AN HETEROGENEOUS INFORMATION PROCESSING USING BIG DATA

Presented By
[1] R . NIVEDHA B.Tech, M.Tech
[1]Assistant Professor,
[1]Department of CSE,
[1]Golden Valley Integrated Campus
Andhra Pradesh

Presented By
[1] S . ARSHIYA SULTHANA B.Tech, M.Tech
[1]Assistant Professor,
[1]Department of CSE,
[1]Golden Valley Integrated Campus
Andhra Pradesh

Abstract— **Big Data though it is a hype up-springing many technical challenges that confront both academic research communities and commercial IT deployment, the root sources of Big Data are founded on data streams. It is generally known that data which are sourced from data streams accumulate continuously making traditional batch-based model induction algorithms infeasible for real-time data mining or high-speed data analytics in a broad sense. In this paper, a novel data stream mining methodology, called Stream-based Holistic Analytics and Reasoning in Parallel (SHARP) is proposed. SHARP is based on principles of incremental learning which span across a typical data-mining model construction process, from lightweight feature selection, one-pass incremental decision tree induction, and incremental swarm optimization. Each one of these components in SHARP is designed to function together aiming at improving the classification/prediction performance to its best possible. SHARP is scalable, that depends on the available computing resources during runtime, the components can execute in parallel, collectively enhancing different aspects of the overall SHARP process for mining data streams. It is believed that if Big Data are being mined by incrementally learning a data mining model, one pass at a time on the fly, the large volume of such big data is no longer a technical issue, from the perspective of data analytics. Three computer simulation experimentations are shown in this paper, pertaining to three components of SHARP, for demonstrating its efficacy.**

Keywords- Data stream mining methodology; Cache-based data stream classifier; CCV feature selection; Meta-heusristics

I. INTRODUCTION

Recently a lot of news in the media advocates the hype of Big Data that are manifested in three problematic issues. They are the 3V challenges known as: Velocity problem that gives rise to a huge amount of data to be handled at an escalating high speed; Variety problem that makes data processing and integration difficult because the data come from various sources and they are formatted differently; and Volume problem that makes storing, processing, and analysis over them both computational and archiving challenging.

In views of these 3V challenges, the traditional data mining approaches which are based on the full batch-mode learning may run short in meeting the demand of analytic efficiency. That is simply because the traditional data mining
model construction techniques require loading in the full set of data, and then the data are partitioned according to some divide-and-conquer strategy; two classical algorithms are CART decision tree induction [1] and Rough-set discrimination [2]. Each time when fresh data arrive, which is typical in the data collection process that makes the big data inflate to bigger data, the traditional induction method needs to re-run and the model that was built needs to be built again with the inclusion of new data.

In contrast, the new breed of algorithms known as data stream mining methods [3] are able to subside these 3V problems of big data, since these 3V challenges are mainly the characteristics of data streams. Data stream algorithm is not stemmed by the huge volume or high speed data collection. The algorithm is capable of inducing a classification or prediction model from bottom-up approach; each pass of data from the data streams triggers the model to incrementally update itself without the need of reloading any previously seen data. This type of algorithms can potentially handle data streams that amount to infinity, and they can run in memory analyzing and mining data streams on the fly. It is regarded as a killer method for big data hype and its related analytics problems. Lately researchers concur data stream mining algorithms are meant to be solutions to tackle big data for now and for the future years to come [4][5].

Although there are not short of algorithms in the computer science and machine learning areas for incremental learning, a holistic approach in summing up different aspects of data stream mining with the aim of improving the ultimate accuracy performance is needed. In this paper, we propose a novel data stream mining methodology. It is called Stream-based Holistic Analytics and Reasoning in Parallel (or SHARP in short) which is based on principles of incremental learning and lightweight processing. SHARP is comprised of several components which cover a typical data-mining model construction process. They are lightweight feature selection, one-pass incremental decision tree induction, and incremental swarm optimization. Each one of these components in SHARP is supposed to complement each other towards the common objective of improving the classification/prediction performance as a whole. SHARP is scalable in computation; additional CPUs can be included in parallel for increasing the execution threads of independent

performance enhancement. The benefits of SHARP include attaining the highest possible prediction accuracy while maintaining the computation as lightweight as possible; some components can be run independently thereby allowing parallel processing and scalable solution; and the operation of SHARP is in stochastic manner implying the longer it runs for, the better the performance it can achieve.

