Multi-objective optimization of base classifiers in StackingC by NSGA-II for intrusion detection

Michael Milliken*, Yaxin Bi†, Leo Galway† and Glenn Hawe†
School of Computing and Mathematics, Ulster University, Belfast, United Kingdom
*Email: milliken-m@email.ulster.ac.uk
†Email: {y.bi, l.galway, gi.hawe}@ulster.ac.uk

Abstract—**Multiple Classifier Systems are often found to be useful for improving individual results by combining a set of classifier decisions where a single base level classifier may not achieve the same level of results. However not every set of base classifiers improve results, therefore a selection of a set of classifiers is required. The process of selecting base level classifiers for a multiple classifier system may be performed by the use of a Genetic Algorithm. The aim of this work is the selection of optimal sets of base level classifies using an evolutionary computation approach. In addition, a comparative analysis is made of the performance of the generated ensembles against the individual base level classifiers.**

I. Introduction

An ongoing challenge, for security experts and businesses, in the defense and stability of modern networks are intrusions that either illicitly retrieve data/information, or prevent legitimate access [1]. Given wide varieties of existing intrusions, one key difficulty is providing widespread detection. Intrusion detection depends largely on analysis of attack vectors and purposes. Attack methods and perceived purposes may be useful to determine if they are different enough from normal traffic to accurately state them to be attacks. Accuracy of intrusions depends upon the algorithm employed, when a single algorithm is not adequate, multiple algorithms may be required, determination of how many and which algorithms to employ requires further consideration.

Systems put in place to detect intrusions are known as Network Intrusion Detection Systems (NIDS). An NIDS may function with respect to historical or real time network traffic [2]. In either case, they are utilized to provide some method of detection through analysing all or a specified set of features [2], [3]. A popular approach to NIDS over the last decade has been the use of multi-layered or tiered approaches [4], [5], [6], which incorporate multiple methods within a hierarchy.

The focus of this paper concerns the use of a Genetic Algorithm (GA) to determine an optimal ensemble of base level classifiers (BLC) for the detection of intrusions. The hypothesis is the possible evolution of BLCs for an ensemble to perform better, to some degree, than individual BLCs. To explore this, we measured performances of a set of supervised Machine Learning (ML) algorithms, previously used with a GA [7], on two datasets used for intrusion detection research, notably the NSL-KDD [8] and ISCX2012 [9] datasets.

The developed work involves the evolution, by GA, of sets of ML algorithms, provided by third party sources, with multiple additions made, with their explicit usage in the domain of intrusion detection using data from two different datasets. This work differentiates itself from existing works by the specific usage of listed ML algorithms in the specific domain of intrusion detection with some more recent and large subsets.

The remainder of this paper is outlined as follows: Section II describes some background on GA, ML algorithms and existing NIDS. Following this, Section III, describes the methodology employed for the experiments discussed herein, followed by a presentation of the results obtained in Section IV. Finally, in Section V a brief discussion of the results is given along with a subsequent conclusion in Section VI.

II. Background

As ML research and development continues more potential algorithms are produced. For any given situation one algorithm may perform as well as another algorithm, in this case the need for both algorithms may become less likely, at least for a certain series of instances [7]. Each algorithm may require a certain amount of time for training and classification, the overall amount of time may not simply be linear and predictable. Therefore, implementation of all algorithms may not be desirable or adequate for the problem of accurate classification.

In general, the set of optimal algorithms for a given problem is known to not be all possible algorithms [10]. Thus, selection of an optimal set requires a further optimization process, such as a GA which follows the theory of evolution, producing optimal sets of classifiers with objective evaluation criteria.

NIDS can be evaluated using various different measures. For example, F-Measure (FM), Detection Rate, totals of False Positive (FP) or False Negative (FN), or rates of FP or FN classification instances may be employed. Within this paper, FP is taken to be the erroneous classification of an instance as an intrusion. Conversely, a FN is taken to be the erroneous classification of an instance as normal. With NIDS a balance must be found between FPs and FNs, as these two objectives generally compete. One way to achieve this balance is identify the Pareto-optimal trade-off between FP and FN using multi-objective optimization algorithms. A noted limiting factor of NIDS is the reduction of FPs [11], [12].

A. Ensemble Algorithms

Outputs from multiple BLCs may be combined into a single output, this approach may be referred to as an *Ensemble* or *Multiple Classifier System (MCS)*.

Popular conventional Ensemble methods include Bagging, Boosting and Stacking.

Bagging uses instances of a BLC and a replicated dataset with differences introducing variations, improving average classification result [13].

Boosting uses BLCs sequentially over one dataset. Improvements are induced at each subsequent BLC, weighting changes of misclassified instances, improving overall classification [14].

