ELC-PPW: Ensemble Learning and Classification (LC) by Positional Patterns Weights (PPW) of API calls as dynamic n-Grams for Malware Perception

G.Bala Krishna

CSE Dept., CVR College of Engineering, Hyderabad, INDIA
Email: govind.krishna83@gmail.com

V.Radha

IDRBT, Hyderabad, INDIA
Email: vradha@idrbt.ac.in

K. VenuGopal Rao

Dept. of CSE, GNITS
Hyderabad, INDIA
Email: kvgrao1234@gmail.com

***Abstract* - Most of the trending malware defense mechanisms are remain least effective against zero-day attacks. Machine learning is emerging as a significant choice for accurate identification of such malicious threats. Most of these ML techniques implement n-gram features of fixed size and sequentially patterned. We argue that the n-gram features of variable size, which are patterned by their positional frequency can escalate the detection accuracy. The volume and coherence of the training data is also the influential factor of the detection accuracy. In this regard, this research paper attempts to adopt an ensemble classification approach that considers n-gram of variable size for learning and classifying. Moreover, the n-grams are patterned by their position, instead sequence. In particular, the contribution of the research study is an Ensemble Learning and Classification strategy that uses cuckoo search as binary classifier. Simulation outcomes of these ML algorithms depict huge difference in malware program over the genuine program. The variations found in the classifier efficiency are assessed with respect to changes in the existence of malicious program.**

Keywords* - *API call, Machine learning, malicious software, Intrusion Detection system, DNA configuration, Cuckoo Search, n-gram,

I. INTRODUCTION

The intrusion of malicious software is witnessing strong growth in the recent few years, in particular, with increasing use of internet and new applications. The malicious software writers design the algorithms for multiple purposes including data destruction, intrusion into systems, steal confidential data, hack and compromise on integrity, credibility, and availability of target computer. In addition, malware is being deployed to break into several reputed and highly secure organizations across the globe. Based on the type of attack/entry point, the malicious program is given multiple names, some of which include- Virus, worms, Rootkit, spyware, key loggers, Trojan, DoS, Backdoor, Exploit and adware. With increasing number of devices and users, intruders are developing new series of attacks on a frequent basis. According to the AV-Test malware statistics [1] of year 2018, there are more than 140 million new malwares among 317 million malwares were introduced in 2014. There are around 700 million malwares are trending in 2017, among that nearly 120 million malwares are emerged alone in 2017. Past five year's statistics of malware are visualized in Figure 1, and Figure 2. Accordingly, security researchers worldwide are confronted with new challenges in detecting and neutralizing these new types of malicious programs.

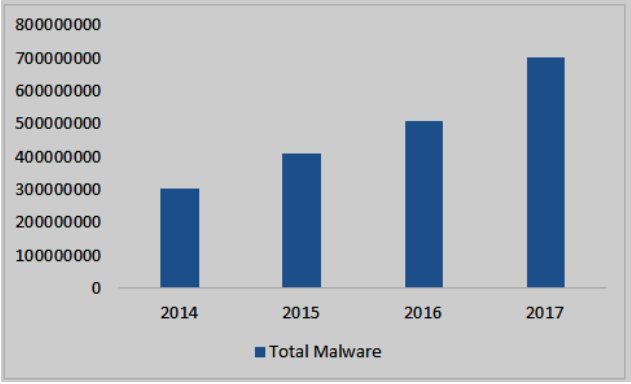

Figure 1: Past 5 year's malware statistics

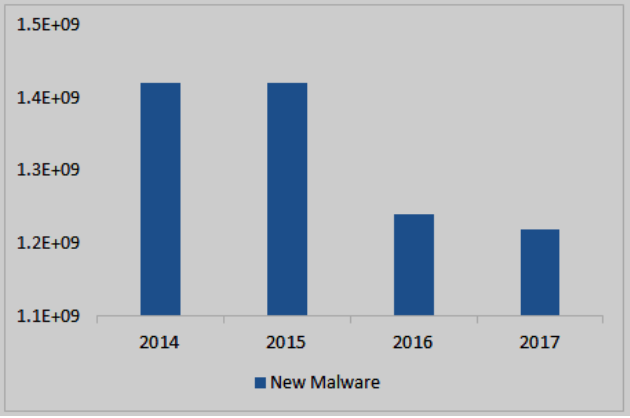

Figure 2: Past 5 years newly emerged malware statistics

Anti-virus programs currently available in the market utilize specific byte sequence models as the signature for detecting malware intrusion. Though the method works efficiently with already known malware sequences, it is largely ineffective to operate against the unknown attacks. In addition, several attackers are focusing on utilizing system zero-day vulnerabilities and the anti-virus vendors are unable to keep up an equivalent detection pace.

According to the Symantec report vendors took 204 days, 22 days and 53 days to neutralize the top three zero-day vulnerability attacks during 2014. As these zero-day attacks are the new type and are unknown to existing detection algorithms, new signatures are to be generated. Accordingly, the prevailing anti-virus detection programs are looking for different approached to handle such zero-day attacks. Further, malware programmers also rely on obfuscation techniques to evade detection mechanisms. Some of the popular obfuscation techniques implemented include encrypting data, compress data, Metamorphic (changes shapes) etc. [2]. Anomaly techniques are developed by security providers to handle such new attack types. However, these anomaly techniques pose a serious threat of high false positive rates (FPR), despite being effective in the detection of unknown malicious software.

Over the past few years, researchers are studying on the application of data mining techniques and machine learning techniques for malware detection. Advancements in machine learning techniques are believed to be a reliable solution for handling new and unknown types of intrusion attacks. These data mining models divide datasets into two categories- training and testing based on random sampling process. The random sampling ensures that test dataset is unknown to training dataset, enabling researchers to evaluate the functioning of the proposed model against new malicious software. Though this approach works with acceptable performance levels, in the real-time scenario, detection models must utilize available malware samples for training purposes so that the classification model can be applied for detecting emerging intrusion. The existing detection models typically mix both old malware and new malicious software in the training dataset and experimental databases and therefore face challenges in performance evaluation.

II. RELATED WORK

Machine learning unites outcomes of multiple classifiers so as to arrive at a single effective classifier. The composite models of classifiers are being studied by most researchers. Some of the recently proposed studies successfully depicted that machine learning process leads to better accuracy ratios and AUC curve of the method. Overall, model performance is improved and low FPR rates are recorded from the model. Accordingly, various researchers are exploring options for deploying different ML techniques-based tools for identifying malicious programs.

