Hierarchical Data Topology Based Selection for Large Scale Learning

Hmida Hmida, Sana Ben Hamida, Amel Borgi, Marta Rukoz

Hierarchical Data Topology Based Selection for Large Scale Learning

Hmida Hmida*†, Sana Ben Hamida†‡, Amel Borgi* and Marta Rukoz†‡
*Université Tunis El Manar, LIPAH, Tunis, Tunisia
†Université Paris Dauphine, PSL Research University, CNRS, UMR[7243], LAMSADE, 75016 Paris, France
‡Université Paris Ouest Nanterre La Défense, France

***Abstract*—The amount of available data for data mining and knowledge discovery continues to grow very fast with the era of Big Data. Genetic Programming algorithms (GP), that are efficient machine learning techniques, are face up to a new challenge that is to deal with the mass of the provided data. Active Sampling, already used for Active Learning, might be a good solution to improve the Evolutionary Algorithms (EA) training from very big data sets. This paper investigates the adaptation of Topology Based Selection (TBS) to face massive learning datasets by means of Hierarchical Sampling. We propose to combine the Random Subset Selection (RSS) with the TBS to create the RSS-TBS method. Two variants are implemented and applied to solve the KDD intrusion detection problem. They are compared to the original RSS and TBS techniques. The experimental results show that the important computational cost generated by original TBS when applied to large datasets can be lightened with the Hierarchical Sampling.**

I. Introduction

Genetic Programming (GP) [1] is a meta heuristic that has been used to solve a wide variety of machine learning problems. Applied for many classification problems, it has made a very competitive results with other approaches like Neural Networks, Decision Trees, etc. Nevertheless, this Evolutionary Algorithm (EA) suffers from an increased computational cost induced mainly by the evaluation step.

Applied to supervised learning and classification problems, GP generates a population of classifiers (genetic programs) by means of genetic operators. The performance of each classifier is measured by a fitness function that needs the evaluation (execution) of every program against the complete training dataset. This leads to a computational overhead increased specially with large datasets. It is proportional to the product of the number of instances in the dataset, the population size and the number of generations that will be carried out during the evolution process. Otherwise, using the full learning data might be impossible when the input data doesn't fit within the main memory which is often the case with Big Datasets.

To alleviate the number of evaluations, sampling the training set has been successfully adopted in several works [2]–[6]. The proposed approaches vary from simple methods based on a random data selection to complex methods using graphs or hierarchical techniques. Topology Based Selection (TBS) [6] is an active data selection algorithm[1] that keeps in the sampled subset the cases that are quite different according to a similarity graph constructed during the learning process. It was applied to solve a set of machine learning problems (thyroid problem, function approximation and the two intertwined spirals problem) and compared to the random sampling and the weighted sampling [9]. It was shown that TBS was able to improve the performance of genetic programming thanks to the data diversity control.

In this paper, we will try to adapt TBS to deal with problems involving large scale datasets using Hierarchical Sampling [2]. The aim is to mix TBS with Random Subset Selection (RSS) [3] in order to reduce the complexity and the computation cost while taking advantage of the strengths of this method.
Our purpose is to apply the TBS on reduced samples that are actively selected along evolution using random sampling or balanced sampling on the whole training data. The aim is to reduce the complexity thanks to the hierarchical strategy and, at the same time, allow the GP to maximize its generalization ability thanks to the data diversity enhanced by the TBS method.

This paper is organized as follows. Section II exposes the RSS, TBS and balanced sampling in addition to Hierarchical Sampling that inspired this work. A description of the proposed RSS-TBS method is given in Section III. The last section exposes intrinsic parameters used and discusses the obtained results under these experimental conditions.

II. Background

Two major classes of sampling techniques can be layed out: static sampling and dynamic sampling. With static sampling, the learner obtains all the input training set at once and stills unmodified across the learning process. This strategy is not efficient when learning from very large data sets, and it might not be applicable with big data sets. Dynamic sampling techniques are tightly related to the learning process evolution. Generally, it depends on a particular feature like unresolved or difficult cases, the number of generations carried out, among others. The presented methods are part of the dynamic sampling class.They are inspired by four techniques in this category: the Random Subset Selection (RSS), the Topologu Based Selection(TBS), the Hierarchical Selection and the Balanced Sampling. These techniques are presented in details in the following section.

