorithm.

### 2.1.4 BAT ALGORITHM

Bat algorithm is a meta-heuristic, nature inspired, swarm based algorithm proposed by (Yang, 2010), and its categorization is illustrated in Figure 2.4. It's an optimization method based on the echolocation behaviour of bats. Micro bats echolocation capability helps them to detect preys, distinguish different kinds of insects.

**How Bat search for prey**

- These bats emit a very loud sound pulse (echolocation) and listens for the echo that bounces back from the surrounding objects as illustrated in figure 2.5.
- Bats use short, frequency-modulated signals to catch prey.

- Each Bat has a constant frequency of emitted pulse which is usually in the region of 25 kHz to 150 kHz.
- Each ultrasonic burst may last typically 5 to 20 ms, and micro bats emit about 10 to 20 such sound bursts every second.
- With time as bat moves toward prey, it changes its velocity and position to get more near about prey.
- As the bat goes near prey, the rate of pulse emission increases which can be up to about 200 pulses per second and loudness decreases.

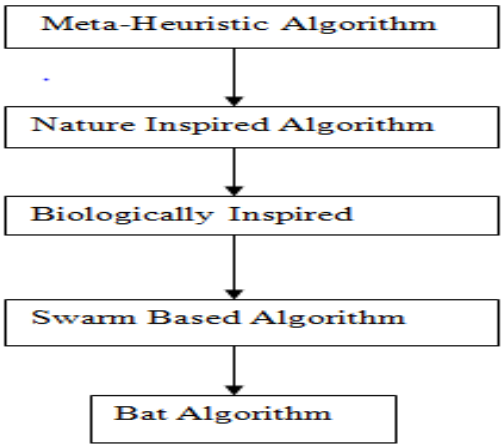


**Figure 2.4: Categorization of Bat Algorithm**

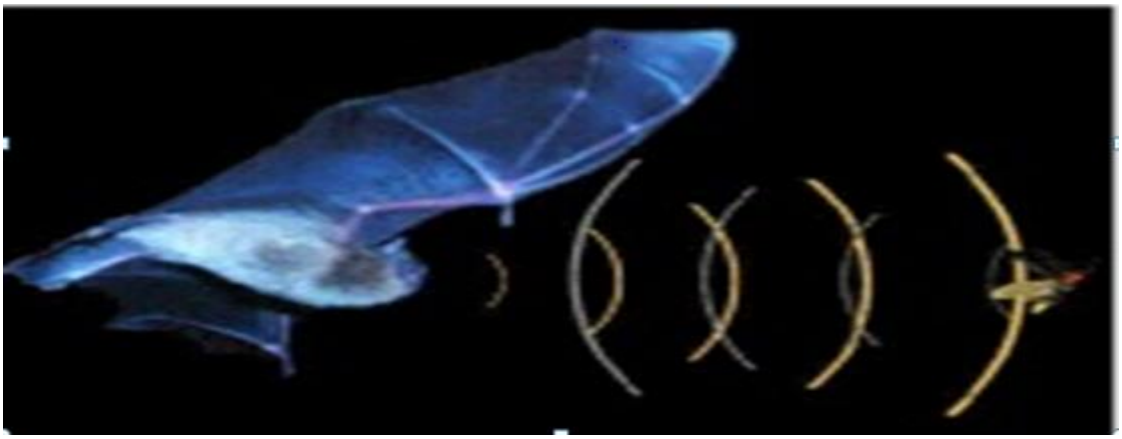

**Figure 2.5: Bat using echolocation to catch its prey**

**This property of bat can be used to propose various algorithms and finally proposed algorithm comes out with following idealized rules:**

- All bats use echolocation to sense distance, and they also 'know' the difference between food/prey
- Bats fly randomly with velocity $v_i$ at position $x_i$ with a fixed frequency $f_{min}$, varying wavelength and loudness $A_0$ to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r [0, 1], depending on prey position.
- We also use the following approximations, for simplicity. In general the frequency f in a range [$f_{min}$, $f_{max}$] corresponds to a range of wavelengths [$\lambda_{min}$, $\lambda_{max}$]. For example a frequency range of [20 kHz, 500 kHz] corresponds to a range of wavelengths from 0.7mm to 17mm.
- Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) $A_0$ to a minimum constant value Amin. Pulse rate increases as the bat approaches prey.

#### 2.1.4.1 PSEUDO CODE OF THE BAT ALGORITHM (BA)

**Objective function** f(x), $x = (x_1 ... x_d)^T$

**Initialize the bat population:**
Initialize position, $x_i$ (i = 1, 2... n bats) and velocity $v_i$
Define pulse rate ($r_i$), loudness ($A_i$) and frequency ($f_i$) for all bats

While (t <Max number of iterations)

Generate new solutions by adjusting frequency,
And updating velocities and locations/solutions [using equations (8) to (10)]

If (rand >$r_i$)
Select a solution among the best solutions
Generate a local solution around the selected best solution [equation 11]

End if
Generate a new solution by flying randomly
If (rand < $A_i$ && f ($x_i$) < f($x_*$))
Accept the new solutions
Increase $r_i$ and reduce $A_i$
End if

Rank the bats and find the current best $x_*$
End while

Post process results and visualization

**Movement of Virtual Bats**

For simulations, we use virtual bats naturally. Their positions $x_i$ and velocities $v_i$ in a d-dimensional search space has to be updated and for that we use following equations. The new Solutions $x_i$ and velocities $v_i$ at time step t is given by:

$$\mathbf{f_i = f_{min} + (f_{max} - f_{min})\beta} \quad \textbf{(8)}$$

$$\mathbf{v_i^t = v_i^{t-1} + (x_i^t - x_*)f_i} \quad \textbf{(9)}$$

$$\mathbf{x_i^t = x_i^{t-1} + v_i^t} \quad \textbf{(10)}$$

Where, β= [0, 1] is a random vector drawn from a uniform distribution. Here $x_*$ is the Current global best location (solution) which is located after comparing all the solutions among all the n bats. We can use either $f_i$ or $\lambda_i$ to adjust the velocity change while fixing the other factor.

For the local search part, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk

$$\mathbf{X_{new} = X_{old} + \epsilon A^t} \quad \textbf{(11)}$$

Where Є is [−1, 1] is a random number, while $A^t$ is the average loudness of all the bats at this time step.

**Loudness and Pulse Emission**

The loudness $A_i$ and the rate $r_i$ of pulse emission have to be updated accordingly as the iterations proceed. As the loudness usually decreases once a bat has found its prey, while the rate of pulse emission increases, the loudness can be chosen as any value of convenience. We can also use $A_{max} = 1$ and $A_{min} = 0$, assuming $A_{min} = 0$ means that a bat has just found the prey and temporarily stop emitting any sound.

$$\mathbf{A_i^{t+1}} = \boldsymbol{\alpha}\, \mathbf{A_i^t}, \quad \boldsymbol{r_i^{t+1}} = \boldsymbol{r_i^0}\,[\mathbf{1} - \mathbf{exp}(-\boldsymbol{\gamma t})], \qquad \textbf{(12)}$$

Where α and ϒ are constants with values: $0 < \alpha < 1$ and $\Upsilon > 0$

Initially, each bat should have different values of loudness and pulse emission rate, and this can be achieved by randomization.

Flow of algorithm, is illustrated in Figure 2.6:

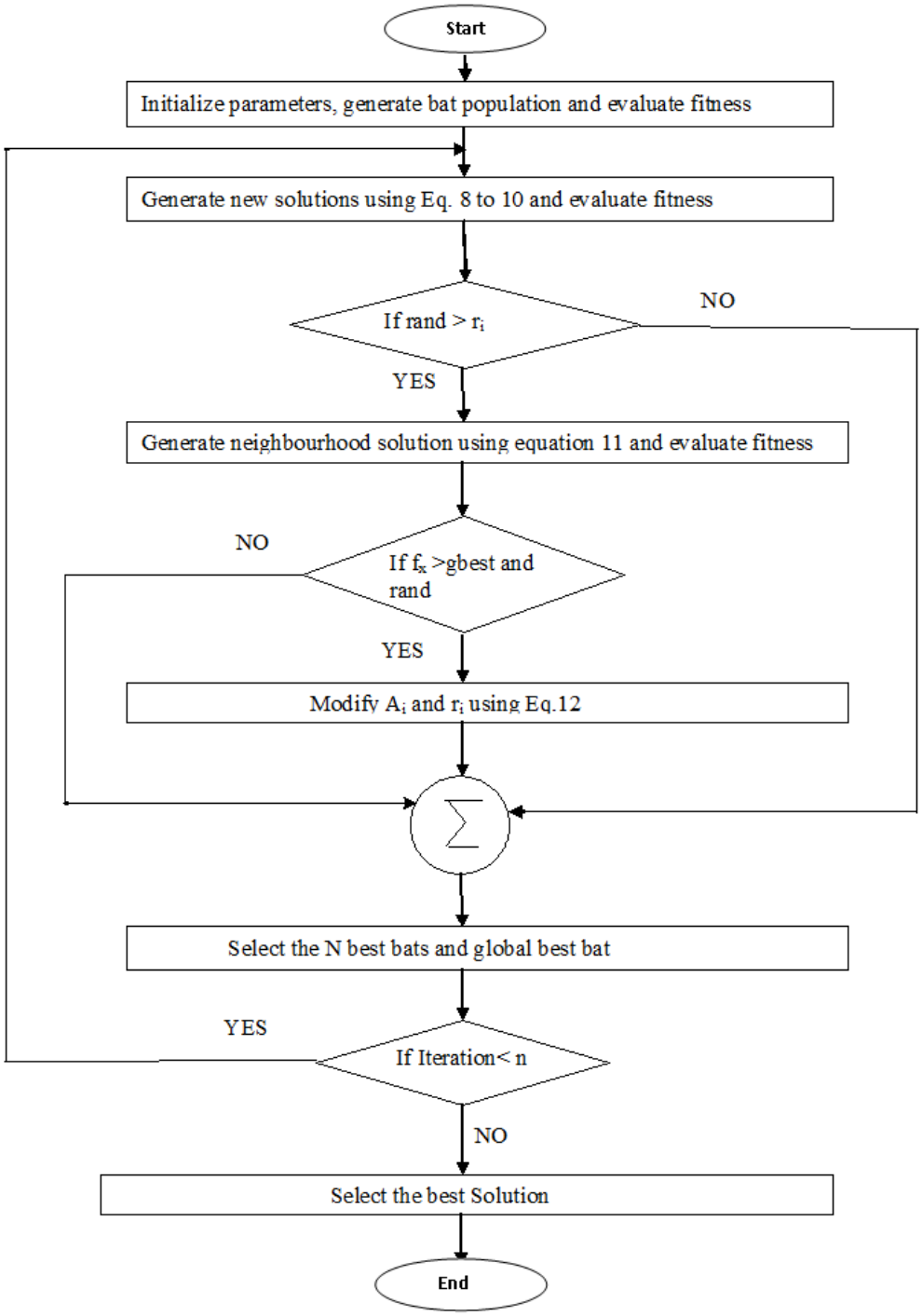


**Figure 2.6: Flowchart of Bat Algorithm**

### 2.1.4.2 **VARIANTS OF BAT ALGORITHM**

In order to improve the performance, many methods and strategies have been attempted to increase the diversity of the solution and thus to enhance the performance, which produced a few good variants of bat algorithm.

- **Fuzzy Logic Bat Algorithm (FLBA):** (Khan, 2011) presented a variant by introducing fuzzy logic into the bat algorithm; they called their variant fuzzy bat algorithm.
- **Multi objective bat algorithm (MOBA):** (Yang X. S., 2011)extended BA to deal with multi objective optimization, which has demonstrated its effectiveness for solving a few design benchmarks in engineering.
- **K-Means Bat Algorithm (KMBA):** (Komarasamy, 2012)presented a combination of K-means and bat algorithm (KMBA) for efficient clustering.
- **Chaotic Bat Algorithm (CBA):** (Lin, 2012) presented a chaotic bat algorithm using Levy flights and chaotic maps to carry out parameter estimation in dynamic biological systems.
- **Binary bat algorithm (BBA):** (Nakamura, 2012) developed a discrete version of bat algorithm to solve classifications and feature selection problems.
- **Differential Operator and Levy flights Bat Algorithm (DLBA):** (Xie, 2013) presented a variant of bat algorithm using differential operator and Levy flights to solve function optimization problems.
- **Improved bat algorithm (IBA):** (Jamil, 2013)extended the bat algorithm with a good combination of Levy flights and subtle variations of loudness and pulse emission rates. They tested the IBA versus over 70 different test functions and proved to be very efficient.

### 2.1.4.3 **APPLICATIONS OF BAT ALGORITHM**

Bat algorithm and its variants have been applied in almost every area of optimization, classifications, image processing, feature selection, data mining. For e.g:

- (Bora, 2012)Optimized the brushless DC wheel motors using bat algorithm with superior results.
- (Yang, X. S., Karamanoglu, M., Fong, S, 2012)Used the bat algorithm to study topological shape optimization in microelectronic applications so that materials of different thermal properties can be placed in such a way that the heat transfer is most efficient under stringent constraints.
- (Jacob, 2014) used Bat algorithm to schedule resources in heterogeneous cloud computing environment with high accurate values as compared to other optimization techniques
- (Abdel-Rahman, E. M., Ahmad, A. R, 2012)Presented a study for full body human pose estimation using bat algorithm, and they concluded that BA performs better than particle swarm optimization (PSO), particle filter (PF) and annealed particle filter (APF).
- (P.R Srivastava, 2014)Proposed a model using the meta-heuristic bat algorithm to estimate the test effort. The proposed model is then used to optimize the effort by iteratively improving the solutions.
- (Lemma, T. A., Bin Mohd Hashim, F, 2011)Used fuzzy systems and bat algorithm for energy modelling, and later Tamiru and Hashim (2013) applied bat algorithm to study fuzzy systems and to model energy changes in a gas turbine.
- (Du, 2012) presented a variant of bat algorithm with mutation for image matching, and they indicated that their bat-based model is more effective and feasible in imagine matching than other models such as differential evolution and genetic algorithms.

## 2.2 PART C (SIMILAR TYPES OF WORK)

Review on COCOMO optimization work which have already been done

- (C.F, 1996)Performed an empirical validation of four algorithmic models (SLIM, COCOMO, Estimates and FPA) using data from projects outside the original model development environments without re-calibrating the models. The results indicate to what extent these models are generalizable to different environments. Most models showed a strong over-estimation bias and large estimation errors with the mean absolute relative error (MARE) ranging from an average of 57 percent to almost 800 percent. Thus we can combine these models with non-algorithmic models such as (PSO, Genetic Algorithm, Fuzzy Logic, Neural Network)(S K Sehra, 2011) and get better results.

While doing this thesis, we have gone through some literature where non algorithmic models are used with COCOMO model, which can be shown as:

- (Basili, 1981) Presented a model process which permits the development of effort estimation model for any particular organization. The model is based on data collected by that organization which captures its particular environment factors and differences in its particular projects. The process provides capability for producing a model tailored to the organization which can be more effective than any model originally developed for other environment. They demonstrated it using data collected for the Software Engineering laboratory at NASA and came to conclusion that

  Effort= a (size in KLOC) $^{b}$ + c $*$ (methodology)

- (F, 2006) Presented two new model structures to estimate the effort required for the development of software projects using Genetic Algorithms (GAs). A modified version of the famous COCOMO model provided to explore the effect of the software development adopted methodology in effort computation. The

performances of the developed models were tested on NASA software project dataset. The developed models were able to provide good estimation capabilities.

- (Anish M, Kamal P and Harish M, 2010) Presented two new models, based on fuzzy logic. Rather than using a single number, the software size is regarded as a triangular fuzzy number. We can optimize the estimated effort for any application by varying arbitrary constants for these models. The developed models were tested on 10 NASA software projects, on the basis of four criterions for assessment of software cost estimation models. Comparison of all the models was done and it is found that the developed models provide better estimation.
- (Reddy, 2010) Proposed three software effort estimation models by using soft computing techniques: Particle Swarm Optimization with inertia weight for tuning effort parameters in COCOMO Model. The performance of the developed models was tested by NASA software project dataset provided by (Basili, 1981). The developed models were able to provide good estimation capabilities.
- (Sheta, 2007) Proposed Differential Evolution (DE) as an alternative technique and powerful tool to estimate the COCOMO model parameters. The performances of the developed models were tested on NASA software project dataset provided by (Basili, 1981). The developed COCOMO-DE model was able to provide good estimation capabilities.
- (Lin J.-C. , 2010) Used Pearson product moment correlation coefficient to select several factors then used K-Means clustering algorithm to software project clustering. After project clustering, he use Particle Swarm Optimization that take mean of MRE (MMRE) as a fitness value and N-1 test method to optimization of COCOMO parameters.
- (Anna Galinina, Olga Burceva, Sergei Parshutin, 2012)Used Genetic algorithm to optimize COCOMO model coefficients which were determined in 1981 by means of the regression analysis of statistical data based on 63 different types of project

data. The proposed algorithm was tested and the obtained results were compared with the ones obtained using the current COCOMO model coefficients. Coefficients optimized by the GA in the organic mode produces better results in comparison with the results obtained using the current COCOMO model coefficients.

- (Vishali, Anshu Sharma, Suchika Malik, 2014) Used Genetic algorithm and Ant Colony Optimization to optimize COCOMO model coefficients which were determined in 1981 by means of the regression analysis of statistical data based on 63 different types of project data. Results were better for GA and ACO as compared to normal COCOMO Model.

# Chapter Three: **OPTIMIZED COCOMO MODEL WITH BAT ALGORITHM**

In this chapter, we describe the proposed methodology by us for optimizing coefficients (a & b) in Intermediate COCOMO Model of Effort estimation (for all types of system i.e. organic, semi-detached and embedded) with Bat Algorithm. We have also given the datasets which have been used.

## 3.1 PROPOSED METHODOLOGY

Bat Algorithm and Intermediate COCOMO Model of effort Estimation have already been discussed in detail in Chapter 2.

Here, we discuss how Bat algorithm is used to find a_best and b_best value for Intermediate COCOMO model (for all types of system i.e. organic, semi-detached and embedded). To derive the new values of coefficients for all types of system (organic, semi-detached and embedded), we have taken NASA 63 dataset and divided it into three sections:

- Dataset for Organic System
- Dataset for Semi-Detached System
- Dataset for Embedded System

Then we have applied Bat Algorithm to each section and calculate new values of a and b for all three types of system (organic, semi-detached and embedded).

