IJCSE International Journal of Computer Sciences and Engineering Open Access
Survey Paper Vol.-6, Issue-12, Dec 2018 E-ISSN: 2347-2693

A CSA based Source Code Plagiarism Detection Approach using Sparse Principle Component Analysis

M. Bhavani[1*], K. Thammi Reddy[2], P. Suresh Varma[3]

[1]Dept. of IT, GITAM institute of Technology, GITAM, India
[2] Dept. of CSE, GITAM institute of Technology, GITAM, India
[3] Dept. of CSE, Adhikavi Nannayya University, Rajamahendravaram, India

[*]*Corresponding Author: bhavani.mm@gmail.com*

Available online at: www.ijcseonline.org

Accepted: 07/Dec/2018, Published: 31/Dec/2018

Abstract-Detection of source code plagiarism is valuable for both the academia and industry. Plagiarism is an approach of unlawfully stealing other person source code or program code which is a serious issue for common open source programming and other software companies. Numerous techniques have been introduced priori for automatic detection of source code plagiarism using Evolutionary Intelligent algorithm like Genetic Algorithm (GA), Particle Swarm Optimization (PSO) etc. These techniques are more susceptible to premature convergence and more time consuming. In this paper, considering the benefits of artificial immune system, source code plagiarism approach is proposed that overcomes the drawbacks of previous genetic algorithm and particle swarm optimization algorithms. The sparse PCA is employed for dimensionality reduction prior to detection approach for obtained sparse matrix. Using CSA, the detection between source codes is computed and fitness evaluation is measured using Normalized Euclidean distance (NED) and Normalized Cumulative Reciprocal Rank (NCRR).The performance analysis of the suggested approach showed that it has better precision and recall values when compared with existing Meta heuristic based Source code plagiarism detection algorithms.

Keywords-Source Code detection, Plagiarism approach, Artificial Immune System, Clonal Selection Algorithm, Sparse PCA.

I. INTRODUCTION

Source Code plagiarism is well-defined as the process of stealing others source code through unlawfully photocopying their information, using code obfuscation approaches for the code to observe diverse and further demanding that it is individual program in a manner that violates the conditions of original authorization. In current years, source code plagiarism has been a severe concern for authentic academician and students and open source societies. It interrupts the intelligent things of software developers and is an austere issue, varying from open source code recycle, product theft to various applications and repackaging. The stolen code could be employed through plagiarists to diminish the price of its software development.

Rendering to a current survey [1], it was discovered that 1083 i.e. 86% of 1260 malevolent app instances were repackaged varieties of genuine apps having malevolent contents. Furthermore, the growing of software provides plagiarists further chances to steal others' code. The surge of open source tasks gives lot of simple objectives for source code robs, as source code is flexible to know and alter compared to working binaries.

Source code plagiarism is a key problem that rises in most of the programming course [3]. Huge quantity of data accessible online causes plagiarism highly flexible to obligate, and this is specifically correct in case of source code. The conventional technique of recognizing copied information in a course is physical examination. This is not merely the tiresome job but characteristically omits code plagiarized from exterior sources or from prior courses provided. Nevertheless, identifying plagiarism manually is fairly time taking. Although mechanically identifying source code plagiarism persists, these incline to concentrate on smaller suggestion sets. Whenever identifying plagiarism in source codes, linguistic precise characteristics are frequently required, particularly to identify progressive plagiarism attack. To overwhelm this issue, numerous spontaneous approaches for identifying source code plagiarism are introduced.

Numerous techniques have been introduced priori for automatic identifying source code plagiarism. In the similar lines, the author has also present two different intelligent source code plagiarism detection approaches using Genetic Algorithm, Particle Swarm Optimization on sparse data matrix. Though GA and PSO are evolutionary intelligent approach, these are more susceptible to premature convergence and more time consuming. In PSO, whenever no superior global best is discovered through another particle for certain time period, entire particles tends towards persisting global best, hypothetically eradicating even the nearby local minimize and find no guarantee in obtaining the global minima. Thus, a clonal selection approach, in this paper is used for detection of Plagiarized codes employing the similarity measures.

The Dimensionality reduction (DR) approach depending on sparse representation considered to be the fieriest investigating themes have attained amazing efficiency in numerous applications in current years. Nevertheless, it's a challenge for prevailing sparse representation aided approaches to resolve non-linear issues pertaining to the restriction of in search of sparse representation of information in original domain. Hence, a sparse PCA technique is employed for the reduction of features in the preprocessed sparse matrix prior to detection phase.

A brief introduction to source code detection and its importance along with the motivation for the suggested methodology is given in this section. The section 2 briefly discusses the existing methodologies in source code detection techniques. The proposed CSA-based detection approach is briefly illuminated in the section 3. The experimental outcomes and its analysis for the proposed approach is given in section 4 followed by conclusion and references given in section 5 and section 6 correspondingly.