The remaining of the paper is structured as follow. Section 2 describes the SHARP methodology and its components in full details. Section 3 presents three computer simulation experimentations for validating the efficacy of SHARP components for data mining big data. Section 4 concludes the paper.

II. SHARP METHODOLOGY

To the best of the authors' knowledge there is no methodology for data stream mining in the academic literature, especially that covers an optimizer for fine-tuning the performance in real-time. A model of SHARP processes which aims to shed light in the methodology of data stream mining is shown in Figure 1. It includes several components that work cooperatively together during the data stream mining operation. The components are; 1. Cache Receiver (CR); 2. Incremental Classifier (IC); 3. Incremental Feature Selection Module (IFS); 4. Factor Analysis Module (FA); and 5. Swarm Optimizer (SO). The methodology offers a holistic approach which takes care of most if not all the possible aspects in data mining for improving performance. These five components are meant for achieving the following objectives respectively: CR- objective is to subside the problems of missing/incomplete data; IC- objective is to enable stream forecasting/prediction/classification by incremental learning manner; IFS and FA- objective is to understand the reasons and influences of the respective data attributes towards the predicted class; and SO- objective is to fine-tune the parameter values including selecting the optimal feature subset regularly. All these components contribute to the overall performance improvement, and they can function concurrently as the data stream in.

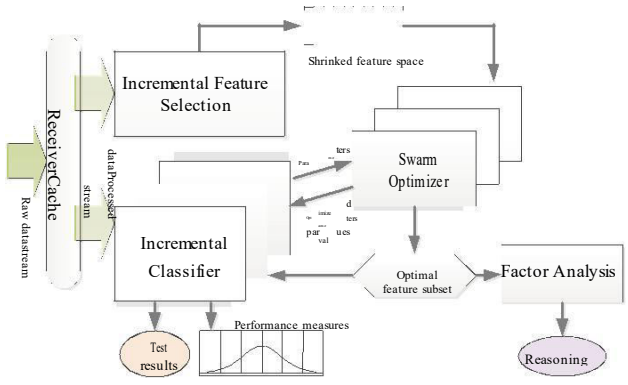

Figure 1: SHARP processes

The multiple rectangles in Figure 1 that represent IC and SO respectively, depict the possibility that these processes can run concurrently over some parallel computing devices.

The components are briefly described as follow. It is acknowledged that there exist many possible solutions or algorithms for implementing these components. Our discussion however highlights only some of the state-of-the-arts developed from our previous projects. The methodology serves as an abstract guideline on the possible integration of several data streaming components, discussing their functions (rather than implementation), interfaces and advantages.

A. Cache Receiver (CR)

The cache receiver is a front-end pre-processing mechanism that holds certain amount of data from the incoming data stream for a while. The main function is to minimize the latency of data arrivals as it is possible and likely that data streams that are being aggregated from various sources would be received at different speeds. CR acts as a delay regulator and buffer allowing opportunities for efficient data cleansing mechanism to operate upon, in real-time. It is not uncommon that data streams are stained with noise and incomplete information; techniques have been proposed and studied previously. The techniques [6] mainly centered on delivering and synchronizing the cache-buckets, estimating missing data, and detecting and alleviating concept-drift problems etc. CR also handles other basic data pre-processing tasks similar to those for traditional data mining in the KDD process. Data stream is partitioned into two portions, one is for training that goes to the IFS and the other one is for testing at the IC.

B. Incremental Classifier (IC)

Many choices exist when it comes to data stream mining algorithms such as those which are available on Massive Online Analysis (MOA) [7] developed by University of Waikato, New Zealand. Some popular algorithms include but not limited to: Decision Stump, Hoeffding Tree, Hoeffding Option Tree, Hoeffding Adaptive Tree, and ADWIN etc. The algorithms have a common design basis that works by incremental learning approach. The model gets rebuilt partially by only seeing enough samples that are qualified (or biased) for growing an additional decision tree branch (or rule). In such way, the model in induced progressively from scratch to a full-grown mature decision tree which has seen enough data stream samples, being able to recognize the mappings between the attributes and the target classes.

