Enhanced Green Accelerated Hoeffding Trees for Improved Data Stream Classification

Hadjer Imene Bensaoula, Sarah Nait Bahloul

Abstract

The proliferation of real-time, infinite data streams necessitates efficient online learning approaches. Hoeffding Trees (HT), which extend traditional decision trees using the Hoeffding bound, offer robust stream classification but face high computational costs. While the Green Accelerated Hoeffding Tree (GAHT) addresses energy efficiency concerns, its prediction accuracy can be improved by addressing its inherent limitations in combining Hoeffding bounds with information gain metrics for incrementally growing the tree. This study successfully develops enhanced GAHT variants through optimized Hoeffding bound stability and node splitting mechanisms. Our empirical evaluation demonstrates that the usage of these new variants improves predictive performance over the state-of-the-art GAHT, without compromising its energy efficiency.

Keywords: Big Data, Data Stream, Online Learning, Incremental Classification, Hoeffding Trees, Hoeffding Bound.

MSC 2020: 68T01, 68T05, 68T09, 68Q32, 62R07.

1 Introduction

In many fields of activity, such as healthcare, finance, and social networks, deriving patterns and discovering interesting trends from real-time data, known as a data stream, is gaining widespread attention and prominence, as it allows decision-makers to extract knowledge in real time, and enables analysts to detect and handle sudden, real-time changes occurring in data.

doi:10.56415/csjm.v33.09

Unlike batch data processing, several restrictions must be considered while building a data stream based model, as no multiple passes over the data are allowed, in addition to its infinite and unbounded nature making it impossible to store the entire stream, which in turn allows the data to evolve and change over time, requiring the model to be constantly updated to prevent it from becoming obsolete [1]. All of these constraints can not be fulfilled by traditional data mining methods, as they require the entire data set to be provided to build the model, whereas updating the model at the arrival of new stream instances using batch learning approaches, requires the model to store historical data and merge it with the new stream instances so that it can be retrained, which is significantly expensive in terms of storage and computing resources [2] [3]. To overcome this, stream mining and online learning methods have been introduced to efficiently generate models in a single pass over stream instances, without the necessity of retraining the existing model on both historical and new stream instances during updates.

In the present work, we were more passionate about stream classification using an energy-efficient extension of Hoeffding Trees (HT) [4], which is an incremental decision tree capable of building the prediction model by making a single pass over the stream data. The introduction of the HT was initially observed within the implementation of the Very Fast Decision Tree (VFDT). Through the use of the Hoeffding Bound (HB), the VFDT splits the leaf nodes to grow the model incrementally as stream instances continue to arrive. The latter performs node splits only after meticulously determining the optimal attribute, upon which the decision is fixed, and the subsequent decision remains unchanged. The authors of [5] point out that if HT rigorously seeks to find the optimal splitting condition, this will cause the algorithm to struggle to perform splits when information is evenly distributed across attributes. This will either result in split delays, slowing model growth and impacting model availability, or will cause the model to grow its leaves based on the use of a tie-breaking threshold, which is likely to compromise its efficiency. To solve this problem and many others, a sequence of improvements was made to the HT, yielding the emergence of the Hoeffding Any Time Tree [5], based on its Extremely

Fast Decision Tree (EFDT) implementation. To avoid delaying splits, EFDT transforms a leaf into an internal node as soon as a beneficial split is identified (not necessarily the optimal split). It then refines the subsequent structure of the tree by replacing existing splits with better alternatives when they are identified, enabling faster learning by initially choosing an attribute and modifying it if necessary.

Throughout the HT improvements, it was observed that improving the model's predictive performance came at the cost of heavy computations, involving significant hardware resources and higher energy consumption. Accordingly, when improving any of the subsequent HT models, along with improving predictive performance, reducing energy consumption emerged as yet another priority. As part of this, several studies were carried out on the VFDT and EFDT implementations to ensure their energy efficiency, resulting in the adaptation n_{min} VFDT [6], the Strict Very Fast Decision Tree (SVFDT) [7], and the Green Accelerated Hoeffding Tree (GAHT) [8]. While VFDTn_{min} and SVFDT have been shown to successfully reduce energy and memory usage, respectively, they are both based on the VFDT implementation, inheriting its aforementioned drawbacks. In contrast, the GAHT extends EFDT with lower power consumption, where results from [8] demonstrate that GAHT significantly reduces the power consumption of EFDT by up to 70%, yet still achieves competitive results for EFDT.

The original implementation of the GAHT model uses the Hoeffding bound along with the information gain to calculate the split merits of a given attribute. However, this combination has been widely criticized in the literature for its inefficiency with nominal data. Considering the unpredictable nature of real-world data streams and the inevitable presence of nominal attributes in these streams, this inefficiency severely limits the effectiveness and applicability of the GAHT model.

The purpose of this research is to address the aforementioned limitation within the Green Accelerated Hoeffding Tree (GAHT), with a particular focus on improving its accuracy rate while maintaining its energy efficiency when processing real-world data streams. To achieve this, we will first address the Hoeffding Bound stability challenge. We will explore Hoeffding bound variants, which have been documented in the literature to provide superior performance compared to the original

bound used in GAHT. Subsequently, we will investigate the most effective tree node splitting mechanisms for tree growth, aiming to find the optimal combination that enhances GAHT's predictive accuracy. This comprehensive approach is designed to contribute to the development of more efficient and accurate online machine-learning models.