Stacking uses BLCs in parallel over one dataset [15]. Each BLC predicts class probabilities as input for a regression model per class. Each regression model classifies each instance [16]. Using class probabilities rather than predictions improve overall performance. A Stacking variation, StackingC [17], improved efficiency by using only the probability of a specific class for each linear model for that class, rather than considering the other class probabilities as well [17].

B. Ensemble-based NIDS

Within research literature, some NIDS have utilized Ensemble approaches, improving upon performance of a single BLC.

Octopus-IIDS [5], described as an Intelligent IDS, implements a Kohonen network to split data into attack classes, then SVMs to reclassify instances as *Normal* or *Attack*.

Hidden Markov Models with Payl (HMMPayl) [18] performs analysis on payloads, as well as using Hidden Markov Models (HMMs) initially, performing multiple initial classifications, then forming a final classification from the initial classification results.

Work described in [19] forms clusters with K-Means Clustering (KMC), after which Nave Bayes (NB) corrects data previously misclassified by the clustering. This approach increases classification results as well as efficiency by grouping the data.

C. Genetic Algorithms

A GA is an evolutionary algorithm that mimics the process of natural selection to evolve a population of candidate solutions to an optimization problem [20], [21].

The evolutionary aspect of the algorithm is related to modelling evolutionary processes whereby two chromosomes are 'mated', producing 'child' chromosomes based on 'parent' chromosome representations. Improvements can arise by selection and crossover (i.e. 'mating') of the fittest chromosomes. Mutations may also be applied, whereby a gene of a chromosome is randomly modified to introduce diversity within the population of chromosomes.

Objective functions of a GA generate values utilized to compare performance of decoded chromosomes, such that the optimal or dominating chromosome may be identified. Values may be maximized or minimized depending upon need; for instance, maximising accuracy and minimizing error. GAs with one objective are Single-Objective GAs (SOGA), those with two or more are Multi-Objective GAs (MOGA).

A SOGA optimizes a single objective, improving to a singular end. However, a MOGA optimizes multiple objectives and in some cases the objectives compete, meaning there exists no 'global' solution thus a concept of how a solution 'dominates' another is used to rank solutions, each set of solutions that do not 'dominate' each other form a 'pareto' front, multiple such fronts may exist.

D. Multi-Objective Algorithms

Non-dominated Sorting Genetic Algorithm II (NSGA-II) [22], an improved version of NSGA creates an initial population of chromosomes and evaluates those chromosomes using a defined problem. Chromosomes in pairs create a child chromosome population, using crossover and mutation, which is then evaluated. Evaluated chromosomes are all compared to each other, providing the domination information of each chromosome, i.e. which chromosomes dominated and which were dominated by an individual chromosome. Non-dominated chromosomes form the first or 0 subset; these pareto optimal chromosomes are removed from any dominating set of remaining chromosomes. This process continues, finding all possible and existing pareto fronts for the problem until all chromosomes have been assigned to a front.

NSGA-III is an improved version of NSGA-II, able to more efficiently evolve solutions where problems include a large number of objectives.

MOEA/D is a Multi-Objective Evolutionary Algorithm that decomposes optimization problems into sub-problems, at each generation the best solution of each sub-problem forms the new generation, in this way the complexity of the overall problem is reduced.

E. Intrusion-based NIDS using GAs

GAs are also used to choose an optimal set of variables. In [23] a GA has been implemented and used for the detection of intrusions from the KDD'99 dataset and is compared. A multi-class approach was taken, performing well on Denial of Service (DoS) data, however not as a well on Normal data, although the performance on DoS and User to Root data is stated as better than that of the KDD'99 competition winner.

In [24] a set of features and the parameters of SVMs kernel function is evolved to optimize the detection rate of the resultant SVM. The proposed system is said to sometimes outperform the KDD'99 competition winner.

In [25], a GA named Archive-based Micro Genetic Algorithm 2 (AMGA2) is used to optimize the features of an NB and compared against other methods using NB as the base algorithm. Experiments are carried out on both KDD'99 and the more recent ISCX2012 datasets, although low numbers of instances are selected from ISCX2012.

III. Methodology

The focus of the work presented in this paper is investigation and determination of optimal sets of BLCs, using NSGA-II, for use in an ensemble, StackingC, employed for the detection of network intrusions.

BLCs and an efficient ensemble method is implemented, as described in [7]. a particular distinction is the difference of domain upon which the set of BLCs are used, in this case network intrusion detection. Additionally compared to some current literature, a more recent dataset, ISCX2012, is also implemented with large number of instances, providing investigation into the effects of large scales of data with StackingC in an evolutionary context.