II. LITERATURE SURVEY

Source code plagiarism in an action of employing function deprived of appropriately quoting the original author [4]. It is an evolving problem in Computer Science (CS) foremost owing to higher submission incidence [5] and recognition complexity [6]. Therefore, to handle this problem, numerous source code plagiarism detection methods are suggested [7].

In [8], exploits implementation traces of Java byte codes are exploited to identify plagiarism with a hypothesis that resemblance could be measured using dynamic behavior. This methodology is not real as both byte codes could be implemented that is not effective with time, particularly on NP-complexity byte codes. However, in the programming course, maximum programs continuously have identical dynamic behavior pertaining to task constraint. In [9], Java byte codes are explored for identifying plagiarism through employing official approach to define byte code similarity. However, using official technique might produce time-efficacy, disadvantage that is not appropriate for distinguishing a heaps of plagiarism circumstances.

In plagiarism detection [10], the other methodology was suggested that are accurately fascinated in stop words in texts. Specified an article and a list of stop words, the text would be lessened to appearance of stop words in the document. The interest of this illustration is the existences of the stop words reveal indices of the syntactic structure of the document, which is probable to remain stable during the procedure of plagiarism of a passage, i.e., when one tries to plagiarize a passage of the text, the most common act is to replace words and expressions by synonyms. Whereas remaining in the scope of plagiarism detection, [11] are concerned in the study of intrinsic plagiarism that aims to recognize potential plagiarism by investigating undeclared changes in the writing style of a document.

The method of semantic plagiarism detection [12] uses the similarity of the chains of characters depending on the fuzzy semantics. The strategy was produced through four principle steps. The initial step is pre-processing comprises of tokenization, deletion of stop words. The second step is to recover a list of candidate documents for each suspect document using the Jaccard algorithm and the shingle algorithm [13]. Another methodology [14] that associates the semantic similarity model with one of the vector models with the Vector Regression (SVR) regression to differentiate the semantic similarity score from the given sentence pairs. This method starts by pre-processing the suspect document, removing hyphens, punctuations etc., the remaining words have been tokenized, lemmatized, labeled according to the Parts of Speech and annotated with labels, constructing pieces.

Additionally, [15] distinguished the way that scholastics frequently have their very own customized impression of plagiarism definitions and policies which may not be consistent with their University's plagiarism policy. Additionally, [16] initiate that scholastics did not often pursue the University's policy by dealing with plagiarism due to concerns about confronting students and not feeling protected by University procedures and discovered that scholastics felt sensitivity for the effect that formal strategies would have on understudies.

III. PROPOSED CLONAL SELECTION BASED PLAGIARISM DETECTION APPROACH

In this section, a novel approach is introduced for the source code plagiarism detection using intelligent heuristic approaches and sparse based dimensionality reduction. Two different similarity measures such as normalized Euclidean distance and normalized cumulative reciprocal rank are used in the detection phase to detect many to many related source code documents. Most of the large scale higher dimensional source code data available now a day are sparse. For this purpose, a sparse PCA based dimensionality reduction is employed. The intelligent detection is accomplished using CSA which has higher benefits compared to GA and PSO. The diagrammatic representation of the proposed approach is specified in Fig 2. This approach is segregated into three phases. They are:

- ➢ Source Code Pre-processing
- ➢ Dimensionality reduction on sparse matrix
- ➢ Intelligent Detection approach

A Source Code Pre-processing

This phase is particularly employed to pre-process the source code document as to improve the retrieval of semantic information for code recognition where the irrelevant and unwanted information such as meaningless terms and characters, symbols or words etc. are removed. This phase is necessary to minimize the dimension of the information to further effectively seize the semantic depiction of every source-code file. The goal of this module is to accumulate the large number of source code into a processed format to detect the plagiarized source code relevantly.

Pre-Processing the source code can be of two forms such as pre-processing constraints that explicit to source code sources and parameters that are not specific to source code files. Pre-processing constraints that specific to source code involves:

- Eliminating commentaries
- Merging or separating terms comprising of compound words
- Eliminating source code identifiers containing complex parameters that joined two words together found within terms and treated it as single term such as ‘student name’ to ‘studentname’.
- Plotting alternative words to a single form like function being plotted to procedure
- Reorganization the procedure according to the order of its function calling
- Eliminating entire tokens that does not have the lexicon of the target language such as eliminating entire words which are not language reserved words

Certain conceivable pre-processing constraints not specific to source-code achieves involves:

- Eliminating words present in single document or entire document as these words preserves no additional knowledge regarding the association among the documents.
- Eliminating words merely comprising of numerical symbols,
- Eliminating syntactical tokens like semi-colons, colons, comma etc.
- Eliminating words comprising of a one alphabets
- Translating upper case alphabets to lower case.