### 3.1.1 DATASETS

Each dataset consist of Project No, its size in KLOC, Effort Adjustment Factor and actual effort for its development.

**a) Dataset for Organic System**

**Table 3.1: NASA 63 dataset for Organic systems**

| PROJECT NO. | KLOC | EAF | Actual Effort |
|---|---|---|---|
| 1 | 132 | 0.320461 | 243 |
| 2 | 60 | 0.998141 | 240 |
| 3 | 16 | 0.656169 | 33 |
| 4 | 4 | 1.865036 | 43 |
| 5 | 25 | 0.85243 | 79 |
| 6 | 9.4 | 1.657303 | 88 |
| 7 | 15 | 0.68887 | 55 |
| 8 | 60 | 0.372242 | 47 |
| 9 | 15 | 0.358804 | 12 |
| 10 | 6.2 | 0.387744 | 8 |
| 11 | 3 | 0.964898 | 8 |
| 12 | 5.3 | 0.254454 | 6 |
| 13 | 45.5 | 0.587344 | 45 |
| 14 | 28.6 | 1.069813 | 83 |
| 15 | 30.6 | 1.336619 | 87 |
| 16 | 35 | 0.872678 | 106 |
| 17 | 73 | 0.824729 | 126 |
| 18 | 24 | 1.28037 | 176 |
| 19 | 10 | 2.304555 | 122 |
| 20 | 5.3 | 1.154275 | 14 |
| 21 | 4.4 | 0.77736 | 20 |
| 22 | 25 | 1.089608 | 130 |
| 23 | 23 | 1.006967 | 70 |
| 24 | 6.7 | 2.125489 | 57 |
| 25 | 10 | 0.386126 | 15 |

**b) Dataset for semi-detached system**

**Table 3.2: NASA 63 Dataset for Semi Detached**

| PROJECT NO. | KLOC | EAF | Actual Effort |
|---|---|---|---|
| 1 | 293 | 0.842266296 | 1600 |
| 2 | 1150 | 0.675539854 | 6600 |
| 3 | 77 | 0.908416597 | 539 |
| 4 | 13 | 2.810694546 | 98 |
| 5 | 2.14 | 0.994394537 | 7.3 |
| 6 | 62 | 3.439167383 | 1063 |
| 7 | 13 | 2.178793679 | 82 |
| 8 | 23 | 0.380665662 | 36 |
| 9 | 464 | 0.758080034 | 1272 |
| 10 | 8.2 | 1.376017605 | 41 |
| 11 | 28 | 0.446598709 | 50 |

**c) Dataset for Embedded system**

**Table 3.3 : NASA 63 Dataset for Embedded system**

| PROJECT NO. | SIZE (KLOC) | EFFORT ADJUSTMENT FACTOR (EAF) | Actual Effort |
|---|---|---|---|
| 1 | 113 | 2.288114989 | 2040 |
| 2 | 6.9 | 0.531284635 | 8 |
| 3 | 22 | 5.509905793 | 1075 |
| 4 | 30 | 2.013772319 | 423 |
| 5 | 29 | 1.730150413 | 321 |
| 6 | 32 | 1.730150413 | 218 |
| 7 | 37 | 0.936262003 | 201 |
| 8 | 3 | 4.945017866 | 60 |
| 9 | 3.9 | 3.043530256 | 61 |
| 10 | 6.1 | 2.374955594 | 40 |
| 11 | 3.6 | 1.947463587 | 9 |

| | | | |
|---|---|---|---|
| 12 | 320 | 3.271167233 | 11400 |
| 13 | 299 | 3.487908449 | 6400 |
| 14 | 252 | 0.846066335 | 2455 |
| 15 | 118 | 0.96815931 | 724 |
| 16 | 90 | 0.702502121 | 453 |
| 17 | 38 | 1.163900531 | 523 |
| 18 | 48 | 0.952487929 | 387 |
| 19 | 1.98 | 0.994394537 | 5.9 |
| 20 | 390 | 0.569092582 | 702 |
| 21 | 42 | 2.301870948 | 605 |
| 22 | 23 | 1.476736523 | 230 |
| 23 | 91 | 0.301677206 | 156 |
| 24 | 6.3 | 0.340097967 | 18 |
| 25 | 27 | 2.660867206 | 958 |
| 26 | 17 | 3.306315857 | 237 |
| 27 | 9.1 | 1.053619034 | 38 |

### 3.1.2 MODEL DESCRIPTION

The following is the methodology employed to tune the parameters in each proposed modes (organic, semi-detached and imbedded).

**Input:** Size of Software Projects, Measured Efforts, Effort Adjustment factor-EAF.

**Output:** Optimized coefficients a_best, b_best and $f_{min}$(Least MMRE of all projects)

**Step 1:** Initialize the bat population $X_i$ (i = 1, 2... n) and $V_i$ , where $X_i$ represents the position or solution and $V_i$ represents the velocity of Bats. Each bat tries to find value for a & b such that MMRE of all project decreases with iteration and after all iterations we get the bat with least MMRE as the best Bat and its values as result.

**Step 2:** Define pulse frequency $f_i$ at $x_i$. Thus each bat will have frequency. We have to set $F_{min}$ and $F_{max}$ according to our problem as detectable range should be chosen such that it is comparable to the size of the domain of interest.

**Step 3:** Initialize pulse rates $r_i$ and the loudness $A_i$, where (i= 1 to n). The rate of pulse can simply be in the range of [0, 1] where 0 means no pulses at all, and 1 means the

maximum rate of pulse emission. For setting loudness we can use $A_{max} = 1$ and $A_{min} = 0$, assuming $A_{min} = 0$ means that a bat has just found the prey and temporarily stop emitting any sound.

**Step 4:** Repeat the following steps 5 to 9 until number of iterations specified by the user Exhaust.

**Step 5:** for i = 1, 2… n do // for all the Bats

**Step 6:** Generate new solutions by adjusting frequency, and updating velocities and locations/solutions [equations (8) to (10)]

**Step 7:** For each bat position with values of tuning parameters (a and b), evaluate the fitness function. The fitness function here is Mean Magnitude of Relative Error (MMRE). Thus we are calculating MMRE for each Bat and considering all projects at one time for each bat. The objective in this method is to minimize the MMRE by selecting appropriate values for a and b and then select the least MMRE among all bats as the final result. All Bats fitness is stored in array Fitness (i)

**Step 8:** if (rand > $r_i$), then
Generate a new solution around the current global best Solution using equation 11 and evaluate its Fitness as F_New.

**Step 9:** If (rand < Ai && F_New < Fitness (i))
Accept the new solution and update the Fitness (i) =F_New
Increase $r_i$ and reduce $A_i$ using equation (12)

**Step 10:** Post process the result

**Step 11:** Stop

**The proposed model is implemented in MATLAB for all modes.**

### 3.1.3 PROPOSED MODES

1) **Organic mode with Bat Algorithm:** In this we have taken 25 organic type projects from NASA 63 Dataset and then applied the model described in 3.1.2 to get values a_best, b_best and fmin (Mean Magnitude of Relative Error). In the whole process we are calculating a_best, b_best and fmin (MMRE) for all bats and then taking the least fmin among all bats as global best value and its corresponding best value as final a_best and b_best.

For tuning the parameters, we have taken the values as illustrated in Table 3.4.

**Table 3.4: Parameters value for Organic Mode**

| **Parameter** | **Value** |
|---|---|
| Dimension (d) | 2; ( a and b) |
| F_Min | 0 |
| F_Max | 4 |
| Lower Bound | 0 |
| Upper Bound | 4 |
| No. of Iterations | 400 |
| No. of Bats | 27 |
| Pulse Rate Range | [0,1] |
| Amplitude Range | [0,1] |
| Alpha(α) | 0.976 |
| Gamma(ϒ) | 0.976 |

2) **Semi Detached mode with Bat Algorithm:** In this we have taken 11 semi-detached type projects from NASA 63 Dataset and then applied the model described in 3.1.2 to get values a_best, b_best and fmin (Mean Magnitude of

Relative Error). In the whole process we are calculating a_best, b_best and fmin (MMRE) for all bats and then taking the least fmin among all bats as global best value and its corresponding best value as final a_best and b_best.

For tuning the parameters, we have taken the values as illustrated in Table 3.5.

**Table 3.5: Parameters value for Semi-detached Mode**

| **Parameter** | **Value** |
|---|---|
| Dimension (d) | 2; ( a and b) |
| F_Min | 0 |
| F_Max | 4 |
| Lower Bound | 0 |
| Upper Bound | 4 |
| No. of Iterations | 400 |
| No. of Bats | 27 |
| Pulse Rate Range | [0,1] |
| Amplitude Range | [0,1] |
| Alpha(α) | 0.976 |
| Gamma(ϒ) | 0.976 |

3) **Embedded mode with Bat Algorithm:** In this we have taken 27 embedded type projects from NASA 63 Dataset and then applied the model described in 3.1.2 to get values a_best, b_best and $f_{min}$ (Mean Magnitude of Relative Error). In the whole process we are calculating a_best, b_best and $f_{min}$ (MMRE) for all bats and then taking the least fmin among all bats as global best value and its corresponding best value as final a_best and b_best.

