AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF SCIENCE AND TECHNOLOGY

FALL 2022-2023

RESEARCH PROPOSAL

Fault detection in software using Biological Techniques

SUBMITTED BY:

1) ASIF SHARIF AKASH (ID: 20-42647-1)

2) MD. YEASIN RAHAT (ID: 20-43097-1)

3) TAREK HOSSAIN (ID: 20-43018-1)

SUPERVISED BY:

DR. AFZORA NAHAR

TABLE OF CONTENT

1. INTRODUCTION:

The most significant and essential element in a software development life cycle to assure software quality is the software testing and maintenance phase. It is highly desired for software to match consumer expectations because of the rising demand for software and potential for increased reliance on it. Software changes may be made repeatedly depending on demand and modification. The adjustments could alter or include additional specifications for high-quality software [1].

However, it adds to the workload for engineers and developers. The hunt for further techniques that aid in early fault detection is of interest to the researchers. It will lower the cost of various testing methods used in the development and maintenance phases. Regression testing's scope is always crucial because it is used to ensure that modifications are accurate and do not adversely affect the software's unmodified portions. It can be expensive to thoroughly run test suites that were previously accessible from older versions of software.

Xiaoxing Yang et.al (2015) proposed the description of construction of previous work and new study that is beneficial for the construction of software defect prediction model. There are two aspects to be considered in this paper. First is the new application of learning-to-rank technique to real world data sets which will help in predicting the software defect. The second is the comprehensive evaluation of the procedure. It is seen through the empirical studies that the performance measures of the learning-to-rank approach are very effective as compared to the already existing approaches. Xiao may have gave the idea but it limits the idea in software fault detection field. LTR may detect the fault but interface design is not included at his proposal [1].

1.1 PROBLEM STATEMENT:

⇒ Boltzmann Learning is only used to detect software quality and efficiency.
⇒ No interface design was introduced for Learn-To-Algorithm to detect software faults.
⇒ Test cases using particle swarm optimization is limited to severe cases only.

1.2 RESEARCH QUESTIONS:

⇒ What issues have various testing methods employing biological technique resolved?
⇒ What are the advantages and disadvantages of various biological approaches used in software engineering?
⇒ What are the various validation methods for detection and prediction?

1.3 RESEARCH OBJECTIVES:

⇒ To find the biological strategies for optimization
⇒ To identify the issue utilizing methods and procedures
⇒ To simulate and interpret the experimental data

1.4 LIMITATION OF THE STUDY:

⇒ The algorithm of LTR is limited to only software interface design
⇒ Test cases more than 10 cannot have appropriate results of showing faults
⇒ Boltzmann method may give the idea of global energy to sustain the approach but lacks in LTR sometime.
⇒ Particle swarm optimization limits to genetic field only.

2. BACKGROUND STUDY:

2.1 BIOLOGICAL TECHNIQUES:

Methods or procedures used to examine living things are referred to as biological techniques. They comprise techniques, approaches, procedures, and tools for biological research that are experimental and computational. Some techniques used for software detection are as follows:

2.1.1 FINGERPRINTING: By using a fingerprinting technique, a dual modular redundant (DMR) processor pair's execution can be distinguished from one another. A hash-based signature allows one to view the history of operations performed by a processor. The contrasts between two mirrored processors can be seen through comparisons of the obtained fingerprints [2].

2.1.2 PROCEDURE DUPLICATION: The programmer in this system replicated the majority of the crucial processes. When these operations are run on two distinct processors, the results are compared. The programmer selects the procedures that will be reproduced. The outcomes obtained here have also undergone a fair examination. There is a manual alteration made to the codes, and the system may have a number of faults [2].

2.1.3 ERROR DETECTION BY DUPLICATED INSTRUCTIONS: The computed results from the master and shadow instructions are put through a number of comparisons before being written into the memory. In the event that a mismatch occurs, the application restarts and also jumps to an exception handler [2].

3. METHODOLOGY & ANALYSIS:

3.1 LEARN-TO-RANK:

The various machine learning techniques that are designed for training the models present within a ranking task are known as learn-to-rank techniques. There are three different categories present within this approach. They are:

1) The pointwise approach
2) The pairwise approach
3) The listwise approach

3.2 BOLTZMANN LEARNING:

Boltzmann machines are networks of symmetrically connected units that depend on stochastic decisions to be on or off. The Boltzmann machines' straightforward learning techniques are used to identify the complicated distributions connected to the observed data. There are various scientific tasks that are used here for learning.