[1]derived from *Active Learning* [7], [8]: 'any form of learning in which the learning program has some control over the inputs on which it trains.'

A. Random Subset Selection (RSS)

The selection of fitness cases is based on a uniform probability among the training subset. This stochastic selection helps to reduce any bias within the full dataset on evolution. In *Random Subset Selection* **RSS** [9], at each generation g, the probability of selecting any case i is equal to $P_i(g)$:

$$\forall i : 1 \leq i \leq T, \quad P_i(g) = \frac{S}{T}. \tag{1}$$

where T is the size of the full dataset and S is the target subset size. The sampled subset have a fluctuating size around S. Fixed Random Selection (**FRS**) [5] is a very similar method with a fixed number of cases selected at every generation.
Stochastic Sampling **SS** [10] is another method using the same probabilistic selection to construct subsets for each individual per generation.

The single parameter of this category of sampling is the subset size, whether a crisp or flexible target, it is set like other GP parameters (population size, crossover probability, etc.) and most likely by several GP runs.

B. Hierarchical Sampling

This kind of sampling is based on multiple levels of sampling methods inspired by the concept of a memory hierarchy. It combines several sampling algorithms that are applied in different levels.

Robert Curry and Malcom Heywood conceived an extension to Gathercole's Dynamic Subset Selection algorithm (DSS) [9] into a 3 levels hierarchy [12]. At level 0, the dataset was first partitioned into blocks that were sufficiently small to reside within RAM alone. Then, at level 1, blocks were then chosen from this partition based on RSS or DSS. Finally, at level 2, the selected block is considered as the full dataset on which DSS was applied for several rounds. Depending on the level 1 algorithm, we have **RSS-DSS** hierarchy or **DSS-DSS** hierarchy. Besides DSS parameters, new parameters are added: level 0 block size, level 1 number of iterations, level 2 iterations, maximum level 2 iterations and tournament iterations. Only level 2 iterations is calculated by:

$$I_b(i) = I_{max} * E_b(i-1). \tag{2}$$

$I_b(i)$: number of level 2 iterations on block b in i^{th} level 1 iteration.
I_{max}: maximum level 2 iterations.
$E_b(i-1)$: error rate over block b for the best case at previous iteration.
A modified DSS in level 2 is used where two roulette wheels exist per block, one is used to control the selection of patterns with respect to age and the other to difficulty, the roulette wheels being selected in proportion to the corresponding probability for age and difficulty (2 additional parameters).
For the DSS-DSS hierarchy, block weight and block selection probability are defined in [12].
Tested against KDD-99 [13] and Adult dataset [12], both algorithms realize competitive results in very shorter time. DSS-DSS outperforms RSS-DSS in these experiments.

An extension to this work is made in [2] with the *Balanced Block DSS* by altering mainly the level 0 block selection with the goal of obtaining a balanced block in level 1 with respect to a fixed ratio. Experimental results showed that Balanced Block algorithm is able to approach the classification performance of the CasGP[2] algorithm, whilst retaining the computational speedup of the original RSS-DSS algorithm.

C. Topology Based Selection

Inspired by the idea that structure influences the efficiency of heuristics working on it, this method consists at building a topology of the problem from the knowledge acquired by individuals about fitness cases. Lazarczyk [6] suggests a *Fitness case Topology Based Sampling* **TBS** in which relationship between fitness cases in the training set is represented by an undirected weighted graph. Vertices are fitness cases and edges have a weight measuring a similarity or a distance induced from individuals' performance. Then, cases having a tight relationship with respect to a threshold cannot be selected together in the same subset assuming that they have an equivalent difficulty for the population. Edge values start at 0 and are updated after evaluation phase by increasing the weight of all edges between solved cases by each individual and decreasing the weight of all edges by a loss rate λ.
This algorithm uses a unique subset for all the individuals renewed each generation after updating the topology graph.This subset is constructed by selecting randomly a case from a candidates set (initially equal to full dataset) until reaching the target size or the candidates set is empty. All fitness cases connected to it with an edge weight exceeding the threshold are removed from candidates.
The threshold is calculated during evolution process, using a binary-search like algorithm, to have a value adapted to the current topology: it must not be too high or too low.