In [3], researchers developed five databases by identifying bytes n-grams, function and PE features along with specific feature extraction metrics. All the five databases were processed through five base inducers. Further, ensemble schemes like majority voting, Naïve Bayes are implemented to integrate these classifiers. Results of these works depict that of different ensemble schemes utilized for combining, troika model and stacking model showed superior performance, followed by weighted distribution, Bayesian combination. The others were inferior in performance as these models relied on single feature view, unlike other models which used combined feature views.

In [4], researchers implemented machine learning to identify malicious programs from the imbalanced gray list. The model applied API calls along with interpretable strings to develop 8 classifiers. It applied SVM machine rules and association programs over feature view combinations and training databases. Later, it used a simple voting method to integrate different classifiers. The method shows superior performance compared to the single classifier.

The study in [5] proposed a new model by segregating API calls and applying API n-grams of these categories for constructing the base classifier. Outcomes of these classifiers are integrated through BKS scheme. An accuracy ratio of 98.7% and False Positive Ratio of 9.1% were recorded.

Combination of features generated from API interface/ DLLs were applied to 12 common characteristics found in malicious programs [6]. SVMAR algorithm was engaged to develop classifier. The results had superior performance as compared to existing models. However, the method is limited to use in the experimental environment and could not be extended to real-time applications as it incorporated six tools for manual feature extraction.

Various permutations are identified to produce three opcode structures from all the structures included in the classifier construction [7]. Based on popular and veto vote mechanism, estimations generated by the classification models are integrated. In case of at least a single classifier identifying the malicious program, veto vote confirms the program coding as a malware regardless of other results. Accordingly, the technique had large false positives reported but the intrusion detection rate was quite large.

Other ML-based programmers relied on mining techniques such as in [8]. They mined for attributes from PE headers and application program interface. Several ML programs have been performed on the same dataset to construct different classification models. Both binary techniques and weighted machine learning techniques are proposed for choosing the best classification model subset. Later, these multiple models are integrated to detect malicious attacks. Simulation outcomes recorded 99.7% detection rate while the false positive ratio of 0.1%.

A similar method for extracting features is put forward in [9]. For partitioning of feature space, Random subspace

method is engaged. Every partitioned subspace functions to construct the classifier. A new algorithm selects classifiers and allocates weights to these classifiers. Later, based on weighted voting, chosen classifiers are integrated. The method also showed improved results over standard machine learning programs.

N-gram features along with application program interface (SVM) characteristics have been implemented in [10]. The results of classification models are integrated on the basis of D-S approach. Detection rates of 98.73% have been observed in the simulation phase.

The study in [11] relied on API calls and Dalvik calls for identification of malicious software. Both static and dynamic variants of both these models are engaged for the detection process. Android platform was preferred by the researchers. By writing multiple programs for constructing classification models, they aimed to improve performance of the proposed approach. From these multiple programs, the one with maximum precision and sensitivity was chosen on the basis of heuristic technique. Popular vote mechanism was implemented to combine results of multiple models. Experimental results depict detection rate of 91.47% and sensitivity ratio of 100%.

The study in [12] focused on Authorization characteristics and API-call characteristics in an Android platform. Multiple approaches including J48, decision stumps, and tree-based approaches are deployed for building classification models. CDF technique is utilized for combining the outcomes of these models. The model results depicted 98.9% detection ratio.

To speed up detection process, [13] relied on Bloom filter to produce malware signatures and lowered computation by lowering feature space. The model identified malware and assigned it to the respective family. Features have been abbreviated through CFGs. These CFGs should follow specific rules including- same blocks count, directed edges and instruction sequences in the block. Based on these set of rules, CFGs identify similarities in a family. Average similarity of 82.3% and 19.7% were observed in same and different families respectively.

A novel method aiming at an effective classification of malicious programs was proposed in [14]. It generates malware features by computing times of every library. To evaluate this approach, multiple malware programs from leading anti-virus vendors like Kaspersky, Norton were used. To precisely identify similarities and differences among malware families, API calls per DLL are gathered along with imported functions. Later, Euclidean metric and MLP are engaged for classification.

An increasing trend towards auto-scale up malware attacks is being observed in several applications. Hence, certain malicious program varieties tend to share same characteristics. Accordingly, it is important to categorize the attack program into similar or different classes for accurate detection. In [15], authors suggested reflecting the attack code in pictorial format to accurately identify and categorize malicious programs of different families. The researchers then disintegrated the picture to modify it into the graphical representation. The approach relied on the resemblance of changed histograms to observe the graph-to-graph differences and sameness [16]. Though acceptable detection rates are posed in experimental results, the efficiency of the method was quite low.

Deep observation of the proposed approaches in the current literature suggests that integrating several characteristics yield improved detection rates and efficiency as compared to the approaches involving single characteristic-views. Nevertheless, these research studies are assessed based on cross-validation. In addition, the manuscripts are assessed over databases that contain existing and new samples. Therefore, it can be misleading to consider the effectiveness of these studies in the real-time environment where complete new and unknown attacks are present. The details of approach and constraints of these contemporary models listed in table 1.

Accordingly, unlike previous models, our recent contribution called EBC-CS [17] depicted a novel classification strategy that relies on n-gram API-call series patterns of the dynamic size that outperformed the contemporary models observed in contemporary studies. However, the detection accuracy and misclassification rate left with considerable margins, which is due to dense volume of the training set, where the correlation of the API-call projections are considerably diversified from one record to other. Moreover, though the n-gram volume is dynamic, but only calls in the sequence are regarded as n-grams. With respect to this, the contribution of this manuscript is extending the EBC-CS, where multiple classifiers are used and n-grams are formed not by the sequence of calls, instead n-grams of varying volume are formed by the call positional patterns.