After the pre-processing is performed, the Source Code Pre-processing phase forms the Vector Space Model (VSM) that represents the source code data samples. In the VSM, a term-by-file matrix is represented as $A_{m\times n} = [a_{ij}]$ where every row i have the rate of processed terms such as terms obtained in source-code document next to pre-processing, and every column j signifies the source-code document. Therefore, every element aij of A comprises of the rate at which the vocabulary term i occurs in a source-code file j. From the term-by-file matrix, the normalized term frequency is obtained by applying probability inverse global weighting method (IDFP) [17] to modify the rate of terms relating to the whole group of source-code archives. Similarly, document length normalization is performed to adjust the frequencies depending on the dimension of every document file. The two estimations are given below:

$$g_i = \log\left(\frac{N-n_i}{n_i}\right) \qquad (1)$$

$$l_i = \frac{1}{\sqrt{(\Sigma_i(g_i \cdot a_{ij}))^2}} \qquad (2)$$

Here g_i is the probability inverse global weighting method, N is number of source code files in the group, n_i is the number of source code files where the word i occurs. l_i represents document length normalization. After the evaluation, every entry of the matrix A is updated as:

$$a_{ij} = a_{ij} \times g_i \times l_i \qquad (3)$$

B Dimensionality Reduction

In this section, the higher dimensional sparse matrix A_{mXn} is reduced to lower dimensional sparse matrix employing sparse PCA algorithm using Iterative elimination approach. A prominent limitation of PCA is the deficiency of sparsity. Classically, entire loadings of principal components are nonzero. From modeling view point, even though the interpretability of linear amalgamations is typically flexible for lower dimensional information, it could become further complex whenever number of variables becomes large. Thus, sparse PCA based approach is employed in this paper which also known as Iterative Elimination (IE) algorithm. This methodology is inspiring through famous Recursive Feature Elimination (RFE) approach in learning philosophy and feeble thresholding technique. In this procedure variables are repetitively eradicated using a ranking strategy that could either be the minimum absolute value strategy or the much-cultured approximated minimal variance loss (AMVL) standard.

Based on this hypothesis, the iterative elimination approach for sparse PCA eradicates smaller part of variables at single time and re-iterates this approach till the preferred sparsity is achieved. The notion of iterative elimination is inspired through RFE approach.RFE is a backward FE approach which is presented to the support vector machines in [18]. The rudimentary notion of this approach is, direct variable position (rendering to certain assessed strategy) might be uneven. Nevertheless, variable graded as minimum significant is infrequently amongst the higher significant ones. Thus, this could be eradicated it initially and re-rank the continuing variables. Owing to the decrease in the size, the ranking improves.

In sparse PCA, it is time taking to evaluate the precise variance loss for entire variables. The assessed quantities are adopted as ranking strategy employing the absolute values of loadings or higher restrictions of variance losses. Beneath this strategy, if a variable graded to be the smallest significant one, it does not essentially use the smallest variance. Nevertheless, its support is comparatively smaller and eliminating it would not consequence in huge variance loss. Thus, it is possibly not amongst the preferred top *k* key variables. This is the location where RFE could perform better. The iterative elimination approach for sparse PCA is given as:

1. Initialize $\Sigma^{(t)} = \Sigma, S_t = \{1,2, \ldots .p\}, R_t = \theta$

2. In step t,evaluate the highest eigenvalue $\lambda_1^{(t)}$and consistent Eigen vector$v_1^{(t)}$. Discover the smallest vital variablei_t: if MAV strategy is employed,

$$i_t = arg\ \min_{i\epsilon S_i} |v_{1,i}^{(t)}| \qquad (4)$$

or, if AMVL strategy is employed

$$i_t = arg\ \min_{i\epsilon S_i} \frac{(v_{1,i}{}^{(t)})^2(\lambda_1^{(t)} - \Sigma_{ii}^{(t)})}{1-(v_{1,i}{}^{(t)})^2} (5)$$

Update$S_{t+1} = S_t \backslash i_t, R_{t+1} = R_t \cup \{i_t\}$ and$\Sigma^{(t+1)} = \Sigma(S_{t+1}, S_{t+1})$;

3. Stop until $|S_t| = k$and output $u_1$with $u_1(S_t) = v_1^{(t)}$and $u_1(R_t) = 0$.

Iterative elimination is feasible for issues comprising of high size. Higher *p*, smaller *n* issue denotes to an issue with higher dimensional information however restricted with interpretations. In every generation, it is essential to evaluate its highest singular value and consistent singular vectors.