To simplify selection and implementation of algorithms for the ensemble, algorithms within Weka are used to form BLCs. In this case StackingC, the BLCs and Meta Level Classifier (MLC), in this case Multi-response Linear Regression, each coming from Weka. In addition, JMetal, an object-oriented Java-based framework for multi-objective optimization with metaheuristics, has been utilized for encoding and decoding chromosomes for the GA, providing an initial population of chromosomes and assisting in catching potential issues with child chromosomes; in the case of this work, ensuring a minimum number of BLCs in initial populations and handling when chromosomes decode to an empty set.

While it is known that NSGA-III more current GAs exist, NSGA-II is a robust and popular approach adequate for two objectives. The work herein did not require more than two, so NSGA-III was not required to replace NSGA-II which was more easily available for implementation from JMetal. Development of a GA was not in focus thus a generic NSGA-II is implemented.

A. *StackingC with pre-trained BLCs*

StackingC required a number of changes to provide the efficiency found when using pre-trained BLCs. Objects and methods were added to Stacking and StackingC (the latter depending upon the former), maintaining trained BLCs, indices selecting required BLCs for the current chromosome and the meta data with which to train each MLC. The reason for implementing a more efficient StackingC follows from the potentially unlimited number of BLCs that could be included.

When StackingC is used over multiple iterations of a set of BLCs, each set would require training on the full dataset. However, building from the full dataset requires time; the amount of time then determined by considering each BLC, as well as training data, compounding the issue of time required to fully train each BLC in the resulting ensemble. Thus due to a combination of, firstly, size of current and potential future BLC sets, secondly, training datasets and, thirdly, time required to train individual BLCs, a change from training each BLC for each ensemble iteration to only training once was required.

B. *NSGA-II using BLC evaluations*

Binary encoding of the set of BLCs was selected, presence or absence of a BLC is represented by a 1 or 0 respectively. An implementation of NSGA-II capable of handling binary encodings was utilized. Subsequently, creation and evolution of chromosomes provide sets of BLCs, where these sets are subsets of the entire possible set of BLCs. Comparisons of chromosomes involved minimization of both FP and FN values. It is assumed that, two sets of BLCs that perform well may be evolved and produce at least one set of BLCs that can perform better than the prior two sets.

C. *Evolving sets of BLCs for StackingC*

The use of a GA would facilitate the process of evolving a chromosome evaluated on pre-selected objectives, evaluation provided by the more efficient StackingC. Within this section such a system is described, it is also illustrated in Fig.1. The illustration provides, in 10 parts total, the methodology of the experiment from an initial StackingC and population generation followed by descriptions of the evolution of the population.

An initial iteration of StackingC is run on training data, training the MLC but also producing a set of Trained BLCs, from the full set of BLCs, and the Meta Data, representing classifications from each BLC, used to train the MLC, as shown in parts 1-3 of Fig.1 respectively. Trained BLCs and Meta Data are represented in parts 2 and 3 respectively, these are maintained throughout.

Subsequently, as illustrated in part 4 of Fig.1, NSGA-II through the use of a problem description creates multiple solutions forming the initial population of chromosomes.

As shown in part 5 of Fig.1 the GA decodes the chromosome to produce an array of indices indicating which BLC should be present in a set. This array is used against the full set of Trained BLCs to provide the set of Reduced BLCs, given by part 6 of Fig.1.

Both the array and reduced set are passed to a new StackingC instance, given by part 7 of Fig.1, to set the required BLCs and the indices of data from the Meta Data initially used to train the MLC, as shown in part 8 of Fig.1.

The combination of these three objects provides the creation of a StackingC model without retraining the whole set of BLCs or recreating meta data with which to train the MLCs, as shown in part 8 of Fig.1. Meta data is only adapted to represent the new instance of StackingC, training each MLC required within StackingC only with portions of the meta data represented by indices provided by NSGA-II.

Testing Data passed to the reduced set of BLCs is used to produce meta data for the testing of Trained MLCs. Classification and evaluation of the ensemble, described by the chromosome, are used to update the solution's objective values and subsequently the solution in the population, as shown in part 9 of Fig.1. The process described by parts 5-9 of Fig.1 is repeated for each chromosome in the initial population, after which the process described in part 10 is performed.