TABLE 1: LIST OF CONTEMPORARY MODELS IN RECENT LITERATURE, THEIR APPROACH AND CONSTRAINTS

Ye, Yanfang et al. [4]	Machine learning approach to identify malicious programs from the imbalanced gray list that applied on API calls along with interpretable strings to develop 8 classifiers.	Though, the method shows superior performance compared to the single classifier, the detection accuracy relied on the sequence of the features.
Shanqing et al. [5]	A new model that segregating API calls and applying API n-grams of these categories for constructing the base classifier.	Though, the results had superior performance as compared to existing models, the method is limited to use in the experimental environment and could not be extended to real-time applications as it incorporated six tools for manual feature extraction.
Landage et al. [7]	Various permutations are identified to produce three opcode structures from all the structures included	the technique had large false positives reported though the attack detection rate was quite large
Krawczyk et al [9]	Random subspace method is engaged to partition the feature space, such that each partitioned entails with a classifier	Selective classifiers only work, but uncertainty in selection of classifiers evinced, which is due to the method of voting strategy that adopted.
Zhang et al. [10]	N-gram features along with application program interface (SVM) characteristics have been implemented	Sequential pattern dependency observed, considerable false negative and false positive rate evinced against dense training set with diversified positions of the calls in given call sequences
Ozdemir et al [11]	Relied on API calls and Dalvik calls for identification of malicious software.	Limited to android platform, dependent of call sequence patterns, not significant to learn from large volume of training set with considerable diversity in call sequences.
Sheen et al. [12]	Focused on Authorization characteristics and API-call characteristics in an Android platform. Multiple approaches including J48, decision stumps, and tree-based approaches are deployed for building classification models.CDF technique is utilized for combining the outcomes of these models.	Though the detection rate is 98.9%, it limited to android environment only. Detection accuracy is not significant, if learning data volume is high.
Kang et al. [13]	Relied on Bloom filter to produce malware signatures and lowered computation by lowering feature space. The features are defined as Call Flow Graphs. Certain rules build about Call Flows depicted in Graphs	The detection accuracy is not significant since, average similarity of 82.3% and 19.7% were observed in same and different families respectively.
Gonzalez et.al [14]	Method aiming at an effective classification of malicious programs that generates malware features by computing times of every library	An increasing trend towards auto-scale up malware attacks is being observed in several applications, hence, certain malicious program varieties tend to share same characteristics. Accordingly, it is important to categorize the attack program into similar or different classes for accurate detection.
Han et al. [15]	Reflecting the attack code in pictorial format to accurately identify and categorize malicious programs of different families. Approach relied on the resemblance of changed histograms to observe the graph-to-graph differences and sameness [16].	Though acceptable detection rates are posed in experimental results, the efficiency of the method was quite low.
Krishna, et al. [17] [EBC-CS].	A novel classification strategy that relies on n-gram API-call series patterns of the dynamic size that outperformed the contemporary models observed in contemporary studies.	The detection accuracy and misclassification rate left with considerable margins, which is due to dense volume of the training set, where the correlation of the API-call projections are considerably diversified from one record to other. Though the n-gram volume is dynamic, but only calls in the sequence are regarded as n-grams.

III. PROCESSES AND RESOURCES

A. The Data Structure

The database DS with call sequence volume of $|DS|$ is utilized for educating the model for determining two parameters - *esms* and *esbs* . Every dissimilar 'two-calls' of the corresponding record is regarded as the feature while all 'two-calls' in series are regarded as the feature.

A call series set D is divided into $M = \{mcs_1, mcs_2, ..., mcs_{|MCS|}\}$ to depict malware and $B = \{bcs_1, bcs_2, ..., bcs_{|BCS|}\}$ to depict benign series. The characteristic sets csp_M of corresponding databases M and csp_B of corresponding database B include call series observed in databases M and B as characteristics. These characteristic sets are depicted as-

$csp_M = \{mp_1, mp_2, ...mp_{|csp_M|}\}$ // depicts the call series structure so that $mp_i = \{c_1, c_2, ..., c_{|mp_i|}\}$ observed in database M . Highest calls in the series of structure mp is m , where,

The notation m represents parent data. Based on set theory, $\{mcs_i \exists mcs_i \in M\}$ *highest count of call series structures of volume* k are $m-k+1$, where the notation m is determined as $\{mcs_i \exists mcs_i \in M\}$.

$csp_B = \{bp_1, bp_2, ...bp_{|csp_B|}\}$ // denotes the call series structures, such that the call series structure $bp_i = \{c_1, c_2, ..., c_{|bp_i|}\}$ observed in data of genuine call series set B .

B. Bi-Directional Combined Variance Prediction

For neutralizing the count of call series structures which are regarded as changing volume n-grams, the corresponding covariance between parameters depicting each call location of the series in the storage is considered as attack-vulnerable or genuine for each and every n-gram. These n-grams are the desired characteristics representing the covariance of data on attack-vulnerable and benign data points selected.

To predict the variance between different call series consisting data of the n-gram value of both malicious and genuine data points of the selected training database, the approach relied on ANOVA bi-directional combined variance prediction. The prediction technique is selected on the basis of experimental outcomes of the studies- [18] and [19]. The technique chooses the best characteristics related to both malware and benign wares for the selected training database.

Disparity data between two different vectors is denoted through the proposed approach as shown below-Eq-1:

$$bpve = \frac{\left(\langle v_1 \rangle - \langle v_2 \rangle\right)}{\sqrt{stdv(v_1) + stdv(v_2)}} .... \quad \text{(Eq) 1}$$

Where, $\langle v_1 \rangle, \langle v_2 \rangle$ denotes the mean values of vectors $v1, v2$ while $v1, v2$ depict call series represented as n-grams data associated to corresponding malware and benign ware in the training database.

- $stdv(v_1), stdv(v_2)$ Denote the MSD value of $v1, v2$ respectively.

The bpve is calculated by the proportion of mean variation of vectors $v1, v2$ to square root of sum of $stdv(v_1), stdv(v_2)$.

In addition, the probability ratio is calculated using the t-table [20], [21]. As this ratio is smaller than the threshold level, call series of volume n depicting corresponding vectors pertain to desired characteristics.

C. Cuckoo Search (CS)

In [22], researchers used natural component driven meta-heuristic methods for achieving optimal or desired features. These are regarded as the best programs for handling optimization tasks. The study assesses the fitness value of an intrusion API call series and benign call series through the implementation of CS. The approach is built on the basis of Cuckoo bird which puts its own eggs in approximately similar nests of different species.

The bird's logic of nesting involves the three basic conditions-

Its egg is similar to a required result and is placed randomly in a selected nest. The condition is that at an instance, only single egg is placed in a nest.

Those nests with better quality pass on to succeeding stages

Actual bird owning the nest will identify the Cuckoo species' egg with the probability of $\in$ [0, 1].