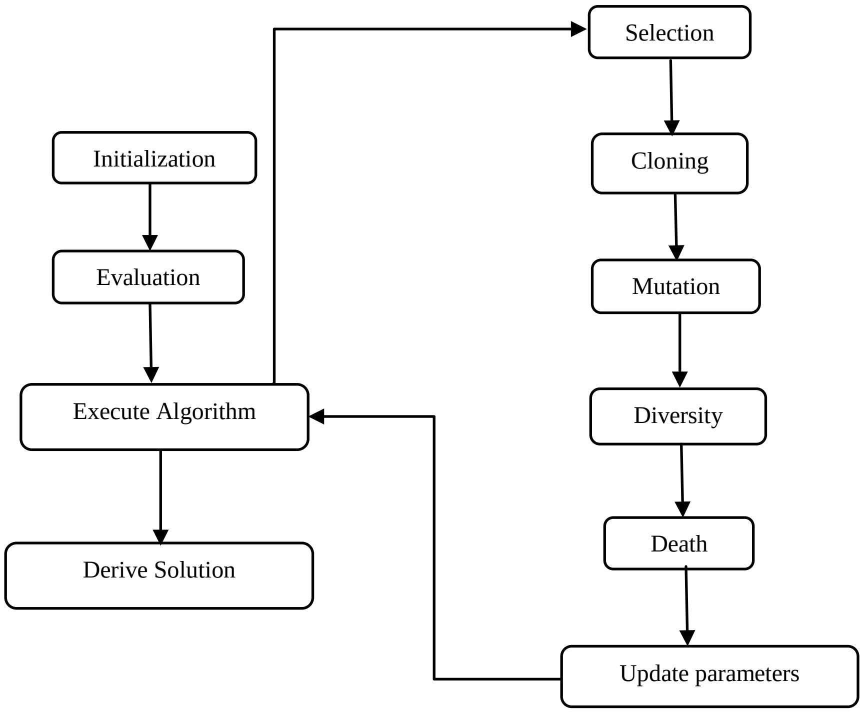

Fig 1: Block Diagram Traditional CSA

C Intelligent Detection Phase

In this section, pair of source codes are detected that are accurately like one another. This detection is achieved using robust meta-heuristic algorithm known as Clonal Selection Approach**.** This is an exceptional group of AIS that employs the clonal selection portion to be the key technique. This procedure was primarily suggested to resolve non-linear functions through [19, 20]. The sparse matrix A_{mXn}

obtained from the Sparse PCA is employed for the detection of pair of source code.AIS is an evolving domain of study in computational intelligence. Most of the previous effort in the development of AIS was employed through genetic and evolutionary evaluating approaches [21]. GA and AIS are the variants of evolutionary approach however the vital difference amongst them is the way the population emerges. In GA, the population is obtained employing crossover and mutation. Nevertheless, in AIS, regeneration is asexual where every child generated through a cell is the precise copy of its parent. Both approaches employ mutation to modify the offspring of cells to preserve diversity in the population [22]. The following offers a comprehensive approach of CSA for detection:

1. *Initialization*: This comprises of inhabiting the antibody group i.e. the minimized sparse matrix with ***S*** arbitrarily nominated antigens (deprived of replacement in the course of initialization) i.e. the elements within the sparse matrix. The Initial dimension of population (S) describes the count of antigens using which to obtain the antibody population, here $S \in (0, p]$ and p is the complete partition dimension.

2. *Loop***:** This includes executing the key stages of approach for Giterations that expresses the complete count approach generation to accomplish, where a unique generation observes that system is visible to entire identified antigens. The factor regulates the quantity of learning the system would function on the domain specific. The iterations varying to higher might consequent in system over-learning the issue or in getting jammed on a locally optimum result.

3. *Selection and Pruning***:** This comprises of exhibiting the complete population to the group of antigen and computing fitness values for every antibody. A group of n antibodies are picked from complete pool having high affinity or having antigen. Those antibodies having a fitness value less than threshold are pruned from picked selected group and base antibody population. The affinity evaluation is made as below:

Affinity Evaluation: The evaluation of each particle i.e. the source code file in the search space is evaluated with the fitness evaluation functions such as Normalized Euclidean Distance between the pair of selected individual's source code documents as to obtain the value of similarity between them. The Normalized Euclidean Distance is given as:

$$ED(x, y) = \frac{1}{2}\left(\sum_{i=1}^{k} \sqrt{\frac{((x_i - \mu(x)) - (y_i - \mu(y)))^2}{\sigma(x) - \sigma(y)}}\right)$$

(6)

The Similarity Measure that is employed for the proposed approach is Normalized Cumulative Reciprocal Rank which is evaluated as:

$$NCRR = \sum_{i=1}^{D}(plag(D_i) \times \frac{1}{i}) \div \sum_{i=1}^{|R|} \frac{1}{i}$$

(7)

Here D is the group of obtained document pairs, R is pair of acknowledged plagiarized document pairs and $plag(d)$ returns 1 for a plagiarized document pair and 0 for a non-plagiarized pair. The fitness values range from 0.0 to 1.0. NCRR similarity measures shows the proportion of the retrieved documents with respect to the relevant documents.

4. *Cloning and Mutation***:** The chosen group is further cloned and mutated employing fitness proportional processes. The number of clones made from every n picked antibodies is proportionate to its fitness using a rank aided measure. Through first sorting the set of selected antibodies is achieved in ascending order by their affinity to the antigen. The ordered list is then iterated, and the number of clones formed from each antibody is considered as follows:

$$num_{clones} = [\frac{\beta.N}{i} + 0.5] \qquad (8)$$

Whereβ is a clonal factor, *N* is the dimension of antibody pool, and i is antibody present rank where $i \in [1, n]$. The complete count of clones obtained for every antigen exhibited to system is consequently evaluated as:

$$N_c = \sum_{i=1}^{n}[\frac{\beta.N}{i} + 0.5] \qquad (9)$$

Where$N_c$is the complete count of clones, *n* is the count of picked antibodies. The Clonal factor (β) **a**grees a scaling factor for the number of clones created for selected antibodies. Assuming an N value of 100, and a β value of 0.5, then using Equation 8, the number of clones created for the for the most stimulated antibody would be 200. Common values are $\beta \in (0,1]$. The lower the value, the more search in the local area (in relation to current antibodies) is performed by the algorithm.

