Dynamic Insider Threat Detection Based on Adaptable Genetic Programming

Duc C. Le, A. Nur Zincir-Heywood, Malcolm I. Heywood
Faculty of Computer Science, Dalhousie University
Halifax, Nova Scotia, Canada
Email: lcd@dal.ca, zincir@cs.dal.ca, mheywood@cs.dal.ca

***Abstract*—Different variations in deployment environments of machine learning techniques may affect the performance of the implemented systems. The variations may cause changes in the data for machine learning solutions, such as in the number of classes and the extracted features. This paper investigates the capabilities of Genetic Programming (GP) for malicious insider detection in corporate environments under such changes. Assuming a Linear GP detector, techniques are introduced to allow a previously trained GP population to adapt to different changes in the data. The experiments and evaluation results show promising insider threat detection performances of the techniques in comparison with training machine learning classifiers from scratch. This reduces the amount of data needed and computation requirements for obtaining dependable insider threat detectors under new conditions.**

***Index Terms*—insider threat detection, cyber-security, dynamic environment**

I. Introduction

Insider threat is one of the most dangerous cyber-security problems that organizations may face. The threat represents a wide range of malicious activities as performed from "inside" the organization. Data exfiltration, disclosure of classified information, intellectual property theft, and IT sabotage are notable examples of insider threats [1]. According to recent reports, not only 53% of organizations and 42% of U.S. federal agencies encounters insider threats each year, the attacks are also becoming more frequent [2], [3].

The fact that malicious insiders are authorized to use the computer systems and are familiar with the organization's security procedures creates many challenges in applying machine learning for insider threat detection. This is a highly unbalanced data problem, where data representing malicious behaviours are rare and usually not well documented. Moreover, dynamics in corporate environments represent another major source of challenge. Firstly, user behaviours, both of normal users and malicious insiders, may change and develop over time. More importantly, a permanent deployment environment may not be assured for machine learning solutions. Different changes may appear in the application environments and data collection and processing procedures. Examples of such changes in insider threat detection might include:

- New features, which may be extracted from new data sources with new information, such as new sensor types, or a new/updated monitoring system.
- New output classes, which result from newly discovered data behaviours, e.g. new malicious insider types.

Essentially, the dynamics of the deployment environment generate changes in feature and/or label space for machine learning applications. This, in turn, often leads to incompatibilities or inadequate performance of traditional machine learning deployments [4].

In this work, the use of Linear Genetic Programming (GP) is explored for addressing the challenges. Specifically, this work studies methods to allow a trained GP population to evolve on expanded feature (input) space and label (output) space. In doing so, the aim is to learn from additional features and classes, while preserving the previous performance on legacy data aspects. The Linear GP based approach is evaluated using multiple releases of a publicly available insider threat detection dataset.

The rest of the paper is organized as follows. Section II summarizes related work in machine learning applications for insider threat detection and learning in dynamic conditions. Section III formally states the challenges and presents our proposed solutions based on Linear GP. This section also details the employed datasets, while Section IV presents the experiments and evaluation results. Finally, conclusions are drawn and the future work is discussed in Section V.

II. Related Work

Cyber-security in general and intrusion detection in particular presents a rich set of opportunities for evolutionary computation. Song et al. in [5] applied Linear GP for intrusion detection on a large and highly imbalanced dataset (KDD-99). Haddadi et al. employed Symbiotic Bid-based GP for botnet detection and showed the advancement of GP over rule based and packet payload inspection based systems [6]. In [7], Sen and Clark applied genetic programming and grammatical evolution for intrusion detection in mobile network environments, with a focus on power efficiency of the solution. In general, Wu and Banzhaf surveyed applications of computational intelligence, from artificial neural networks, fuzzy systems, evolutionary computation, artificial immune systems, to swarm intelligence, and soft computing in intrusion detection [8].

In insider threat detection specifically, Le et al. benchmarked GP methods, based on both static and active learning approaches [9]. On the other hand, many other non-evolutionary approaches have been applied to insider threat detection, where most of them are based on anomaly detection. Approaches proposed to date include graph based anomaly detection [10], [11], Hidden Markov Model and Gaussian Mixture Model [12]–[14], deep neural network [15], decision tree and self-organizing map [16]–[18]. A general review of such applications can be found in [19].

Several works have been proposed for dealing with changing dynamics in machine learning deployment environments. For streaming data, where main challenges are non-stationary data and limited label budget, team-based GP has been successfully applied through the use of suitable sampling policy and archiving policy [4], [20]. In [21], Haddadi and Zincir-Heywood examined the effectiveness of Symbiotic Bid-Based GP and other machine learning methods for botnet detection under botnet evolution conditions. On evolving feature space, recently Manzoor et al. in [22] proposed xStreams, a technique for outlier detection in feature-evolving data streams through the use of a streaming random projection scheme and ensemble of half-space chains.

III. Methodology

In this section, we present and discuss the approach employed to explore the capabilities of GP for malicious insider detection in corporate environments.

A. Problem Statement

In real world, the changes in the deployment environment may cause the data generated by the applications and services to evolve in both the number of features and the number of classes (different behaviours). These changes could essentially render many deployed machine learning solutions, which assume a fixed feature space and output classes, obsolete. Specifically, additional time and effort, and training examples will need to be invested to create new models and to ensure that the new models perform adequately on the new data.