Experimental results on classification problems (intertwined spirals [14] and the thyroid problems) demonstrated improvements in mean fitness value through generations compared to DSS and SSS. However, the additional computational cost of TBS caused by the threshold calculation algorithm and the topology update has not been measured. Moreover, this method has not been tested on a large dataset.

D. Balanced sampling

Balanced sampling [15] is a method aiming to improve classifiers accuracy by correcting the original dataset imbalance within majority and minority class instances. It has some methods based on the minority class size and thus reduce the number of instances like the methods studied in this paper. The following sampling methods are used with GP:

- **Static Balanced Sampling:** is a static sampling method that selects cases with uniform probability from each class without replacement until obtaining a balanced subset with equal number of majority and minority class instances of the desired size at every generation.

[2]Cascaded GP: uses the RSS-DSS algorithm to provide the basis for building cascades of GP classifiers, up to a predefined number of layers

- **Basic Under-sampling:** select all minority class instances and then an equal number from majority class randomly. This method produces balanced subsets having a reduced size especially when the gap between minority and majority classes is very important. We integrated this method in our approach to take advantage of its contribution to reduce the sample size, while keeping a balanced sample (see Section III-B).
- **Basic Over-sampling:** select all majority class instances and then an equal number from minority class randomly with replacement.
- **Under-sampling A:** Multiple balanced samples are created at each generation with Basic Under-sampling method. At every generation, all individuals are evaluated against all of the sample sets and are attributed the average fitness across all the sample sets.
- **Under-sampling B:** Identical to Under-sampling A but uses the minimum fitness instead of the average.
- **Over-sampling A:** Creates several balanced samples using Basic Over-sampling. The final fitness is the average on all subsets.
- **Over-sampling B:** Uses the same sampling method as Over-sampling A and the minimum fitness.

III. Hierarchical Sampling with TBS

GP based learning algorithms using TBS are limited to those in the original paper [6] where training set does not exceed few thousands of cases. TBS accelerates evolutionary search with a high diversity and good average fitness of the evolved population. This sampling method relies on a complete undirected graph having its edges updated after each evaluation phase. It is expected that the cost of maintaining this topology for a high volume of datasets will be very important. In this section, we propose a method inspired by the Hierarchical Sampling to overcome this additional cost by combining the use of RSS, TBS and Basic Under-Sampling (BUSS). The main idea is to apply TBS on different training sets with reduced size that are selected randomly from a fixed set of blocks. Blocks are created proportionally to the training set (the RSS-TBS method) or balanced according to BUSS (the BUSS-RSS-TBS method). We describe below the two methods in details.

A. RSS-TBS

At level 0, the full training set is first partitioned into blocks with a fixed size and then saved into hard disk. The number of cases per class is proportional to the initial training set. Then, at level 1, RSS is applied to randomly select a block. At the last level, previously selected block is sampled with TBS.
The general algorithm is shown below (Algorithm 1).

At step 13, the number of TBS iterations is calculated using Equation 2 in which the error rate is for the actual best individual of the run.
We used a modified TBS that calculates the threshold through the steps described by Algorithm 2

Algorithm 1 RSS-TBS

```
1: Divide dataset into blocks {level 0}
2: repeat {level 1 iterations}
3:    Conduct Block Selection {level 1 algorithm: RSS}
4:    repeat {level 2 iterations <= maximum iterations}
5:       Conduct Subset Selection {level 2 algorithm: TBS}
6:       repeat {Tournament iterations}
7:          Conduct tournament selection
8:          Train tournament individuals on Subset
9:       until Tournament End
10:      Update topology graph
11:      Apply genetic operators
12:   until level 2 iterations = calculated in 13
13:   Update level 2 iterations to be performed on this block
      at next selection
14: until level 1 end
```

Algorithm 2 Threshold calculation

```
1: Use the lowest edge weight as the initial threshold to have
   the smallest possible subset
2: repeat
3:    Apply TBS selection using current threshold on the
      remaining patterns
4:    Remove the selected patterns
5:    if the subset size is too small then
6:       Threshold value ← next edge weight
7:       Previously excluded patterns are added to candidates
8:    end if
9: until target size reached
```