5. *Insertion*: The produced clones are injected within the central antibody population. *n is the* arbitrarily preferred antigens from set are injected into population, where *n* is the count of antibodies in the chosen group from step 3.
6. *Final Pruning***:** This refers to the organizing antibody population for production application. The population is proposed to antigen population an ending time, fitness scores are ready, and pruning is according to the threshold value.
7. *Classification*: The antibody population is the group of examples. For the defined unclassified data example, it is exhibited to population. The ***k*** finest matches (high affinity) are preferred, and the major vote for class of antigens is applied to unclassified example.

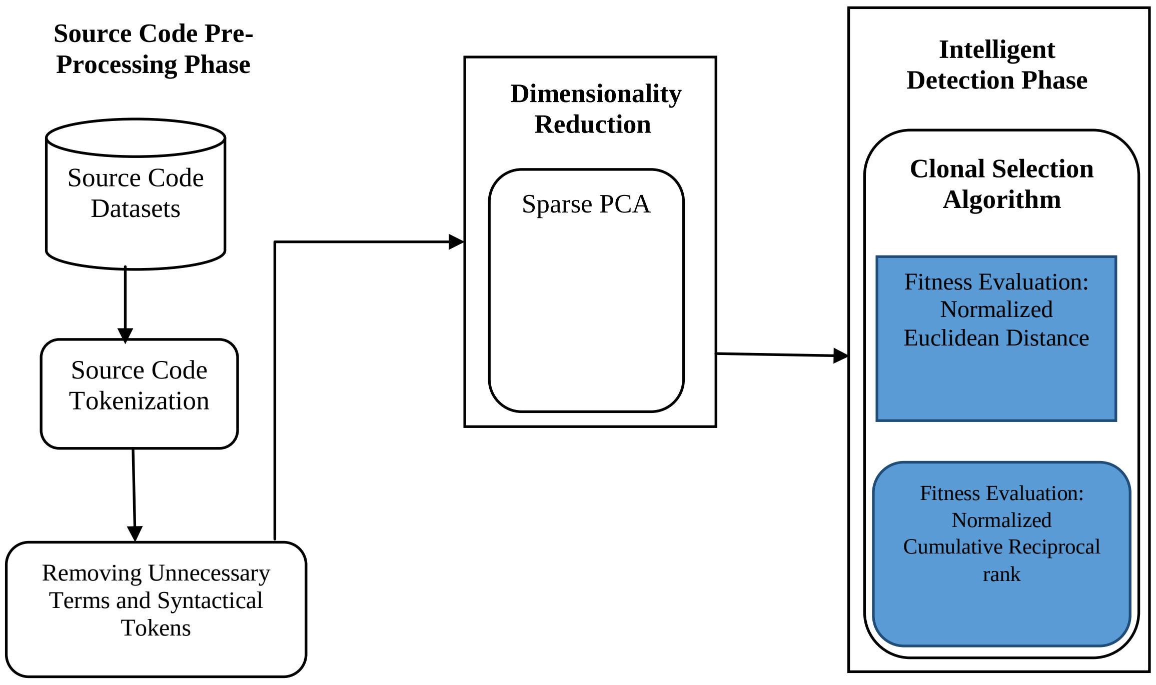
Source Code Pre-Processing Phase
Source Code Datasets
Source Code Tokenization
Removing Unnecessary Terms and Syntactical Tokens
Dimensionality Reduction
Sparse PCA
Intelligent Detection Phase
Clonal Selection Algorithm
Fitness Evaluation: Normalized Euclidean Distance
Fitness Evaluation: Normalized Cumulative Reciprocal rank

Fig 2: Block Diagram of the Proposed Source Code Plagiarism Detection approach

IV. EXPERIMENTAL RESULTS AND ITS ANALYSIS

The Experimental Results for the proposed CSA based Source Code Detection System is carried out using different kinds of datasets one in Java language and the other in C language. Specifically, the source codes with different sizes are considered and the proposed methodology irrespective to the size of the source code. The proposed approach is compared with the existing detection systems such as detection system using incremental genetic algorithm by means of sub graphs [23] and An Iterative Genetic Algorithm Based Source Code Plagiarism Detection Approach Using NCRR Similarity Measure [24] and Particle Swarm Optimization based Source Code Plagiarism system.