For tuning the parameters, we have taken the values as illustrated in Table 3.6.

**Table 3.6: Parameters value for Embedded Mode**

| Parameter | Value |
|---|---|
| Dimension (d) | 2; ( a and b) |
| F_Min | 0 |
| F_Max | 3 |
| Lower Bound | 0 |
| Upper Bound | 3 |
| No. of Iterations | 400 |
| No. of Bats | 27 |
| Pulse Rate Range | [0,1] |
| Amplitude Range | [0,1] |
| Alpha(α) | 0.976 |
| Gamma(ϒ) | 0.976 |

# Chapter Four: **RESULTS AND DISCUSSION**

During the experiments, the initial population of 27 bats was generated. Then the optimization of the COCOMO model coefficients was performed using the proposed algorithm. Based on the fact that each of the three modes of COCOMO model has its own coefficients, experiments were performed using datasets according to each mode. Experiments were performed by changing the Bat algorithm parameters (No. of iterations, No. of Bats, Fmin, Fmax, Lower Bound, Upper Bound, Pulse Rate, Loudness, alpha and gamma). Value used for parameters is already discussed in chapter 3.

## 4.1 ORGANIC MODE EXPERIMENTS

In experiments using organic mode datasets, the best result was achieved using 400 iterations and 27 bats. As a result of algorithm execution, we got different values at each execution and mostly the MMRE with Bat Execution was less than that of COCOMO.

**Solution:**

**MMRE by COCOMO: 0.3720**

**Fmin (Least Mean Magnitude of Relative Error) by Bat Algorithm: 0.3093**

**a_best=3.63**

**b_best=0.916**

Figure 4.1 depicts how bats are searching for coefficient a and b in the given range [0-4], while execution and Figure 4.2 depicts MMRE for all 27 bats. Bat no. 1 has got the least MMRE for Organic projects.

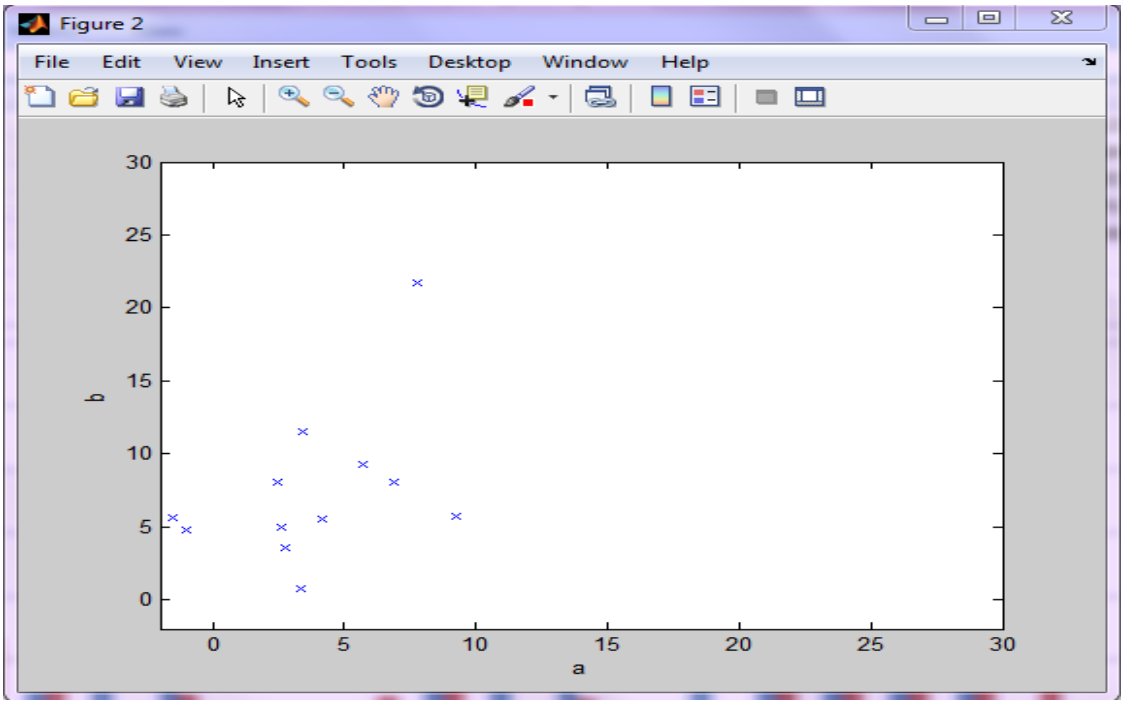


**Figure 4.1: Plot of Bats searching for a_best and b_best for Organic Projects**

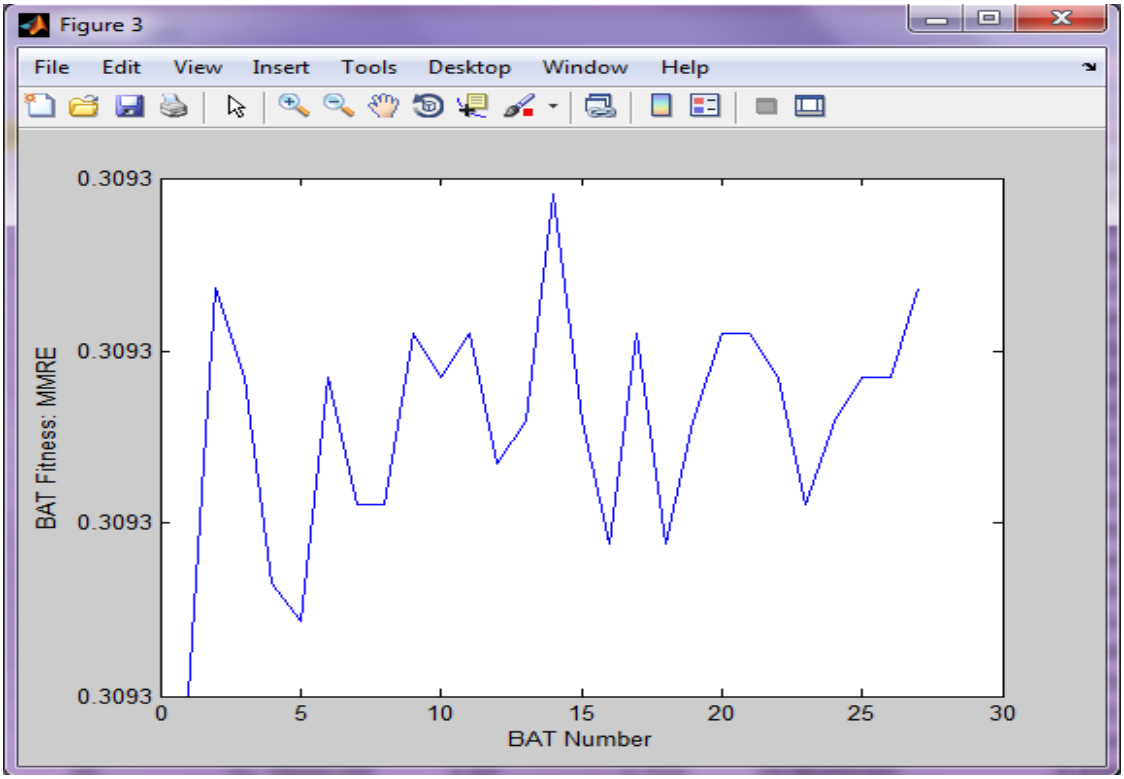


**Figure 4.2: MMRE of organic projects for all bats.**

Table 4.1 depicts comparison among the organic mode projects real effort, predicted development effort using Bat coefficients and current COCOMO model coefficients:

**Table 4.1: Comparison of MRE of Actual and Proposed Model for Organic Mode**

| No | Actual Effort | COCOMO Effort | BAT-COCOMO Effort | MRE (COCOMO-Actual) | MRE(BAT-COCOMO) |
|---|---|---|---|---|---|
| 1 | 243 | 172.79365 | 101.88905 | 0.2889150 | 0.580703482 |
| 2 | 240 | 235.18016 | 154.12925 | 0.0200826 | 0.357794791 |
| 3 | 33 | 38.591500 | 30.192302 | 0.169439 | 0.085081734 |
| 4 | 43 | 25.585869 | 24.10353 | 0.4049797 | 0.439452602 |
| 5 | 79 | 80.102434 | 59.030763 | 0.0139548 | 0.252775149 |
| 6 | 88 | 55.76171 | 46.848280 | 0.3663441 | 0.467633176 |
| 7 | 55 | 37.86021 | 29.877447 | 0.3116324 | 0.456773684 |
| 8 | 47 | 87.706981 | 57.48023 | 0.8661059 | 0.222983747 |
| 9 | 12 | 19.719825 | 15.561931 | 0.6433188 | 0.29682761 |
| 10 | 8 | 8.4276470 | 7.4865632 | 0.0534558 | 0.064179592 |
| 11 | 8 | 9.7860786 | 9.5814469 | 0.2232598 | 0.197680872 |
| 12 | 6 | 4.6908216 | 4.2555196 | 0.2181963 | 0.290746719 |
| 13 | 45 | 103.50336 | 70.394317 | 1.3000748 | 0.564318161 |
| 14 | 83 | 115.78230 | 83.800340 | 0.3949675 | 0.009642655 |
| 15 | 87 | 155.29772 | 111.38714 | 0.7850313 | 0.280312039 |
| 16 | 106 | 116.75499 | 82.248309 | 0.1014621 | 0.22407255 |
| 17 | 126 | 238.75377 | 152.41286 | 0.8948712 | 0.209625924 |
| 18 | 176 | 115.26760 | 85.411355 | 0.3450704 | 0.51470821 |
| 19 | 122 | 82.744103 | 68.943559 | 0.3217696 | 0.434888856 |
| 20 | 14 | 21.278888 | 19.304235 | 0.5199205 | 0.378873977 |
| 21 | 20 | 11.786842 | 10.963051 | 0.4106578 | 0.45184741 |
| 22 | 130 | 102.38993 | 75.45533 | 0.2123851 | 0.41957434 |
| 23 | 70 | 86.692084 | 64.604766 | 0.2384583 | 0.077074765 |
| 24 | 57 | 50.117267 | 44.060559 | 0.1207496 | 0.227007731 |
| 25 | 15 | 13.863695 | 11.551427 | 0.0757536 | 0.229904808 |
| | | | | **∑MRE=9.300857605** | **∑MRE=7.734484583** |
| | | | | **MMRE:9.300857/25 =0.3720** | **MMRE:7.73448/25= 0.3093** |