With the population evaluated, if the stopping condition is not met, which in this case is the evaluation of 30 generations of the population, a child population is evolved. The set from which the child set evolved is the parent population. Each child population chromosome is subject to processes of parts 5-9 of Fig.1. Optimal chromosomes from parent and child set

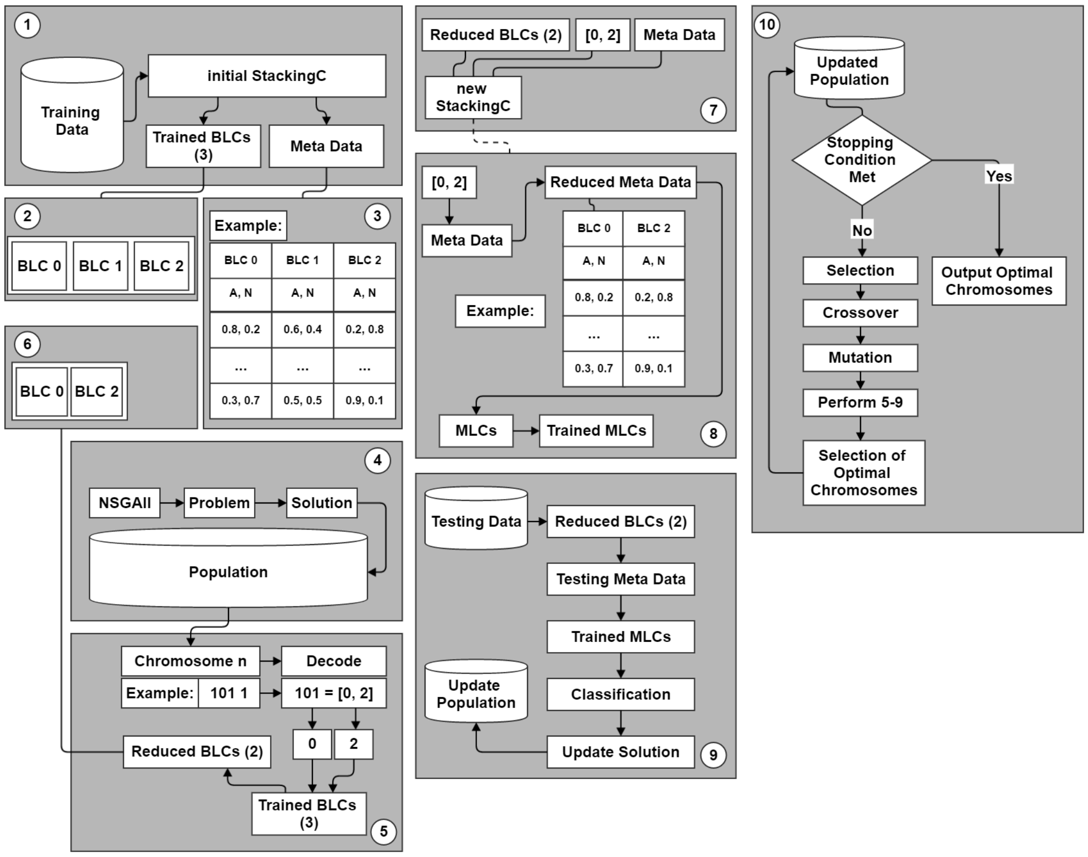

Fig. 1. Illustration of efficient StackingC training and subsequent evolution by NSGA-II

are found by measure of domination, as described previously, to form a new Updated Population of parent chromosomes, the process repeats until the specified number of generations has been passed.

D. Datasets

As previously mentioned, the work presented in this paper employs two primary datasets: NSL-KDD and ISCX2012. NSL-KDD has been pre-processed into training and testing datasets. By contrast, ISCX2012 has not been pre-processed, hence required training and testing datasets to be subsequently generated. Consequently, the ISCX2012 dataset was split in two ways: UUJN - using only the Distributed Denial of Service (DDoS) scenario from the dataset, with a similar distribution of classes as NSL-KDD; and Yassin2013 - using an approximation of the dataset used by [19] with a higher number of attack instances in the training set. Table I gives brief description of datasets.

IV. Results

A number of experiments were performed to determine the optimal sets of BLCs for StackingC, with a focus on detecting network intrusions. Each column in Table II represents the point of a chromosome at which the presence of the associated BLC is determined, also represented are the changes made to the possible position of each BLC in a chromosome as the iteration sets change. The optimal sets of BLCs depicted in Table IV are referred to as ensembles.

During the experiments, each dataset was used with the full set of BLCs. Some adjustments to the full set of BLCs was required in order to allow results from each dataset to be obtained. This was due to different feature types, number of instances as well as how each BLC performed on each dataset.

One issue faced was the time complexity of training; NSL-KDD was trained in an adequate [Define adequate here] amount of time and was able to produce results on the full set of BLCs. However, UUJN and Yassin2013 were found to require extended periods of time to be trained. Extended periods such that some BLCs required scores of hours before completing training and in some cases did not finish. Therefore, it was only possible to obtain results from a subset of the BLCs for these two datasets, as indicated in Table II. Consequently, BLCs that experienced extended training times were removed from the full set and a new subset/full set was produced. This new set experienced no such issues with UUJN, however still some issues were found with Yassin2013 for one BLC. In response, again, the BLC causing issues was removed and a new subset/fullset was produced.