In formalizing the problem, this study assumes that in deployment is a machine learning classifier $\mathscr{C}$ working on a data with feature set $\mathcal{F}$ to output to a set of categories $\mathcal{C}$. Then anticipated or not, changes in environment and data collection create a different data stream with feature space $\mathcal{F}_1$, where $\mathcal{F} \subset \mathcal{F}_1$, and / or a larger output space (set of classes) $\mathcal{C}_1$, where $\mathcal{C} \subset \mathcal{C}_1$. Without having to train a completely new classifier to accommodate the changes, there are two major challenges in evolving $\mathscr{C}$ to $\mathscr{C}'$ working under the new environment: (i) Learning from newly introduced features, $\mathcal{F}_1 \setminus \mathcal{F}$, and (ii) Learning to classify new classes, $\mathcal{C}_1 \setminus \mathcal{C}$, while maintaining current performance on legacy features and classes.

B. Linear Genetic Programming based Proposals

In this work, we select an evolutionary based method, specifically Linear GP, as the prime candidate for fulfilling the stated challenges. It is assumed that GP can both perform feature construction based on existing and new features, as well as scale to additional classes without invalidating legacy solutions. Indeed, the population based approach might provide a suitable means for incrementally shifting from legacy solutions to new solutions. GP has been successfully applied in streaming data tasks, under concept change and limited label budget conditions, [9], [20]. In this section, we describe Linear GP and our methods in order to incorporate newly introduced information from larger feature and output spaces.

1) Linear Genetic Programming: Linear GP is a variant of GP where programs in a population are represented in a linear structure, as a sequence of instructions from an imperative programming language [23]. The execution of a Linear GP program follows a graph-based data flow, where each instruction is executed based on the defined arithmetic operations and operands, which can be registers, constants, or input values; and the output is taken at the end of the program as the values of the designated registers. The design allows the register content to be reused multiple times during execution, and structurally noneffective code (introns) to exist. Introns may reduce the effect of variation on the effective code and allows neutral variations in terms of fitness change [23].

In this work, Linear GP is trained through generations using subsets of the original training data. Based on fitness values from the evaluation of the population on a data subset, selection and variation operators are performed to reproduce the population for the next generation. Since the data is extremely skewed, multi-objective selection is employed to address two objectives simultaneously: (i) Maximizing the detection rate (over all classes), and (ii) Maximizing the accuracy. This is done through the use of Pareto ranking. At each generation, higher ranked individuals are selected to produce offsprings through variation operators to replace the worst ranked individuals. The variation operators include: (i) Crossover, where blocks of instructions are swapped between pairs of parents, (ii) Micro mutation, where a part of a selected instruction is modified (target register, operands, or operator), and (iii) Macro mutation, where a selected instruction is replaced, deleted, or a random instruction is inserted.

2) Learning from an expanded feature space: In Linear GP, the input feature space to each GP program is defined by a set of read only input registers [23]. An instruction set selectively reads values and potentially performs feature construction by manipulating the content of 'general purpose registers' (GPR). The program and the output is defined by mapping the value of some subset of the GPR to class labels. The design essentially allows Linear GP program to take any amount of expansion in feature space through expanding the register space accordingly.

Initially, new features have no effect on the execution and output of a program. However, employing appropriate changes in variation operators, such as mutation, new features in $\mathcal{F}_1 \setminus \mathcal{F}$ might be incorporated into programs through the training generations. This allows a previously trained population to evolve on an expanded feature space. On modifying the variation operators to allow exploration of an expanded feature space, we examine the following two methods:

(i) Simply adjust the variation operators to take into account the expanded feature space. In this case, variation operators will immediately consider a feature $f_1 \in \mathcal{F}_1 \setminus \mathcal{F}$ with the same probability as $f \in \mathcal{F}$ to include in an instruction.

(ii) Introduce a bias toward selecting a feature $f_1 \in \mathcal{F}_1 \setminus \mathcal{F}$ in the variation operators upon detecting feature space expansion. In this initial work, we examine a scheme where new features in $\mathcal{F}_1 \setminus \mathcal{F}$ are selected with higher probability, $p_{f_1} = \alpha_t \times p_f$, where p is the probability that a register is selected by the variation operators and/or sampled by a new instruction. Let α_t be a generation dependent factor that starts at $\alpha_0 > 1$ and gradually decreases to 1 at generation G_f.

3) Learning to classify a new class: Similar to the feature space expansion, new output classes of a Linear GP program can be accommodated by creating new output registers that corresponds to the new classes. Initially the introduction of new output register has no effect on output of a GP program, as they are not involved in any trained Linear GP programs. Eventually, through selection and variation operators, the GP population may learn to recognize the new class. To accelerate the process, we investigate several methods for putting higher pressure on the Linear GP population such that new classes are more likely to be detected:

(i) A custom cost matrix can be used to reward individuals that succeed in classifying the new class. For example, the cost of misclassifying an exemplar of class $c_1 \in \mathcal{C}_1 \setminus \mathcal{C}$ can be set to $E_{c_1} = \beta_t \times E_c$, where E_c is misclassification cost of a class $c \in \mathcal{C}$. β_t is a time dependent factor, that, similar to α_t, starts at β_0 and reduces to 1 at G_c.

(ii) Using multi-objective selection, choose the detection rate of recently introduced classes in $\mathcal{C}_1 \setminus \mathcal{C}$ as an objective to select individuals for reproduction. In this work, we apply an objective swapping scheme, where the detection rate of the new classes are switched among other objectives in the first G_c generations upon output space expansion.

C. Dataset and Feature Extraction

This work employs the CERT insider threat datasets [24], which are a publicly available datasets for research, development, and testing of insider threat mitigation approaches. Two releases, 4.2 and 5.2, of the CERT datasets (hereafter R4.2 and R5.2) are used in the experiments. These releases simulate organizations with 1000 and 2000 employees, respectively, over the period of 18 months. Two main sources of information in the datasets are (i) the users' activity logs, which include log in/off, web access, file, email, and thumb drive connect, and (ii) the organization's structure and users' information.

