Software Requirement Prioritization using Artificial Bee Colony Algorithm

By:

Ayet E Noor

MS-18-CSE

Supervisor

Dr. Wasi Haider Butt

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

September, 2022

Software Requirement Prioritization using Artificial Bee Colony Algorithm

By:

Ms. Ayet E Noor

MS-18-CSE

A thesis submitted in partial fulfillment of the requirements for the degree of MS Software Engineering

Approved by:

_____

Supervisor:

Dr. Wasi Haider Butt

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY, ISLAMABAD

September, 2022

DECLARATION

I declare that this research work titled “Software Requirement Prioritization using Artificial Bee Colony Algorithm” is my original work. The material which has been taken from other sources has been properly acknowledged or referred otherwise. No part of this thesis has been submitted in candidature of any degree.

_____

Ms. Ayet E Noor

MS-18-CSE

00000276305

Countersigned:

_____

Dr. Wasi Haider Butt

THESIS SUPERVISOR

LANGUAGE CORRECTNESS CERTIFICATE

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical, and spelling mistakes. The thesis is also according to the format given by the university.

_____

Ms. Ayet E Noor

MS-18-CSE

00000276305

_____

Dr. Wasi Haider Butt

THESIS SUPERVISOR

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

All the glory and praise belong to Allah Almighty, who gave me the courage, tolerance, knowledge and competence to accomplish this research and to endure and finish it satisfactorily. Undeniably, I wouldn't have been able to achieve it without HIS blessings.

I wish to express my extreme gratitude to my supervisor Dr. Wasi Haider Butt, for his endless motivation, and constant support and encouragement throughout my research. His supervision has guided me in my research and writing of this thesis. I could not have asked for a better advisor and mentor for my MS study.

I would also like to acknowledge and thank Dr. Arslan Shaukat and Dr. Farooque Azam for being on my thesis guidance and evaluation committee and also for their assistance and judicious recommendations.

My acknowledgements would not be complete without expressing my gratitude towards my biggest source of strength, my parents. I am generously grateful towards my parents who raised me when I was not capable of walking and continued to support me throughout ever department of my life. I express my greatest regards to my beloved sisters who have been my support system and have tolerated my varying temperament at through my life.

In the end, I would like to show my appreciation to all my friends and every other individual who has supported me in one way or another through this entire period.

Thanks for all your encouragement!

Dedicated to my loving parents and adored siblings whose incredible encouragement and support led me to this wonderful academic accomplishment

ABSTRACT

Requirement Prioritization plays an important role in the software development. To achieve effective outcome, a variety of techniques have been employed to prioritize software requirements. There are two disagreeing aspects; maximizing the satisfaction of the customers and minimizing the costs of development, that we face while selecting an optimal set of requirements and it is categorized as NP-hard problem. If we focus on the acquiring the approval from the customer, the price of the development increases whereas when we focus on cost, there are high chances of reduction in customer satisfaction. To solve this problem of prioritization, many meta-heuristic algorithms have been used and researches on including many genetic and evolutionary algorithms. Some focus on single-objective where others develop solutions for multiple objectives. Some of the simple methods such as hierarchy methods, ranking and numerical assignments have been discussed as well. In our research, we focus on the artificial bee colony algorithm to provide effective results for requirement optimization. We have proposed a method to prioritize requirements using the clustering technique with the help of artificial bee colony algorithm. The requirements with similar importance are assembled together in same groups. Our goal is to improve the prioritization results and reduce the errors obtained in the results as much as possible while maintaining the best outcome. The algorithm is executed on selected datasets of RALIC and the results are analyzed. In this research, a benchmark study is used to compare the results of the prioritization. The experimental results show that using clustering with ABC algorithm, the prioritization of requirements has been effectively improved.

Key Words: *Software requirements; Prioritization; Artificial Intelligence; ABC Optimization; Clustering; Quality Solutions.*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

CHAPTER 1: INTRODUCTION

1.1 Background

Requirement Engineering is an essential phase in the process of software development. An increasing number of methods and techniques have been employed to control the requirements until now. Larger software systems have much more complex development processes with numerous requirements. And in such cases, it is mostly impossible to implement all the elicited requirements due to the lack of time and finances. Subsequently, it is imperative to select an ideal subset of the formative prerequisites that can give most extreme fulfillment to the clients, with negligible time and taken a toll for the improvement group. [1] And to obtain an ideal set of requirements, we have to select techniques to prioritize the requirements. Efficient optimization algorithms are almost needed in every aspect of technology to deal with the increasing and complex optimization problems. For our research, we focus on the Artificial Bee Colony algorithm for optimizing the requirements set. The optimization algorithm was developed by Karaboga, [2] is based on the intellectual behavior of honey bee swarms when searching for a quality food source and it is applied to solve numerical optimization problems. ABC is a population-based algorithm which is relatively easy-to-use and is a quick search method. [3]

1.2 Aim & Objective

Requirement optimization has a dynamic role in the progress of large and complex software systems. There are two approaches for optimization i.e., single objective and multi objective. In the former type of optimization approaches, only one objective is considered whereas in latter approaches, more than one goal is considered to be optimized. And there are researches that analyze the multi objective optimization algorithm, such as genetic algorithms, neural networks and evolutionary algorithms etc. It is important to optimize the requirements to make them efficient and effective that provides customer satisfaction within minimum budget. The basis objective our research is to perform the optimization of requirement set using optimization algorithm i.e., Artificial Bee Colony algorithm that is a swarm-based population algorithm. The aim is to facilitate the software development industry by providing them with a process to get optimal set of requirements.

1.3 Scope of Proposed Work

Scope of our proposed research work includes the study of the artificial bee colony algorithm in order to find how to improve the optimization with regards to requirements set. We will tune the algorithm and look for a method that can help us in achieving our required optimal set of requirements. Benchmark case study will used for further validation and comparison of results.

1.4 Title of Research

"*Software Requirement Optimization using Artificial Bee Colony Algorithm*"

1.5 Motivation behind the Research

In software development, we deal with different kinds of requirements which vary according to the software system to be developed. In case of large and complex systems, there are so many requirements that are gathered by the time of implementation. It is improbable to implement all these requirements keeping in mind the different constraints for the system, such as cost, time and effort constraints. Therefore, we need to narrow the requirements by prioritizing them. We cannot simply prioritize it manually as it is time taking, and it is likely that several important requirements might skip out which affect the working of the end product. To address the above-mentioned problems, we implore the use of optimization algorithm.

1.6 Structure of Thesis

The study consists of the following sections. Chapter 1 includes an introduction of the thesis, and objectives of the research and motivation for topic selection and structure of this thesis. Chapter 2 contains a detailed review of research related to the selected topic. Chapter 3 elaborates the applied research method and implementation of our proposed idea. Chapter 4 outlines the results and validation to the applied approach. Chapter 5 summarizes the research findings that concludes the thesis along with idea for further improving the existing work under the heading future work.

Chapter 2

Literature Review

CHAPTER 2: LITERATURE REVIEW

2.1 Overview

In the process of Requirement Engineering, the focus is on recognize the stakeholders and their priorities and supporting them in a way that allows for analysis, communication and then later on execution. [4] In this process, we define and document the requirements for the desired system. It is important to perform requirement engineering process to achieve a better-quality output in the field of software engineering. As accurately documented requirements are essential for any software development. Failing to achieve an appropriate set of requirements may result in failure of the software product. In the world of innovation and new technology, many techniques and methods have been proposed to deal with the requirements, which has made the process of requirement engineering much easier and effective.

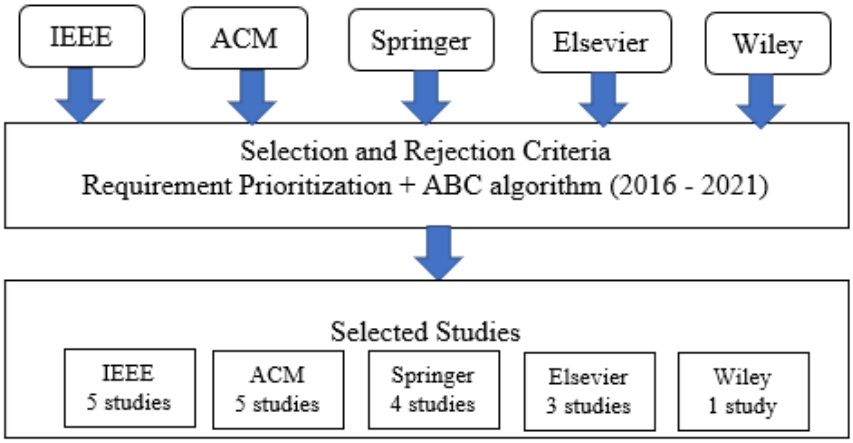

Figure 1: Research Process

The result of our research process is shown. Through sequential steps, the search process is carried out. For example, as shown in figure, the accepting and rejecting criteria were set beforehand. And according to those particular criteria, only those papers were taken into account that were accepted.

2.1.1 Requirement Prioritization

As prioritization assists in dealing with difficult decision issues, requirement prioritization helps in recognizing the most important requirements from the set by separating the few important ones from the many insignificant requirements. The following actions are supported by the requirement prioritization process. [5]

- to manage conflicting demands, concentrate the negotiation process, and settle difference among clients and stakeholders.
- to determine the importance of each condition in order to offer the best value at an affordable cost.
- to estimate expected customer satisfaction
- to design and pick the best collection of requirements to be implemented in upcoming releases.
- to choose just a subset of the specifications while creating a solution to satisfy the client(s).

2.1.2 Approaches to Requirement Prioritization

With the help of an extensive literature review, we have stumbled upon many different techniques to prioritize the requirements that are described in detail, in [5], [6] [7], [8], [9]. A few of these are briefly described below.

- *Analytical Hierarchy Process (AHP)*

It has been a widely used efficient and organized way of making decisions used for the prioritization of the software requirements. This method uses a matrix for the comparison of the requirements to determine the rate of parameters such as cost and effort. [5] As the set of requirements grow larger, the contrasts increase drastically therefore it is not recommended for systems with bigger set of requirements. The output achieved from AHP prioritization is a list of weights on a ratio scale. AHP is, generally, found to be clear, easily understood, and provides a good indication of the cost/value ratio for prioritizing requirements. [13] Research has discovered frameworks to making correct decisions in different fields using AHP. [27] Similarly, the analytical hierarchy process has also been employed for the prioritization of the non-functional requirements. [42]

- *Numerical Assignment*

Numerical Assignment is an important technique for prioritizing the requirements in which requirements are assigned one of the many clusters that are made on the basis of their priority. multiple clusters of requirements are created. On the basis of their priority given to them by

the customers, the requirements then fall under the particular grouping. [6] The requirements in divergent groups depend on a priority measure so that while decision making, the clients can refer to that to group the requirements accordingly. [28]

- *Minimal Spanning Tree (MST)*

Software requirements can easily be ordered on the basis of priority with the help of spanning trees. The requirements are plotted via directed graph and then these graphs are transformed into spanning trees. High priority requirements reduce the different kinds of interrelated dependencies and ensure that projects are completed on schedule. [7] The concept of MST method considers the perfect and reliable decisions to ensure the lack of redundancy, thereby minimizing the evaluations being made. Consequently, we can minimize the number of times we have to compare intensely as compared with AHP. However, it is difficult to spot conflicting assessments. [30]

- *Planning Game*

In XP Programming, a planning game is a prioritization technique for software requirements. To prioritize requirements, it combines the numeral assignment prioritization approach and ranking prioritizing technique. In the planning game, the users' requirements are gathered, and then these requirements are put on the story board. Customers then sort these requirements into three heaps depending on their perception of their relevance. [5] Whereas, the developers are asked to do the same based on factors that might affect the end product such as effort, cost or time required to fulfil the particular requirement. In this way, the requirements are narrowed down based on their priority with respect to every stakeholder.