Thus three BLC set iterations were of appropriate use for

TABLE I
CLASS DISTRIBUTIONS OF DATASETS USED FOR EXPERIMENTS

Dataset	**Features**	**Training**			**Testing**		
		Normal	*Attack*	*Total*	*Normal*	*Attack*	*Total*
NSL-KDD	42	67,343	58,630	125,973	9,711	12,833	22,544
UUJN	16	35,223	30,666	65,889	4,571	6,040	10,611
Yassin2013	15	63,765	8,968	72,733	19,115	37,159	56,274

TABLE II
PRESENCE OF BASE LEVEL CLASSIFIERS FROM WEKA

Base Level Classifier		**Base Level Classifier Set**		
Name	*Type*	*One*	*Two*	*Three*
NaiveBayesUpdateable	bayes	10000000000000	10000000000	1000000000
PART	rules	01000000000000	01000000000	0100000000
J48 (pruned)	trees	00100000000000	00100000000	0010000000
J48 (unpruned)	trees	00010000000000	00010000000	0001000000
DecisionStump	trees	00001000000000	00001000000	0000100000
DecisionTable	rules	00000100000000	00000100000	0000010000
ClassificationViaRegression	meta	00000010000000		
RandomForest	trees	00000001000000		
RandomTree	trees	00000000010000		
VFI	misc	00000000010000	00000010000	0000001000
ConjunctiveRule	rules	00000000001000	00000001000	0000000100
JRip	rules	00000000000100	00000000100	0000000010
NNge	rules	00000000000010	00000000010	
HyperPipes	misc	00000000000001	00000000001	0000000001
Total Classifiers		14	11	10

each dataset with some overlap between which datasets and which sets of BLCs provided results. NSL-KDD produced results with each BLC set iteration, UUJN produced results with the second and third BLC set iterations and Yassin2013 only produced results with the third BLC set iteration. This is shown in Tables III and IV.

A. On NSL-KDD Results

The results of each BLC set iteration, as given in Table III, show that overall there is a single data point that exists as an optimal set of FP and FN values, which is achievable by multiple unique ensembles. This is true across each BLC set iteration where the FP and FN values obtained remain the same. All ensembles across NSL-KDD achieve 248 FPs and 3263 FNs, while some of the individual BLCs achieve similar results. It can be seen from Table IV that the ensembles perform best regarding FN where in all but one case the individuals achieve around 700 more FNs.

Considering the results obtained from NSL-KDD, shown in Table III and Table IV, it can be determined that the ensembles outperform the majority of BLCs, in terms of FP as well as FN. An exception, however, is found with the HyperPipes BLC, which outperforms any of the ensembles in terms of FP, achieving 56 FP compared to 248 FP achieved by any ensemble. The data points from BLCs and optimal ensembles are presented in Fig.2. It is evident from positions of BLCs in comparison to the optimal ensemble that the latter outperforms the former. Similar figures exist for UUJN and Yassin2013, however due to space constraints are not shown here.

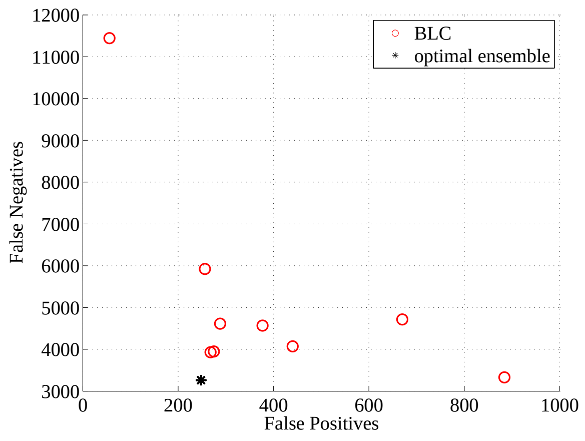

Fig. 2. NSL-KDD Results

B. On UUJN Results

As previously discussed, in terms of the UUJN dataset, there are only results from BLC set iterations Two and Three. Across both of these set iterations there are four data points found, each with at least two unique ensembles. The ensembles outperform a number of BLCs in terms of both FP and FN. However, while some of the ensembles outperform a number of BLCs, both PART and JRip produce similar results and one, J48 unpruned, produces the same results.