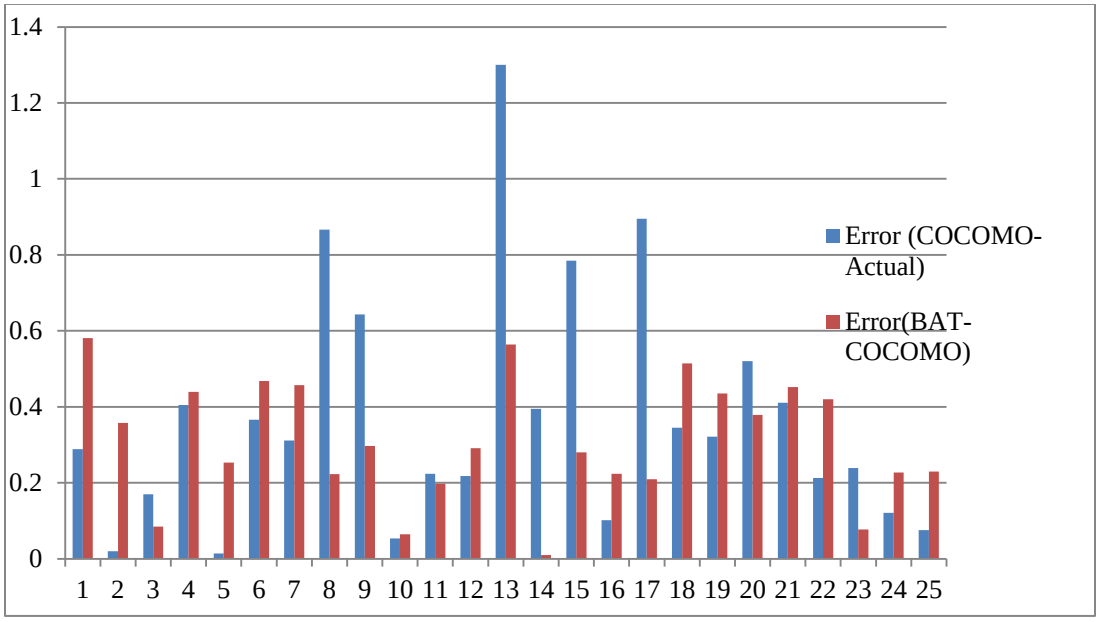


**Figure 4.3: Comparison of MRE with COCOMO Model and Proposed Bat Model for all organic projects.**

From Table 4.1 we can draw bar chart as depicted in Figure 4.3, which shows comparison of MRE for all organic projects by Actual COCOMO Model and Bat-COCOMO Model.

Results show that MMRE by proposed Bat-COCOMO Model (0.3093) is less than that of Original COCOMO Model (0.3720). From figure we can also see that MRE of most of the projects by proposed Bat-COCOMO Model is less as compared to that of Original COCOMO Model.

## 4.2 SEMI DETACHED MODE EXPERIMENTS

In experiments using semi-detached mode datasets, the best result was achieved using 400 iterations and 27 bats. As a result of algorithm execution, we got different values at each execution and mostly the MMRE with Bat Execution was less than that of COCOMO.

**Solution:**

**MMRE by COCOMO: 0.2337**

**Fmin (Least Mean Magnitude of Relative Error) by Bat Algorithm: 0.2157**

**a_best= 2.9383**

**b_best= 1.1009**

Figure 4.4 depicts how bats are searching for coefficient a and b in the given range [0-3], while execution and Figure 4.2 depicts MMRE for all 27 bats. Bat no. 23 has got the least MMRE for semi-detached projects.