Performance Evaluation Measures:

Recall and Precision are two benchmarked and utmost recurrently employed metric in information retrieval system to estimate. These measures are exploited to estimate the efficiency of plagiarism detection. For the reason of estimation, the terms are specified below:

i. Suspicious pairs: Every suspicious pair, sp, comprises of documents that are being refereed through human graders as suspicious. A class of suspicious pairs is given as $SP = \{sp_1, sp_2, \dots . sp_x\}$ where the complete amount of known suspicious pairs in a class (i.e., data sample) is $x = |SP|$.

ii. Innocent pairs: These are the pairs that does not segment any suspicious identity type however is identified as suspicious by proposed and existing detection systems.

iii. Detected pairs: These pairs are obtained by means of the proposed Detection approaches. A class of identified pairs is referred as $DF = SD \cup NS = \{sd_1, sd_2, \dots sd_x\} \cup \{ns_1, ns_2, \dots ns_y\}$.

Here $SD \subseteq S$. The complete count of detected file pairs is $|DF|$. The complete count of suspicious pairs identified is given by $|SD|$, and the total amount of innocent file pairs identified is referred as $|NS|$.

Recall is given as R, where $R \in [0,1]$, is the percentage of suspicious pairs that are recognized depending on limit value, $\emptyset$. Recall is 1.00 whenever entire doubtful pairs are recognized.

$$Recall = \frac{|SD|}{|SP|} = \frac{number\_of\_suspicious\_file\_pair\_detected}{total\_suspicious\_file\_pairs}$$

Precision is given as P, where $P \in [0,1]$, is the percentage of suspicious pairs that are recognized in the group of document pairs identified. Precision is 1.00 whenever each document pair identified is doubtful.

$$Precision = \frac{|SD|}{|DF|} = \frac{number\_of\_suspicious\_file\_pair\_detected}{total\_file\_pairs\_detected}$$

The complete performance of every device is estimated through merging precision and recall metrics. Considering, it to be a unique measurement for estimating the efficiency of device for plagiarism detection, weighted total of precision and recall would be evaluated as

$$Fscore_{\beta} = \frac{(\beta^2 + 0.1)(Precision \times Recall)}{\beta^2 \times (Precision + Recall)}, \in [0, 1]$$

The β coefficient obtains a way to bias $Fscore$ in direction of Precision or Recall. Specifically, the value β=0.5 biases it in the direction of precision, value β = 1.0evaluates precision and recall similarly and value β = 2.0 biases in the direction of recall. In the experimentations, entire three conditions are verified to define the comparative efficiency of several algorithms while highlighting recall or precision, and both. Henceforth, to penalize false negatives further powerfully compared to false positives through picking a value β> 1, therefore provides higher weight age to Recall.

Experimental Results

The precision, recall and $Fscore$ with different β are evaluated for the proposed CSA based Source Code Detection System against the existing approaches given in [23, 24] for two different sets of data samples separately. Table 2 represents the performance measures of the proposed approach that are matched with the existing detection system using the java programming source code data samples.

Table 2: Comparison of Performance Measures on Java Source Code Data Samples

Performance Measures	Proposed Clonal Selection Approach	Particle Swarm Optimization Approach	Iterative Genetic Algorithm based Approach	Incremental Genetic Algorithm based Detection System
Precision	0.99	0.98	0.97	0.96
Recall	0.01	0.02	0.03	0.04
$Fscore_{0.5}$	1.00	1.00	1.00	0.96
$Fscore_{1.0}$	1.00	0.99	0.98	0.96
$Fscore_{2.0}$	0.99	1.00	0.99	0.95

Table 3 represents the performance measures of the suggested method that are matched with the existing detection system using the C programming source code data samples. From table 2 and table 3, it is obviously witnessed the performance measure of suggested methodology is higher when matched with the other two techniques. It can also be inferred that the computational complexity of the proposed approach is also less compared to the other two approaches.

Table 3: Comparison of Performance Measures on C Programming Source Code Data Samples

Performance Measures	Proposed Clonal Selection Approach	Proposed Particle Swarm Optimization Approach	Proposed Iterative Genetic Algorithm based Approach	Incremental Genetic Algorithm based Detection System
Precision	0.99	0.97	0.96	0.96
Recall	0.01	0.03	0.04	0.04
$Fscore_{0.5}$	1.00	1.00	1.00	0.97
$Fscore_{1.0}$	0.99	1.00	0.99	0.98
$Fscore_{2.0}$	1.00	0.99	0.99	0.98

The Analysis of the proposed approach is also performed by means of the Fitness Functions or the similarity measures that employed in the proposed approach. The average Normalized Euclidean Distance (NED) and Normalized Cumulative Reciprocal Rank (NCRR) measures is used to analyze the proposed approach specifically the CSA with Pre-processing against Simple CSA without any kind of Pre-processing. The comparison is accomplished against the number of generation or iterations employed in the proposed Particle Swarm Optimization and Iterative Genetic Algorithm. Fig 3 and Fig 4 refers to the Average NED for Java and C source code data samples respectively.