From Table III and Table IV it may be observed that PART achieved 1 FP with 10 FN to an ensembles 0 FP with FN 10 and JRip achieved 2 FP with 7 FN to an ensembles 1 FP with 6 FN. J48 unpruned achieved the same results as some

TABLE III
INDIVIDUAL BASE LEVEL CLASSIFIER RESULTS

Base Level Classifier	**Dataset**					
	NSL-KDD		*UUJN*		*Yassin2013*	
	FP	FN	FP	FN	FP	FN
NaiveBayesUpdateable	670	4714	67	66	728	37152
PART	275	3948	1	10	11	37159
J48 pruned	263	3900	0	16	60	37159
J48 unpruned	268	3930	3	3	60	37159
DecisionStump	440	4072	65	66	3	37158
DecisionTable	256	5923	92	25	415	37156
ClassificationViaRegression	659	4713				
RandomForest	267	4206				
RandomTree	278	4381				
VFI	377	4567	37	12	54	37069
ConjunctiveRule	884	3328	65	66	0	37159
JRip	288	4613	2	7	11	37159
Nnge	697	3994	430	4		
HyperPipes	56	11443	445	9	211	37069

TABLE IV
OPTIMAL ENSEMBLES AND RESULTS FOUND FOR EACH DATASET AND RELEVANT BLC SET ITERATION

Dataset	**Base Level Classifier Set Iterations**								
	One			***Two***			***Three***		
	Chromosome	*FP*	*FN*	*Chromosome*	*FP*	*FN*	*Chromosome*	*FP*	*FN*
NSL-KDD	11110000011000	248	3263	01111000000	248	3263	0111101001	248	3263
	11110000000001			01111000001			1111001001		
	01110000010001			01111010000			1111101001		
	11110000010001			11111000001			0111001001		
	11110000000000			01110000000			0111000000		
	01110000010000			11111010000			1111101000		
	11110000010000			01111010001			0111101000		
				11111000000			0111001000		
							0111000001		
UUJN	**N/A**			11100101011	0	10	0110110001	0	11
				11100101010			0110010001		
				10110101000	1	6	1011110001	1	6
				10110101001			0011110001		
				00110101001			1011110000		
				00110101000			1011010000		
				00111100000			1011010100		
				10011111011	2	4	0011100001	3	3
				00011111011			0001111001		
				10010001011	3	3	0001100101		
				00010111000			0011010001		
				10010101000			1001100101		
				10010011011			0011000000		
				00010101000			1011100100		
				10010011001			1011000100		
				10010111011			1011000000		
				10011101000			1011100001		
							0001000101		
							0001010000		
Yassin2013	**N/A**						0000100100	0	37159
							1000100100		
							1000100101		
							1000000101		
							0000100101		
							0000000101		
							0000100001	3	37158
							1000100000		
							0000100000		
							1000100001		
							0000111001	43	37069
							0000011001		
							0000011100		
							0000011000		

ensembles of 3 FPs with 3 FNs as evidenced in Tables III and IV. The optimal ensembles achieve low FPs ranging from 0-3. While these values match some FPs produced by BLCs, the FNs indicate that those same instances achieve higher results and, in some cases, the optimal ensembles achieve lower FNs. For example, PART with 1 FP and 10 FNs is outperformed by an ensemble with 0 FPs and 10 FNs. While this difference may not be large it is still indicative of a better performing ensemble in some cases.

C. On Yassin2013 Results

Finally, with regard to the Yassin2013 dataset, results were only obtained for BLC set iteration Three. As there exists only one set of results for this dataset the comparisons between the optimal sets and BLCs are somewhat simplified. Consequently, there are a small number of optimal data points found, each with at least four unique ensembles. Considering the results from each individual BLC, as well as comparisons between the optimal ensembles, it can be observed that some BLCs are able to achieve the same result as optimal ensembles. In particular, this is the case with the BLCs ConjunctiveRule and DecisionStump, which achieve 0 FPs with 37159 FNs and 3 FPs with 37158 FNs respectively. Comparisons also show that while both the set of individual BLCs and optimal ensembles achieve 37159, 37158 or 37069 FNs, the number of FPs vary more, with the optimal ensembles outperforming most of the BLCs in terms of the number of FPs alone. Optimal ensembles achieve 0, 3 or 43 FPs, where optimal ensembles that achieve 0 or 3 FPs perform better than most individual BLCs where the number of FPs are typically found to be 11 and above, three of which achieve over 200 FPs.