2.1.3 Requirement Optimization using Artificial Intelligence

With the advance of technology, the fields such as artificial intelligence propose challenging yet efficient and effective methods to complete our task and achieve quality results with higher accuracy and precision. The development time has been reduced as well as there are many different kinds of tools available to complete the tasks for the developers. To obtain efficient and accurate results, developers have resorted to use machine learning algorithms instead of manual labor.

Therefore, where manually obtaining a set of optimal requirements is difficult with high chances of inaccuracy, the use of complex algorithms has taken the lead. Clearly, with the use

of machine learning algorithms, the development time has been reduced evidently as well as the quality of results. Here, we discuss some of those algorithms that have been used in research to prioritize the requirements.

- *Parallel Multi-Objective Artificial Bee Colony Algorithm*

Many researches have made use of this particular algorithm to optimize the software requirement. In [20][21][22], single-objective ABC algorithm and in [23][24][25], multi-objective ABC procedure has been employed to resolve the particular issue. However, in [1] , an advanced approach of master slave model is suggested to solve the NRP problematic issue in an architecture where the memory is being shared among models. [14] With the help of multiple core machine, a module is selected as the master, while all the other modules run the algorithm together and autonomously. Each module produces an output colony for the algorithm. After execution, the NDS sets are sent to the master core which removes any repeated solutions. The combination of the generated solutions produces a larger number of NDS in comparison to other models. With the help of this method, the output solutions achieved have the best quality as compared to the output from the other modules. The time required to achieve the perfect solution has been decreased while the quality of solution has been improved.

- *Fuzzy Artificial Chemical Reaction Optimization Algorithm*

Work has been done on requirement prioritization using the artificial chemical reaction technique. A meta-heuristic optimization algorithm based on artificial chemical reaction is used to optimize the requirements. [10] The property of molecules interrelating with each other is employed as when they transformed into those molecules that have less energy which is considered as a fitness function. The algorithm begins with the molecules in a population. At that point, some particles are dominated whereas others are generated through the chemical reaction and when the criterion of problem is satisfied, the algorithm terminates. There exist various operators such as crossover, mutation etc., that results in examining the complete search space and discover the best solutions in the minimum time. Using the NDS record technique, the fine solutions that are found during every run are stored in the collection.

- *Non-Dominated Sorting Genetic Algorithm (NSGA)*

Greer and Ruhe [15] have employed the use of genetic algorithm to select the optimal set of requirements. Assume, that we have a collection of requirements and the group of customers that are involved in the development. There is a set of weights that are associated to all the customers that indicates the level of their importance. V is the set of values that are assigned to the requirements by the customers. The two goals for the system requirements can be expressed as: maximizing the customer satisfaction and minimizing total cost requirements. [16] A multiple objective evolutionary algorithm was proposed in the study to solve the system requirement selection problem.

Using the required conditions, the objective function of the proposed method is formulated. The NSGA will keep all the results at hand and it is restructured at the completion of each generation with the NDS of the population. Although genetic algorithms, in general, cannot guarantee optimality [15], the identical and superfluous, both such, solutions are kept uninvolved to protect from getting identical solutions to provide a chance for new solutions and more diversity.

- *Swarm Intelligence Evolutionary Algorithm*

According to [17], the ideas from both NSGA-II, i.e., the concept of organizing the solutions in the population into various sets based on the dominance, and also from PAES, i.e., idea of ND archive is to hold the restructured the best solutions found during execution for the multi-objective ABC in this work are included. Utilizing the three phases of ABC approach, it incorporates the operators for a verification process to be checked efficiently.

A probability vector is generated that produces the best outcome; a chance to be opted in the next stage. Also, the results achieved from the suggested method show less distribution for each boundary. In conclusion, we can clearly declare that the improvement in results obtained by the approach is indeed prominent.

- *Fuzzy Logic Inference System*

The Mamdani implication method is used to combine fuzzy rules to regulate the capability of each and every requirement for development in the next developing iteration. [18] For this evaluation, the fitness of every objective or condition, few input variables and one output variable, all fuzzy variables, for development in the next release are used and for each requirement characteristics, namely the development time, the development cost and the

satisfaction rate, an input fuzzy variable is defined. The requirements are evaluated in a circuit for better fitness in order to look at the restrictions. Then, the testing outcomes of the suggested algorithm are analyzed with those of the genetic algorithm. The chosen outcome depicts more fitness or satisfaction from the suggested technique as compared to the genetic algorithm.

- *Multi Objective Ant Colony Optimization*

Here, a comparison analysis is executed with NSGA-II and the GRASP algorithm to evaluate the ACO system after the algorithms are revised to the problem. One ant will develop a solution depending upon the weighted results of the matrices of every objective. Each goal will have a different pheromone matrix. The weights that ants assign to the different attributes determines the importance of the prioritization criteria. It has to be scattered evenly across the different regions. Ever curve that is being included in the best solution repository, the pheromone is updated. It is highlighted that the accurate values for the ordering of the indicators achieved by the algorithm when used in search. [19]

- *Clustering Algorithms*

Research has shown that requirements are also being prioritized into group of their importance by the means of clustering algorithms. The stakeholders are asked to rate the requirements according to their priority and with the help of clustering technique, the requirements are classified into groups such as very important, important or less important. According to [46][47], the requirements are prioritized by using k-means clustering with using the weights of attribute of requirements as the input producing centroids in result to classify the requirements.

Clustering technique has also been helpful in the process of regression testing in order to prioritize the test cases. [48][49] By using the technique, the test cases are classified into clusters which are then prioritized with respect to some important aspects. In [50], it has also been used for performance evaluation of test case prioritization. Clustering has been used for prioritization in different scenarios such as in risky monitoring systems [51] and in fault prediction [52] and many others such as in [53][54][55][56][57].

- *Parameter Tuning and Feature Selection*

Hyperparameter tuning is a method of selecting set of optimal hyperparameters for the particular algorithm in use. The technique has been used in many researched such as [38]

whereas [37] provides a detailed overview for the similar process for different algorithms. Research has been on the use of ABC algorithm using hyperparameter tuning for other purposes such as discussed by Deniz Ustun et al [39].

2.2 Research Gap

With an extensive and detailed study of the literature, we have come to the conclusion that there exist some gaps that needs to be bridged. Requirement engineering is a widely spread domain and a lot of work has been done its multiple branches. Our research focuses on the topic of prioritizing those requirements. Many studies have been conducted on multiple algorithms and different techniques or methods in this domain to prioritize the requirements. Even though, work has been done on the algorithm that we have in mind but, according to our best knowledge, no work has been done on clustering the requirements using artificial bee colony algorithm for large datasets.

Following a thorough review of existing research on requirement prioritization methods, we have concluded that no research yet has combined the two techniques to achieve better results. We suggested a solution to acquire the benefits of two techniques that will help in achieving far better accuracy in prioritization. The technique further uses multiple process for further accuracy to be obtained.

Chapter 3

Proposed Methodology & Implementation

CHAPTER 3: PROPOSED METHODOLOGY & IMPLEMENTATION

3.1 Problem Definition

Requirement Engineering is the basic step in the development of the software products. Proper work must be done on requirements to achieve the desired results in the end product. Negligence at this phase in the development of software can lead to increase in costs, increase in development time and reduction in quality which will result in failure of customer satisfaction. Customers and stakeholders play a major role in the process of requirement elicitation. There are a lot of approaches and procedures that ease the process of gathering requirements. Recognizing the customer requirement accurately is important which is being done by different new methods. [26]

The bigger challenge after elicitation is prioritizing the requirements. The need arises as because of the lack in resources and time and cost, and in most cases, it is not possible to implement all the requirements. It might be the case that the requirements conflict with each other and that is why it is impossible to develop them. It has to be kept in mind that; our main goal is to satisfy the customer. Therefore, a set of optimal requirements must be chosen from the originally proposed development requirements that provides maximum satisfaction to the customer with minimum cost and effort for the development process. In complex projects, it is difficult to select a small collection of requirements. It is important to have techniques that can help the software developers to select such requirements that are important and needed for the end result. Researches have made to help with this particular problem using different methods and algorithms.

There are two conflicting objectives; customer satisfaction and development costs. As, if the customer satisfaction increase, and cost is reduced, it is clear that by doing so the quality will be compromised. Less requirements will be developed in return which will impact the customer satisfaction. As opposed to that, if a large set of requirements are developed and satisfaction of the customer is taken into account, it is obvious that it will lead to higher costs.

Our research aims to propose a method that can help in determining the prioritized selection of requirements that must be included in the successive releases during development of the software. Requirements with the highest priority that results in maximized customer satisfaction and requires minimum time and efforts to be developed. We discuss our proposed

approach in detail which will help us to improve the prioritization of the requirements to achieve the required results. The research methodology is explained in this chapter.

3.2 Research Methodology

In the figure below, we have shown the research methodology that has been followed for this study. As an initial idea, software prioritization of requirements was chosen. And after studying different approaches for the particular objective, the use of artificial bee colony algorithm was selected as our tactic to deal with this problem. Datasets were collected after conducting the extensive literature review. The artificial bee colony algorithm is modified in order to prioritize the requirements with the help of clustering technique on the basis of their importance.

In our approach, we use the implementation of the artificial bee colony algorithm to perform the task of clustering the requirements. As we observe the clustering problem and we realize that we can easily convert it into an optimization task. Since a well-defined optimization problem uses a search space and a set of input decision variables. An objective function is also used to either maximize or minimize the results on basis of the relationship between decision variable or some constraints. In the artificial bee colony algorithm, the food searching behavior of bees is implemented to look for ideal solutions for the problem. Each bee can be considered as a whole solution for the particular clustering problem at hand and it can then represent a complete set of candidate centroids. As a result, the requirements are then classified into groups based on the level of their importance. There are many other approaches that can be used to achieve the maximum level of accuracy for prioritizing the requirements; these include feature selection and parameter tuning.

After the implementation of the proposed approach, we achieve the clusters of requirements. For the partitional clustering approach, we have to increase the outer distance between two discrete groups or clusters and decrease the inner distance within a group. We have selected the Sum of squared errors as our objective function for this approach to chance the errors for the predicted values.

The research methodology for the suggested approach is described in given figure. Later on, every step is discussed in detail in this chapter.