Out of the many implementation, lately there is a decision tree design called "Cache-based Classifier" (CBC) which is claimed to be able to detect and overcome the problem of concept-drift [8]. In this particular design, CBC has an auxiliary data cache similar to CR from which data are copied to a Decision Table Classifier (DTC) and a Main Tree Classifier (MTC). While the MTC remains as the main classifier from which the output result is derived, the main function of DTC is only to test if a concept drift has occurred hence the need of refreshing the MTC is assured. By this logic, MTC is spared from excessive updating and its accuracy is not diluted unless a concept drift happens at the incoming data streams. A loss function is defined by counting simple statistics of the frequencies of agreement

and disagreement between the predicted outputs by DTC and MTC respectively. The counting method is in Eqn. 1 and 2.

$$\begin{aligned} \mathrm{MTC}(X) \rightarrow \widehat{y_k} &\begin{cases} = y_k \Rightarrow T_k^{MTC} = T_k^{MTC} + 1 \\ \neq y_k \Rightarrow F_k^{MTC} = F_k^{MTC} + 1 \end{cases} \\ \mathrm{DTC}(X) \rightarrow \widehat{y_k} &\begin{cases} = y_k \Rightarrow T_k^{DTC} = T_k^{DTC} + 1 \\ \neq y_k \Rightarrow F_k^{DTC} = F_k^{DTC} + 1 \end{cases} \end{aligned} \quad (1)$$

$$\begin{aligned} Loss_k^{MTC} &= T_k^{MTC} - F_k^{MTC} \\ Loss_k^{DTC} &= T_k^{DTC} - F_k^{DTC} \end{aligned} \quad (2)$$

where y_k is the predicted value at k^{th} iteration, the *Loss* variables are counters for the classifiers, T is the number of times both predicted values are in agreement, F is the count of otherwise.

A coefficient P_k is used as a normalized single factor for deciding whether a decision tree should grow by inducing an additional tree path. P_k is defined in Eqn. 3 and 4.

$$\frac{\sum_{i=1}^{I}\sum_{j=1}^{J} n_{ijk}}{N} = P_k \times \frac{Loss_k^{DTC}}{Loss_k^{MTC}} \quad (3)$$

$$P_k = \frac{Loss_k^{MTC} \times \sum_{i=1}^{I}\sum_{j=1}^{J} n_{ijk}}{Loss_k^{DTC} \times N + 1}. \quad (4)$$

The values of the variables are remembered between the current and the previous steps, and they do get updated when fresh data arrives. If it were detected that *current* $P_k <$ *previous* P_{k-1} , it indicates that the accuracy of MTC is declining, and the need for updating to be updated by ε_k which is defined by Eqn. 5, materializes.

$$m_k = \frac{1}{\frac{1}{P_k^t} + \frac{1}{P_k^{t-1}}}, \delta' = \frac{\delta}{P_k^t + P_k^{t-1}}, \epsilon_k = \sqrt{\frac{1}{2m_k} \times \ln\left(\frac{4}{\delta'}\right)} \quad (5)$$

This dual concept of test-first-before-update leads to good accuracy, compact tree size and fast processing. Furthermore, CBC is scalable in nature implying the possibility of parallel processing. For instance, the DTC and MTC can be made as independent processes. The design of CBC can be extended by incorporating with multiple MTC's like an ensemble tree which is very common in traditional data mining. Each MTC could well be implemented by different incremental learning algorithms; the prediction result is to be taken from one winner among all the models that outperforms the rest.

C. Incremental Feature Selection Module (IFS)

The objective of this module is twofold; one is supposed to shrink down the total combination of feature subsets by simple selection algorithm, the other is to reason about the importance of the attributes with respective to the predicted classes, such as attribute scoring. There are plenty of available algorithms for incremental feature selection. Some popular ones include Grafting [9] which is based on the heuristic of gradient descent in function space, some is based on rough set theory on dynamic incomplete dataset [10], and the incremental feature ranking method over dynamic feature space [11], to just name a few.