V. Discussion

As stated in Section IV, the optimal ensembles found with NSL-KDD outperform the majority of individual BLCs. Comparing them on their individual FPs and FNs, 100% of the ensembles outperform 93% of the BLCs in regards to FPs while 100% of the ensembles outperform 100% of the BLCs in regards to FNs. Thus, overall, it is shown that 93% of the optimal ensembles strictly outperform any of the BLCs. The greatest distinction between BLCs and optimal ensembles may be the difference found with the single BLC, HyperPipes, that is not strictly improved upon; the FPs of the BLC is approximately 22% of any ensemble's FPs, the FNs from the BLC is approximately 350% of any ensemble's FNs. Hence while the FPs may be of more importance for NIDS, the FNs would have more of an effect being instances where an intrusion was not caught. The performances found with UUJN differ in comparison to NSL-KDD, the differences between optimal ensembles and individual BLCs are not as definitive. With four and three possible optimal ensemble results for BLC set iterations Two and Three respectively, individual comparisons are more varied. For BLC set iterations Two and Three the domination of BLCs by at least a single optimal ensemble is approximately 63% and 54% of individual BLCs respectively. The performance when considering Yassin2013 is more easily defined. Two BLCs, DecisionStump and ConjunctiveRule, achieve results equal to those of two different sets of optimal ensembles. Those optimal ensembles are least likely to dominate given results equal to individual BLCs, thus remains the third optimal ensemble, strictly dominating 60% of the individual BLCs.

Given that adjustments to the sets of BLCs were based on the usability of the datasets, it has been shown that each BLC did not perform equally across each individual dataset based on the results. A clear example of this is with the BLC named ConjunctiveRule; from results obtained from both the NSL-KDD and UUJN datasets it may be observed that any optimal ensemble outperforms ConjunctiveRule. Instead, from results obtained from Yassin2013 it may be observed that ConjunctiveRule performs just as well as at least one of the ensembles, example 0000100100.

Where comparisons across each dataset can be made directly, over BLC set iteration Three, the last three columns of Table IV, it can be seen that each ensemble found to be optimal is unique; an optimal set of BLC found for NSL-KDD is not found for UUJN or Yassin2013, this is also true for UUJN and Yassin2013. Accordingly, it may be the case that the data points utilized by each dataset may be less compatible with some BLCs than with others. A particular BLC may perform better when trained on NSL-KDD rather than UUJN; BLCs that do not perform as well may be removed from subsequent generations of chromosomes by the evolutionary process.

When considering the patterns of the presence of the BLCs, over set iteration Three, as given in Table IV, it can be seen that for each dataset there is at least one BLC, namely JRIP, that does not appear in any of the optimal ensembles. JRip is dominated by at least one ensemble whenever used with NSL-KDD, UUJN or Yassin2013. Consequently, it is likely that future experiments may be able to exclude JRip as it would appear to play no part in the generation of optimal ensembles.

With the removal of a BLC, NNge, between set iteration Two and set iteration Three, there is a slight decrease in performance with UUJN. In set iteration Two an optimal ensemble, example 11100101011, achieves 0 FPs and 10 FNs, while in set iteration Three an optimal ensemble, example 0110110001, achieves 0 FPs and 11 FNs. While these example ensembles differ in the presence of more than one BLC, it may be observed that removal of NNge allows production of similar but worse results by the remaining BLCs. Furthermore, a different optimal ensemble, example 10011111011, is no longer produced. Hence, achieving 2 FPs and 4 FNs no longer occurs in set iteration Three, perhaps indicating that while the addition of a BLC can improve an optimal ensemble, it may also introduce a wider range of unique optimal ensembles from a set of BLCs including the additional BLC.

Comparisons can be made against some similar works. Comparing against [19] shows less FPs however more FNs, all evolved ensembles more accurately detect intrusions but less accurately classify normal instances. It should be noted that the Yassin2013 dataset used here is only an approximation of the dataset used in [19] and not an exact replica, this may provide some explanation where an exact replication could provide more accurate comparisons.

VI. Conclusion

The hypothesis was that a GA should be able to evolve sets of BLCs for use with an MCS to produce an ensemble that is able to outperform a number of the individual BLCs that comprise the ensemble. Results show that evolved optimal ensembles perform better than individual BLCs, the number of optimal ensembles that perform better would appear to partly be based on the dataset.

When considering the BLC set iterations, no singular optimal set of BLCs is produced that may be applied with impunity across the varied datasets. However, NSGA-II is capable of evolving sets of BLCs to optimal ensembles for each dataset. Individual BLCs require some analysis before benefits may be produced, especially where prolonged training time requires removals of BLCs, indicating that datasets may dictate inclusion of BLCs. Sets of BLCs may also need to be adequately large to provide optimal variations of optimal ensembles and subsequent data points from evaluation measures. Thus the benefits of optimization of StackingC could rely on combinations of BLCs and datasets.

Future works could include further investigation and analysis into a number of aspects; selection of individual BLCs better suited for varied datasets, implementation of a more recent GA, additional objective functions.

References

[1] A. Kumar and E. B. Fernandez, "Security patterns for intrusion detection systems," in *1st LACCEI International Symposium on Software Architecture and Patterns (LACCEI-ISAP-MiniPLoP2012), Panama City, Panama*, 2012.