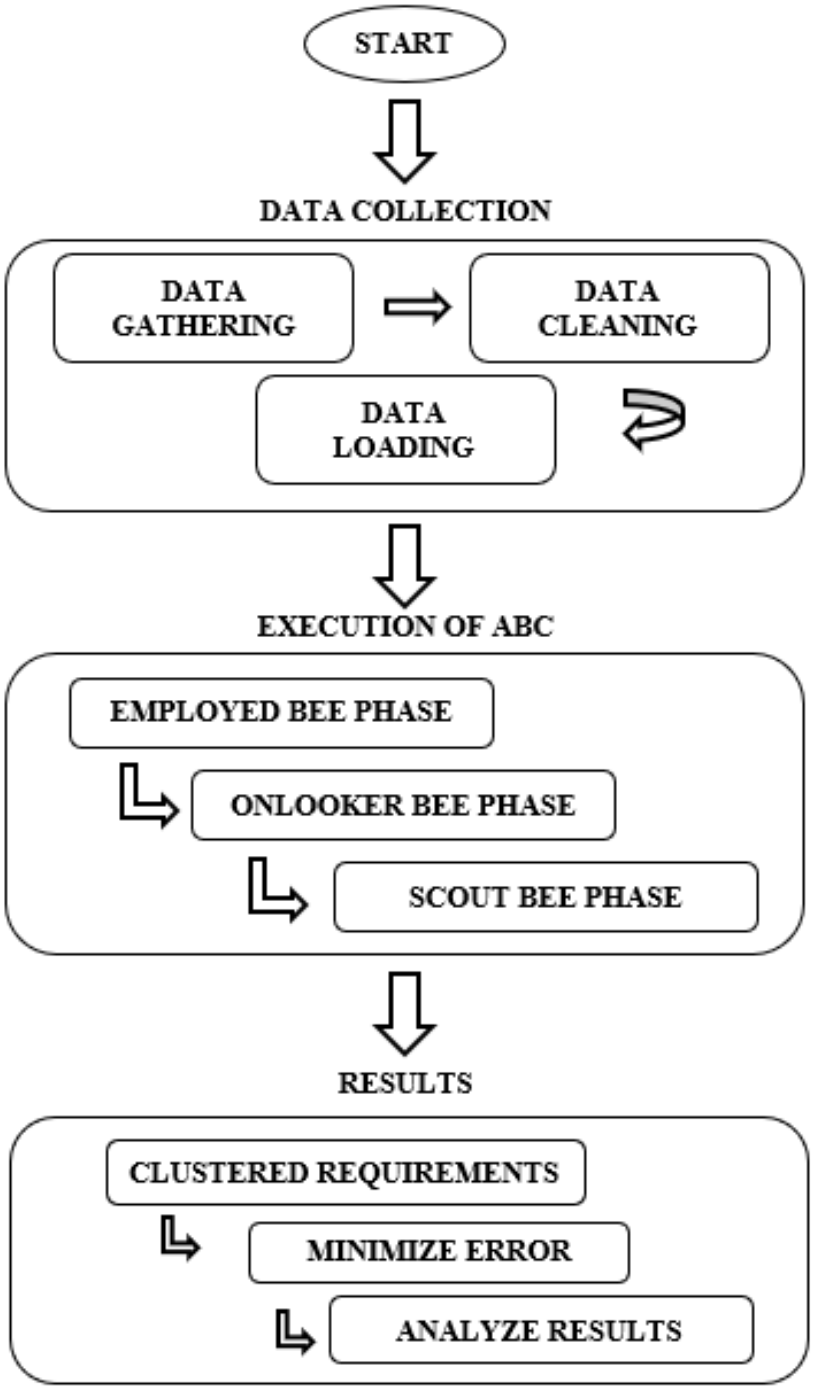

Figure 2: Proposed Methodology

3.2.1 Data Collection

RALIC datasets were used to validate the proposed method. It contains various datasets of stakeholders and their requirements on a real software project. RALIC stands for replacement access, library and ID card. [43] The datasets are publicly available at [58]. The

three aspects of the requirements datasets, PointP, RateP and RankP were used. RankP is a ranked profile which contains customer priorities for each requirement ranging from 1 to X based on the level of difficulty and effort required for the requirement. RateP is a rated profile that is a dataset that contains the stakeholder rate for a list of requirements. And PointP is a point test profile where stakeholders give points to the requirements according to their priority. [44].

After the collection of the datasets, the requirements with missing values were given the value of 0 as a way of preliminary processing the datasets. A snippet of these datasets is shown below.

c.1	0	4.5454	0	0	0
c.1.1	0	0	0	0	0
c.1.2	0	0	0	0	0
c.2	10	18.181	0	0	0
c.2.1	0	4.5454	0	0	0
c.2.2	0	0	0	0	0
c.2.3	0	0	0	0	0
c.2.4	0	0	0	0	0
c.2.5	0	0	0	0	0
c.2.6	0	0	0	0	0
c.2.7	10	4.5454	0	0	0
c.2.8	0	0	0	0	0
c.2.9	0	4.5454	0	0	0
c.2.10	0	4.5454	0	0	0
c.3	0	4.5454	0	0	0
c.3.1	0	4.5454	0	0	0
c.3.2	0	0	0	0	0

Table 1: Point P Dataset

The above given table shows the snippet of point test profile; rating of five customers for one of the general requirements and its subset of specific requirements. It can be seen that the missing values were rated 0.

Table 2 shows the rating of the same five customers the same set of requirements but for a ranked profile. The complete set of requirements were marked ranging from 0 to 5. Whereas the Table 3 shows the priority of those five customers for the similar set of requirements ranging from 0 to 5 for a rated dataset.

c.2	0.128205	0.4375	0	0	0
c.2.1	0	0.096491	0	0	0
c.2.2	0	0.096491	0	0	0
c.2.3	0	0.096491	0	0	0
c.2.4	0	0.096491	0	0	0
c.2.5	0	0.096491	0	0	0
c.2.6	0	0.096491	0	0	0
c.2.7	0.080952	0.096491	0	0	0
c.2.8	0	0.096491	0	0	0
c.2.9	0	0.096491	0	0	0
c.2.10	0	0	0	0	0
c.3	0	0	0	0	0
c.3.1	0	0	0	0	0
c.3.2	0	0	0	0	0
c.4	0.025641	0	0	0	0
c.4.1	0.021429	0	0	0	0
c.4.2	0.021429	0	0	0	0
c.4.3	0.021429	0	0	0	0
c.4.4	0.021429	0	0	0	0
c.4.5	0.021429	0	0	0	0
c.4.6	0.021429	0	0	0	0
c.5	0	0	0	0	0

Table 2: Rank P Dataset

c.1	5	5	5	4	0
c.1.1	0	0	0	0	0
c.1.2	0	0	0	0	0
c.2	5	5	5	4	0
c.2.1	0	0	0	0	0
c.2.2	5	5	5	4	0
c.2.3	5	5	5	4	0
c.2.4	5	5	5	4	0
c.2.5	0	0	0	0	0
c.2.6	0	0	0	0	0
c.2.7	5	5	5	4	0
c.2.8	5	5	5	4	0
c.2.9	5	5	5	4	0
c.2.10	5	5	5	4	0
c.3	5	5	5	4	0
c.3.1	0	5	5	4	0
c.3.2	5	5	5	4	0
c.4	5	5	5	4	0

Table 3: Rate P Dataset

3.2.2 Execution of Artificial Bee Colony Algorithm

Artificial Bee Colony algorithm has been used for problems where there is a need to achieve optimal results, where the objective is to find the finest solution from among all the feasible solutions. The problem at hand is represented by the objective function; a rule that describes the problem and the different variables in relation. The purpose is to find the variables that affect the objective function i.e., either minimize or maximize it which is our fitness value.

ABC algorithm was encouraged by the intellectual searching performance of honey bees. [40] It replicates the collective action of bees at the time of finding the food, which in our case would be a solution, motivation for finding the and searching the place, the authority of the food source and looking for new food sources. Here, the food source means the position in search space i.e., the solution for our problem.

The algorithm makes use of bees as agents to look for the optimal solution. These bees are categorized into further groups depending upon the type of task it should perform. Initially, the amount of food sources available is equivalent to the total number of employed bees. The quality of food sources is judged by the value of objective function at that particular position.

In the artificial bee colony algorithm, the problem undergoes through the three main phases in artificial bee colony algorithm. As specified in the Figure 2, there are three types of bees. But as we are executing the algorithm, there are actually only two types of bees, i.e., employed and onlooker. Scout bee is an exploratory behavior exhibited by both of these bees.

The general structure of the artificial bee colony algorithm is shown in the figure below. It shows the phases involved in the algorithm and the different tasks that are assigned to the bees are also displayed in the figure.

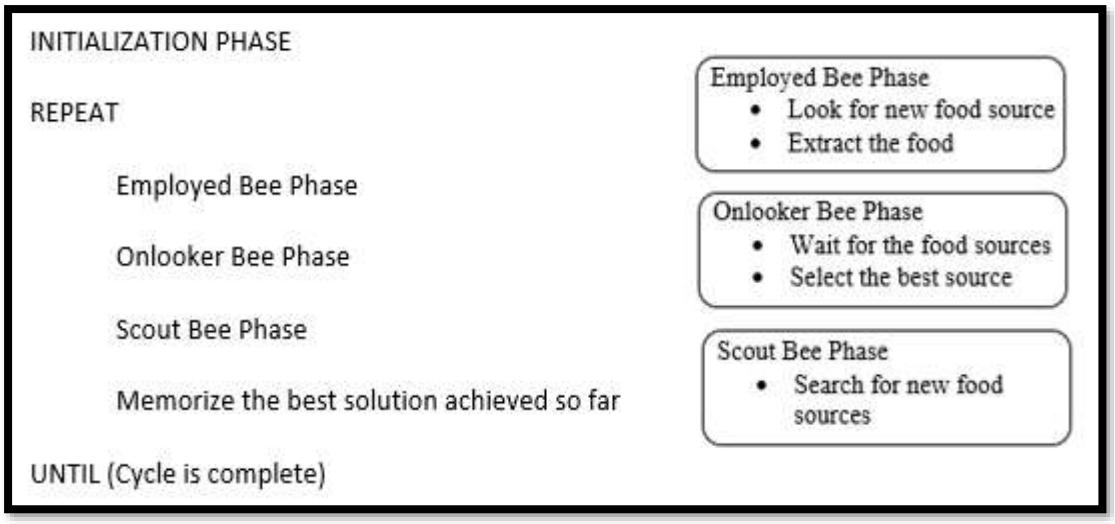

Figure 3: Scheme of ABC Algorithm

The brief explanation of each of the different phases of the artificial bee colony algorithm is discussed.

- **Initialization Phase**

First, the algorithm produces an initial population of *SW* solutions at random, i.e., the food positions where the size of the swarm is denoted by *SW*.

Suppose $X_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}$; signifies the i^{th} solution in the bee swarm where n denotes dimension size.

After the values of the collection of the food sources are assigned with values accordingly, we assign the control parameters with values according to our conditions at this stage. There can be multiple control parameters such as colony size, limit or numbers of bees, that can affect the output of the algorithm. As each of the food source is the solution to the problem; which is equal to the number of the bees available. These values are to be optimized according to the objective function used for the problem.

After the initialization, the solutions are exposed to a repetition of iterations, D = 1, 2, …, MCN of the search processes to look for any improvement.

- **Employed Bee Phase**

In this phase, the employed bee looks for the food sources and extract the food. In the algorithm, this behavior corresponds to each employee bee X_i generating a new position V_i in the neighborhood and evaluating if the new position has better quality solution.

$$V_{ij} = X_{ij} + \Phi_{ij}(X_{ij} - X_{kj})$$

The algorithm uses the given expression to produce a new candidate solution where $k \in \{1, 2, \dots, SN\}$ and $j \in \{1, 2, \dots, D\}$. Both of these values are selected at random whereas Φ_{ij} is a random number between [-1, 1]. Based on the greedy selection, the outcome at this position is memorized or ignored in comparison to the old solution. If the fitness value of the new solution V_{ij} is better then, the previous value is updated with the new one i.e., V_{ij}. Otherwise, it remains unchanged.

After the search process of all the employed bees is complete, the outcome of the search is shared with the other bees through a special dance called waggle dance. In this dance, the bees indicate the direction and distance of the food source from the hive and the quality of the food source.

- **Onlooker Bee Phase**

In this phase, the onlooker bee analyzes the work of the employed bee. They check the progress and evaluate the food being gathered. In the algorithm, the onlooker bee looks for the best outcome from the employed bee phase and try to improve those food sources. Here, the food source is chosen by the onlooker bee on the basis of the probability value associated with it.

The probability value p_i is calculated using the following equation which considers the fitness value of the food source.

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n}$$

In the expression, fit_i is the fitness value of the solution *i* in the swarm whereas *SN* is the total number of food sources which is equal to the number of employed bees, as discussed before. After specified number of cycles which is called the limit, the onlooker bee indicates

that the source has been exhausted, which means that no further solutions with better results could be found.

Here, the onlooker bee uses the same previous expression to update the food source if the fitness value is better in comparison to the old one otherwise it is kept unchanged.