The global energy, E, in a Boltzmann machine is identical,

$$E = -(\sum_{i,j} w_{i,j} s_i s_j + \sum_i \theta_i s_i)$$

Where w_{ij} is the connection strength between i and j unit, s_i is the state θ_i is the bias of unit i in the global energy function [1].

3.3 PARTICLE SWARM OPTIMIZATION:

A novel technique for solving optimization problems is particle swarm optimization, which is compared to genetic algorithms [3]. PSO is used to optimize a complex problem, such as the traveling salesman problem or the job scheduling problem. PSO is also used in software testing because creating test cases is an NP-complete problem and testing is a complex problem. It is also simple for optimization it has only two basic equation-1 is for velocity and equation-2 is for new position.

v[] = wv[] + c1 * r1() * (pbest[] - present[]) + c2 * r2() * (gbest[] - present[])---(1)

present[] = present[] + v[]----------(2)

Here, the constants w, c1, c2, r1, r2, pbest, and gbest stand for local best and global best of particle, respectively. The test case for software testing is a problem particle that is managed by an objective function with the goal of focusing on satisfying the requirement. If we compare this particle to a genetic algorithm, it is a chromosome. In comparison to genetic algorithms, PSO has produced better results in terms of convergence speed. Through PSO, test case optimization works very well for defect detection in a shorter amount of time [3].

3.4 PROPOSED ALGORITHM:

Init population P(t)

Evaluate P(t);

t:=0;

```
Network ConstructNetworkLayers()
InitializeWeights(Network,test cases)
For(i=0; i=testcases; i++)
        SelectInputPattern(Input fault values)
        ForwardPropagate(P)
        BackwardPropagateError(P)
        UpdateWeights(P)
End
Return(P)
        While not done do
                t:=t+1;
        P':=test case P(t);
recombine P'(t);
        mutate P'(t);
                evaluate P'(t);
                P:=survive P.P(t);
                end
```

3.5 FLOWCHART:

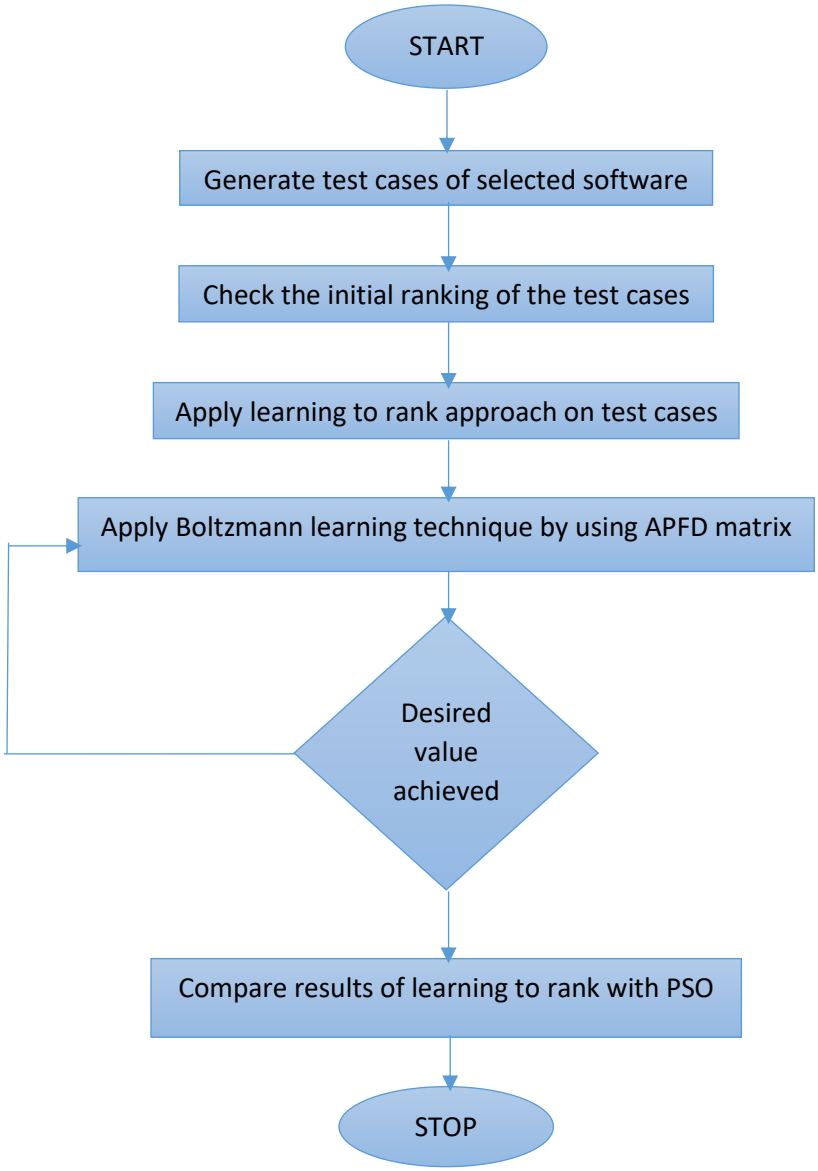

Figure-1: Proposed flowchart

4. SIMULATION RESULT:

MATLAB is used to implement the Learn-to-Rank and Improved Learn-to-Rank algorithms. The implementation given in Table-1 takes into account the dataset. [5]