Upon detecting the intrusion egg, the owner discards the particular nest and passes on for building anew nest. Nevertheless, given the resource constraint, the total of new nests is limited to a predetermined constant.

However, all the steps discussed above need not be compulsory as the CS utilized in the suggested method is for detecting the fit of characteristics for a selected API call series data point. Accordingly, the study suggests building nests in the conventional method and implements a search function in a randomized pattern. Conventional CS restricts one egg in one situation but the study clones the task to multiple nests and drops an egg in all possible places. Further, fit of each egg is predicted for all nests.

D. The database

The datasets api-mds [23], CSDMC2010_API [24], and ICARIS09_API [25] are combined and used for the experimental study, which are of 23,819 malicious program samples, and 11181 benevolent samples (Mixture of windows-OS calls and different trusted algorithms. The complete quantum of entries in the last database is 35000.

IV. EVOLVING BINARY CLASSIFIER MODELS THROUGH CUCKOO-SEARCH FOR MALICIOUS SOFTWARE IDENTIFICATION

The chapter describes the complete procedure of the suggested method involving selecting desired or characteristics, constructing classification model, training procedure and categorizing the record to benign or intrusion.

A. Selecting Optimal Characteristics

The dataset D is divided into subsets $M, and\ B$ denoting malware and benign API call series respectively. All the data points in these subsets are of different volumes and denote the call series patterns. This manuscript projects the attributes that are the patterns of the calls by their position in API call series. The volume of these positional patterns is two and above. The potential volume of positional patterns is the $2^m - 1$, where m is the maximum size of the call series in given set. The overall positional features depicted from the call series of a maximum length m are $2^n - 1$ and they can be projected as:

$$\left\{\begin{array}{l}\{(p_1),(p_2),...,(p_m)\},\\ \left\{\begin{array}{l}(p_1,p_2),(p_1,p_3),...,(p_1,p_m),\\ (p_2,p_3),(p_2,p_4),...,(p_2,p_m),...,\\ (p_{m-1},p_m)\end{array}\right\},\\ \left\{\begin{array}{l}(p_1,p_2,p_3),(p_1,p_2,p_4),...,(p_1,p_2,p_m),\\ (p_2,p_3,p_4),(p_2,p_3,p_5),...,(p_2,p_3,p_m),\\ ...,\\ (p_{m-2},p_{m-1},p_m)\end{array}\right\},\\ ....,\\ \{(p_1,p_2,p_3,....,p_m)\}\end{array}\right\}$$

Further, for every feature, the values depicted in the records of the given set are projected as a vector as $f(p_1) = \{(v_1),(v_2).....(v_s)\}$, where, the vector $f(p_1)$ represents values depicted in the specific record set at position p_1. Similarly, the vector:

$$f(p_i, p_j, p_k) = \left\{(v_i, v_j, v_k)_1, (v_i, v_j, v_k)_2, ..., (v_i, v_j, v_k)_s\right\}$$

is depicting the values projected for call positions i, j, k in the records of the considered set.

This process is applied for both genuine and malevolent record sets considered for learning process. For choosing the optimal attributes of both types of record sets, the study evaluates the correlation among the values projected for every feature (positional pattern)representing malicious call series group M with the values projected for the same feature (positional pattern) from benign call series subset B. In case of high correlation values, the corresponding locations are regarded as features that are associated with correlation evaluation procedure and are regarded as undesired features, removed from M and B subsets. The sequence is iterated till all the characteristics will high correlation is removed from either subset. The remaining attributes of the subset M are considered as the required optimal characteristics with different volumes extending from 2 to maximum volume of the corresponding call series with respect to the subset M. On a similar basis, the remaining characteristic in the subset B are also considered as optimal attributes of n-grams with dynamic size.

The values projected for all of the optimal characteristics (positional patterns) of call sequence sets $M, and\ B$ will be denoted further as $csp_M, and\ csp_B$ in corresponding sequence. In addition, the process explores the values projected for each positional pattern of both malicious and benevolent sequence sets will be assigned a unique index, which is presented as below:

step 1. $for\{pp_i^M, pp_i^B \exists 1 \le i \le P\}$ Begin // where the notation pp_i^M indicates the positional pattern of index i of malware call sequence group, pp_i^B indicates the positional pattern of the index i of benevolent class sequence set.

a. $pp_i^{M|B} = pp_i^M \cup pp_i^B$ // pools all unique entries found for feature pp_i in both attack and genuine sequence sets.

b. Then replace all of the entries in pp_i^M with their respective index in $pp_i^{M|B}$. In addition, replace all of the entries in pp_i^B with their corresponding indexes in $pp_i^{M|B}$

step 2. End // of step 1

The pseudo code representation of the optimal attribute choosing through Bi-directional Combined Variance that assesses the correlation between values projected for a positional pattern in attack and normal call sequence group scan be found in appendix-A

B. Partitioning the Training Set into Multiple Groups

The entries depicted for divergent optimal positional patterns for each call series of the respective malevolent and genuine call series groups will be projected as a record in the order of positional patterns size.

These records of malware call sequence set, which are of the values projected for optimal positional patterns will consider as ppr_M , similarly the records of the genuine call series set, which are of the values projected for optimal positional patterns will consider as ppr_B.

Cluster the records depicted as set ppr_M in to dynamic number of groups, which is done by using X-means [26] model that depicts the volume of clusters dynamically. Similarly, cluster the records depicted in ppr_B in to dynamic number of groups using X-Means algorithm.

C. Ensemble Classification Model for Malicious Software Identification

The entire ensemble classifier operating procedure involves nest construction for each cluster of the optimal positional structures of the respective malevolent and normal groups. Further, performs hierarchical search of the ensemble set of nests related to both the malware and genuine attributes, which is to present the positional structures of the particular entry that is with respect to evaluating the fit to malware attack and to benign ware. Description of the classification model function is provided in the paragraphs below.

1) Learning Stage

This paragraph details the training part of the ML classifier procedure based on CS approach discussed earlier. The training stage constructs the nests in hierarchical order of the positional patterns size from max to min, so that every nest refers to a unique positional pattern. Such hierarchies will be built for all clusters of the malicious and genuine entries of the optimal positional patterns depicted. Further these nests of the corresponding hierarchies will be filled with the entries depicted for the respective positional pattern of the particular nest in corresponding cluster. The provided nests of a corresponding cluster are organized in hierarchical structure, with the positional structures with highest volume m will be in the top most stage of the hierarchy. In the same manner, the positional structures of volume $m-1$ will be presented at next level so that the positional structures of volume $m-i$ will be included at $(i+1)^{th}$ stage. The positional pattern of volume1 (the least volume) will be staged at the bottom most plane in the hierarchal structure. The nests represented by the equal volume of positional patterns will at same stage, which are in the order of left to right.