The data releases are constructed using a number of different models including connection graphs, topic models, behavioural models, and psychometric models [24]. In each release of the dataset, in addition to an expanded organization size and structure, the source models are also updated to provide new information that is unavailable in previous releases. Furthermore, additional insider threat scenarios are added. In R4.2 and R5.2, there are 3 and 4 insider threat scenarios, respectively. These range from data exfiltration (scenario 1), intellectual property theft (scenarios 2 and 4) to IT sabotage (scenario 3) [24]. In short, the datasets provide an ideal condition for examining the aforementioned approaches, in both the feature space and the output space expansions.

From the data sources, pre-processing steps and feature extraction are performed. In this work, we extract features from each session of a user's activities, starting from a log on to a corresponding log off action. The session data has been shown to provide high malicious insider detection rates and low detection delay [18]. There are two main categories of features, depending on data sources. The first feature category is user activity features, which can be frequency features, such as the *number of website visited, number of user connections after hours, or the number of file access on a shared PC*, or statistical features, such as the mean and standard deviation of *email attachment size, file size, or visited website word count* in each session. The second category includes user and session information, e.g user's role, department, or session time, duration, and PC, which are designed to provide context for machine learning algorithms.

Table I presents a summary of the training and testing data in R4.2 and R5.2 datasets. To reflect real-world conditions, where ground truth is limited [25], [26], we use data belonging to the first half (50%) of the duration of each release and from a limited set of users (400) for training the machine learning algorithms. The second half of each release is used as the testing data (part). This resembles an extremely imbalanced data problem, where data from all malicious classes accounts for only 0.5% of the training dataset, and 0.2% of the testing dataset.

IV. Experiments and Evaluation Results

We conduct two main experiments to explore the performance of Linear GP. These are: (i) Feature space and (ii) Output space (new behaviours) expansion conditions. The main parameters of Linear GP are summarized in Table II. The performances of GP based classifiers are measured using detection rates,

$$DR_c = 100 \times \frac{tp_c}{tp_c + fn_c},$$

where tp_c and fn_c are true positive and false negative counts for class c. Class-wise detection rate is used as an objective in multi-objective selection:

$$DR = \frac{1}{|\mathcal{C}|} \times \sum_{c=1}^{|\mathcal{C}|} DR_c$$

Seeing that normal class dominates 99.5% of the data, classification accuracy is essentially the same as DR_{Normal}. Additionally, with this in mind, the best individual of GP population post training is selected with at least 99% training accuracy to keep low false positive rates. Results are presented in terms of mean and standard deviation from 10 runs in each experiment. All the experiments are performed on a lab

TABLE I
CERT DATASET SUMMARY

Data		$\|\mathcal{F}\|$	$\|\mathcal{C}\|$	Exemplar count				
				Normal	Scenario 1	Scenario 2	Scenario 3	Scenario 4
R4.2	training	107	4	101023	23	427	16	
	testing			217469	45	482	16	
R5.2	training	190	5	107747	25	489	10	78
	testing			497038	40	537	23	522

TABLE II
PARAMETERS OF LGP

Parameter	Value
Population size	2000
Data subset size	400
Number of generations	300
Crossover rate	0.6
Micro mutation rate	0.8
Macro mutation rate	0.8
Function set	$\{+, -, \times, /, >, \sin, \exp, \log\}$
Maximum program length	200

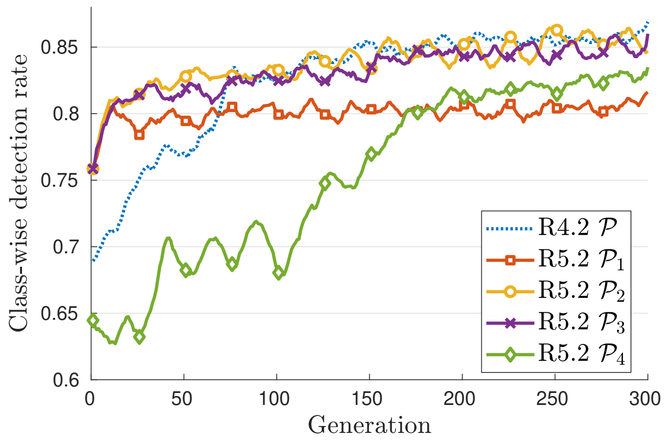

Fig. 1. Experiment 1 Class-wise detection rates of the populations on training subsets by number of generations

TABLE III
EXPERIMENT 1: R5.2 TEST RESULTS OF GP POPULATIONS TRAINED WITH INCREASING GENERATIONS.