- **Scout Bee Phase**

After the employed bees have collected food at their respective initial food sources and onlooker bees are done with analysis of better-quality food source, it checks if the food source is exhausted. If it is, then either of the bees become a scout bee and explores randomly in search for a new food source. The abandoned food source is then replaced with new food source that is discovered using the following expression.

$$x_i^j = x_{min}^j + rand(0,1)(x_{max}^j - x_{min}^j)$$

Here, random function implies that value between 0 and 1 is used randomly whereas x_{min}^j and x_{max}^j are the lower and upper limits of the solution i. In the algorithm, if there is no improvement in the new solution, the specific position is discarded and bees start with the random selection of food source. In conclusion, the process of random search of food source is repeated again and again until the termination conditions are met i.e., the number of repeated cycles indicated for the specified problem.

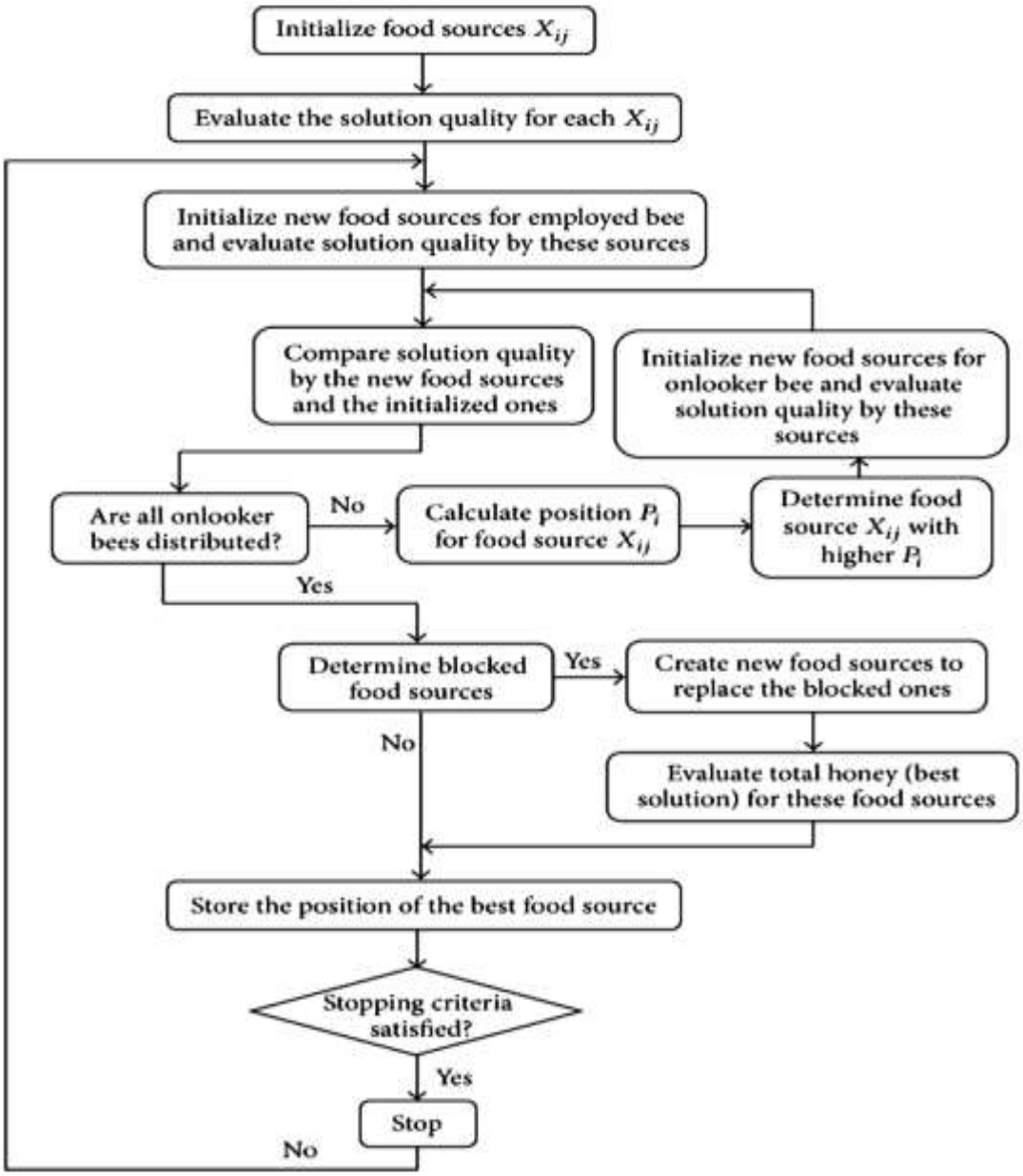

Figure 4: Flow Diagram of ABC Algorithm

The above figure displays all the steps covered in the implementation of the artificial bee colony algorithm. [60] As discussed before, and as per the flowchart, the initialization is performed of the food sources. These food sources are then evaluated by different types of bees based on their own varying criteria to select the best quality food source at the end.

The algorithm is repeated again and again until a specified number of cycles or trials are complete; which is the limit, a control parameter to determine if the exhausted food source can be further improved or it should be abandoned.

There are a few selection processes that are followed in the working of the artificial bee colony algorithm. First of all, the employed and the onlooker bees carry out a local search in the region around the hive. And then there is a global search done by only the onlooker bees to look for other food sources, if there are any. Greedy selection is performed by the bees when they compare the previous solution with the new solution based on their fitness value.

And the end, the scouts perform the random selection process once the food sources are exhausted to look for new food sources.

3.3 Clustering Technique using Artificial Bee Colony Algorithm

Clustering is a technique that performs grouping of data having similar patterns. It is an optimization problem where the objective is to classify the provided data instances into a particular number of groups to evaluate the distance between those instances. [45] It is an unsupervised learning technique used for statistical data analysis.

We can solve our problem by finding a k-partition of the original requirements dataset. And the groups that are formed as results would have to make sure that no requirement must belong to more than one group which indicates that the intersection between these groups must be empty. The union of these groups or clusters would ultimately results in the original requirements dataset.

Since a well-defined optimization problem requires a search space and a set of input decision variables, we can use our artificial bee colony algorithm to form clusters. The algorithm uses bees as agent and if we look at each bee as a solution then it can represent the candidate centroid. And we have our *d*-dimensional variables and *k*-partitions, then each bee will be $k \cdot d$ dimensional vector.

In the clustering technique, we either have to increase the outer distance between the distinct clusters or decrease the inner distance between the data objects within a cluster. We have selected sum of squared errors (SSE) as the objective function to further improve our results. As we know, that the sum of the squared errors is a measurement which calculates the square of the distance between each data object and the closest centroid to it.

$$SSE = \sum_{k=1}^{K} \sum_{\forall x_i \in C_k} \|x_i - \mu_k\|^2$$

The above given equation is used to calculate the sum of squared errors for the requirement dataset. The x_i in the equation indicates to the ith data point which is compared with the centroids to determine the centroid that is near to it.

Sum of squared error (SSE) indicates the difference between the prediction and the true value. And our goal is to minimize this function through our algorithm to get better results.

The files included carrying out the proposed approach has been displayed in the figure below. The main phases are covered and the objective function has also been presented separately.

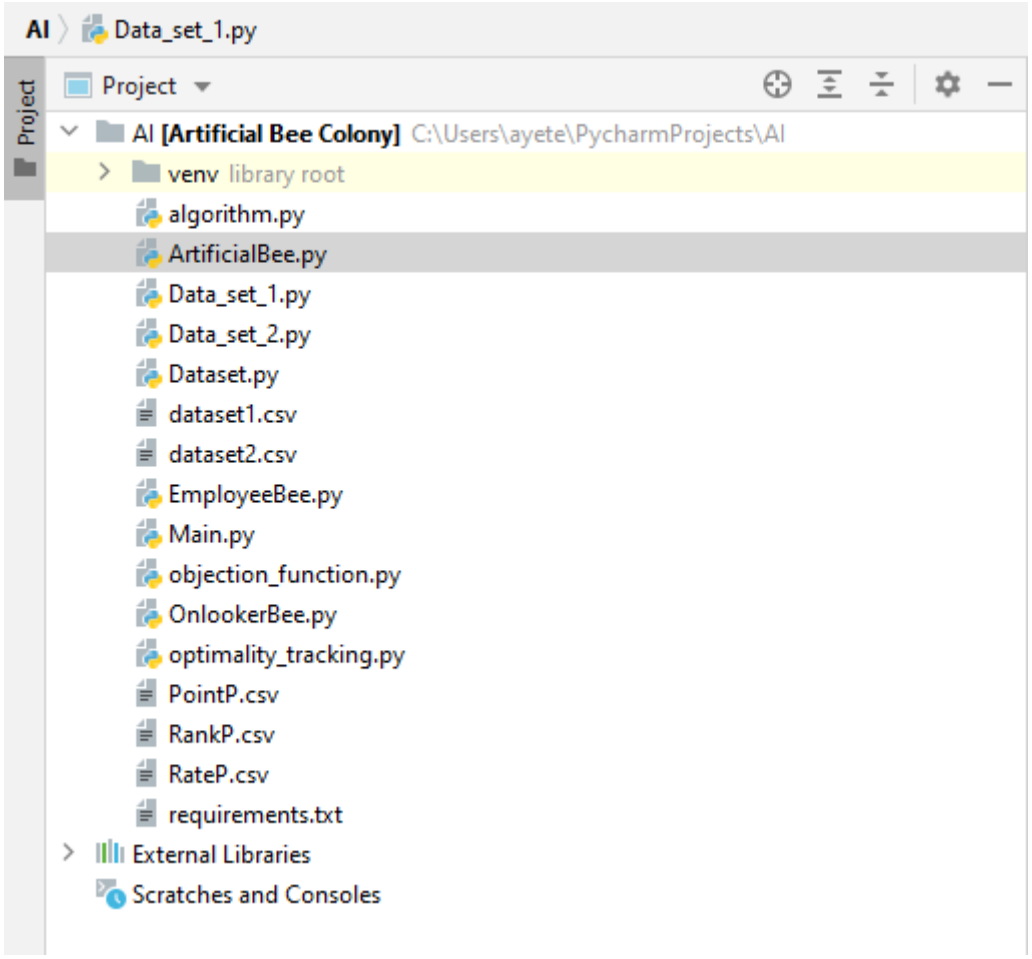

Figure 5: Project files

Chapter 4

Results & Discussions

CHAPTER 4: RESULTS & DISCUSSION

After performing all the steps specified in the research methodology, we achieve our required results. The requirement datasets, RALIC, has been used to validate our proposed solution. All the requirements along with their priorities specified by multiple stakeholders were analyzed by the algorithm and the result shows that the requirements are classified into clusters.

As previously discussed, the datasets were collected and then pre-processes as a means of cleaning the data and filling out the missing values.

4.1 Results of ABC Clustering

Three aspects of the RALIC dataset have been used for the validation of the proposed methodology. These datasets include the rated profile, ranked profile and the point test profile named as RateP, RankP and the PointP respectively. These profiles contain the customers' priorities for the given requirements as discussed previously.

The proposed algorithm was executed on the RankP, RateP and PointP datasets and the output files generated as results are displayed respectively. It consists of four output files each, which are discussed later on. At first, the algorithm was executed on each of the dataset for 1000 iterations. The number of clusters to be formed is kept at 20. So, for each of the dataset, the algorithm performs 1000 iterations and as a result, it groups the data objects into clusters of 20, based on the level of their importance.

The results of the algorithm include the original data spread across the plot and then the grouping of the data objects is displayed. Afterwards, the partitions of the data objects are displayed where the centroids are marked for each cluster.

The objective function used in our methodology is the sum of squared error (SSE) which the difference between the predicted value and real value. The output of SSE for the three datasets are also displayed in this section.

- **Ranked Profile – RankP**

Ranked profile contains the responses of the stakeholders based on their priorities of the requirements. The stakeholders prioritized the requirements based on the importance of the requirement towards them, 1 being the most important.