Based on the aforementioned analysis of the functioning of nest concept, for reflecting the positional patterns of the each cluster related to malicious records, and each cluster related to benevolent records, the presented model constructs ensemble nest hierarchies:

$$MH = \{mh_1, mh_2, .., mh_{mc}\}, BH = \{bh_1, bh_2, .., bh_{bc}\}$$

corresponding to the malicious, and benevolent positional patterns, where set MH is having mc (maximum cluster count of the malicious positional patterns) number of nest hierarchies, and set BH is having bc (maximum number of clusters formed from benevolent positional patterns) number of hierarchies.

After constructing the ensemble nest hierarchies with suitable entries of the positional orders dropped as Cuckoo eggs in corresponding nests and a simultaneous CS is commenced to evaluate fit of considered data point for malware-vulnerability and genuine ware, which are discussed in the following paragraphs.

2) Classifying Stage

The fit of corresponding data point predicts on the count of feasible nests identified in corresponding machine learning hierarchies- MH, BH . Regarding this, for all nests, an egg of corresponding nest is similar to that of call series orders in the respective set, then fit of the respective data point associated to hierarchy will be increased by 1 count.

This procedure provides fit associated to malware and genuine ware of the corresponding data point. In addition, the proportion of this fit for any data point will compute the mean of fit associated to nest count in respective hierarchy. Further, the root MSD of fit relates to either of the hierarchies to be computed. Then, both these parameters are utilized to validate if the considered data point is intrusion prone of not.

The evaluation method for fitness calculation is provided in Appendix-B.

3) Identifying the Entry state

The fitness values related to ensemble hierarchies MF, BF and respective deviations obtained with respect to malicious and benevolent ensemble hierarchies MH, BH for considered input entry R_{must} utilize to classify the entry R is prone to attack or genuine. The classification label must determine on the basis of conditional flow that presented in Appendix-C.

V. SIMULATION STUDY OF THE SUGGESTED APPROACH

Simulation of the suggested method is executed through test database discussed in section D. To identify the extent of performance of the proposed method, which included the

positional patterns as features that are optimized by Bi-directional Combined Variance, which are further utilized in the learning of the ensemble classification model constructed on the principles of EBC-CS search, the simulation study outcomes extracted and compared to the other benchmark models called EBC-CS for malware perception [17], and API-Call series (ACSA) through DNA sequence arrangement method [23].

The classifier functioning procedure on said dataset using the proposed and other contemporary models selected were done in 10 times. Further, the performance evaluation of the suggested study and other methods suggested by various researchers observed through classification evaluation parameters [27] such as PPV (Positive predictive value, or precision), NPV (Negative Predictive Value), TPR (True Positive Rate, or Sensitivity), TNR (True Negative Rate, or Specificity), FPR (False Positive Rate or fallout), FNR (False Negative Rate, or Missing Rate), Classification Accuracy, and Misclassification Rate. The outcomes gathered from the simulation stage executed on proposed model ECL-PPW, and the contemporary models EBC-CS, and ACSA were depicted in Table I.