Table-1: Properties of dataset

Attributes	Values
Number test cases	10
Repeated test cases	No
Fault in the Test cases	Yes
Number of application	1

The suggested algorithm is put into practice, and an interface is created for it, as illustrated in the images below:

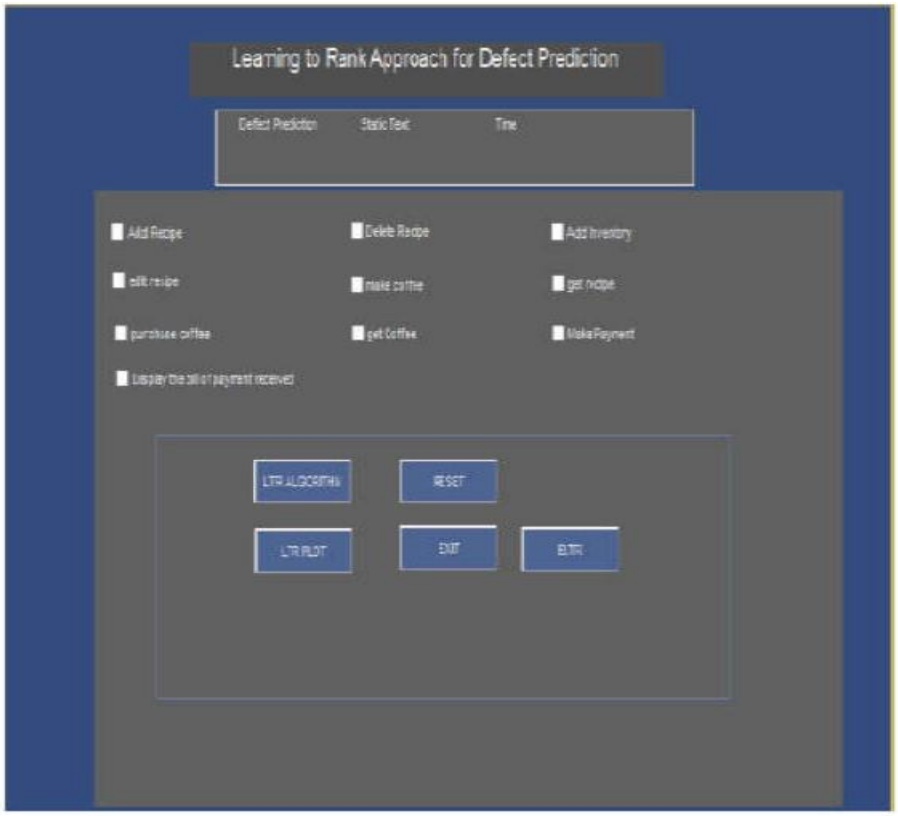

Figure-2: Interface is designed for implementation

The interface for the Learn-to-Rank and improved rank learn algorithms is designed, as seen in figure 2. The UI displays ten test cases. Here, the suggested and existing algorithms are put into practice. The outcome is examined in relation to the defect detection rate parameter [4].

5. WORK SCHEDULING:

REFERNCE:

[1] S. S. JAGATJOT SINGH, "FAULT DETECTION TECHNIQUE FOR TEST CASES IN SOFTWARE," *International Journal of Advances in Electronics and Computer Science*, vol. 4, no. 8, pp. 2393-2835, 2017.

[2] B. D. Cagatay Catal, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," *Information Sciences*, p. 1040–1058, 2009.

[3] P. R. Sushant Kumar, "A Comprehensive Analysis for Software Fault Detection and Prediction using Computational Intelligence Techniques," *International Journal of Computational Intelligence Research,* vol. 13, no. 1, pp. 65-78, 2017.

[4] G. A. ·. A. Selamat, "A survey on software fault detection based on different prediction approaches," *Vietnam J Comput Sci,* vol. 1, pp. 79-95, 2014.

[5] C. B. ,. a. M. Y. Jiang Y., "Techniques for evaluating fault prediction models," *Proc. Empiric. Softw,* vol. 13, no. 5, pp. 561-595, 2008.