One of the latest state-of-arts called Clustering Coefficients of Variation (CCV) is relatively simple hence suitable for lightweight computation in SHARP. CCV is founded on a basic belief that a good attribute in a training dataset should have its data vary sufficiently wide across a range of values, so that it is significant to characterize a useful prediction model. The coefficient of variation (CV) is expressed as a real number from - to + and it describes the standard deviation of a set of numbers relative to their mean. It can be used to compare variability even when the units are not the same. In general CV informs us about the extent of variation relative to the size of the observation, and it has the advantage that the coefficient of variation is independent of the units of observation. The coefficient of variation, however, will be the same over all the features of a dataset as it does not depend on the unit of measurement. So you can obtain information about the data variation throughout all the features, by using the coefficient of variation to look at all the ratios of standard deviations to mean in each feature. Intuitively, if the mean is the expected value, then the coefficient of variation is the expected variability of a measurement, relative to the mean. This is useful when comparing measurements across multiple heterogeneous data sets or across multiple measurements taken on the same data set – the coefficient of variation between two data sets, or calculated for two attributes of measurements in the case of feature selection, can be directly compared, even if the data in each are measured on very different scales, sampling rates or resolutions. In contrast, standard deviation is specific to the measurement/sample it is obtained from, i.e. it is an absolute rather than a relative measure of variation. In statistics, it is sometimes known as measure of dispersion, which helps compare variation across variables with different units. A variable with higher coefficient of variation is more dispersed than one with lower CV. Readers are referred to [12] for the formulation and details. In the case of SHARP, CCV helps to shrink the feature space by eliminating the disqualified features (by the CV principle) and the combinations of such features. The feature space that has been reduced in size will then been used by the SO for finding the most optimal feature subset by metaheuristics search algorithms.

D. Factor Analysis Module (FA)

IFS and FA are usually work together (or in parallel as in SHARP), having IFS to produce the selected features and FA offers insights about the significance of attributes to the predicted classes. In general, this method is to correlate a large number of features in a dynamic dataset with an outcome variable, such as the predicted class. Computationally this is done by scoring each feature by some statistical means (correlation is one of them). The other types of feature scoring exist such as gain ratio, information gain, Chi-square evaluation, etc. that have similar methods for scoring. As a result of FA, a list of features sorted by values in ascending or descending order would be produced; their rankings could be visualized too. It offers insights to users about the importance of each attribute for inquisitives.

E. Swarm Optimizer (SO)

A swarm optimizer is essentially a search module that looks for optimal parameter values of the classifier in use, and the optimal feature subset for the IC module. SO takes the output of IFS which is a reduced search space of feasible combinations of feature subsets as input. Multiple search agents, which often are inspired by natural phenomena or behaviors of biological creatures, scout over the search space for the optimum solution. The search operation is in parallel as these multiple agents are working autonomously but collectively through some stochastic process. The search iterates through generations, thereby evolving the solution to an optimum one at the end. Some researchers call such methods meta-heuristics as it is meant to be a high-level strategy (therefore the name meta-) that guides the underlying heuristic search in achieving a goal. In the case of SHARP, SO is an optimized implemented by Swarm Search [13] which is a latest search method for finding the optimum feature subset using meta-heuristics from large datasets.

Swarm Search is particular useful for datasets that are characterized by a very large amount dimensionalities, so called features. Although the meta-heuristics is generic which is able to integrate any type of bio-inspired optimization algorithms into any type of classifier (at least theoretically), the work by [13] tested 9 different combinations – three classifiers: neural network, decision tree, and Naïve Bayes, and three bio-inspired optimization algorithms: Wolf Search Algorithm, Particle Swarm Optimization and Bat Algorithm.

For SHARP that demands for incremental learning and therefore progressive search for the best feature subset, Swarm Search should be configured to embed with only incremental algorithms, like those mentioned in Section IIB. Depending on the setup at IC which may be an ensemble method where multiple classifiers are being tested, the most accurate model got selected, SO should operate in parallel too having each execution thread corresponds to each candidate classifier as in IC. The composite optimization applies where the possible parameters values and the possible feature subsets search space are blended together into a large search space, over which the Swarm Search attempts to find the best combination. This approach was pioneered by Iztok et al in [14].

III. EXPERIMENTS

In order to validate efficacy of SHARP, three sets of experimentation are shown hereafter, with each set tests on some of the core components of SHARP such as IC, IFS and SO. The datasets being used are big data, not only in volume but they are large in number of features that pose great computational challenges in data stream mining. All the experiments run on a Windows 7 64-bit workstation with Intel Quad 2.83 GHz processor and 8 Gb RAM.

A. Testing Performance of CBC as IC

In this experiment, CBC is tested versus two state-of-the-art data streams mining algorithms, namely HOT [15] and ADWIN [16]. The representative big data is Cover-type data that are available for free download from the UCI data archive (www.ics.uci.edu/~mlearn). It has 42 categorical attributes and 12 continuous attributes, for predicting seven types of cover lands. The number of instances is 581,012.