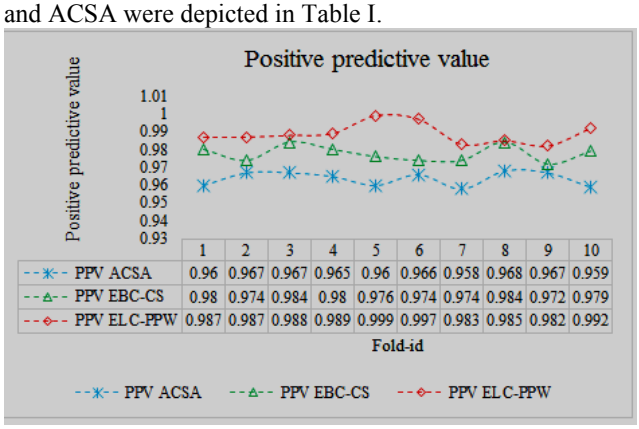

Figure 3: Positive predictive values of EBC-CS vs. ACSA

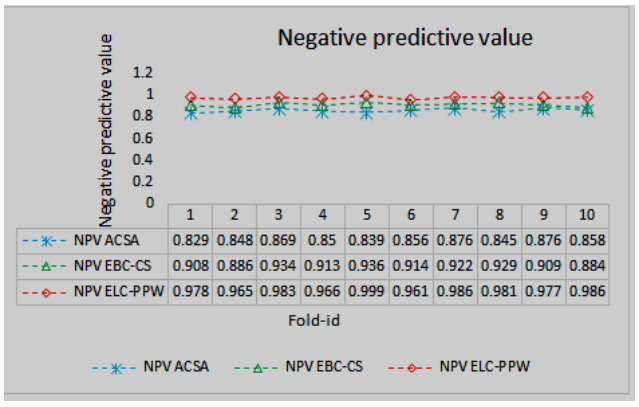

Figure 4: Negative predictive values of EBC-CS vs. ACSA

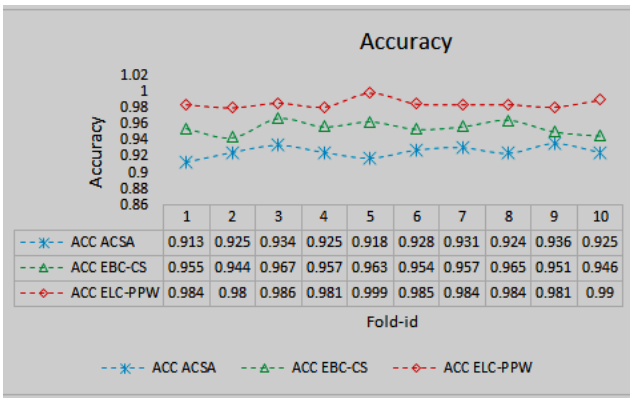

Figure 5: The classifier precision ratios of EBC-CS vs. ACSA

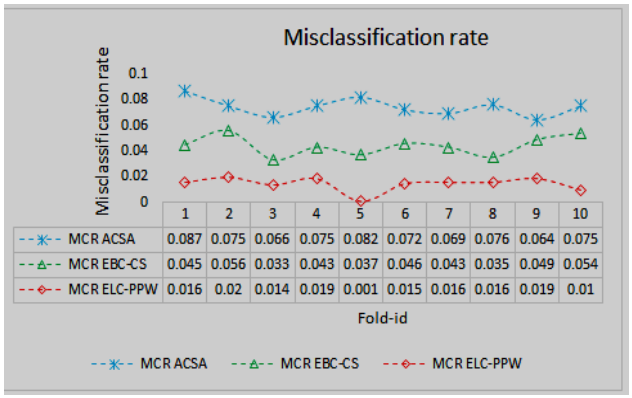

Figure 6: The misclassification rate of EBC-CS vs. ACSA

Estimation precision rate of ELC-PPW, EBC-CS, and ACSA identified from the simulation study depicted in Figure 5 ensure that the accuracy rates for ELC-PPW is relatively steady and much larger at over 98% that compared to EBC-CS with classification accuracy $\geq 0.94 \&\& < 0.97$ and ACSA with classification accuracy $\geq 0.91 \&\& < 0.93$.

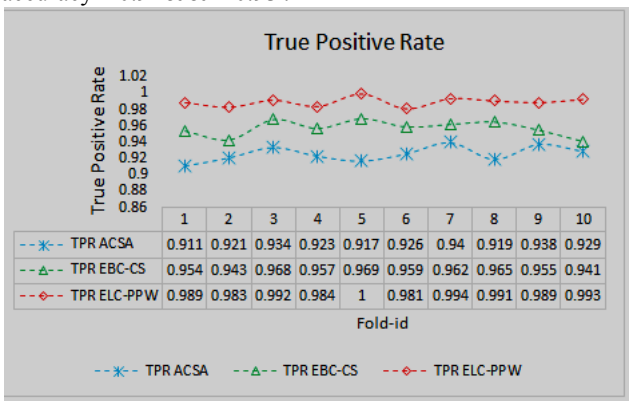

Figure 7: The sensitivity (malware prediction rate) of EBC-CS vs. ACSA

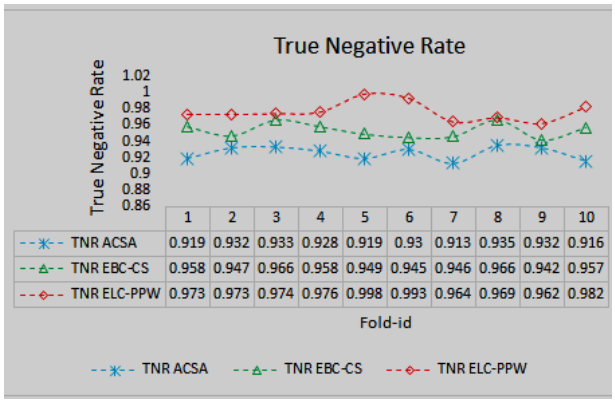

Figure 8: The specificity (salubrious state prediction rate) of EBC-CS vs. ACSA

The above two charts present the efficiency edge of the ELC-PPW over EBC-CS, and ACSA towards sensitivity and specificity those refers the significance of malware prediction and significance of genuine state estimation respectively. The suggested approach clearly outperformed the other two model EBC-CS, and ACSA in this regard, which is evincing malicious estimation rate of more than 0.98, and genuine call sequence prediction rate greater than 0.97. Whereas, the other two contemporary models EBC-CS, and ACSA evincing the malware prediction rate "$\geq 0.94 \&\& < 0.97$", and "$> 0.91 \&\& \leq 0.94$" respectively. Similarly, the benevolent call sequence prediction rate observed for EBC-CS, and ACSA are $> 0.94 \&\& < 0.97$ and $> 0.91 \&\& \leq 0.93$ respectively.

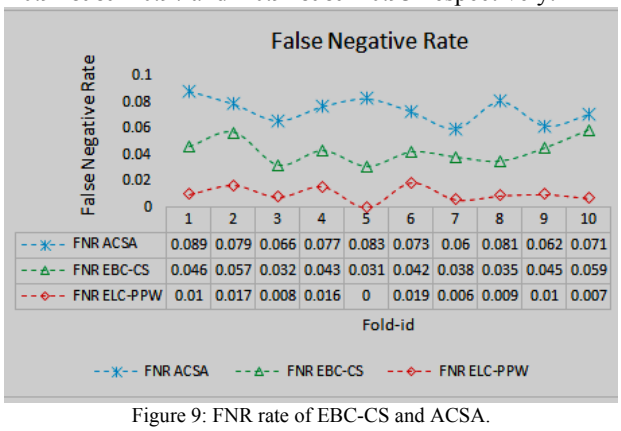

Figure 9: FNR rate of EBC-CS and ACSA.

The following Figures 9, 10depict the FNR and FPR, which depicts the estimation failure rate malicious call sequence and benevolent call sequence respectively observed for ELC-PPW, EBC-CS and ACSA. From the depicted results of fall-out observed for ELC-PPW is considerably low that compared to the fall-out observed for EBC-CS, and ACSA.

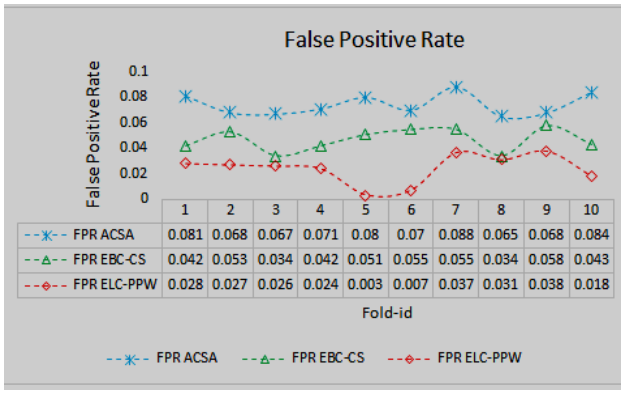

Figure 10: The fall-out of ELC-PPW, EBC-CS, and ACSA observed from 10-fold classification.