# generation	Population	Normal DR	Insider threat DR
0	$\mathcal{P}, \mathcal{P}_{1-3}$	98.72 ± 0.40	21.46 ± 2.89
20	$\mathcal{P}_1$	98.13 ± 0.26	29.97 ± 3.15
	$\mathcal{P}_2$	97.96 ± 0.14	39.62 ± 5.03
	$\mathcal{P}_3$	97.98 ± 0.52	35.98 ± 2.00
	$\mathcal{P}_4$	99.91 ± 0.12	11.70 ± 7.21
50	$\mathcal{P}_1$	98.11 ± 0.22	36.59 ± 8.62
	$\mathcal{P}_2$	97.99 ± 0.02	45.36 ± 5.30
	$\mathcal{P}_3$	98.15 ± 0.12	38.41 ± 3.92
	$\mathcal{P}_4$	99.57 ± 0.07	15.74 ± 7.31
100	$\mathcal{P}_1$	97.97 ± 0.37	44.37 ± 4.71
	$\mathcal{P}_2$	97.91 ± 0.04	49.98 ± 1.60
	$\mathcal{P}_3$	98.04 ± 0.31	44.06 ± 6.44
	$\mathcal{P}_4$	99.51 ± 0.31	22.51 ± 7.74
200	$\mathcal{P}_1$	97.70 ± 0.29	46.60 ± 3.16
	$\mathcal{P}_2$	98.01 ± 0.04	52.54 ± 0.84
	$\mathcal{P}_3$	97.61 ± 0.16	49.12 ± 7.24
	$\mathcal{P}_4$	98.10 ± 0.57	39.65 ± 1.31
300	$\mathcal{P}_1$	97.60 ± 0.45	46.21 ± 1.63
	$\mathcal{P}_2$	97.67 ± 0.19	53.52 ± 1.39
	$\mathcal{P}_3$	97.98 ± 0.25	49.65 ± 5.78
	$\mathcal{P}_4$	98.13 ± 0.21	50.52 ± 0.14

machine equipped with dual Intel Xeon E5-2650v4 CPUs and 128 GB of RAM. Linear GP is implemented in Matlab R2018b [27].

A. Experiment 1 - Evolving GP on expanded feature space

In this experiment, a GP population which has been trained on R4.2 is then evolved on R5.2 in different settings for understanding of the evolution under feature space expansion. To focus exclusively on learning from the newly introduced features, we assume a binary classification task, where the two classes are: normal and malicious (insider threat). Based on III-B2, four different GP populations ($\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4$) are evolved under the expanded feature space. $\mathcal{P}_1, \mathcal{P}_2$, and $\mathcal{P}_3$, are evolved from population $\mathcal{P}$, which was trained for 300 generations on R4.2. Note that $\mathcal{P}$ achieved 98.39% and 51.50% normal and malicious DRs on R4.2 testing part (Table V). Population $\mathcal{P}_1$ uses the same (old) feature subspace $\mathcal{F}$ (as of R4.2) for evolution. Based on the two strategies described in III-B2, $\mathcal{P}_2$ and $\mathcal{P}_3$ are retrained on $\mathcal{F}_1$ (R5.2 full feature space), without and with a bias towards new features, respectively. The bias in variation operators of $\mathcal{P}_3$ towards newly introduced features is controlled by $G_f = 50$ and $\alpha_0 = 2$. Finally, a population $\mathcal{P}_4$ is trained from scratch on R5.2 as a baseline for comparisons.

Class-wise detection rates of the GP populations over *training* generations (on subsets of R5.2 and R4.2 training data) are shown in Figure 1. Table III presents *test* performance after 0, 20, 50, 100, 200, and 300 generations on R5.2 for the Linear GP training scenarios.

Figure 1 clearly shows that $\mathcal{P}_{1-3}$ maintained the performance of previously trained population $\mathcal{P}$. Furthermore, the populations evolved from that to adapt to changes in R5.2. Even at 0 generation (no data samples from R5.2 has been used to train $\mathcal{P}_{1-3}$), these GP populations are already better than $\mathcal{P}_4$, in terms of DR, after 100 generations. Using the initial advantage, $\mathcal{P}_{2-3}$ maintain clearly better performances than $\mathcal{P}_4$ during the rest of the training generations. Similarly, Table III shows that after just 100 generations (using about 40% of R5.2 training data), populations $\mathcal{P}_2$ and $\mathcal{P}_3$ achieved nearly 50% insider threat detection rate on R5.2 test data. Even after only 50 generations (20% of training data), population $\mathcal{P}_2$ was able to obtain 45% in malicious class detection rate. In general, these results show that retraining a Linear GP population ($\mathcal{P}$) with techniques for adaptation to R5.2 are suitable for obtaining better results than a population trained from scratch on R5.2 ($\mathcal{P}_4$), using less data and training time.

On the three GP populations that evolve from $\mathcal{P}$ on R5.2, $\mathcal{P}_1$ gives the worst results. Given enough training time, $\mathcal{P}_4$ was also able to surpass $\mathcal{P}_1$ (after 200 generations). Thus, it is clear that using only legacy features ($\mathcal{F}$) of new data may result in inability to learn new information (introduced in $\mathcal{F}_1 \setminus \mathcal{F}$). This, in return, hampers the detection of malicious behaviours in the new data. On the other hand, comparing results of $\mathcal{P}_2$ and $\mathcal{P}_3$, which evolve on R5.2 based on methods presented in § III-B2, it is shown that $\mathcal{P}_2$ achieved marginally better results overall. In this case, it appears that introducing bias to increase the usage of newly introduced features (as in $\mathcal{P}_3$) does not improve results over simply training GP normally with all features in $\mathcal{F}_1$. While this may indicate that GP with expanded input register space was able to incorporate newly introduced features well enough, better designed schemes to allow GP evolving on unseen feature spaces may be needed to further improve the performance.

Finally, results of the GP populations on test part of R5.2 (Table III) are slightly different from that observed on training R5.2 data (Figure 1). Different distribution of malicious behaviours in training and testing R5.2 data (Table I), especially in insider threat scenario 4, is likely the reason. Under the binary classification setting, where all malicious classes are handled in the same manner, it is harder to overcome this issue. In the next experiment, Linear GP is run under multi-class classification setting to examine its ability to learn from an expanded output space condition (new behaviours).