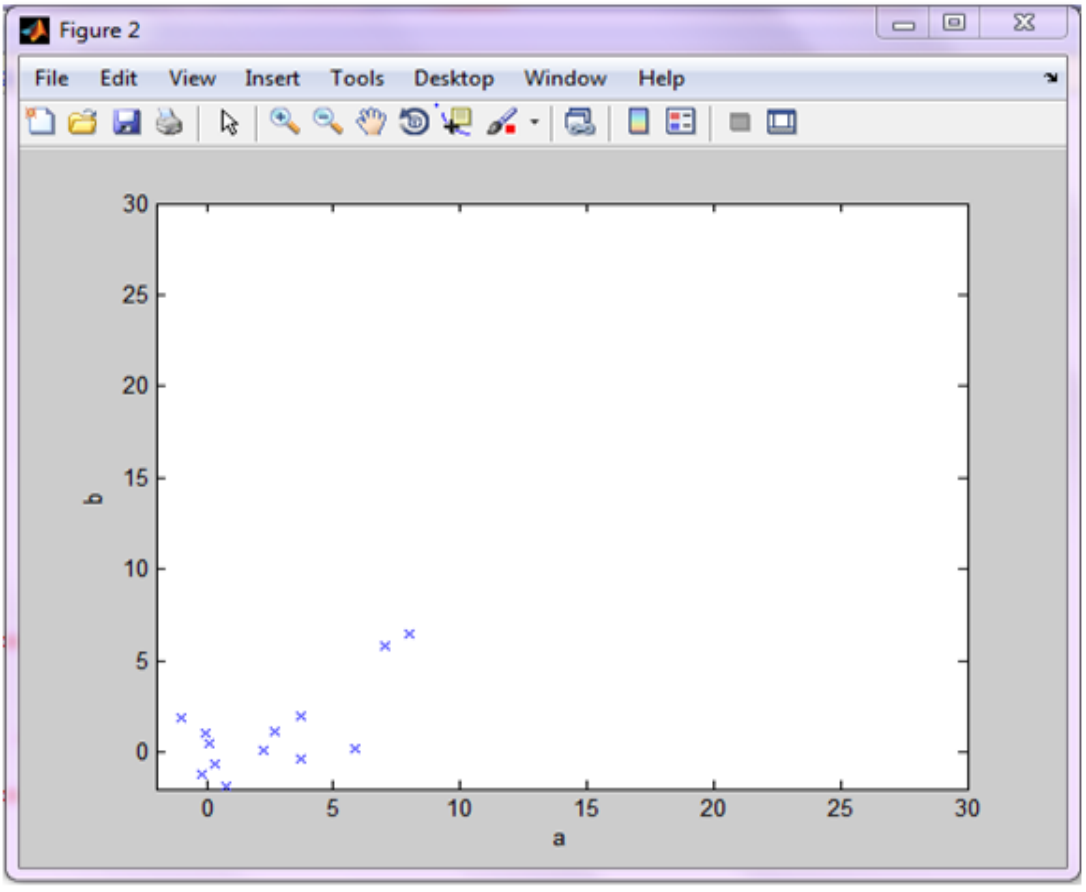


**Figure 4.4: Plot of Bats searching for a_best and b_best for semi-detached projects**

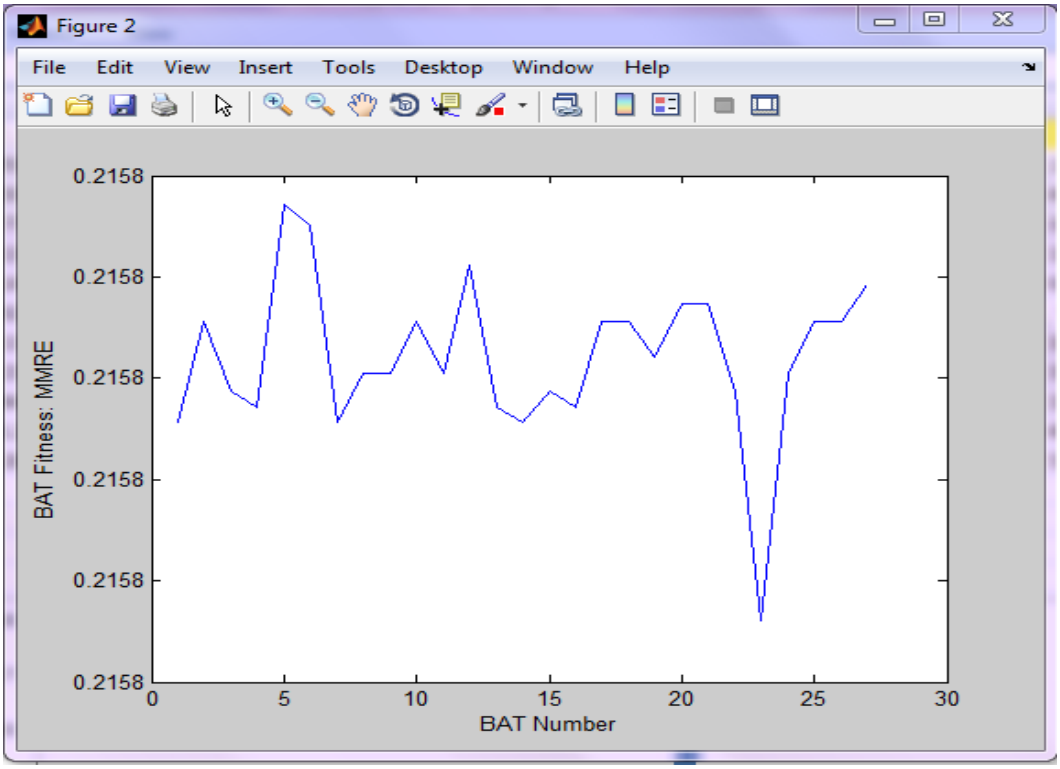


**Figure 4.5: MMRE of semi-detached projects for all bats**

Table 4.2 depicts comparison among the semi-detached mode projects real development effort, the predicted development effort using Bat coefficients and the predicted development effort using current COCOMO model coefficients.

**Table 4.2: Comparison of MRE of Actual and Proposed Model for Semi-detached Mode**

| No. | Actual Effort | COCOMO Effort | BAT Effort | MRE(COCOMO-Actual) | MRE( Bat-COCOMO) |
|---|---|---|---|---|---|
| 1 | 1600 | 1463.7348 | 1286.234 | 0.085165 | 0.196103 |
| 2 | 6600 | 5429.4246 | 4648.034 | 0.177359 | 0.295752 |
| 3 | 539 | 353.4076 | 318.5802 | 0.344327 | 0.408942 |

| | | | | | |
|---|---|---|---|---|---|
| 4 | 98 | 149.1253 | 139.0753 | 0.521687 | 0.419135 |
| 5 | 7.3 | 6.99428 | 6.75161 | 0.041878 | 0.075121 |
| 6 | 1063 | 1049.6711 | 950.1528 | 0.012538 | 0.106159 |
| 7 | 82 | 115.5989 | 107.8083 | 0.409743 | 0.314736 |
| 8 | 36 | 38.2648 | 35.2992 | 0.062912 | 0.01946 |
| 9 | 1272 | 2204.6337 | 1920.351 | 0.733202 | 0.509710 |
| 10 | 41 | 43.57292 | 40.99565 | 0.062755 | 0.000106 |
| 11 | 50 | 55.957166 | 51.42684 | 0.119143 | 0..28536 |
| | | | | **∑MRE= 2.570714** | **∑MRE= 2.373768** |
| | | | | **MMRE=2.570714/11 =0.2337** | **MMRE=2.373768/11 =0.2157** |

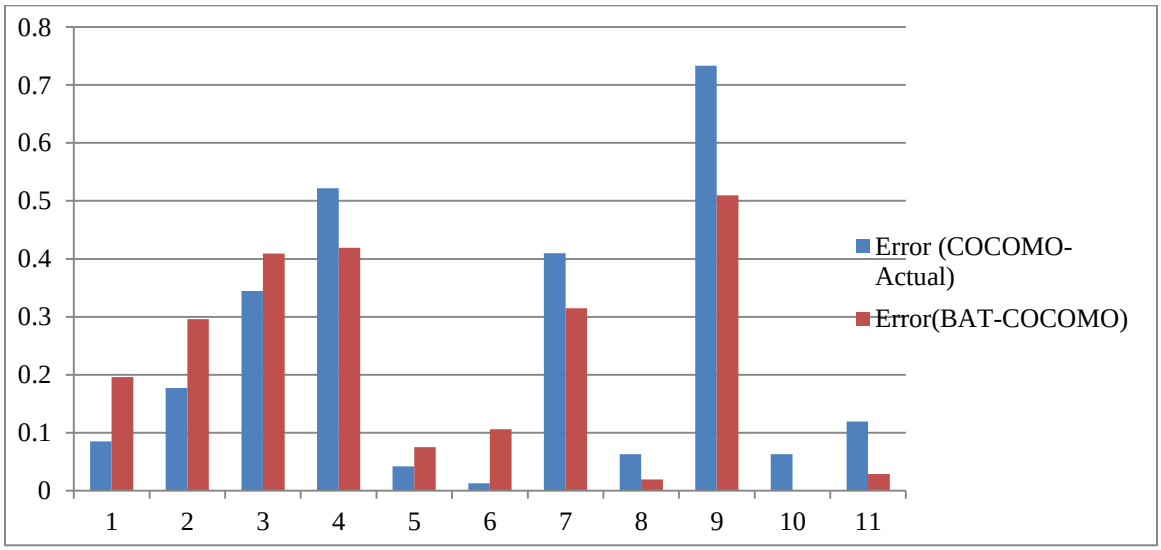


**Figure 4.6: Comparison of MRE with COCOMO Model and Proposed Bat Model for all semi-detached projects.**

From Table 4.2 we can draw bar chart as depicted in Figure 4.6, which shows comparison of MRE for all semi-detached projects by Actual COCOMO Model and Bat-COCOMO Model.

Results show that MMRE by proposed Bat-COCOMO Model (0.2157) is less than that of Original COCOMO Model (0.2337). From figure we can also see that MRE of most of the projects by proposed Bat-COCOMO Model is less as compared to that of Original COCOMO Model.

## 4.3 EMBEDDED MODE EXPERIMENTS

In experiments using embedded mode datasets, the best result was achieved using 400 iterations and 27 bats. As a result of algorithm execution, we got different values at each execution and mostly the MMRE with Bat Execution was less than that of COCOMO.

**Solution:**

**MMRE by COCOMO: 0.3921**

**Fmin (Least Mean Magnitude of Relative Error) by Bat Algorithm: 0.3826**

**a_best= 2.8908**

**b_best=1.1689**

Figure 4.7 depicts how bats are searching for coefficient a and b in the given range [0-3], while execution and Figure 4.8 depicts MMRE for all 27 bats. Bat no. 5 has got the least MMRE for embedded projects.