VI. CONCLUSION

The research work attempts to propose a method of selecting n-grams of varying size, which are the positional structures of the API calls as optimal features, and performing ensemble learning and classification for malware estimation in API-call series. The objective of the depicted model is maximal accuracy and lowest false alarms. So as to select the optimal features, unlike the traditional approaches, those selects n-gram features that are sequential patterns with static volume, the proposed model projecting the positional patterns (patterning the calls based on their position in call sequence) and selecting optimal features among the positional patterns projected, In order to choose the most desired attributes, the study relies on the statistical model termed "Bi-directional Combined Variance Prediction", and further build an ensemble classifier that uses the Cuckoo-search as binary classifier. The significance of the suggested ensemble classification strategy is that trains the multiple classifiers for each set of highly coherent positional patterns as optimal features. This practice evinced the high accuracy in classification rate and less fall-out. The simulation results convey that the suggested approach is both scalable and strong and has higher efficiency as compared to other existing methods suggested in the field including EBC-CS, and ACSA against diversified and large training set of API-call series. The next stage researchers and scholars can make efforts to enable the evolutionary strategies in regard to choose best attributes.

REFERENCES

[1] https://www.av-test.org/en/statistics/malware/ (accessed on 2/1/2018)
[2] Tsyganok, Ksenia, et al. "Classification of polymorphic and metamorphic malware samples based on their behavior." Proceedings of the Fifth International Conference on Security of Information and Networks.ACM, 2012.
[3] Menahem, Eitan, et al. "Improving malware detection by applying multi-inducer ensemble." Computational Statistics & Data Analysis 53.4 (2009): 1483-1494.

[4] Ye, Yanfang, et al. "Intelligent file scoring system for malware detection from the gray list."Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.ACM, 2009.
[5] Shanqing, Guo, et al. "A Malware Detection Algorithm Based on Multi-view Fusion." Proc. of ICONIP.Vol. 10.
[6] Lu, Yi-Bin, et al. "Using multi-feature and classifier ensembles to improve malware detection." Journal of CCIT 39.2 (2010): 57-72.
[7] Landage, Jyoti, and M. P. Wankhade. "Malware detection with different voting schemes." Compusoft 3.1 (2014): 450.
[8] Sheen, Shina, R. Anitha, and P. Sirisha. "Malware detection by pruning of parallel ensembles using harmony search." Pattern Recognition Letters 34.14 (2013): 1679-1686.
[9] Krawczyk, Bartosz, and MichałWoźniak. "Evolutionary cost-sensitive ensemble for malware detection." International Joint Conference SOCO'14-CISIS'14-ICEUTE'14. Springer, Cham, 2014.
[10] Zhang, Bo-yun, et al. "Research on virus detection technique based on ensemble neural network and SVM."Neurocomputing 137 (2014): 24-33.
[11] Ozdemir, Mehmet, and Ibrahim Sogukpinar. "An android malware detection architecture based on ensemble learning." Transactions on Machine Learning and Artificial Intelligence 2.3 (2014): 90-106.
[12] Sheen, Shina, R. Anitha, and V. Natarajan. "Android based malware detection using a multifeature collaborative decision fusion approach." Neurocomputing 151 (2015): 905-912.
[13] Kang, Boojoong, et al. "Fast malware family detection method using control flow graphs." Proceedings of the 2011 ACM Symposium on Research in Applied Computation.ACM, 2011.
[14] Gonzalez, Lilia E., and Roberto A. Vazquez. "Malware classification using Euclidean distance and artificial neural networks." Artificial Intelligence (MICAI), 2013 12th Mexican International Conference on. IEEE, 2013.
[15] Han, KyoungSoo, et al. "Malware analysis using visualized images and entropy graphs." International Journal of Information Security 14.1 (2015): 1-14.
[16] Strelkov, V. V. "A new similarity measure for histogram comparison and its application in time series analysis." Pattern Recognition Letters 29.13 (2008): 1768-1774.
[17] Krishna, G. Bala, V. Radha, and K. Venugopala Rao., " Evolutionary Binary Classification using Cuckoo Search for Malware Perception in API Call Sequences," 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), 2017, ISBN:978-1-5090-6620-9.
[18] Budak H, Taşabat SE. A MODIFIED T-SCORE FOR FEATURE SELECTION.
[19] Kummer O, Savoy J, Argand RE. Feature selection in sentiment analysis.
[20] Sahoo PR, Theorems TM. Functional Equations. World Scientific; 1998.
[21] http://www.sjsu.edu/faculty/gerstman/StatPrimer/t-table.pdf, 2017
[22] Yang XS, Deb S. Cuckoo search via Lévy flights. InNature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on 2009 Dec 9 (pp. 210-214). IEEE.
[23] Ki, Youngjoon, Eunjin Kim, and Huy Kang Kim. "A novel approach to detect malware based on API call sequence analysis." International Journal of Distributed Sensor Networks 11.6 (2015): 659101.
[24] http://csmining.org/index.php/malicious-software-datasets-.html
[25] http://nexginrc.org/Datasets/Default.aspx
[26] Pelleg, Dan, and Andrew W. Moore. "X-means: Extending K-means with Efficient Estimation of the Number of Clusters." ICML. Vol. 1. 2000.
[27] Powers DM. Evaluation: from precision, recall and F-measure to ROC, unforcedness, nakedness and correlation.

Appendix-A

The pseudo code representation of the optimal attribute choosing through Bi-directional Combined Variance that assesses the correlation between values projected for a positional pattern in attack and normal call sequence groups.

Loop 1: $for\{pp_i \exists 1 \le i \le P\}$ // The index i refers the positional pattern that is regarded as an attribute, and the symbol P is the highest count of positional patterns projected.

$\left\langle pp_i^M \right\rangle = \frac{\sum_{j=1}^{|pp_i^M|} \{e_j \exists e_j \in pp_i^M\}}{|pp_i^M|}$ the average $\left\langle pp_i^M \right\rangle$ of the entries observed for positional pattern pp_i^M of malicious sequence set M

$\left\langle pp_i^B \right\rangle = \frac{\sum_{j=1}^{|pp_i^B|} \{e_j \exists e_j \in pp_i^B\}}{|pp_i^B|}$ the average $\left\langle pp_i^B \right\rangle$ of the entries observed for positional pattern pp_i^B of benevolent sequence set M

$$rmsd(pp_i^M \square \quad pp_i^B) = \sqrt{\frac{\sum_{j=1}^{|pp_i^M|} \left(e_j - \left\langle pp_i^M \right\rangle\right)}{|pp_i^M| - 1} + \frac{\sum_{j=1}^{|pp_i^B|} e_j - \left\langle pp_i^B \right\rangle}{|pp_i^B| - 1}}$$

$t(pp_i^M \square \quad pp_i^B) = \frac{\left(\left\langle pp_i^M \right\rangle - \left\langle pp_i^B \right\rangle\right)}{rmsd(pp_i^M \square \quad pp^B)}$ // predicting the Bi-directional pool-variance value of the i^{th} positional pattern of both types of sequence groups.