B. Experiment 2 - Evolving GP to recognize new malicious behaviours

This experiment is aimed at studying the evolution of a GP population – which has been trained on R4.2 – under output space expansion in R5.2. Based on the results of the previous experiment, the GP populations in this experiment are trained on all feature set $\mathcal{F}_1$ of R5.2, without feature bias, for up to 300 generations. There are 4 populations in this experiment, $\mathcal{Q}_1$, $\mathcal{Q}_2$, $\mathcal{Q}_3$, and $\mathcal{Q}_4$. $\mathcal{Q}_{1-3}$ are evolved from population $\mathcal{Q}$ that was trained on R4.2, and $\mathcal{Q}_4$ is trained from scratch on R5.2. Population $\mathcal{Q}$ achieved 98.37% normal DR and 100%, 47.86%, and 45.83% DRs on three insider threat scenarios, respectively. Based on that, $\mathcal{Q}_{1-3}$ is trained on R5.2 with expanded output register vectors to accommodate the the appearance of new class in R5.2, insider threat scenario 4. Population $\mathcal{Q}_1$ assumes a normal training process without any bias. $\mathcal{Q}_2$ and $\mathcal{Q}_3$ training

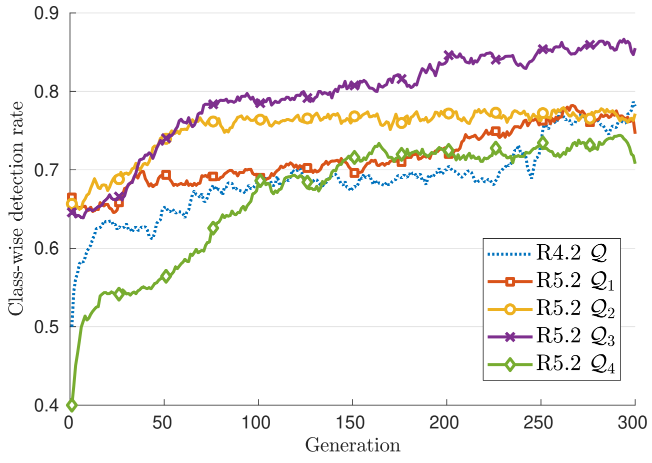

Fig. 2. Experiment 2 Class-wise detection rates of the populations on training subsets by the number of generations

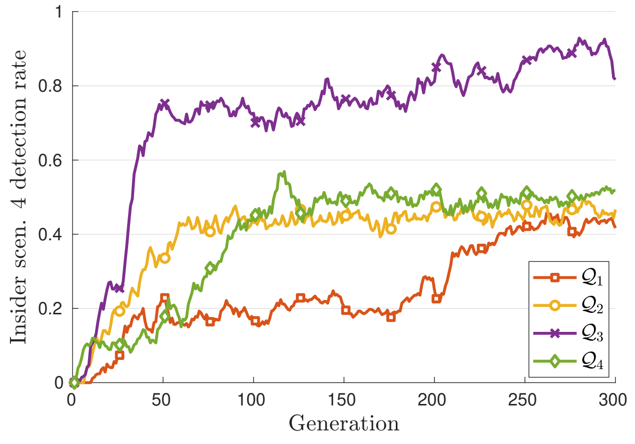

Fig. 3. Experiment 2 Insider threat scenario 4 detection rates of the populations on training subsets by the number of generations

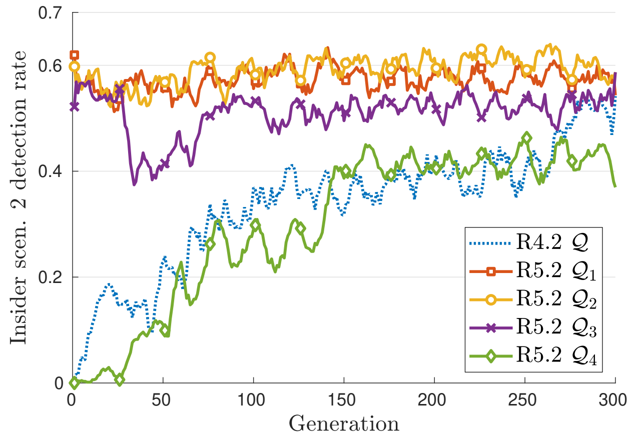

Fig. 4. Experiment 2 Insider threat scenario 2 detection rates of the populations on training subsets by the number of generations