B. BUSS-RSS-TBS

This method differs from the previous one by level 0 block creation. In fact, BUSS is performed at this level and then block size becomes calculated according to minority class.
KDD-99 have a normal class and 4 attack classes. The subset selection uses the U2R attack (52 patterns) class as minority class and then randomly selects an equal number of patterns from the remaining 3 attack classes gathering 208 attack patterns. In a first experiment, 52 normal were added obtaining 260 patterns in a single block. In the second experiment, 208 normal patterns were chosen to get a final level 0 block of 416 patterns. RSS and TBS are then applied in the same way as in Section III-A

IV. Experiments and Results

A. Learning Data

We used here a large dataset, which was firstly created for the competition held at the Fifth International Conference on Knowledge Discovery and Data Mining KDD-99 and is about intrusion detection problem. The original training dataset contains 5 million lines conveying data about connections from the simulated traffic and their labels as Normal connections and four attack classes. Another set, called corrected set is constructed with the same way and used as test dataset. The

TABLE I
KDD-99 DATASETS COMPOSITION

Class	Number of Patterns	
	10% Training Set	*Test Set*
Normal	97278	60593
Dos	391458	229853
probe	4107	4166
R2L	1126	16347
U2R	52	70
Total Attacks	396743	250436
Total Patterns	494021	311029

training process is run on the derived 10% KDD-99 dataset to learn how to classify normal connections and attacks. The best individual of each run is then tested on the test set. Both datasets are presented in Table I.

B. Cartesian GP (CGP)

CGP is a form of GP, proposed by Julian Miller [16], that represents genetic programs as directed acyclic graphs and mostly inspired by digital circuits called FPGA. In this graph, nodes representing functions have many inputs and one output, and are layed into two dimensional matrix very similar to neural networks. This form of GP has the following advantages [17]:

- Unlike trees, it allows more than one path between any pair of nodes permitting the reuse of previous results.
- Highly competitive with other GP methods.
- Does not bloat.
- Is easy to implement.
- Can have multiple outputs and then solve many types of problems.

We use here a CGP implementation done by David Oranchak [17] as a contribution package to Sean Luke's ECJ [18]. ECJ is an open source evolutionary computation framework implementing evolutionary algorithms like GP, Evolution Strategies, Genetic Algorithms etc., and offering the most complete range of GP representations (Koza, linear, grammatical, etc.).

1) CGP parameters: The implementation of CGP involves different parameters that determine its efficiency. The final design of CGP parameters used in this work is summarized in Table II.

2) Terminal and function sets: The terminal set includes input variables which are the 41 KDD-99 features set with 8 randomly generated constants.
The function set includes basic arithmetic, comparison and logical operators reaching 17 functions (see Table III).

C. Performance Metrics

By the end of each run, the best individual based on the fitness function is tested on the whole dataset and then on the test dataset. Results are recorded in a confusion matrix from

TABLE II
CGP PARAMETERS

Parameter	Value
Population size	512 for RSS, and RSS-TBS 128 for TBS
Subpopuations number	1
Number of generations	500 for RSS 100 for TBS depending on the RSS iterations for RSS-TBS
CGP nodes	300
Inputs/Outputs	49/1
Tournament size	4
Crossover/Mutation probability	0.9/0.04
Fitness	Minimize classification errors

TABLE III
TERMINAL AND FUNCTION SETS

Function (node) set	
Arithmetic operators:	$+, -, *, \%$
Comparison operators:	$<, >, <=, >=, =$
Logic operators:	AND, OR, NOT, NOR, NAND
Other :	NEGATE, IF (IF THEN ELSE), IFLEZE(IF $<=0$ THEN ELSE)
Terminal set	
KDD-99 Features	41
Ephemeral Random Constants	8 in $[-2, 2[$

which Accuracy, True Positive Rate (TPR) and False Positive Rate (FPR) are calculated.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ patterns}. \quad (3)$$

$$TPR = \frac{True\ Positives}{True\ Positives + False\ Negatives}. \quad (4)$$

$$FPR = \frac{False\ Positives}{False\ Positives + True\ Negatives}. \quad (5)$$

The training time is the amount of elapsed seconds between the first and the last generation (see Table II for the number of generations).

D. Specific Parameters for Sampling Methods

The values of additional parameters brought by each sampling method are assigned with respect to the original method paper. Some of theses methods have been modified in order to fit in a comparable context. The Table IV shows each method configuration.