$if\left(p\left(t(pp_i^M \square \quad pp_i^B)\right) < p\tau\right)$

// In the context of degree of probability $p\left(t(pp_i^M \square \quad pp_i^B)\right)$ being detectedfor $t(pp_i^M \square \quad pp_i^B)$ is smaller than the predefined threshold (typically 0.01, 0.05 or 0.1) provided

a. $opp_M \leftarrow pp_i^M$ // then the positional pattern pp_i^M of the malevolent call series group observed to be optimal and saved in superset opp_M .

b. $opp_B \leftarrow pp_i^B$ // then the positional pattern pp_i^B of the benevolent call sequence set are found to be optimal and saved in superset opp_B .

End

End

Appendix-B

The evaluation method for fitness calculation is provided in following steps
Let R be the considered call series data point to be categorized as malware or benign
Let eR be the set of entries representing all possible positional structures estimated from the corresponding call series entry R .

Loop 1: $\overset{mc}{\underset{i=1}{\forall}}\{mh_i \exists mh_i \in MH\}$ Begin// For each hierarchy of the positional patterns of malicious set

$mf_i = \sum_{h=1}^{|mh_i|}\sum_{j=1}^{|mh_i(h)|}\sum_{k=1}^{|eR|}\{1 \exists e_k \in nst_{\{h,j\}} \wedge nst_{\{h,j\}} \in mh_i \wedge pp(nst_{\{h,j\}}) \cong pp(e_k)\}$ //add 1 to fitness mf_i of the given record R related to malicious scope, if egg (call sequence pattern) e_i is compatible to place in nest $nst_{\{h,j\}}$ in hierarchy mh_i and positional structure $pp(nst_{\{h,j\}})$ of the nest $nst_{\{h,j\}}$ is same as the positional structure $pp(e_k)$ of the record e_k .

$mh_i nc = \sum_{h=1}^{|mh_i|} |\, mh_i(h)\,|$ // number of nests in mh_i

$\langle mf_i \rangle = \dfrac{mf_i}{mh_i nc}$ // Finding the fitness ratio $\langle mf_i \rangle$ of the given record in related to malicious scope

$MF \leftarrow \langle mf_i \rangle$ // move the fitness rate $\langle mf_i \rangle$ to the set MF

$d(mf_i) = \dfrac{\sum_{i=1}^{mf_i}\left\{\sqrt{\left(1-\langle mf_i \rangle\right)^2}\right\} + \langle mf_i \rangle * (mh_i nc - mf_i)}{mh_i nc}$ // The sum of absolute variance of the fitness ratio depicted, and max fitness with respect to each nest, (which is 1) for count of nests compatible to the eggs exist in eR, and the fitness rate multiplies by the volume of incompatible nests that is the discrepancy between the sum of count of all nests and sum of count of all suitable nests that signified as $mh_i nc - mf_i$

End

Loop 2: $\overset{bc}{\underset{i=1}{\forall}}\{bh_i \exists bh_i \in BH\}$ Begin //For each hierarchy of the benevolent positional patterns

$bf_i = \sum_{h=1}^{|bh_i|}\sum_{j=1}^{|bh_i(h)|}\sum_{k=1}^{|eR|}\{1 \exists e_k \in nst_{\{h,j\}} \wedge nst_{\{h,j\}} \in bh_i \wedge pp(nst_{\{h,j\}}) \cong pp(e_k)\}$ //append 1 to fitness bf_i of the corresponding entry R associated to malware scope, in case egg (call series structure) e_i is suitable to be put in nest $nst_{\{h,j\}}$ in hierarchy bh_i and positional pattern $pp(nst_{\{h,j\}})$ of the nest $nst_{\{h,j\}}$ is equal to positional pattern $pp(e_k)$ of the entry e_k .

$bh_i nc = \sum_{h=1}^{|bh_i|} |\, bh_i(h)\,|$ // count of nests in bh_i

$\langle bf_i \rangle = \dfrac{bf_i}{bh_i nc}$ // Calculating the fitness rate $\langle bf_i \rangle$ of the corresponding entry with respect to genuine software

$BF \leftarrow \langle bf_i \rangle$ // move the fitness ratio $\langle bf_i \rangle$ to the set BF

$d(bf_i) = \dfrac{\sum_{i=1}^{bf_i}\left\{\sqrt{\left(1-\langle bf_i \rangle\right)^2}\right\} + \langle bf_i \rangle * (bh_i nc - bf_i)}{bh_i nc}$ // The total of absolute value of difference of the fitness rate calculated, and maximum value with respect to every nest (it is considered as 1) for all those nests suitable to the eggs available in eR, and the fitness rate increases by the volume of unsuitable nests, that is the discrepancy between count of all nests and count of suitable ones, depicted as $bh_i nc - bf_i$

End

Appendix-C

The conditional flow that used to determine the class label:

if $(\max(MF) \cong \max(BF))$ Begin
  //the maximum fitness ratio $\max(MF)$ observed from ensemble hierarchies of the positional patterns related to malicious set is approximately equal to the maximum fitness ratio $\max(BF)$ observed from ensemble hierarchies of the positional patterns related to benevolent set.
  if $(d(\max(MF) < d(\max(BF))$ Begin
    Classify the considered call series R as malicious
  End
  Else *if* $(d(\max(MF) > d(\max(BF))$ Begin
    Label the call series R as genuine
  End
  Else //of condition in step 5
  Entry state is unclear// due to the fact that highest fitness ratios and corresponding deviations extracted for either of the hierarchies is equal
End
Else Begin // of condition in step 1
  if $(\max(MF) > \max(BF))$ Begin
    Label the call sequence R as malicious
  End
  Else *if* $(\max(MF) < \max(BF))$ Begin
    Label the call sequence R as benevolent
  End
  Else Begin//of condition in step 15
    Call sequence state is ambiguous// since the root mean square distance and fitness ratios attained for both hierarchies are not meeting the prescribed situations
  End
End