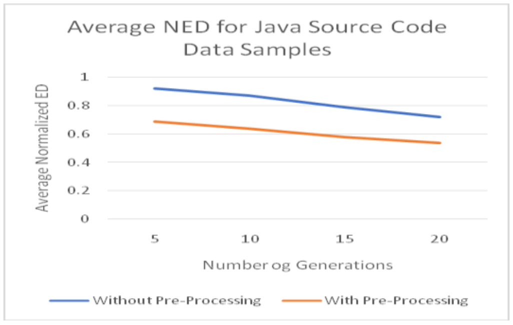

Figure 3: Average NED for Java Source Code Data Samples

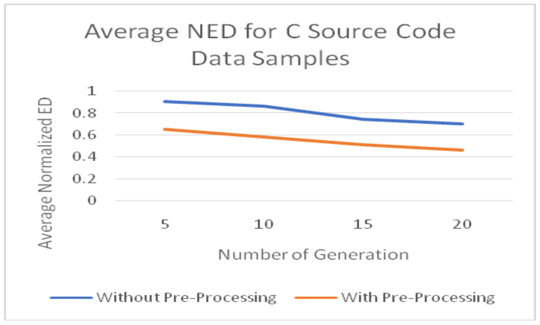

Figure 4: Average NED for C Source Code Data Samples

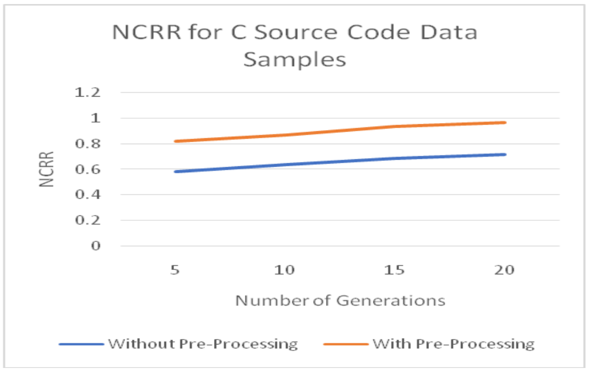

Figure 5: NCRR for Java Source Code Data Samples

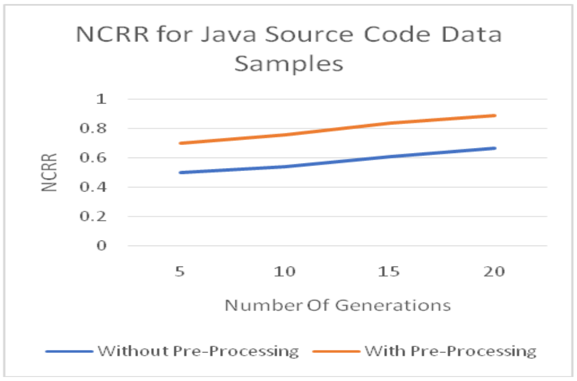

Figure 6: NCRR for C Source Code Data Samples

V. CONCLUSIONS

In this paper, an AIS based source code detection methodology is introduced to detect the plagiarized pair of source code. CSA is an exceptional group of AIS that employs the Clonal selection portion of the AIS as a foremost technique. Clonal selection approach regeneration is asexual where every child generated through a cell is the precise copy of its parent. For the purpose of reducing the sparse matrix obtained from preprocessing phase, sparse PCA algorithm is employed prior to detection phase. The higher dimensional sparse matrix A_{mXn} is reduced to lower dimensional sparse matrix employing sparse PCA algorithm using Iterative elimination approach. The performance analysis of the suggested approach showed that it has better precision and recall values when compared with existing Meta heuristic based Source code plagiarism detection algorithms.

REFERENCES

[1] W. Zhou, Y. Zhou, X. Jiang, and P. Ning, "Detecting repackaged smartphone applications in third-party android marketplaces," in Proc. 2nd ACM Conf. Data Appl. Security Privacy, 2012, pp. 317–326.

[2] S. Burrows, S. M. M. Tahaghoghi and J. Zobel, "Efficient plagiarism detection for large code repositories", Software Practice and Experience, vol.37, pp. 151-175, 2006.

[3] G. Cosma and M. Joy, "Towards a Definition of Source-Code Plagiarism," IEEE Transactions on Education, vol. 51, no. 2, pp. 195 - 200, 2008.

[4] Cosma, G., Joy, M., 2008. Towards a definition of source-code plagiarism. IEEE Trans. Edu. 51 (2), 195–200.

[5] Kustanto, C., Liem, I., 2009. Automatic source code plagiarism detection. In: 2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing. IEEE, pp. 481–486.

[6] Rabbani, F.S., Karnalim, O., 2017. Detecting source code plagiarism on .NET programming languages using low-level representation and adaptive local alignment. J. Inf. Org. Sci. 41 (1), 105–123.

[7]nLancaster, T., Culwin, F., 2004. A comparison of source code plagiarism detection engines. Comput. Sci. Edu. 14 (2), 101–112.