E. Experimental Results

Experiments are performed on an Intel $i7 - 4810MQ$ $(2.8GHZ)$ workstation with $8GB$ RAM running under Windows $8.164 - bit$ Operating System. GP programs are trained to distinguish between normal and attack patterns.

To compare the performance of the implemented sampling methods, six measures are recorded cross the 21 runs performed for each technique: the best and the mean values of the accuracy, TPR and FPR metrics for both training and test datasets. The best measures are illustrated by the Figures 4, 5

TABLE IV
SPECIFIC METHODS' PARAMETERS

Method	Parameter	Value
RSS	Target Size	5000
TBS	Target Size	1000
	Loss Rate	0.7
RSS-TBS	Target Size(level 2 block size)	2500 then 100
	Level 0 block size	5000
	RSS iterations	200 then 500
	Max TBS iterations	20 then 50
	Individuals evaluated per TBS iteration	100 (20% of population size)
BUSS-RSS-TBS	Target Size(level 2 block size)	2500 then 100
	Level 0 block size	260 then 416 (calculated)
	RSS iterations	50 then 40
	Max TBS iterations	40 then 50
	Individuals evaluated per TBS iteration	100 (20% of population size)

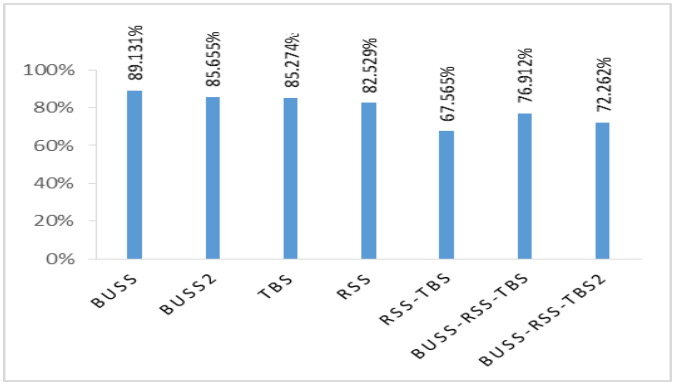

Fig. 1. Mean Accuracy

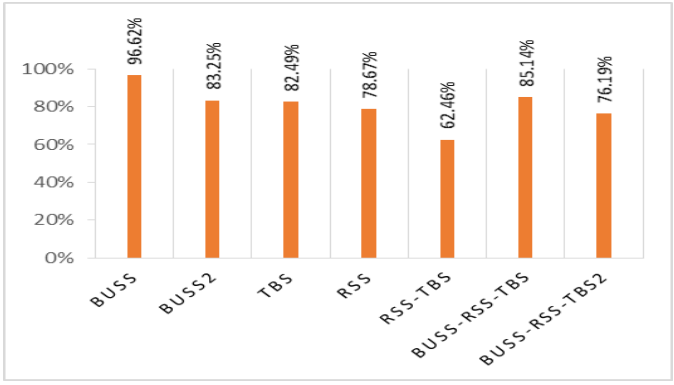

Fig. 2. Mean Recall (TPR)