The CBD is comprised of DTC+MTC with varying window sizes and it uses node-splitting bound computed from the loss function in Eqn 2. The selected window sizes (*ws*) are 50, 200, 500, and 1000. For τ = 0.05 which is a default value by MOA for controlling the learning speed, accuracy is better when *ws* = 50 and *ws* = 500, but the tree becomes larger than it does with the other *ws*. The classifier attains higher accuracy but smaller tree size than HOT. Furthermore, when *ws* = 50, the accuracy is better and the tree size is reduced compared with ADWIN. The performance comparison is shown in Figures 2 and 3.

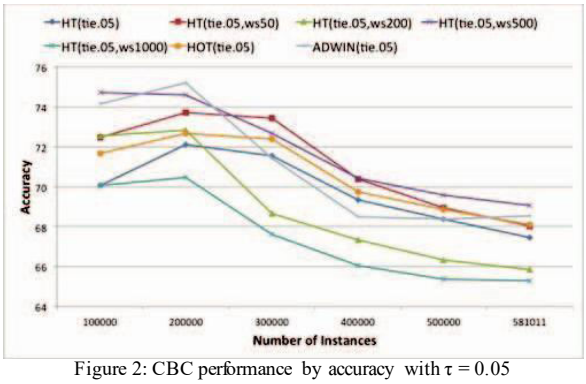

Figure 2: CBC performance by accuracy with τ = 0.05

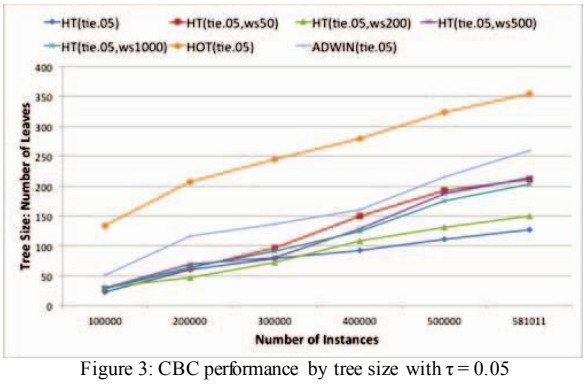

Figure 3: CBC performance by tree size with τ = 0.05

As it can be seen from Figure 2, CBC (which are named as HT that stands for Hoeffding Tree) in the right setting combination of τ and *ws*, is able to outperform HOT and ADWIN. For the tree size in Figure 3, which is presumed to be the smaller the better with less memory space required in run-time, HOT and ADWIN requires more than CBC. One shortcoming however is the requirement of manually setting the right values of parameters for the algorithms to run. Nevertheless it can be shown that CBC is able to achieve good accuracy and compact tree size. This strongly suggests that integrating with SO for finding the right parameters values are imperative, especially when different versions of MTC implemented by different algorithms are running in parallel.

B. Testing Performance of CCV as IFS

In order to validate the effectiveness of a simple and lightweight feature selection algorithm, called CCV, versus the other existing methods, we test them on forty-four different datasets from UCI using four popular classification algorithms in data mining, namely ADABoost, J48 decision tree, Naïve Bayes and SMO. We compare the performance of CCS against the four classical ones like Correlation-based
(CFS), Consistency-based (Consist) and Relief-based (Relief) Subset Evaluation methods, and Principle Component Analysis (PCA). A typical 10-fold cross-validation is used to valid and generate the outputs of performance indicators in each test.

The forty-four datasets are first preprocessed by the different feature selection algorithms. After that they are processed in turn by each one of the four classification algorithms, and then the results are averaged out. The performance results are expressed in terms of Accuracy (number of correctly classified instances over the total number of instances), pre-processing time (which is crucial in SHARP that demands for high-speed processing), and the percentage of attributes being selected. The results are shown in Table 1.

Table 1: Averaged performance results of feature selection methods by classification models built over the forty-four datasets

Feature selection method	Accuracy	Time (ms)	% Selected Features
CCV	0.774926	298.068182	62.46602422
CFS	0.749371	109.727273	11.04521868
Relief	0.754955	56426.1591	66.79021497
Consist	0.737733	1269.81818	9.933283914
PCA	0.728131	3717.29545	75.14208055

As it can be observed from Table 1, CCV rates second after CFS for speed, which is far shorter the time when compared with the other three feature selection algorithms – Relief, Consist and PCA. The accuracy achieved by CCV is the highest, and it can retain a moderate amount of features.