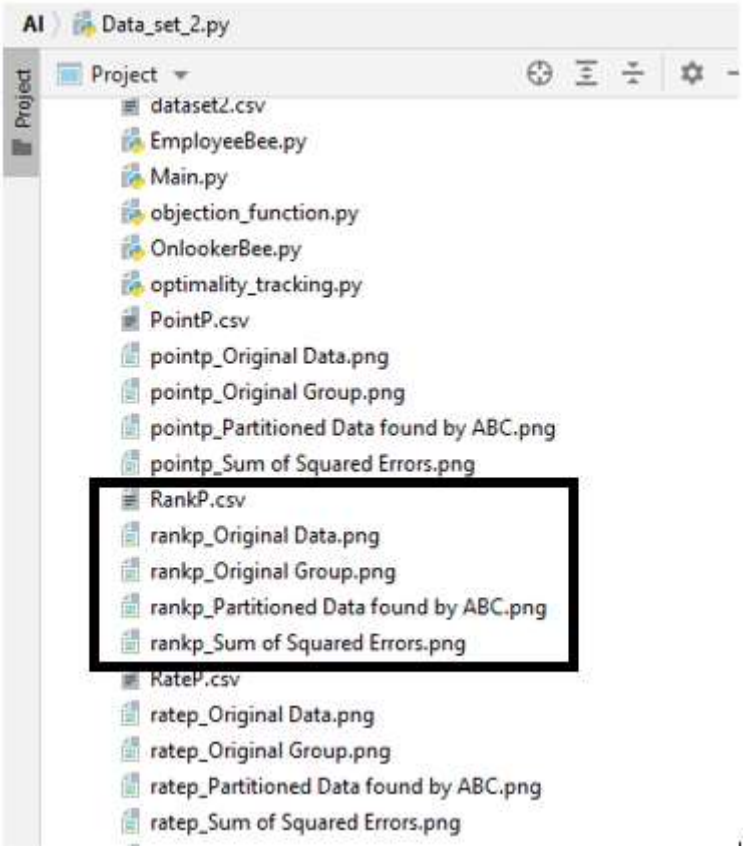

Figure 6: Ranked Profile Outputs

The output that is produced is shown below. It shows how the data objects that follow the similar patterns are grouped together following the clustering technique. The centroids are marked and then further analysis is performed on the results obtained.

The figure given below is a combination of two plots. The first plot shows the data objects scattered on the plot in its original form as per the stakeholder input. The second plot shows that the data objects have grouped together based on their importance. The figure highlights the groups that are formed as a result in different colors.

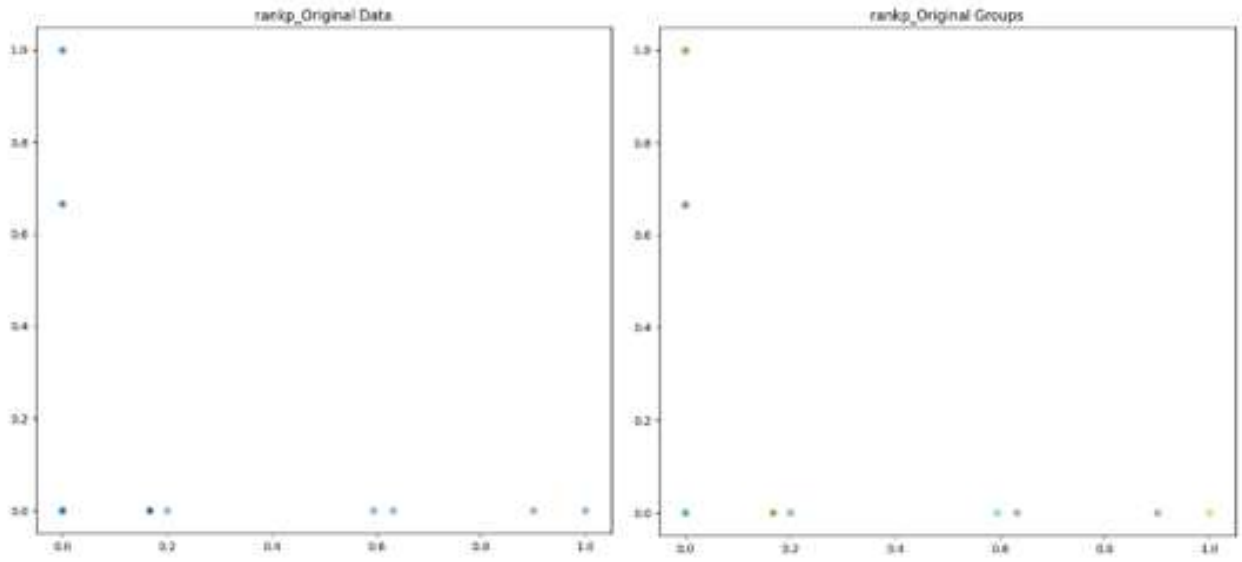

Figure 7: RankP - Grouping of Data

The centroids of each of the clusters are marked in the figure given below. It indicates that the instances that fall near to the particular centroid is clustered together with that centroid accordingly. In other words, data is partitioned by the artificial bee colony algorithm.

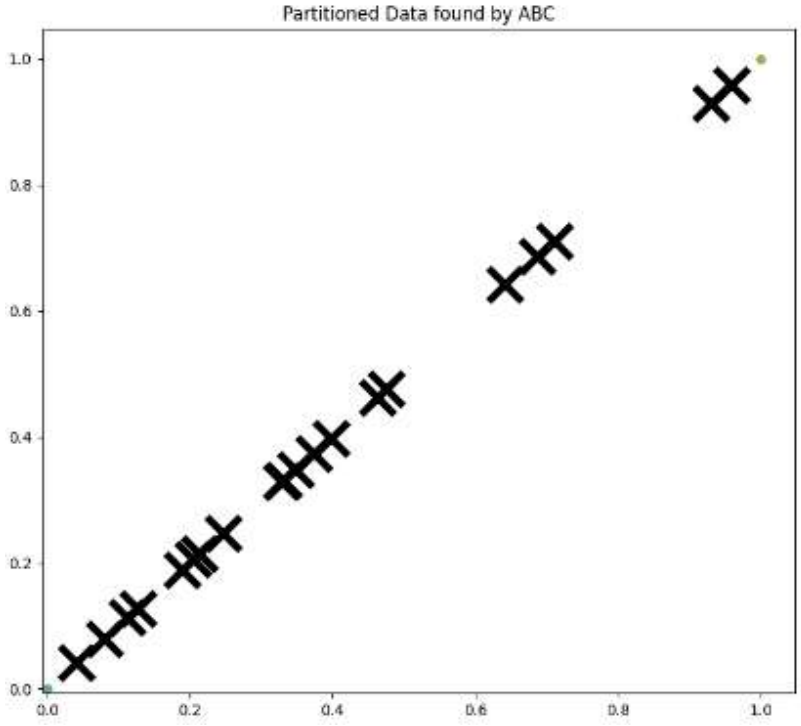

Figure 8: RankP - Partitioned Data

The figure below displays the result of the objective function that is being used in our proposed methodology. The objective function used is the sum of squared errors as discussed in the previous section.

The results of the algorithm after it has completed the 1000 iterations as specified is given below. As displayed in figure, the SSE value remains constant throughout the iterations. However, the error has been minimized as shown in the figure as it displays the best result that can be achieved for RankP dataset.

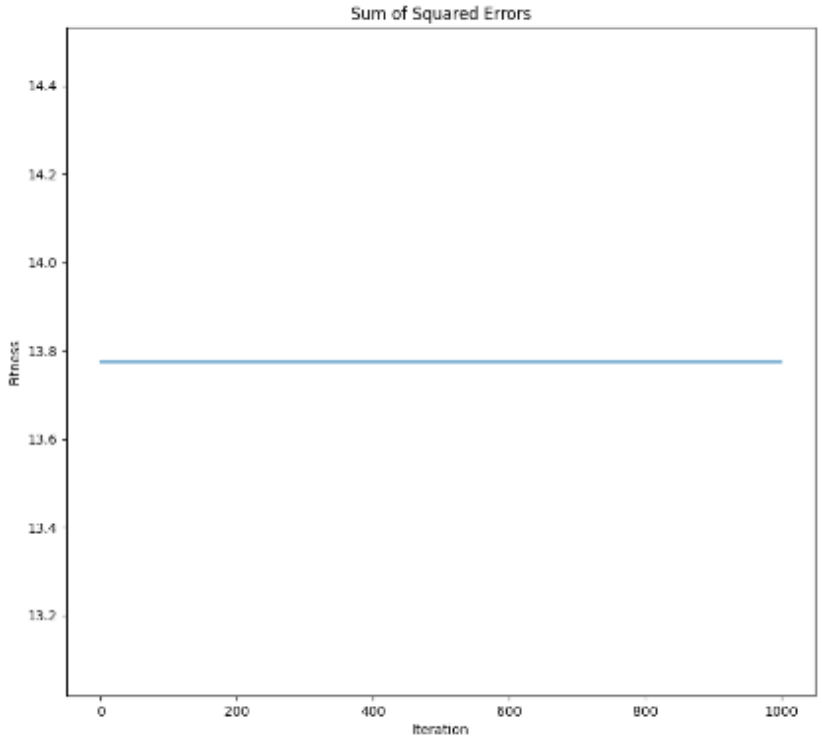

Figure 9: RankP - SSE Output

- **Rated Profile – Rank P**

Rated profile contains the numeral priorities assigned to each requirement by every stakeholder; where the value is based on the importance of the requirement towards them. In other words, the stakeholders rate those requirements high that are more important to them in comparison to those requirements that pertains less priority.

The outputs that are produced have been displayed through the figure. It consists of four files; the original data, grouping of data and data partitions where the centroids are marked. Last figure displays the output of the objective function.

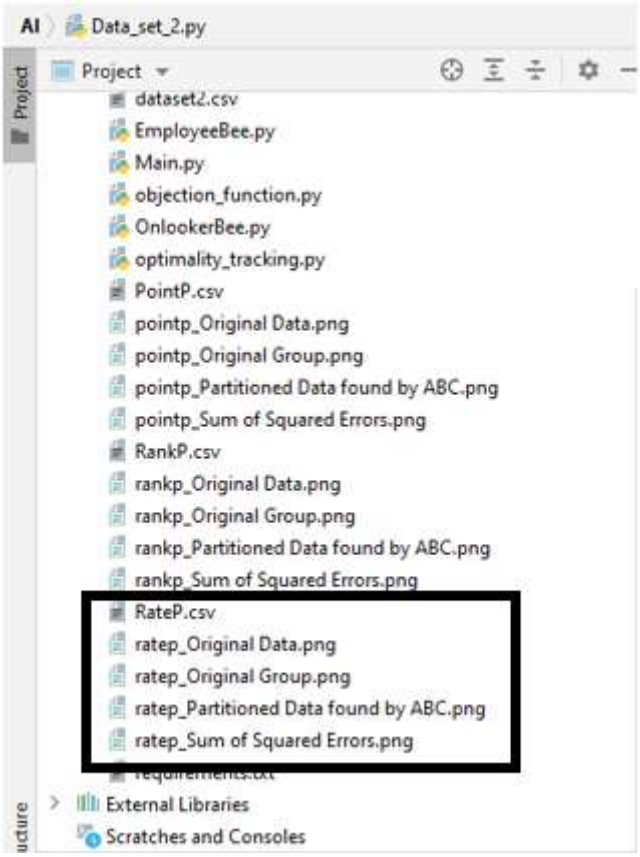

Figure 10: Rated Profile Outputs

The overall outcome of the algorithm depicts that the instances that have the similar rating are assembled together in a particular group following the clustering technique. The centroids are marked and then further analysis is performed on the results obtained.