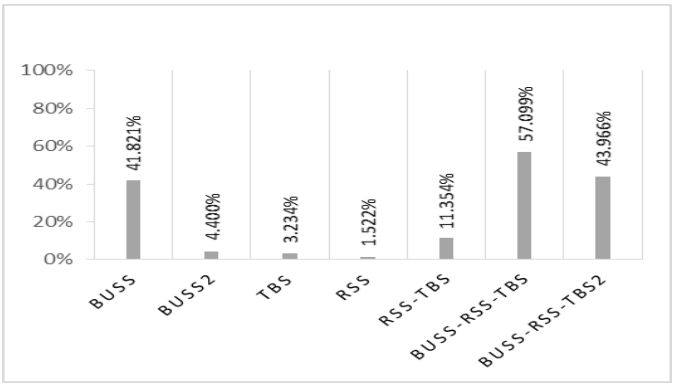

Fig. 3. Mean FPR

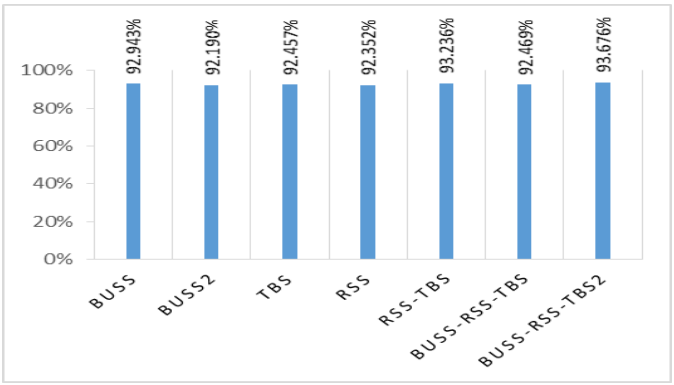

Fig. 4. Best Individual Accuracy

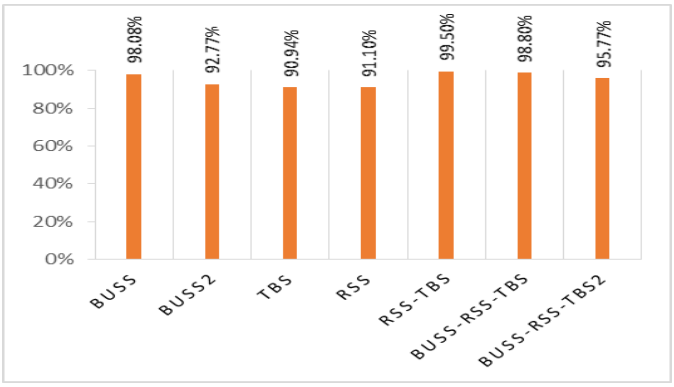

Fig. 5. Best Individual Recall (TPR)

and 6 whereas the mean values are presented in Figures 1, 2 and 3.

Otherwise, to compare the methods according to the computational cost, the spent time to complete each run and to find the best of run individual for all the trials are recorded. Table V illustrates the corresponding mean values for all implemented techniques.

From Table V, it is obvious that all three hierarchical sampling methods are more faster than TBS while having larger population and performing a greater number of generations due to the two level iterations. These methods also spend less time than RSS.

Meanwhile, it's noticeable that the best classifier is found very early according to the whole run. Regarding resulting best classifier quality (Figure 4) in terms of accuracy, RSS-TBS and BUSS-RSS-TBS (2 variants) merely outperforms RSS and TBS. The last two methods have very similar results. At the same time, recall and FPR increased remarkably for the three hierarchical methods. Whenever the second configuration of BUSS is used as in BUSS2 and BUSS-RSS-TBS2 the FPR

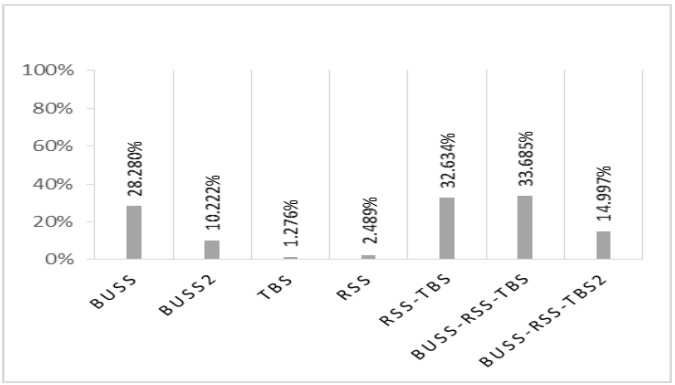

Fig. 6. Best Individual FPR

TABLE V
TIME MEASURES

Method	Mean Run Time	Mean 'Best of Run'Time
BUSS	75.612	20.135
BUSS2	102.428	43.054
TBS	11174.507	1494.977
RSS	1493.788	530.038
RSS-TBS	714.414	9.524
BUSS-RSS-TBS	157.485	1.712
BUSS-RSS-TBS2	236.029	4.687

falls down.

Another evidence, based on Figures 1,2 and 3 is that hierarchical sampling obtains the worst mean performance indicators (accuracy, recall and FPR) assuming that results are very divergent.

V. CONCLUSION

In this paper, we have shown the glaring computational cost of TBS when used within a supervised learning process using GP algorithm. Then we gave a solution to this problem by mounting a cascade of different sampling methods inspired by the RSS-DSS algorithm. In a first scenario, TBS was mixed with RSS and in a second one BUSS is deployed to build first level blocks.