TABLE IV
EXPERIMENT 2 RESULTS OF GP POPULATIONS TRAINED WITH DIFFERENT NUMBER OF GENERATIONS ON R5.2 TESTING PART

# generation	Population	Normal DR	Insider threat DRs			
			Scenario 1	Scenario 2	Scenario 3	Scenario 4
0	$\mathcal{Q}, \mathcal{Q}_{1-3}$	98.32 ± 0.52	96.43 ± 2.69	46.17 ± 11.97	49.69 ± 21.07	0.00 ± 0.00
20	$\mathcal{Q}_1$	98.73 ± 0.52	98.93 ± 2.83	34.51 ± 13.80	57.76 ± 20.73	7.83 ± 20.72
	$\mathcal{Q}_2$	98.70 ± 0.68	100.0 ± 0.00	31.09 ± 16.83	56.52 ± 18.04	6.67 ± 17.64
	$\mathcal{Q}_3$	98.58 ± 0.61	100.0 ± 0.00	41.47 ± 13.22	49.07 ± 23.78	18.13 ± 13.87
	$\mathcal{Q}_4$	99.74 ± 0.08	100.0 ± 0.00	0.22 ± 0.59	46.58 ± 23.17	14.50 ± 19.20
50	$\mathcal{Q}_1$	98.78 ± 0.35	98.93 ± 1.00	33.87 ± 11.41	62.11 ± 20.29	20.86 ± 18.47
	$\mathcal{Q}_2$	98.51 ± 0.56	99.64 ± 0.33	39.67 ± 07.05	56.52 ± 21.00	38.11 ± 14.35
	$\mathcal{Q}_3$	98.67 ± 0.47	100.0 ± 0.00	28.43 ± 16.00	55.28 ± 18.74	46.75 ± 2.09
	$\mathcal{Q}_4$	99.65 ± 0.10	100.0 ± 0.00	0.00 ± 0.00	52.17 ± 10.24	21.59 ± 19.57
100	$\mathcal{Q}_1$	98.61 ± 0.45	100.0 ± 0.00	41.64 ± 9.40	62.11 ± 14.02	15.39 ± 22.37
	$\mathcal{Q}_2$	98.49 ± 0.39	100.0 ± 0.00	38.00 ± 11.04	65.22 ± 12.20	33.63 ± 12.25
	$\mathcal{Q}_3$	98.64 ± 0.29	100.0 ± 0.00	35.87 ± 9.78	64.60 ± 15.28	47.15 ± 3.50
	$\mathcal{Q}_4$	99.46 ± 0.18	100.0 ± 0.00	3.27 ± 3.06	58.38 ± 12.42	38.14 ± 4.96
200	$\mathcal{Q}_1$	98.69 ± 0.35	100.0 ± 0.00	33.26 ± 6.60	68.32 ± 9.63	20.93 ± 19.36
	$\mathcal{Q}_2$	98.45 ± 0.53	100.0 ± 0.00	36.39 ± 18.07	65.22 ± 14.49	39.46 ± 14.87
	$\mathcal{Q}_3$	98.27 ± 0.21	100.0 ± 0.00	48.19 ± 11.74	67.08 ± 13.63	46.74 ± 6.25
	$\mathcal{Q}_4$	98.98 ± 0.45	100.0 ± 0.00	18.00 ± 12.94	60.87 ± 13.89	41.79 ± 13.16
300	$\mathcal{Q}_1$	98.83 ± 0.36	100.0 ± 0.00	30.06 ± 3.97	69.57 ± 10.45	26.75 ± 15.57
	$\mathcal{Q}_2$	98.16 ± 0.51	99.64 ± 0.94	49.60 ± 13.91	69.56 ± 14.49	41.21 ± 16.28
	$\mathcal{Q}_3$	98.30 ± 0.46	100.0 ± 0.00	41.57 ± 16.28	61.49 ± 17.47	50.53 ± 5.24
	$\mathcal{Q}_4$	98.88 ± 0.40	100.0 ± 0.00	31.21 ± 18.21	65.22 ± 4.07	41.56 ± 12.91

follows method (i) and (ii) in § III-B3, respectively. The bias towards the newly introduced class is controlled by $G_f = 50$ and $\alpha_0 = 2$. Furthermore, $\mathcal{Q}_2$ assumes a cost matrix, where the cost of misclassifying a normal instance and a malicious class $m \in \mathcal{C}$ are $E_{\text{Normal}} = 1$ and $E_m = 10$, respectively.

The evolution of the populations, measured by class-wise DR, over R5.2 and R4.2 training subsets are presented in Figure 2. Table IV shows Linear GP results on testing part of R5.2 after 0, 20, 50, 100, 200, and 300 generations.

Similar to experiment 1, pre-trained GP populations, $\mathcal{Q}_{1-3}$, show better results over a population training from scratch, $\mathcal{Q}_4$. At 0 generation, as shown in Table IV, $\mathcal{Q}_{1-3}$ is already able to obtain fairly good DRs on the first 3 malicious insider scenarios, which is inherited from $\mathcal{Q}$. Hence, on R5.2, $\mathcal{Q}_{1-3}$ only need to maintain performance on insider threat scenarios 1-3 while learning to detect the newly introduced scenario. Overall, the best results achieved on R5.2 belongs to population $\mathcal{Q}_3$, which is trained using method (ii) in III-B3, where accuracy, class-wise DR, and insider threat scenario 4 DR are objectives being swapped in the first 50 generations.

In this experiment, introducing bias in selection operator (via cost matrix or swapping objective mechanism) indeed enables the population to learn from the new class better. As shown in Figure 3, at $G_c = 50$ generations, $\mathcal{Q}_2$ and $\mathcal{Q}_3$ achieved much better scenario 4 DR than $\mathcal{Q}_1$ and $\mathcal{Q}_4$. Especially on $\mathcal{Q}_3$, it seems that the swapping objective mechanism puts the highest pressure toward detection of the new class. Hence, when the bias is removed at the 50th generation, the performance on the new malicious insider scenario reaches the highest point using population $\mathcal{Q}_3$. On the other hand, simply retraining a pre-trained population without taking care of the new class, as in $\mathcal{Q}_1$, may prohibit it from learning to recognize the new class (Figure 3). In effect, useful innovations that improve the new class may hurt the overall performance initially, and hence fail to compete and remain in the population.

On the three remaining malicious classes, results of $\mathcal{Q}_{1-3}$ on two insider threat scenarios, 1 and 3, remain stable over training generations. On the other hand, although the same initial scenario 2 detection rate was inherited from $\mathcal{Q}$, the populations showed different performances on this scenario over training generations R5.2 (Figure 4). Population $\mathcal{Q}_3$ initially sees some decrease to insider scenario 2, as it puts high priority in learning to detect insider scenario 4. However, $\mathcal{Q}_3$'s scenario 3 performance quickly recovers after the bias mechanism stops at generation 50.