The figure given below is a combination of two plots. The first plot shows that the instances are scattered on the plot in its original form as per the stakeholder rating. Whereas the second plot shows that those same instances but now they have been grouped together based on their rating.

The plot displays that the instances from particular group appear to be highlighted in a particular color. Different groups are assigned varying colors to differentiate between them on the plot.

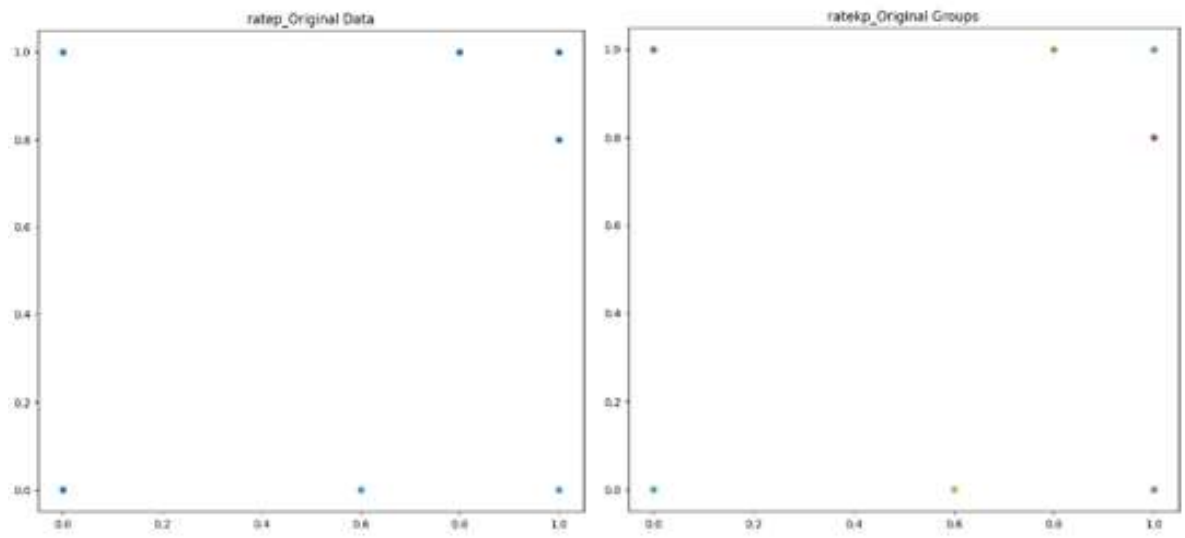

Figure 11: RateP - Grouping of Data

The figure given below shows the centroids marking on the plot. Now that the clusters have been made, the centroids are marked for further assessment of the result. Clusters are formed until the instances in particular groups are closest to its assigned centroid.

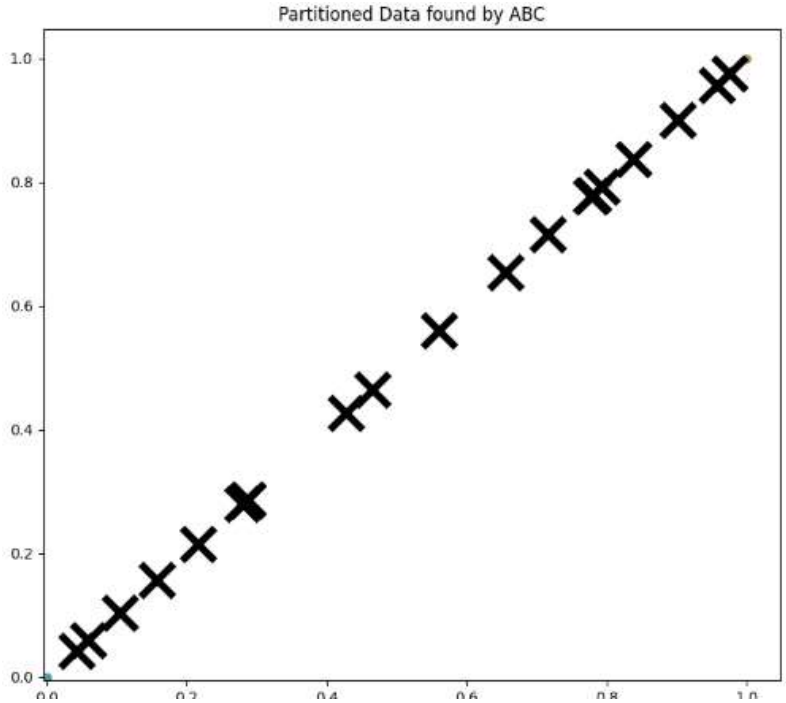

Figure 12: RateP - Partitioned Data

The output of the objective function is displayed in the figure below. The function used is the sum of squared errors as discussed in the previous section. After the algorithm completes

the 1000 iterations as specified, the SSE output in generated which is presented below. As displayed in figure, the SSE plot remains constant throughout the iterations but it doesn't increase. We can easily state that the error has been minimized as the figure below displays the best result that can be achieved for RateP dataset.

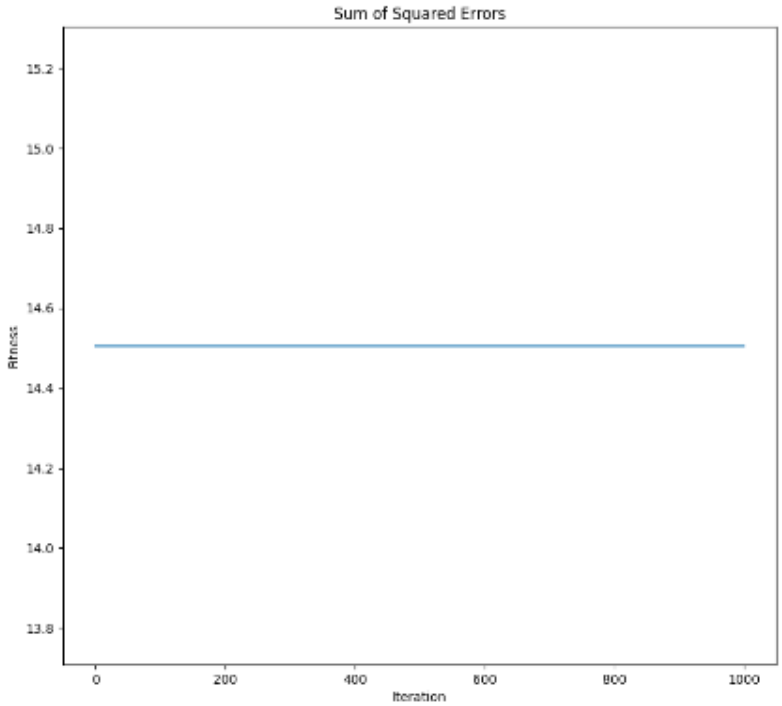

Figure 13: RateP - SSE Output

- **Point Test Profile – Point P**

Point Test Profile or Point P dataset consist of points from the stakeholder. Here, the stakeholders are allocated points to assign to each requirement. The stakeholders regard those requirements with more points that are of high importance to them and less points to those requirements that are less important for them.

The four output files that are generated as a result have been shown through the figure. The four files include, the original data, grouping of data and data partitions and lastly, the figure that displays the output of the objective function.

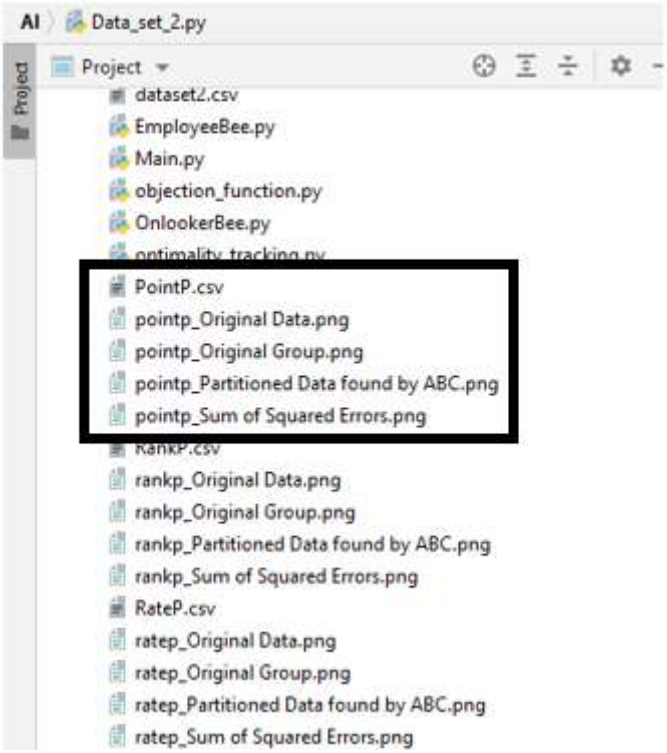

Figure 14: Point Test Profile Outputs

The outcome of the approach depicts that the instances that have more points are grouped together in a particular cluster following the clustering technique. The centroids are marked and then further analysis is performed on the results obtained.

The figure given below is a combination of two plots. The first plot shows that the instances are scattered on the plot in its original form as per the points given by the stakeholders. And the second plot shows that those same instances but now they have been grouped together based on their points.

The plot displays that the instances from particular group appear to be highlighted in a particular color. All the groups are assigned different colors to differentiate between them on the plot.

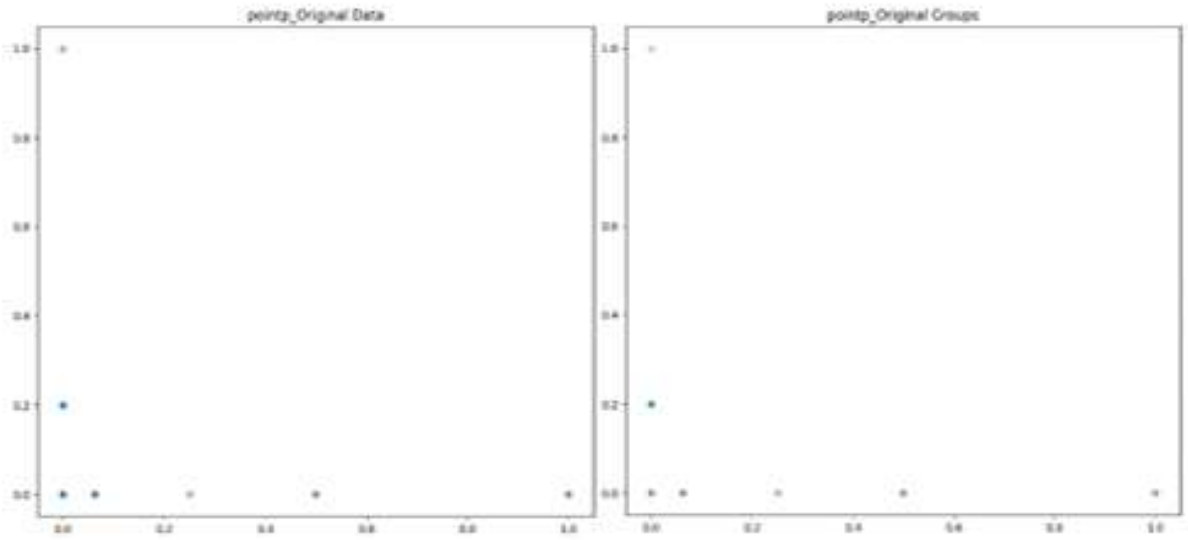

Figure 15: PointP - Grouping of Data

The figure given below shows the centroids marking on the plot. Now that the clusters have been made, the centroids are marked for further assessment of the result. Clusters are formed until the instances in particular groups are closest to its assigned centroid.