Experimental results demonstrates this speed gain with RSS-TBS whilst reaching a slightly better accuracy than the original TBS and RSS. In addition to that BUSS registers an increase in FPR.

Though, the three hierarchical sampling methods suffer from an early convergence and could be handled with using a suitable Fitness Function. Moreover, in this paper we used a topology graph for each level 1 subset generated at each RSS iteration and thus the previous similarity measures is lost. Keeping a full topology graph could save some extra time.

REFERENCES

[1] J. R. Koza, "Genetic programming: On the programming of computers by means of natural selection," *Statistics and Computing*, vol. 4, no. 2, pp. 87–112, 1994.

[2] R. Curry, P. Lichodzijewski, and M. I. Heywood, "Scaling genetic programming to large datasets using hierarchical dynamic subset selection," *IEEE Transactions on Systems, Man, and Cybernetics: Part B - Cybernetics*, vol. 37, no. 4, pp. 1065–1073, 2007.

[3] C. Gathercole, "An investigation of supervised learning in genetic programming," Thesis, University of Edinburgh, 1998.

[4] H. Iba, "Bagging, boosting, and bloating in genetic programming," in *Proc. of the Genetic and Evolutionary Computation Conf. GECCO-99*. San Francisco, CA: Morgan Kaufmann, 1999, pp. 1053–1060.

[5] B.-T. Zhang and J.-G. Joung, "Genetic programming with incremental data inheritance," in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2. Orlando, Florida, USA: Morgan Kaufmann, 13-17 July 1999, pp. 1217–1224.

[6] C. W. G. Lasarczyk, P. Dittrich, and W. Banzhaf, "Dynamic subset selection based on a fitness case topology," *Evolutionary Computation*, vol. 12, no. 2, pp. 223–242, 2004.

[7] L. E. Atlas, D. Cohn, and R. Ladner, "Training connectionist networks with queries and selective sampling," in *Advances in Neural Information Processing Systems 2*. Morgan-Kaufmann, 1990, pp. 566–573.

[8] D. Cohn, L. E. Atlas, R. Ladner, and A. Waibel, "Improving generalization with active learning," in *Machine Learning*, 1994, pp. 201–221.

[9] C. Gathercole and P. Ross, "Dynamic training subset selection for supervised learning in genetic programming," in *Parallel Problem Solving from Nature - PPSN III*, ser. Lecture Notes in Computer Science, vol. 866. Springer, 1994, pp. 312–321.

[10] P. Nordin and W. Banzhaf, "An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming," *Adaptive Behaviour*, vol. 5, no. 2, pp. 107–140, 1997.

[11] C. Gathercole and P. Ross, "Small populations over many generations can beat large populations over few generations in genetic programming," in *Genetic Programming 1997: Proc. of the Second Annual Conf.* San Francisco, CA: Morgan Kaufmann, 1997, pp. 111–118.

[12] R. C. Curry and M. Heywood, "Towards efficient training on large datasets for genetic programming," *Lecture Notes in Computer Science*, vol. 866, no. Advances in Artificial Intelligence, pp. 161–174, 2004.

[13] UCI, "Kdd cup," 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/

[14] K. J. Lang and M. J. Witbrock, "Learning to tell two spirals apart," in *Proceedings of the 1988 Connectionist Models Summer School*. Morgan Kaufmann, 1988, pp. 52–59.

[15] R. Hunt, M. Johnston, W. N. Browne, and M. Zhang, "Sampling methods in genetic programming for classification with unbalanced data," in *Australasian Conference on Artificial Intelligence*, ser. Lecture Notes in Computer Science, vol. 6464. Springer, 2010, pp. 273–282.

[16] J. F. Miller and P. Thomson, "Cartesian genetic programming," in *Proceedings of the Third European Conference on Genetic Programming (EuroGP-2000)*, ser. LNCS, vol. 1802. Edinburgh, Scotland: Springer Verlag, 2000, pp. 121–132.

[17] CGP, "Cartesian gp website." [Online]. Available: http://www.cartesiangp.co.uk

[18] S. Luke, "Ecj homepage." [Online]. Available: http://cs.gmu.edu/~eclab/projects/ecj/