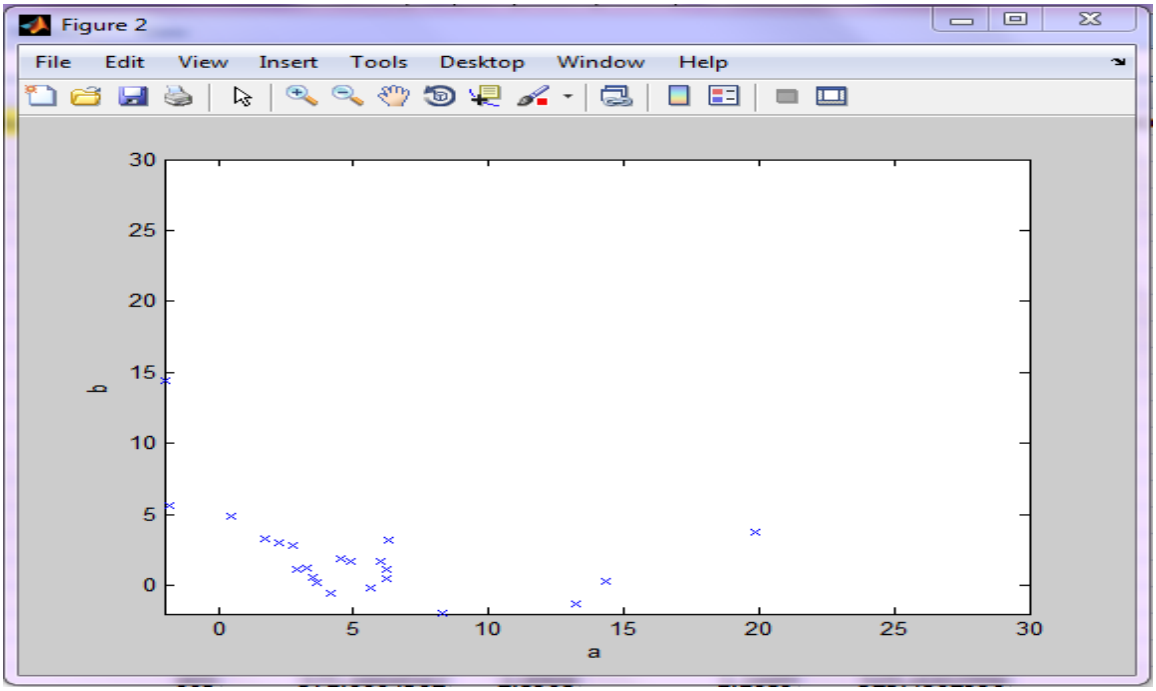


**Figure 4.7: Plot of Bats searching for a_best and b_best for embedded projects**

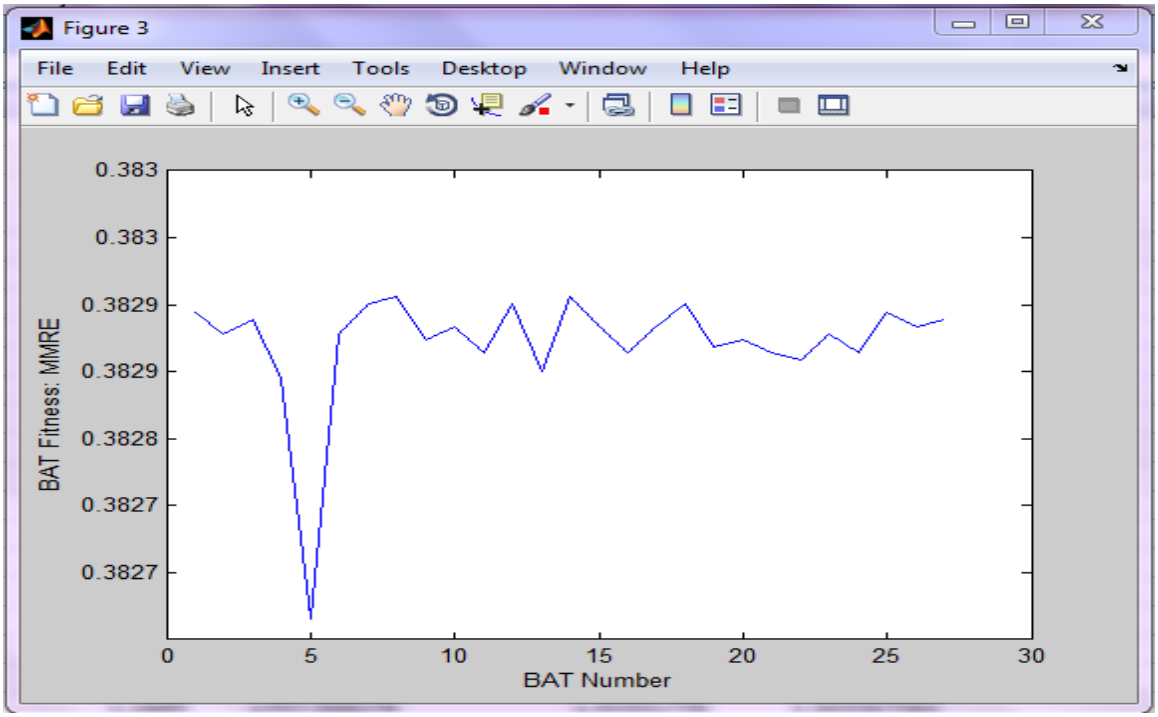


**Figure 4.8: MMRE of embedded projects for all bats**

Table 4.3 depicts comparison among the embedded mode projects real development effort and predicted effort by Bat coefficients and COCOMO model coefficients.

**Table 4.3: Comparison of MRE of Actual and Proposed Model for Embedded Mode**

| NO | Actual Effort | COCOMO Effort | BAT Effort | MRE (COCOMO-Actual) | MRE (BAT-COCOMO) |
|---|---|---|---|---|---|
| 1 | 2040 | 1863.502628 | 1660.883936 | 0.086518319 | 0.185841208 |
| 2 | 8 | 15.10442497 | 14.68506943 | 0.888053122 | 0.835633679 |
| 3 | 1075 | 629.8098112 | 590.6362406 | 0.414130408 | 0.450570939 |
| 4 | 423 | 333.9749579 | 310.1954673 | 0.210461092 | 0.266677382 |
| 5 | 321 | 275.4986592 | 256.1526933 | 0.141748725 | 0.202016532 |
| 6 | 218 | 310.042954 | 287.3900424 | 0.422215385 | 0.318302947 |
| 7 | 201 | 199.7089364 | 184.2834948 | 0.006423202 | 0.083166692 |
| 8 | 60 | 51.74535872 | 51.62890875 | 0.137577355 | 0.139518188 |
| 9 | 61 | 43.63281003 | 43.18084022 | 0.284708032 | 0.292117373 |
| 10 | 40 | 58.23838554 | 56.83888946 | 0.455959638 | 0.420972237 |
| 11 | 9 | 25.36240179 | 25.16224492 | 1.818044644 | 1.795804991 |
| 12 | 11400 | 9290.535617 | 8016.607173 | 0.185040735 | 0.296788844 |
| 13 | 6400 | 9131.214279 | 7895.782472 | 0.426752231 | 0.233716011 |
| 14 | 2455 | 1804.027257 | 1568.265371 | 0.265162013 | 0.361195368 |
| 15 | 724 | 830.5451876 | 739.2439015 | 0.147161861 | 0.021055113 |
| 16 | 453 | 435.4082789 | 390.822677 | 0.038833821 | 0.137256784 |
| 17 | 523 | 256.3387213 | 236.3431168 | 0.509868602 | 0.548101115 |
| 18 | 387 | 277.6559578 | 254.1443206 | 0.282542745 | 0.343296329 |
| 19 | 5.9 | 6.319969602 | 6.387761855 | 0.071181288 | 0.082671501 |
| 20 | 702 | 2049.361082 | 1757.504274 | 1.91931778 | 1.503567342 |
| 21 | 605 | 571.6604562 | 525.4302996 | 0.055106684 | 0.131520166 |
| 22 | 230 | 178.046914 | 166.7418938 | 0.225882983 | 0.275035245 |
| 23 | 156 | 189.4742512 | 170.0137422 | 0.214578533 | 0.089831681 |
| 2 | 18 | 8.669034405 | 8.452228904 | 0.518386978 | 0.530431728 |
| 25 | 958 | 388.8819174 | 362.37844 | 0.59406898 | 0.621734405 |
| 26 | 237 | 277.3582427 | 262.2008711 | 0.170287944 | 0.10633279 |
| 27 | 38 | 41.75334845 | 40.24622374 | 0.098772328 | 0.059111151 |
| | | | | **∑MRE=10.5887** | **∑MRE=10.33226** |
| | | | | **MMRE:10.5887 8/27=0.3921** | **MMRE:10.33226 7/27=0.3826** |

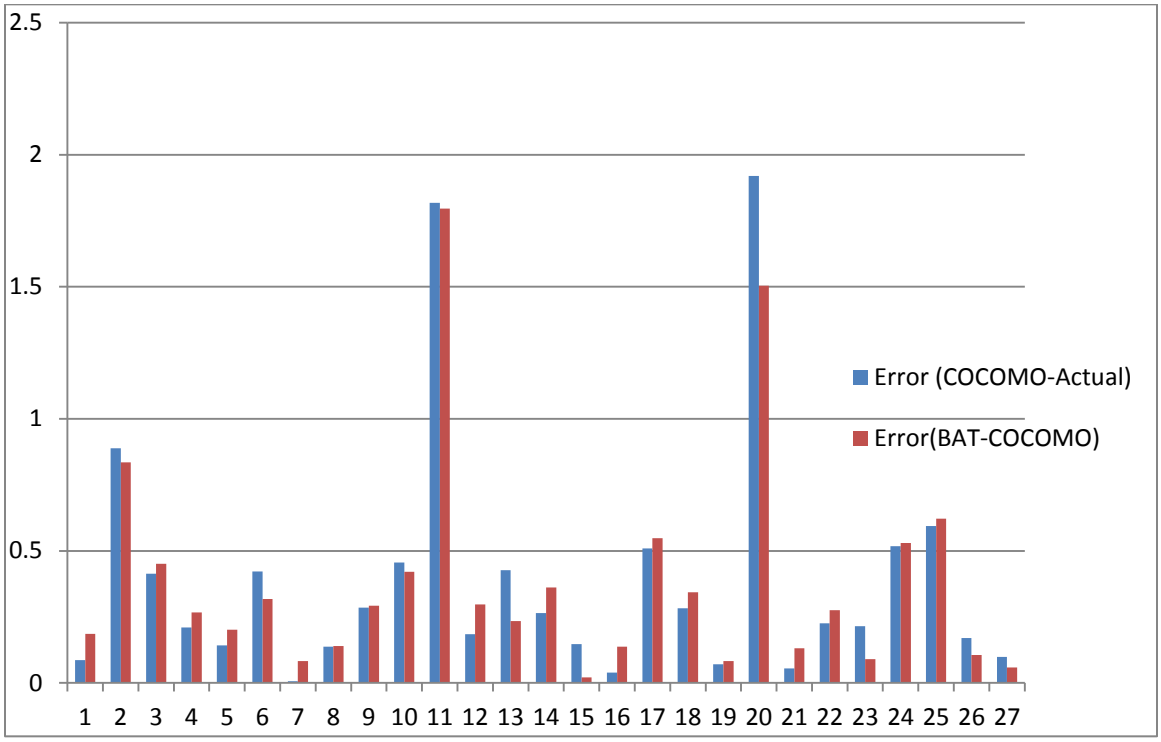


**Figure 4.9: Comparison of MRE with COCOMO Model and Proposed Bat Model for all embedded projects**

From Table 4.3 we can draw bar chart as depicted in Figure 4.9, which shows comparison of MRE for all embedded projects by Actual COCOMO Model and Bat-COCOMO Model.