Finally, similar to experiment 1, there are discrepancies in R5.2 training results (Figures 2, 3, 4) and results on R5.2 testing data Table IV. A possible explanation of this is that not only the distribution of malicious behaviours is different (Table I), but also temporal variations in user behaviours may exist in R5.2 training and testing datasets. In that case, a stream learning method as in [9] may be employed to better adapt the GP population to temporal variations in the data.

C. Comparison to other machine learning algorithms

Popular non-evolutionary machine learning techniques - Random forest (RF) [28], Logistic regression (LR), and Multi-layer perceptron (MLP) - are employed in this work for com-

TABLE V
RESULTS OF LEARNING ALGORITHMS ON CERT DATASETS WITH DIFFERENT TRAINING AND TESTING DATA

Algorithm	Training data	Testing data	Normal DR	Insider threat DR
LR	R4.2	R4.2	99.83	17.09
RF			100	16.39
MLP			99.78	26.39
LGP			98.39	51.50
LR	R4.2	R5.2	99.76	10.42
RF			100	5.14
MLP			99.87	8.99
LR	R5.2	R5.2	99.74	26.4
RF			99.99	30.39
MLP			99.89	28.65
LGP, 100 gen.	R4.2, R5.2	R5.2	98.64	43.90
LGP, 300 gen.			98.30	48.14

parison with the proposed approach. The algorithms, which are widely used in cyber-security applications [25], provide a baseline of non-evolving methods for comparisons against GP [18]. More details on the foundations of these algorithms can be found in [29].

We used implementations of the algorithms in Scikit-learn on Python 3.7 [30]. While parameters of LR are left at default values, we used random search with cross-validation to adjust hyperparameters of RF and MLP. For RF, we tuned the number of individual decision trees (50, 100, and 200), the number of features available for training individual trees ($|\mathcal{F}|$, $\sqrt{|\mathcal{F}|}$, and $\log_2 |\mathcal{F}|$), and the maximum number of leaf nodes in a tree (100, 200, or unlimited). For MLP, Adam optimization algorithm [31] is used for training up to 250 epochs, and the number of hidden layers (1 to 3, each hidden layer has the size set to a half of the previous layer), batch size (32, 64, and 256), $l2$ penalty (10^{-4}, 10^{-2}, and 10^{-1}) are tuned using random search.

Table V presents results of Linear GP and other widely used machine learning algorithms, MLP, RF, LR, on CERT R4.2 and R5.2 datasets. Since it is not possible to retrain MLP, RF, or LR classifiers under both the feature space and the output space expansion conditions (at least using these code bases), test results of these algorithms are shown where the training and the testing datasets are in the same CERT release, or a trained model on R4.2 is used to test on R5.2. In the latter case, the models can work on only a subset ($\mathcal{F}$) of features in R5.2 ($\mathcal{F}_1$). These are done under a binary classification setting. Results of Linear GP on R5.2 in Table V are obtained from population $\mathcal{Q}_3$ in experiment 2.

Overall, there was a clear trade-off between detection performance on normal and malicious insider behaviours. Nevertheless, Linear GP showed much better malicious insider detection rates, while maintaining an acceptable accuracy. As demonstrated in the experiments, a clear advantage of GP is that a properly retrained GP population can adapt to changes in data, where new features and classes are introduced over time. Moreover, in this case, evolving a pre-trained GP population on a new CERT dataset needs only 20% to 40% of training data to achieve good results. This greatly reduces the computational overheads and expedites the deployment of insider threat detection under the new conditions.

V. CONCLUSIONS AND FUTURE WORK

In this work, we examined the application of Linear GP in dynamic insider threat detection task, specifically under feature space and output space expansion conditions. Exploiting unique characteristics of Linear GP, where registers are used for data feature input and class output, and the nature of a population-based method, we applied different techniques to allow a trained GP population to evolve on updated versions of previously used training data, where new features and new classes are introduced. Experiments are performed on multiple releases of a publicly available dataset of enterprise insider threats. Results show that an appropriate bias can be introduced to the variation operators to assist the incorporation of newly introduced information in expanded feature and output spaces. Thus, the trained Linear GP populations can be successfully evolved to the new working environments. This greatly reduces the requirements for computational cost and the number of training examples for a working machine learning model under evolving (changing) conditions. This, then, allows quicker redeployment of the system under new environments. Furthermore, comparisons against other machine learning methods that are widely used in cyber-security show that not only the results that are achieved by the Linear GP population are comparable, but also the method has the advantage of being evolvable and adaptable to dynamics in deployment environments. This is crucial in corporate deployments, while still maintaining good performances under different conditions.

Future work will investigate different and more sophisticated techniques addressing the changes in feature and output spaces, for example task transfer. The use of learning algorithms for streaming data will also be investigated for providing an online process for adapting to change.

ACKNOWLEDGMENT

This research is supported by Natural Science and Engineering Research Council of Canada (NSERC). Duc C. Le gratefully acknowledges the support by Killam Trusts, Mitacs, and Nova Scotia Research and Innovation funding programs. The research is conducted as part of the Dalhousie NIMS Lab at: https: //projects.cs.dal.ca/projectx/.