When CCV is applied in SHARP, it exhibits supposedly the benefits of producing a moderate size of reduced search space (in terms of features and their combinations), at a reasonable run-time speed, and the search space would be likely to produce good accuracy for the classification algorithms in IC.

C. Testing Performance of Swarm Search as SO

Swarm Search is a vital part of SO that searches for the optimal feature subset given the reduced search space of feasible feature combinations by IFS. There are many meta-heuristics available, however in this experiment, the same settings and collection of Swarm Search algorithms are used as in [13]. They are FS-PSO, FS-BAT and FS-WSA integrated with neural network, decision tree and Naïve Bayes classification algorithms. Though this is a preliminary experimentation with the aim of demonstrating the viability of Swarm Search to be used in SHARP, future experiments might be needed by integrating Swarm Search with incremental algorithms, particularly with CBC. Five testing datasets with varying number of features, ranging from 56 to 875, are obtained from UCI. They are called Lung Cancer, Heart Disease, Libra Movement, Hill Valley and CNAE.

The following comparisons show that different couple of meta-heuristics and classification algorithms for different datasets yield variable results, as showing in Figures 4, 5, and 6 for classification error comparison and Figures 7, 8 and 9 for time consumption comparison.

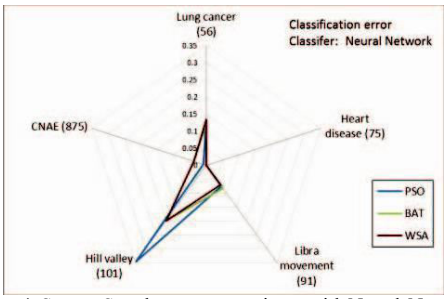

Figure 4: Swarm Search error comparison, with Neural Network

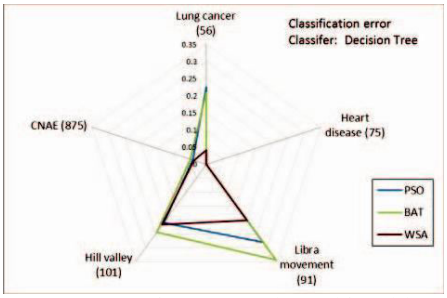

Figure 5: Swarm Search error comparison, with Decision Tree

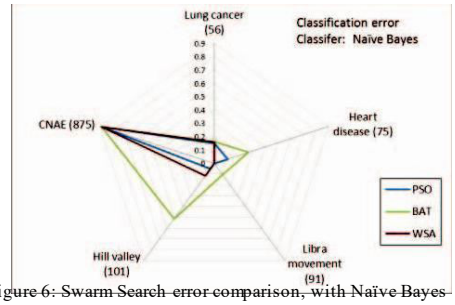

Figure 6: Swarm Search error comparison, with Naïve Bayes

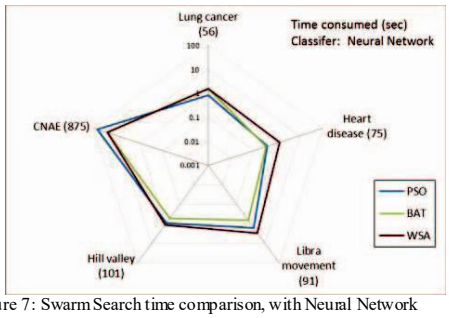

Figure 7: Swarm Search time comparison, with Neural Network

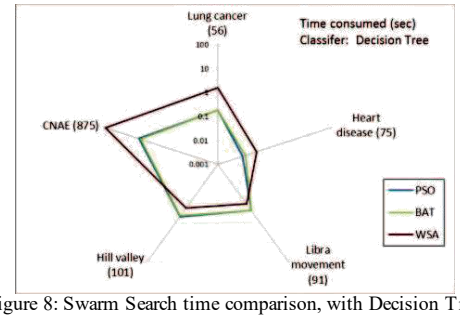

Figure 8: Swarm Search time comparison, with Decision Tree

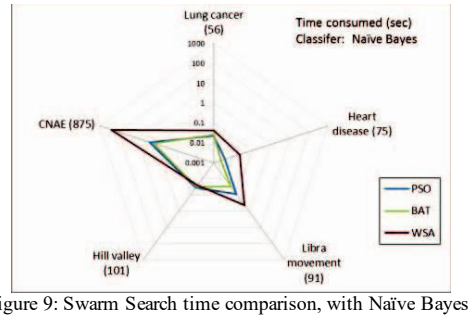

Figure 9: Swarm Search time comparison, with Naïve Bayes

The variable results as shown from the above Figures testify the need of multiple tests that are supposed to run in parallel, from which the best performance collection of meta-heuristics as well as the classification algorithms and the related parameters values are required to be chosen out. This could only be made possible by parallel computation where different combos are tested on individual processor and execution thread. The SHARP methodology allows and encourages SO to operate on some appropriate parallel computing devices.