Results show that MMRE by proposed Bat-COCOMO Model (0.3826) is less than that of Original COCOMO Model (0.3921). From figure we can also see that MRE of most of the projects by proposed Bat-COCOMO Model is less as compared to that of Original COCOMO Model.

# Chapter Five: **CONCLUSION AND FUTURE WORK**

## 5.1 CONCLUSION:

The objective of this research was to optimize the Intermediate COCOMO model coefficients using the Bat Algorithm. The task of the COCOMO coefficient optimization is not new; different methods such as neural networks, fuzzy algorithms, genetic algorithm, Particle swarm optimization etc. were applied to it by a number of scientists. But none have applied Bat Algorithm for it.

The current research proposes a Bat algorithm based method for optimization of the Intermediate COCOMO model coefficients for organic, semi-detached and embedded modes. In a series of experiments, the proposed algorithm was tested and the obtained results were compared with the ones obtained using the current COCOMO model coefficients. The results show that in most cases the results obtained using the coefficients optimized by the proposed algorithm are close and better compared to the ones obtained using the current coefficients. We have Compared results for all modes (organic, semidetached and embedded) and found out that mostly the results by proposed methodology (Bat-COCOMO) produces better results in comparison with the results obtained using the current COCOMO model coefficients.

According to the findings of the research, it should be stated that having the appropriate statistical data describing the software development projects, we can generate new model or optimize existing model such as COCOMO with Bat algorithm, which is amongst the new meta-heuristic Algorithms.

We have concluded the final result of this research work in Table 5.1.

**Table 5.1: Actual and optimized values of a and b for all modes in Intermediate Model**

| Project | Actual a | Actual b | Bat-COCOMO a | Bat-COCOMO b | MMRE by COCOMO | MMRE by Bat-CCOMO |
|---|---|---|---|---|---|---|
| Organic | 3.2 | 1.05 | 3.63 | 0.916 | 0.3720 | 0.3093 |
| Semidetached | 3.0 | 1.12 | 2.9383 | 1.1009 | 0.2337 | 0.2157 |
| Embedded | 2.8 | 1.20 | 2.8908 | 1.1689 | 0.3921 | 0.3826 |

## 5.2 FUTURE WORK:

- A lot of work can be done in Software Engineering using Bat algorithm.
- We can use Bat Algorithm to optimize other software effort estimation technique such as Function Point Analysis, Use case points or COCOMO II model parameters.
- We can compare the results of BAT-COCOMO optimization with other new optimization algorithms (PSO, Genetic Algorithm, Firefly Algorithm, Ant Colony Optimization) etc.
- We can use Bat Algorithm on our own datasets, to get a new model.
- We can use Bat Algorithm in other software Engineering domains such as Software reliability, Testing etc.

# Bibliography

"The Nature of Mathematical Programming. (n.d.).

*Five reason why software projects fail.* (2002, may 20). Retrieved from Computerworld.

(2009). *The 10 laws of chaos.* The Standish group International, Inc.

*Facts about COCOMO And Costar.* (2012). Retrieved from http://www.softstarsystems.com/.

Abdel-Rahman, E. M., Ahmad, A. R. (2012). A metaheurisic bat inspired algorithm for full body human pose estimation. *Ninth Conference on Computer and Robot Vision*, (pp. 369–375).

Abraham A., C. G. (2006). Stigmergic Optimization. *Springer*.

Albrecht, A. (1979). Measuring Application Development Productivity. *In Proc of the IBM Applications Development Symposium* , (pp. 83-92).

Anish M, Kamal P and Harish M. (2010). Software Cost Estimation using Fuzzy logic. *ACM SIGSOFT Software Engineering Notes*, 1-7.

Anna Galinina, Olga Burceva, Sergei Parshutin. (2012). The Optimization of COCOMO Model Coefficients Using Genetic Algorithms. *Information Technology and Management Science*, 45-52.

Banks A., J. V. (2007). A Review of Particle Swarm Optimization- Part I: Background and Development, Natural Computation. *springer*, 467–484.

Basili, J. B. (1981). A meta model for software development resource expenditures. *Fifth International conference on software Engineering*, (pp. 107-129).

Boehm., B. (1981). *Software Engineering Economics.* New Jersey.

Bora, T. C. (2012). Bat-inspired optimization approach for the brushless DC wheel motor problem. *IEEE Trans. Magnetics*, 947-950.

Brajesh Kumar Singh, S. T. (2013). Tuning of Cost Drivers by Significance Occurrences and Their Calibration with Novel Software Effort Estimation Method. *Advances in Software Engineering*.

C.F, K. (1996). An Empirical Validation of Software Cost Estimation Models. *ACM*, 416-429.

Dolado, J. J. (2009). *On the Problem of the Software Cost Function,*. spain.

Du, Z. Y. (2012). Image matching using a bat algorithm with mutation. *Applied Mechanics and Materials*, 88-93.

F, s. (2006). Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects. *Journal of computer science*, 118-123.

F. Ferrucci, C. G. (2010). Genetic programming for effort estimation: an analysis of the impact of different fitness functions. *in Proceedings of the 2nd International Symposium on Search Based Software Engineering (SSBSE '10),* (pp. 89-98). IEEE Computer Society.

Gao, B. W. (1997). ASSESSING SOFTWARE COST ESTIMATION MODELS: CRITERIA FOR ACCURACY, CONSISTENCY AND REGRESSION. *Advanced journal of Information sciences*, 30-44.

*http://en.wikipedia.org/wiki/COCOMO*. (n.d.). Retrieved june 2014, from http://en.wikipedia.org/: http://en.wikipedia.org/wiki/COCOMO

Jacob, L. (2014). Bat Algorithm for resource scheduling in cloud computing enviornment. *International Journal for research in applied sciences and engineering technology*.

Jamil, M. Z.-J. (2013). Improved bat algorithm for global optimization. *Applied Soft Computing*.

Khan, K. N. (2011). A fuzzy bat clustering method for er-gonomic screening of office workplaces,. *Advances in Intelligent and Soft Computing*, 59–66.

Komarasamy, G. a. (2012). An optimized K-means clustering techniqueusing bat algorithm. *European J. Scientific Research*, 263-273.

Lemma, T. A., Bin Mohd Hashim, F. (2011). Use of fuzzy systems and bat algorithm for exergy modelling in a gas turbine generator,. *IEEE Colloquium*, 305–310.

Lin, J. H. (2012). A chaotic Levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems. *J.Computer and Information Technology*, 56–63.

Lin, J.-C. (2010). Applying Particle Swarm Optimization to Estimate Software Effort by Multiple Factors Software Project Clustering. *IEEE*.

M.jorgensen, K. a. (2003). A review of software surveys on software effort estimation. *International symposium on Empirical Software Engineering*, (pp. 223-230).

Michalewicz. (1992). Genetic Algorithms + Data Structures = Evolution Programs. *Springer*.

Molokken, K. F. (2007). Increasing Software Effort Estimation Accuracy- using experiance data, estimation models and checklists. *&th International conference on Quality Software*, (pp. 342-347). portland.

Nakamura, R. Y. (2012). A binary bat algorithm for feature selection. *25th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)* (pp. 291-297). IEEE Publication.

P.R Srivastava, A. B. (2014). An empirical study of test effort estimation based on bat algorithm. *Int. J. Bio-Inspired Computation*, 57-70.

Putnam, L. (1978). A general Empirical Solution to the Macro Software Sizing and Estimating Problem. *IEEE Transactions on Software Engineering* , (pp. 345-360).

Q. Alam, P. (n.d.). Systematic Review of Effort Estimation and cost Estimation. Roorkee: Institute of management studies.

Reddy, P. (2010). Software effort estimation using Particle Swarm Optimization with inertia weight. *International journal of software Engineering* , 12-23.

S K Sehra, Y. S. (2011). SOFT COMPUTING TECHNIQUES FOR SOFTWARE PROJECT EFFORT ESTIMATION. *"International Journal of Advanced Computer and Mathematical Sciences*, 160-167.

Segundo. (2001). SEER-SEM Users Manual .

Shepperd, M. J. (2007). A Systematic Review of Software Development Cost Estimation Studies. *IEEE Transactions on Software Engineering*.

Sheta, S. A. (2007). Software Effort Estimation by Tuning COOCMO Model Parameters Using Differential Evolution. *IEEE congess on evolutionary computation*, 1283-1289.

Vishali, Anshu Sharma, Suchika Malik. (2014). COCOMO model Coefficients Optimization Using GA and ACO. *International Journal of Advanced Research in Computer Science and Software Engineering*, 771-776.

X.S., Y. (2008). *Nature-Inspired Metaheuristic Algorithms.* UK: Luniver. .

Xie, J. Z. (2013). A novel bat algorithm based on differential operator and Levy flights trajectory. *Computational Intelligence and Neuroscience*.

Yang, X. S. (2011). Bat algorithm for multi-objective optimisation. *Int. J. Bio-Inspired Computation*, 267-274.

Yang, X. S., Karamanoglu, M., Fong, S. (2012). Bat aglorithm for topology optimization in microelectronic applications. *Conference on Future Generation Communication Technology*, (pp. 150–155).

Yang, X.-S. (2010). A New Metaheuristic Bat-Inspired Algorithm. *Nature Inspired Coop-erative Strategies for Optimization (NISCO 2010)* (pp. 65-74). Springer.