[8] V. Anjali, T. R. Swapna and B. Jayaraman, "Plagiarism Detection for Java Programs without Source Codes," in The International Conference on Information and Communication Technologies, Kochi, 2014.

[9] A. Cuomo, A. Santone and U. Vilano, "A novel approach based on formal methods for clone detection," in The 6th International Workshop on Software Clones, 2012.

[10] Stamatatos, E. (2011). Plagiarism detection using stop word n_grams. Journal of the American Society for Information Science and Technology, 62(12), 2512-2527.

[11] Stein, B., Lipka, N., &Prettenhofer, P. (2011). Intrinsic plagiarism analysis. Language Resources and Evaluation, 45(1), 63-82.

[12] Alzahrani, S., &Salim, N. (2010). Fuzzy semantic-based string similarity for extrinsic plagiarism detection. Braschlerand Harman.

[13] Gillium, J. (2015). Big data etbibliothèques: traitement etanalyseinformatiques des collections numériques

[14] Banjade, R., Maharjan, N., Gautam, D., &Rus, V. (2016).DTSim at SemEval-2016 Task 1: Semantic Similarity Model Including Multi-Level Alignment and Vector-Based Compositional Semantics. Proceedings of Sem Eval, 640-644.

[15] A. Flint, S. Clegg, and R. Macdonald. Exploring staff perceptions of student plagiarism. Journal of Further and Higher Education, 30:145–156, 2006.

[16] P. Keith-Spiegel, B. G. Tabachnick, B. E. Whitley, and J. Washburn. Why professors ignore cheating: Opinions of a national sample of psychology instructors. Ethics and Behavior, 8(3):215–227, 1998.

[17] W. B. Croft, and D. J. Harper. Using probabilistic models of document retrieval without relevance information. J. Documentation, 35(4): pp. 285-295, 1979

[18] Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. Machine Learning, 46(1-3):389–422, 2002.

[19] L. De Castro, F. J. Von Zuben, 'The CSA with engineering applications,'GECCO 2000, Workshop Proceedings, Workshop on Artificial Immune Systems and Their Applications, Las Vegas, USA, 2000, 36-37.

[20] L. N. De Castro, F. J. Von Zuben, 'Learning and Optimization Using the Clonal Selection Principle,'IEEE Transactions on Evolutionary Computation, Vol. 6, No. 3, June 2002, pp. 239-251.

[21]. Forrest, S. et al.: Using genetic algorithms to explore pattern recognition in the immune system. Evol. Compute. 1, 191–211 (1993).

[22]. Jennifer A. White, Simon M. Garrett: Improved Pattern Recognition with Artificial Clonal Selection? ICARIS 2003, LNCS 2787, Springer-Verlag Berlin Heidelberg 2003. 181-193(2003).

[23] Jinhyun Kim, HyukGeun Choi, Hansang Yun, Byung-Ro Moon, "Measuring Source Code Similarity by Finding Similar Sub graph with an Incremental Genetic Algorithm", In Proceeding of the Genetic and Evolutionary Computation Conference, pp. 925-932, ACM, 2016.

[24] M.Bhavani, Prof K.Thammi Reddy, Prof P.Suresh Varma, "An Iterative Genetic Algorithm Based Source Code Plagiarism Detection Approach Using NCRR Similarity Measure", Journal of Theoretical and Applied Information Technology, 2018.

[25] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (gpca). IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 12, pp. 1945-1959, 2005.

[26] Vanita Jain, Aarushi Jain, Achin Jain, Arun Kumar Dubey, "*Comparative Study between FA, ACO, and PSO Algorithms for Optimizing Quadratic Assignment Problem*", International Journal of Scientific Research in Computer Science and Engineering, Vol.6, Issue.2, pp.76-81, 2018.

[27] S.Arora, P. Shukla, N. Karankar, "*Community Structure Detection in Social Networking Data Using Text Mining Approach*", International Journal of Scientific Research in Computer Science and Engineering, Vol.5, Issue.4, pp.9-15, 2017

Author's Profile

M.Bhavani pursuing her Ph.D in the area of "Data mining" in the Dept. of CSE, JNTUK .She is having teaching experience of 15 years.

Dr. K. Thammi Reddy, currently working as the Head of the Department of CSE at Gandhi Institute of Technology (GITAM) University, Visakhapatnam. He is having vast experience in teaching, Research, Curriculum Design and consultancy. His research areas include Data warehousing and Mining, Distributed computing, etc.

Dr. P Suresh Varma Working as a Professor of Computer Science and Engineering, Faculty of Engineering and Technology, Adikavi Nannaya University. He has supervised the research work of a number of Ph.D and M.Phil scholars and published and lectured extensively on Communication Networks, Data mining, Cloud Computing, Big Data and Image Processing. In 2010 Government of Andhra Pradesh honored with Best Teacher Award in the occasion of Teacher Day. Domain of Research: Technology Trends,Communication,Software Engineering and Quality,3G/4G Network Evolutions, Big Data, Big data analytics for security, Cloud Computing, Communication Protocols, Computer Networks, Data Science, Databases.