REFERENCES

[1] M. L. Collins, M. C. Theis, R. F. Trzeciak, J. R. Strozer, J. W. Clark, D. L. Costa, T. Cassidy, M. J. Albrethsen, and A. P. Moore, "Common sense guide to mitigating insider threats, fifth edition," The CERT Insider Threat Center, Tech. Rep. CMU/SEI-2015-TR-010, 2016.
[2] CSO, U.S. Secret Service, CERT Division of SEI-CMU, KnowBe4, "The 2018 U.S. state of cybercrime survey," IDG, Tech. Rep., 2018. [Online]. Available: https://www.idg.com/tools-for-marketers/2018-u-s-state-of-cybercrime/
[3] Crowd Research Partners, "2018 insider threat report," 2018. [Online]. Available: https://crowdresearchpartners.com/insider-threat-report/
[4] M. I. Heywood, "Evolutionary model building under streaming data for classification tasks: opportunities and challenges," *Genetic Programming and Evolvable Machines*, vol. 16, no. 3, pp. 283–326, 2015.

[5] D. Song, M. I. Heywood, and A. N. Zincir-Heywood, "Training genetic programming on half a million patterns: an example from anomaly detection," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 225–239, June 2005.

[6] F. Haddadi, D. Runkel, A. N. Zincir-Heywood, and M. I. Heywood, "On botnet behaviour analysis using GP and C4.5," in *Proceedings of the Companion Publication of the 2014 Conference on Genetic and Evolutionary Computation*, 2014, pp. 1253–1260.

[7] S. Sen and J. A. Clark, "Evolutionary computation techniques for intrusion detection in mobile ad hoc networks," *Computer Networks*, vol. 55, no. 15, pp. 3441 – 3457, 2011.

[8] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Applied Soft Computing*, vol. 10, no. 1, pp. 1 – 35, 2010.

[9] D. C. Le, S. Khanchi, A. N. Zincir-Heywood, and M. I. Heywood, "Benchmarking evolutionary computation approaches to insider threat detection," in *Proceedings of the ACM Genetic and Evolutionary Computation Conference*, ser. GECCO '18, 2018, pp. 1286–1293.

[10] W. Eberle, J. Graves, and L. Holder, "Insider threat detection using a graph-based approach," *Journal of Applied Security Research*, vol. 6, no. 1, pp. 32–81, 2011.

[11] P. Parveen and B. Thuraisingham, "Unsupervised incremental sequence learning for insider threat detection," in *IEEE International Conference on Intelligence and Security Informatics*, 2012, pp. 141–143.

[12] T. E. Senator, E. Chow, I. Essa, J. Jones, V. Bettadapura, D. H. Chau, O. Green, O. Kaya, A. Zakrzewska, E. Briscoe, R. I. L. Mappus, H. G. Goldberg, R. McColl, L. Weiss, T. G. Dietterich, A. Fern, W.-K. Wong, S. Das, A. Emmott, J. Irvine, J.-Y. Lee, D. Koutra, A. Memory, C. Faloutsos, D. Corkill, L. Friedland, A. Gentzel, D. Jensen, W. T. Young, B. Rees, R. Pierce, D. Huang, M. Reardon, and D. A. Bader, "Detecting insider threats in a real corporate database of computer usage activity," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2013, pp. 1393–1401.

[13] T. Rashid, I. Agrafiotis, and J. R. Nurse, "A new take on detecting insider threats," in *Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats*, 2016, pp. 47–56.

[14] H. Eldardiry, E. Bart, J. Liu, J. Hanley, B. Price, and O. Brdiczka, "Multi-domain information fusion for insider threat detection," in *Proceedings of the 2013 IEEE Security and Privacy Workshops*, 2013, pp. 44–51.

[15] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," in *Proceedings of the AAAI-17 Workshop on Artificial Intelligence for Cyber Security*, 2017, pp. 224–231.

[16] D. C. Le and A. N. Zincir-Heywood, "Evaluating insider threat detection workflow using supervised and unsupervised learning," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy Workshops*, 2018, pp. 270–275.

[17] G. Gavai, K. Sricharan, D. Gunning, J. Hanley, M. Singhal, and R. Rolleston, "Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 6, no. 4, pp. 47–63, December 2015.

[18] D. C. Le and A. N. Zincir-Heywood, "Machine learning based insider threat modelling and detection," in *IFIP/IEEE International Symposium on Integrated Network Management*, Washington DC, USA, 2019.

[19] L. Liu, O. De Vel, Q.-L. Han, J. Zhang, and Y. Xiang, "Detecting and Preventing Cyber Insider Threats: A Survey," *IEEE Communications Surveys & Tutorials*, pp. 1397–1417, 2018.

[20] S. Khanchi, A. Vahdat, M. I. Heywood, and A. N. Zincir-Heywood, "On botnet detection with genetic programming under streaming data label budgets and class imbalance," *Swarm and Evolutionary Computation*, vol. 39, pp. 123 – 140, 2018.

[21] F. Haddadi and A. N. Zincir-Heywood, "Botnet detection system analysis on the effect of botnet evolution and feature representation," in *Proceedings of the Companion Publication of the 2015 Conference on Genetic and Evolutionary Computation*, 2015, pp. 893–900.

[22] E. Manzoor, H. Lamba, and L. Akoglu, "xstream: Outlier detection in feature-evolving data streams," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1963–1972.

[23] M. Brameier and W. Banzhaf, *Linear Genetic Programming*, ser. Genetic and Evolutionary Computation. Boston, MA, USA: Springer US, 2007.

[24] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *Proceedings of the 2013 IEEE Security and Privacy Workshops*, 2013, pp. 98–104. [Online]. Available: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=509801

[25] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1153–1176, Secondquarter 2016.

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

[27] *MATLAB version 9.5.0.944444 (R2018b)*, The Mathworks, Inc., Natick, Massachusetts, 2018.

[28] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, 2001.

[29] E. Alpaydin, *Introduction to Machine Learning*. Cambridge, MA, USA: The MIT Press, 2014.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, Nov. 2011.

[31] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.