[2] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 303–336, 2014.

[3] M. A. Ambusaidi, X. He, and P. Nanda, "Unsupervised Feature Selection Method for Intrusion Detection System," *2015 IEEE Trustcom/BigDataSE/ISPA*, 2015.

[4] D. Bolzoni, S. Etalle, and P. Hartel, "Poseidon: a 2-tier anomaly-based network intrusion detection system," in *Information Assurance, 2006. IWIA 2006. Fourth IEEE International Workshop on*. IEEE, 2006, pp. 144–156.

[5] P. M. Mafra, V. Moll, J. da Silva Fraga, and A. O. Santin, "Octopus-iids: An anomaly based intelligent intrusion detection system," in *Computers and Communications (ISCC), 2010 IEEE Symposium on*. IEEE, 2010, pp. 405–410.

[6] E. Menahem, L. Rokach, and Y. Elovici, "Troika–an improved stacking schema for classification tasks," *Information Sciences*, vol. 179, no. 24, pp. 4097–4122, 2009.

[7] A. Ledezma, A. Sanchis, and F. J. Ord, "Genetic Approach for Optimizing Ensembles of Classifiers," *In proceeding of: Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference*, pp. 89–94, 2008.

[8] M. Tavallaee and E. Bagheri, "A detailed analysis of the KDD CUP 99 data set," *Proceedings of the* . . . , no. Cisda, pp. 1–6, 2009. [Online]. Available: http://nparc.cisti-icist.nrc-cnrc.gc.ca/npsi/ctrl?action=rtdoc{\&}an=15084639

[9] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012.

[10] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling Neural Networks: Many Could Be Better Than All," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[11] S. Axelsson, "The base-rate fallacy and its implications for the difficulty of intrusion detection," in *Proceedings of the 6th ACM Conference on Computer and Communications Security*, 1999, pp. 1–7.

[12] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *2010 IEEE Symposium on Security and Privacy*. IEEE, 2010, pp. 305–316. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5504793

[13] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[14] A. Balon-Perin and B. Gambäck, "Ensembles of decision trees for network intrusion detection systems," *International Journal on Advances in Security Volume 6, Number 1 & 2, 2013*, 2013.

[15] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[16] K. M. Ting and I. H. Witten, "Issues in stacked generalization," *Journal of Artificial Intelligence Research*, vol. 10, pp. 271–289, 1999.

[17] A. K. Seewald, "How to make stacking better and faster while also taking care of an unknown weakness," in *Proceedings of the Nineteenth International Conference on Machine Learning*, ser. ICML '02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 554–561. [Online]. Available: http://dl.acm.org/citation.cfm?id=645531.656165

[18] D. Ariu, R. Tronci, and G. Giacinto, "Hmmpayl: An intrusion detection system based on hidden markov models," *computers & security*, vol. 30, no. 4, pp. 221–241, 2011.

[19] W. Yassin, N. I. Udzir, and Z. Muda, "Anomaly-based intrusion detection through k-means clustering and naives bayes classification," in *Proceedings of the 4th International Conference on Computing and Informatics, ICOCI 2013*, no. 049, 2013, pp. 298–303. [Online]. Available: http://www.icoci.cms.net.my/proceedings/2013/PDF/PID49.pdf

[20] W. Li, "Using genetic algorithm for network intrusion detection," *Proceedings of the United States Department of Energy Cyber Security Group 2004 Training Conference, Kansas City, Kansas*, pp. 24–27, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.3125{\&}rep=rep1{\&}type=pdf

[21] R. H. G. R. H. Gong, M. Zulkernine, and P. Abolmaesumi, "A software implementation of a genetic algorithm based approach to network intrusion detection," *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network*, pp. 246–253, 2005. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1434896

[22] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[23] M. S. Hoque, M. A. Mukit, M. A. N. Bikas, and M. Sazzadul Hoque, "An Implementation of Intrusion Detection System Using Genetic Algorithm," *International Journal of Network Security Its Applications*, vol. 4, no. 2, pp. 109–120, 2012. [Online]. Available: http://www.airccse.org/journal/nsa/0312nsa08.pdf

[24] D. S. Kim, H.-n. Nguyen, and J. S. Park, "Genetic Algorithm to Improve SVM Based Network Intrusion Detection System," *19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers)*, vol. 2, pp. 155–158, 2005. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1423667

[25] G. Kumar and K. Kumar, "Design of an evolutionary approach for intrusion detection." *TheScientificWorldJournal*, vol. 2013, p. 962185, 2013. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-84893827507{\&}partnerID=tZOtx3y1