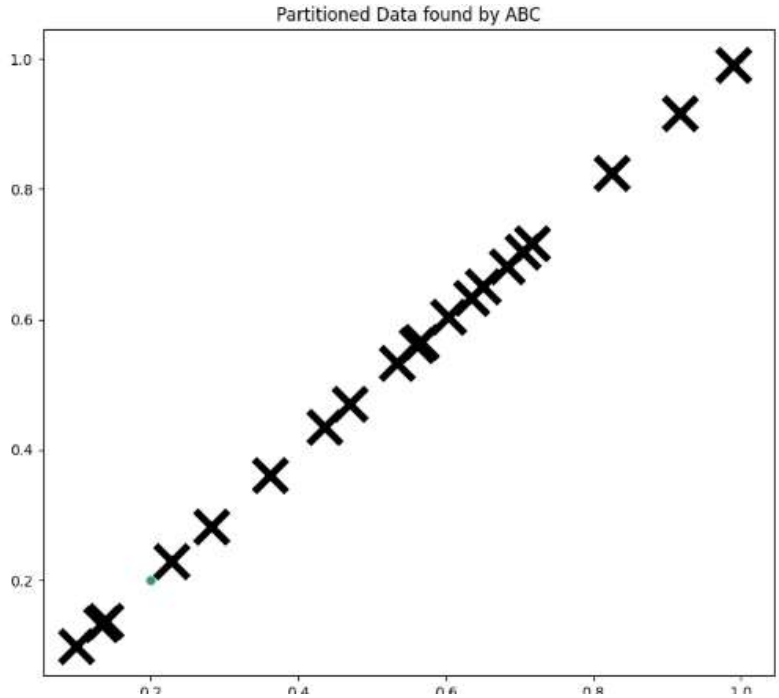

Figure 16: Point P - Partitioned Data

The output of the objective function is displayed in the figure below. The function used is the sum of squared errors as discussed in the previous section. After the algorithm completes

the 1000 iterations as specified, the SSE output in generated which is presented below. As displayed in figure, the SSE plot remains constant throughout the iterations but it doesn't increase. We can easily state that the error has been minimized. The figure below shows the best result achieved by far for the PointP dataset.

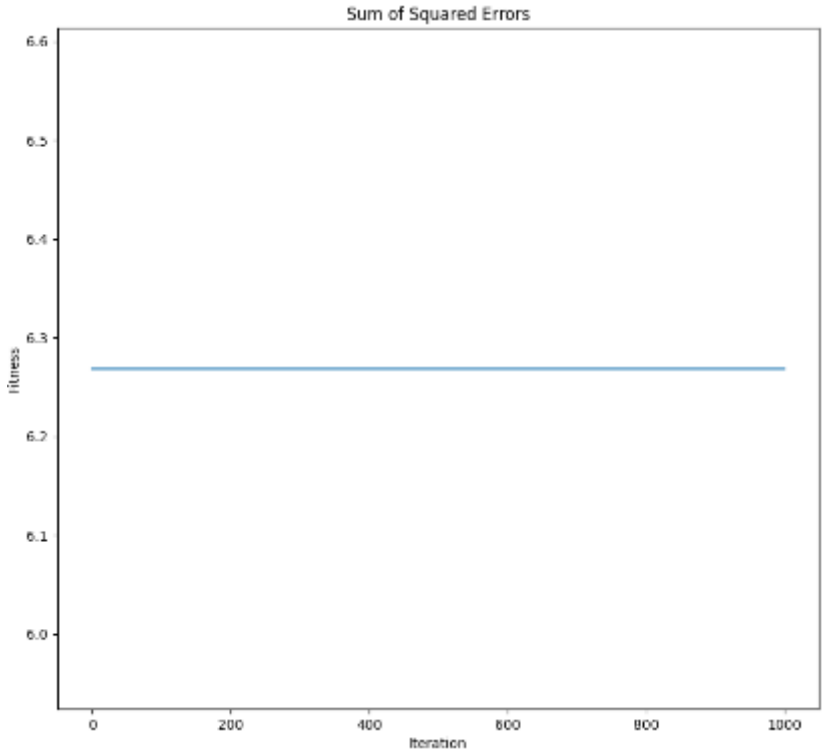

Figure 17: PointP - SSE Output

4.2 Comparison of Clustered Requirements

Analysis of Variance is statistical technique with the help of which we can easily determine the variances of different groups and then perform the comparison between the variance across the mean of such groups.

We have applied the same technique to validate the results achieved so far to compare. This technique has been applied to a subset of the three datasets that we have used in our methodology.

Variables	Obj.	Obj. with missing data	Min	Max	Mean	Std. Dev
RankP	100	0	0.00	100	0.012	5.109
RateP	100	0	0.00	100	2.61	9.733
PointP	100	0	1.98	100	10.32	13.676

Table 4: Datasets Overview

The figure below presents the outcome when the proposed algorithm is on the algorithm for 10 clusters. It shows the clusters containing the requirements with weights and the variance within clusters.

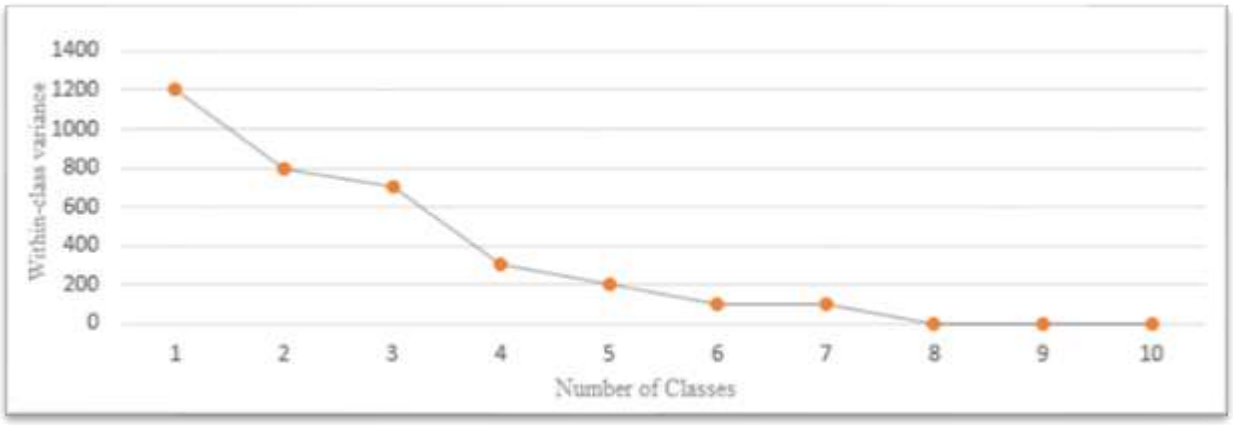

Figure 18: Within-class variance

The table below displays the summary of the datasets which includes the number of objects in these three datasets. The size of each cluster varies from 1 to 30 whereas the mean and the standard deviation as shown in the table lie in the 0-15 and 1-30 boundaries respectively.

Next, we focus on the results of the clusters in the form of centroids. Analysis of the multiple iterations of the algorithm has been discussed. The centroids markings that were

presented before are now discussed in details. Each of the datasets were divided into 10 clusters hence the resulting 10 centroids. The table below shows those centroids for each RankP and RateP and PointP datasets which were computed after the stakeholder priorities. The sum of weights for every cluster is calculated and then the within-class variance is determined.

In [46], the study has been focused on prioritization using the same clustering technique where the variance is determined at the end. A good cluster has comparatively low value of within-class variance. Details of the clusters and its within-class variance achieved from the proposed approach is displayed below in the table.

Class	RankP	RateP	PointP	Sum of Weights	Within-class variance
1	0.23	4.71	6.69	24.00	5.090
2	0.12	4.24	16.65	17.00	6.562
3	0.24	3.45	59.86	14.00	1.009
4	0.11	4.79	24.39	29.00	50.00
5	0.17	3.98	20.45	5.00	1.196
6	0.20	3.55	28.01	12.00	3.609
7	0.39	4.17	86.98	14.00	2.96
8	0.14	1.78	72.01	32.00	1.142
9	0.29	5.10	39.84	8.00	1.00
10	0.43	4.37	19.53	1.00	0.00

Table 5: Centroids Summary

To compare the results from our approach, we had to manually manipulate the data and apply the same technique on the achieved results from clustering to display the slight improvement in variance. Our objective here was to minimize the within-class distance of the instance and as we can clearly see from the table below, that the within class variance is comparatively low.

A comparison table has been presented in the given table to display the outcomes from both the techniques i.e., clustering with the help of ABC algorithm which is the proposed approach and the k-means clustering technique used in the benchmark study. After using the similar datasets, we have obtained the results displayed in the above sections and it is compared with the case study.

	Within-class Variance	
Class	**ABC Clusters (Proposed Approach)**	**Clustering Technique**
1	5.090	7.302
2	6.562	8.282
3	1.009	37.89
4	50.00	172.8
5	1.196	2.393
6	3.609	12.69
7	2.96	3.607
8	1.142	1.992
9	1.00	1.190
10	0.00	0.000

Table 6: Comparison of variance

The above table shows the difference in the results of variance obtained by the both techniques. The within class variance is compared as it depicts the quality of clusters. The lower the within-class variance is, the tighter the cluster is which means that the clusters are formed as a result of high priority basis.

The benchmark study [46] that has been kept in focus while working on our approach had applied cluttering technique for the prioritization of requirement sets. We have applied the clustering technique but with the help of the artificial bee colony algorithm. It has made improvement in the results in comparison to the benchmark in terms of clustering.

Chapter 5

Conclusion & Future Work

CHAPTER 5: CONCLUSION & FUTURE WORK

5.1 Conclusion

In this research, a formal approach for developing a method to help achieve the maximum prioritization of the software requirements is presented. All phases of the prioritization method from data collection to running the particular algorithm and then analyzing the results were discussed in detail. Our research has employed the use of clustering technique with the help of artificial bee colony algorithm to achieve the maximum prioritization of requirements. Three aspects of RALIC dataset have been used in our research for prioritization. Our approach has been validated with the help of these datasets. Each of datasets contained requirements with their numeric priorities assigned by the stakeholders. Our objective was to prioritize the requirements and improve it. After the requirements had been clustered in groups, the mean and the standard deviation for each aspect was calculated to determine the within-class variance at the end. The proposed approach is validated by the benchmark study mentioned before. The experimental outcome of our technique show that it has improved the within-class variance of clusters.

5.2 Future Work

In the future, we intend to further improve the technique with the use of hyperparameter tuning. We can tune the important parameters for the artificial bee colony algorithm, we can aim to improve our results. In addition to that, the resulting clusters can further be analyzed with the ANOVA technique which focuses on variance. Although, the proposed technique is not time consuming, in the future we can aim to modify the approach to work in less time.

The technique has been specially designed for the requirements datasets in the field of requirement engineering. We can, however modify the approach to prioritize other aspect of real life.

References

[1] Alrezaamiri, Hamidreza & Ebrahimnejad, Ali & Motameni, Homayun. (2020). Parallel multi-objective artificial bee colony algorithm for software requirement optimization. Requirements Engineering. 25. 10.1007/s00766-020-00328-y.

[2] Karaboga, Dervis. (2005). An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report - TR06. Technical Report, Erciyes University.

[3] Sharma, Sangeeta & Bhambu, Pawn. (2016). Artificial Bee Colony Algorithm: A Survey. International Journal of Computer Applications. 149. 11-19. 10.5120/ijca2016911384.

[4] Nuseibeh, Bashar & Easterbrook, Steve. (2000). Requirements engineering: a roadmap. 35-46.

[5] Berander, P., Andrews, A. (2005). Requirements Prioritization. In: Aurum, A., Wohlin, C. (eds) Engineering and Managing Software Requirements. Springer, Berlin, Heidelberg.

[6] Khan, Javed & Rehman, Izaz & Khan, Iftikhar & Rashid, Salman. (2015). Comparison of Requirement Prioritization Techniques to Find Best Prioritization Technique. International Journal of Modern Education and Computer Science. 7. 53-59. 10.5815/ijmecs.2015.11.06.