IV. Conclusion

In this paper, a scalable data stream mining called Stream-based Holistic Analytics and Reasoning in Parallel (SHARP) was introduced. SHARP is holistic because it consists of several components and they target to improve different aspects of data mining functions such as smoothing the input data streams, reducing the feature search space, finding the optimum feature subset, optimizing parameter values for the classifiers, and allowing incremental classifiers to go ensemble by spawning different classifiers in parallel. Preliminary experiments for three individual components have been tested and demonstrated superiority over existing methods. In the future, it is planned that all the components would be fully integrated and tested as a holistic data stream mining system that can produce the best possible performance. It is anticipated that SHARP is capable of eliminating some of the key problems in Big Data especially those associated with high-dimensionality and infinite and continuous data streams.

References

[1] Quinlan, J. R., C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993

[2] Ping-Feng Pai, Tai-Chi Chen, "Rough set theory with discriminant analysis in analyzing electricity loads", Expert Systems with Applications 36 (2009), pp.8799–8806

[3] Mohamed Medhat Gaber, Arkady Zaslavsky, Shonali Krishnaswamy, "Mining data streams: a review", ACM SIGMOD Record, Volume 34 Issue 2, June 2005, pp.18-26

[4] Wei Fan, Albert Bifet, "Mining Big Data: Current Status, and Forecast to the Future", SIGKDD Explorations, Volume 14, Issue 2, pp.1-5

[5] Arinto Murdopo, "Distributed Decision Tree Learning for Mining Big Data Streams", Master of Science Thesis, European Master in Distributed Computing, July 2013

[6] Hang Yang, "Incremental Decision Tree Learning for Streams of Imperfect Data", PhD Thesis, Department of Computer and Information Science, University of Macau, 2014

[7] Bifet, Albert; Holmes, Geoff; Kirkby, Richard; Pfahringer, Bernhard (2010). "MOA: Massive online analysis". The Journal of Machine Learning Research, Volume 99, pp.1601–1604

[8] Hang Yang, Simon Fong, "Improving the Accuracy of Incremental Decision Tree Learning Algorithm via Loss Function", 2013 IEEE 16th International Conference on Computational Science and Engineering, pp.910-916

[9] Simon Perkins, Kevin Lacker, James Theiler, "Grafting: Fast, Incremental Feature Selection by Gradient Descent in Function Space", Journal of Machine Learning Research, Volume 3 (2003), pp.1333-1356

[10] Wenhao Shu, Hong Shen, "Incremental feature selection based on rough set in dynamic incomplete data", Pattern Recognition, Volume 47, Issue 12, December 2014, pp.3890–3906

[11] I. Katakis, G. Tsoumakas, and I. Vlahavas, "On the Utility of Incremental Feature Selection for the Classification of Textual Data Streams", PCI 2005, LNCS 3746, Springer, pp.338–348

[12] Simon Fong, Justin Liang, Raymond Wong, Mojgan Ghanavati, "A Novel Feature Selection by Clustering Coefficients of Variations", ICDIM, 2014, pp.205-213

[13] Simon Fong, Suash Deb, Xin-She Yang, Jinyan Li, "Feature Selection in Life Science Classification: Metaheuristic Swarm Search", IT Professional, Volume 16, Issue 4, July-Aug. 2014, pp.24-29

[14] Janez Brest, Borko Boskovic, Ales Zamuda, Iztok Fister, Efrén Mezura-Montes, "Real Parameter Single Objective Optimization using self-adaptive differential evolution algorithm with more strategies", IEEE Congress on Evolutionary Computation, 2013, pp.377-383

[15] B. Pfahringer, G. Holmes and R. Kirkby. "New Options for Hoeffding Trees", Advances in Artificial Intelligence, Springer, 2007, pp. 90-99

[16] A. Bifet, R. Gavaldà. "Learning from time-changing data with adaptive windowing". SIAM International Conference on Data Mining, 2007