[7] Yaseen, Muhammad & Mustapha, Aida & Ibrahim, Noraini. (2019). Prioritization of Software Functional Requirements: Spanning Tree based Approach. International Journal of Advanced Computer Science and Applications. 10. 10.14569/IJACSA.2019.0100767.

[8] Narendhar, Mulugu, and K. Anuradha. "Different Approaches of Software Requirement Prioritization." International Journal of Engineering Science Invention, Vol. 5 (2016): 38-43

[9] Kitchenham, Barbara & Brereton, Pearl & Budgen, David & Turner, Mark & Bailey, John & Linkman, Stephen. (2009). Systematic literature reviews in software engineering-A systematic literature review. Information and Software Technology. 51. 7-15. 10.1016/j.infsof.2008.09.009.

[10] Alrezaamiri, Hamidreza & Ebrahimnejad, Ali & Motameni, Homayun. (2019). Software requirement optimization using a fuzzy artificial chemical reaction

optimization algorithm. Soft Computing. 23. 10.1007/s00500-018-3553-7.

[11] Chaves-González, Jose M. & Pérez-Toledano, Miguel & Navasa, Amparo. (2015). Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm. Knowledge-Based Systems. 83. 10.1016/j.knosys.2015.03.012.

[12] Sagrado, José & Águila, Isabel & Orellana, Francisco. (2015). Multi-objective ant colony optimization for requirements selection. Empirical Software Engineering. 20. 577-610. 10.1007/s10664-013-9287-3.

[13] Requirements Prioritization Case Study Case Using AHP by Nancy Mead https://resources.sei.cmu.edu/asset_files/WhitePaper/2013_019_001_297260.pdf

[14] Bagnall AJ, Rayward-Smith VJ, Whittley IM (2001) The next release problem. Inf Softw Technol 43(14):883–890

[15] Greer D, Ruhe G (2004) Software release planning: an evolutionary and iterative approach. Inf Softw Technol 46(4):243–253

[16] Marghny, M & El-Hawary, H & Dukhan, Wathiq. (2017). An Effective Method of Systems Requirement Optimization Based on Genetic Algorithms. 15. 10.12785/isl/060102.

[17] Chaves-González, Jose M. & Pérez-Toledano, Miguel & Navasa, Amparo. (2015). Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm. Knowledge-Based Systems. 83. 10.1016/j.knosys.2015.03.012.

[18] Alrezaamiri, Hamidreza & Ebrahimnejad, Ali & Motameni, Homayun. (2019). Solving the next release problem by means of the fuzzy logic inference system with respect to the competitive market. Journal of Experimental & Theoretical Artificial Intelligence. 32. 1-18. 10.1080/0952813X.2019.1704440.

[19] Sagrado, José & Águila, Isabel & Orellana, Francisco. (2015). Multi-objective ant colony optimization for requirements selection. Empirical Software Engineering. 20. 577-610. 10.1007/s10664-013-9287-3.

[20] Ebrahimnejad A, Tavana M, Alrezaamiri H (2016) A novel artificial bee colony algorithm for shortest path problems with fuzzy arc weights. Measurement 93:48–56

[21] Xiang Y, Peng Y, Zhong Y, Chen Z, Lu X, Zhong X (2014) A particle swarm inspired multi-elitist artificial bee colony algorithm for real-parameter optimization. Comput Optim Appl 57(2):493–516

[22] Yurtkuran A, Emel E (2015) An adaptive artificial bee colony algorithm for global optimization. Appl Math Comput 271:1004–1023

[23] Chaves-González JM, Vega-Rodríguez MA, Granado-Criado JM (2013) A multiobjective swarm intelligence approach based on artificial bee colony for reliable DNA sequence design. Eng Appl Artif Intell 26(9):2045–2057

[24] Rubio-Largo Á, Vega-Rodríguez MA, González-Álvarez DL (2015) Multiobjective swarm intelligence for the traffic grooming problem. Comput Optim Appl 60(2):479–511

[25] Delgarm N, Sajadi B, Delgarm S (2016) Multi-objective optimization of building energy performance and indoor thermal comfort: a new method using artificial bee colony (ABC). Energy Build 131:42–53

[26] Thew S, Sutcliffe A (2018) Value-based requirements engineering: method and experience. Requir Eng 23(4):443–464

[27] Reddy, Jogannagari Malla, S. V. A. V. Prasad, and Kothuri Parashu Ramulu. "A Critical Analysis and Evaluation of Requirement Prioritization using Analytical Hierarchy Process." International Journal Of Engineering And Computer Science 6, no. 6 (2017).

[28] Garg, Umang, and Abhishek Singhal. "Software requirement prioritization based on non-functional requirements." In Cloud Computing, Data Science & Engineering-Confluence, 2017 7th International Conference on, pp. 793-797. IEEE, 2017

[29] B. Basturk, D.Karaboga, An Artificial Bee Colony (ABC) Algorithm for Numeric function Optimization, IEEE Swarm Intelligence Symposium 2006, May 12-14, 2006, Indianapolis, Indiana, USA.

[30] Yaseen, Muhammad. (2020). Prioritization of Software Functional Requirements: A Novel Approach using AHP and Spanning Tree. International Journal of Advanced Trends in Computer Science and Engineering. 9. 51-56. 10.30534/ijatcse/2020/09912020. Srivastava, Shweta. (2014). Weka: A Tool for Data preprocessing, Classification, Ensemble, Clustering and Association Rule Mining. International Journal of Computer Applications. 88. 10.5120/15389-3809.

[31] Ismail, Nur Hafieza & Ahmad, Fadhilah & Aziz, Azwa. (2013). Implementing WEKA as a Data Mining Tool to Analyze Students' Academic Performances Using Naïve Bayes Classifier. 10.13140/2.1.4937.8565.

[32] Desai, Aaditya & Rai, Sunil. (2013). Analysis of Machine Learning Algorithms using Weka.

[33] Sathish, C. & Srinivasan, K.. (2021). An artificial bee colony algorithm for efficient optimized data aggregation to agricultural IoT devices application. Journal of Applied Science and Engineering (Taiwan). 24. 927-936. 10.6180/jase.202112_24(6).0013.

[34] Kumar, Anil & Kabra, Gaurav & Mussada, Eswara & Dash, Manoj & Rana, Prashant. (2019). Combined Artificial Bee Colony Algorithm and Machine Learning Techniques for Prediction of Online Consumer Repurchase Intention. Neural Computing and Applications. 31. 877–890. 10.1007/s00521-017-3047-z.

[35] Schiezaro, Mauricio & Pedrini, Helio. (2013). Data feature selection based on Artificial Bee Colony algorithm. EURASIP Journal on Image and Video Processing. 2013. 47. 10.1186/1687-5281-2013-47.

[36] Yu, Tong & Zhu, Hong. (2020). Hyper-Parameter Optimization: A Review of Algorithms and Applications.

[37] Pannakkong, Warut & Thiwa-Anont, Kwanluck & Singthong, Kasidit & Parthanadee, Parthana & Buddhakulsomsiri, Jirachai. (2022). Hyperparameter Tuning of Machine Learning Algorithms Using Response Surface Methodology: A Case Study of ANN, SVM, and DBN. Mathematical Problems in Engineering. 2022. 10.1155/2022/8513719.

[38] Del Sagrado J, Del Aguila IM, Orellana FJ (2015) Multi-objective ant colony optimization for requirements selection. Empir Softw Eng 20(3):577–610

[39] Erkan, Uğur & Toktas, Abdurrahim & Ustun, Deniz. (2022). Hyperparameter optimization of deep CNN classifier for plant species identification using artificial bee colony algorithm. Journal of Ambient Intelligence and Humanized Computing. 1-12. 10.1007/s12652-021-03631-w.

[40] http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm

[41] Thant Z. Win et al 2020 IOP Conf. Ser.: Mater. Sci. Eng. 769 012060

[42] Lim, S.L., Finkelstein, A.: StakeRare: using social networks and collaborative filtering for large-scale requirements elicitation. IEEE Transactions on Software Engineering 38(3), 707–735 (2012)

[43] Lim, Soo Ling. (2011). Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation.

[44] Kaur, J., Gupta, S., Kundra, S.: A kmeans clustering based approach for evaluation of success of software reuse. In: Proceedings of International Conference on Intelligent Computational Systems, ICICS 2011 (2011)

[45] P Achimugu, A Selamat, and R Ibrahim. “A Clustering Based Technique for Large Scale Prioritization during Requirement Elicitations”, Advances in Intelligent Systems and Computing Journal Volume 287 pp 623-63

[46] Setiani, Novi & Dirgahayu, Teduh. (2016). Clustering technique for information requirement prioritization in specific CMSs. 1-6. 10.1109/ICODSE.2016.7936107.

[47] Haraty, Ramzi & Mansour, Nashat & Moukahal, Lama & Khalil, Iman. (2016). Regression Test Cases Prioritization Using Clustering and Code Change Relevance. International Journal of Software Engineering and Knowledge Engineering. 26. 733-768. 10.1142/S0218194016500248.

[48] Arafeen, Md & Do, Hyunsook. (2013). Test Case Prioritization Using Requirements-Based Clustering. Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation, ICST 2013. 312-321. 10.1109/ICST.2013.12.

[49] S. Chaudhary and A. Jatain, "Performance Evaluation of Clustering Techniques in Test Case Prioritization," 2020 International Conference on Computational Performance Evaluation (ComPE), 2020, pp. 699-703, doi: 10.1109/ComPE49325.2020.9200083.

[50] Sjerps, R.M.A., Brunner, A.M., Fujita, Y. et al. Clustering and prioritization to design a risk-based monitoring program in groundwater sources for drinking water. Environ Sci Eur 33, 32 (2021).

[51] L. Xiao, H. Miao, W. Zhuang and S. Chen, "Applying Assemble Clustering Algorithm and Fault Prediction to Test Case Prioritization," 2016 International Conference on Software Analysis, Testing and Evolution (SATE), 2016, pp. 108-116, doi: 10.1109/SATE.2016.25.

[52] S. Iqbal, R. Naseem, S. Jan, S. Alshmrany, M. Yasar and A. Ali, "Determining Bug Prioritization Using Feature Reduction and Clustering With Classification," in IEEE Access, vol. 8, pp. 215661-215678, 2020, doi: 10.1109/ACCESS.2020.3035063.

[53] A. Hudaib and F. Alhaj, "Self-Organizing Maps for Agile Requirements Prioritization," 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS), 2019, pp. 1-5, doi: 10.1109/ICTCS.2019.8923075.

[54] L. Xiao, H. Miao, W. Zhuang and S. Chen, "An empirical study on clustering approach combining fault prediction for test case prioritization," 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), 2017, pp. 815-820, doi: 10.1109/ICIS.2017.7960105.

[55] J. Chen et al., "An Adaptive Sequence Approach for OOS Test Case Prioritization," 2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2016, pp. 205-212, doi: 10.1109/ISSREW.2016.29.

[56] Z. Khalid and U. Qamar, "Weight and Cluster Based Test case Prioritization Technique," 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2019, pp. 1013-1022, doi: 10.1109/IEMCON.2019.8936202.

[57] http://www0.cs.ucl.ac.uk/staff/S.Lim/soolinglim/Datasets.html

[58] Lim, S.L., Harman, M., Susi, A.: Using Genetic Algorithms to Search for Key Stakeholders in Large-Scale Software Projects. In: Aligning Enterprise, System, and Software Architectures, pp. 118–134 (2013)

[59] Shah, Habib & Tairan, Nasser & Mashwani, Wali & Alsewari, AbdulRahman & Jan, Muhammad Asif & Badshah, Gran. (2017). Hybrid Global Crossover Bees Algorithm for Solving Boolean Function Classification Task. 467-478. 10.1007/978-3-319-63315-2_41.