The subsequent sections are organized as follows: Section 2 provides a literature review on data stream mining, online learning, and Hoeffding Trees. Section 3 details the Green Accelerated Hoeffding Tree (GAHT), our baseline model. Section 4 outlines our research contributions, addressing the limitations of Hoeffding Trees and presenting our proposed solutions. Section 5 describes the experimental setup, including software and datasets, and analyzes the results. Finally, Section 6 summarizes our key findings and concludes the study with a glimpse into future work.

2 Literature Review

In a dynamic data stream environment where non-stationarity is a norm, the incoming data is more likely to evolve over time and dynamically change its underlying distribution unpredictably, a situation known as "Concept Drift" [9]. Consequently, processing and gaining insights from the potentially infinite streams of high-frequency data requires the use of incremental learning approaches capable of continuously updating the model, thus ensuring efficient real-time learning, known as online learning.

Being a widely recognized supervised learning task among the data mining approaches [10], classification has also served the online learning community to accurately predict the labels of the emerging stream instances, on the basis of a model built incrementally from the previous labeled training instances, and updated using the new ones. Data stream classifiers differ from batch data classifiers by the update property, a crucial phase that involves automatic adjustments as new data becomes available, and can be known as a self-improvement mechanism [11] [12], preventing the model from becoming obsolete over time due to the concept drift challenge [10].

Consequently, building a stream classifier requires learning, testing,

and updating tasks. Among various stream classification approaches, the Hoeffding Trees [4] algorithm stands out as our baseline method, given its proven effectiveness in addressing the unique challenges of data stream mining through its advanced decision tree architecture.

2.1 Hoeffding Trees

The Hoeffding Trees (HT) were introduced to address the challenge of classifying high-velocity non-static data. As stream instances arrive at a high frequency, Hoeffding Trees have the ability to update the existing tree using only new instances, without having to store and examine previous instances [4], which effectively minimizes learning costs in terms of optimizing the use of storage and computing resources, since data streams are infinite [13]. Furthermore, this incremental update mechanism prevents the existing model from becoming obsolete, as data properties can change over time due to the concept drift challenge [1].

Growing an incremental decision tree in a stream processing scenario entails performing splits using only a subset of the stream, rather than the entire stream instances, as for the batch decision trees.

This gives rise to the challenge of identifying the correct attribute to perform a split while turning a leaf into a node [14], as a better split attribute may appear within future stream instances. Consequently, the Hoeffding Bound (see equation 1) was proven to be the appropriate measure to choose the right split attribute for a given leaf node.

In the Very Fast Decision Tree (VFDT) [4], the state-of-the-art implementation of Hoeffding Trees, the Hoeffding Bound provides a statistical guarantee that the mean value of a random variable contained in a range R will not differ from its estimated mean by more than ε after sampling n independent observations, all within an error rate of δ:

$$\varepsilon = \sqrt{\frac{R^2 \cdot \ln\left(\frac{1}{\delta}\right)}{2n}}. \tag{1}$$

The major strength of this bound is that it only depends on: (i) the values range R, (ii) the number of instances n, (iii) the expected confidence within the split $(1 - \delta)$. This guarantees with a probability of $(1 - \delta)$ that the chosen attribute is the right one to make the split [15].

The VFDT's tree-building process initiates with a root node (initially a leaf) and processes stream instances characterized by a set of attributes $\{X_1, X_2,..., X_k\}$. Upon their arrival, the instances are gathered within the root node that keeps track of the number of instances using the variable n_t, which gets compared with another parameter fixed by the user, called n_{min}, which indicates the minimum required samples to calculate a split. Once the n_t value reaches the n_{min} value, the algorithm uses a splitting criterion, commonly the information gain ratio, and calculates its value Gain(X_i) for each attribute X_i. Afterwards, the algorithm estimates the difference between the two attributes X_a and X_b having the best gain ratio value as follows:

$$Gain(X_a) - Gain(X_b) > \varepsilon. \tag{2}$$

Once the equation (2) is satisfied, the root node will grow a child node based on the attribute X_a, which will be omitted from the next child node split candidate attributes. Otherwise, the algorithm employs a user-defined tie threshold parameter, denoted as τ, to bound the ε value, and choose X_a as a split attribute if equation 3 is achieved:

$$\varepsilon < \tau. \tag{3}$$

This process will be performed recursively to fully grow the tree.

However, due to the potentially infinite nature of the data stream, the tree will grow in an unlimited and unrestricted manner, more so, if no parameters such as the tree maximum depth are set. In case that the memory reaches its limit, the tree system temporarily frees up space by deactivating child nodes having less impact on the model; however, if an inactive node is found to have a higher classification accuracy later on, it will resume its activity [16].

2.2 Hoeffding Trees: State of the Art

The Very Fast Decision Tree (VFDT) initially operated under the assumption of stationary data distributions, which proved insufficient for handling evolving streams due to concept drift caused by unpredictable changes in the statistical distribution of the data. To address this limitation, the Concept-adaptive Very Fast Decision Tree (CVFDT) was

introduced in [17], enhancing the VFDT by constructing alternative subtrees that replace the main tree when they exhibit lower error rates.

In addition, a special category of HT for dealing with concept drift, called Hoeffding Adaptive Tree (HAT), has emerged in [18] to perform on evolving data streams. HAT is combined with drift detection methods, many of which are well detailed in [19]. Other notable contributions include the One-class Very Fast Decision Tree (OcVFDT) [20], designed to handle class imbalance but lacking adaptability to concept drift. The Enhanced Decision Tree (EDT) [21] integrated Naive Bayes to reduce time complexity but encountered issues with overfitting. The Optimized Very Fast Decision Tree (OVFDT) [22] and the Efficient Concept-adaptive Very Fast Decision Tree (ECVFDT) [23] targeted class imbalance and various forms of concept drift, respectively. The Vertical Hoeffding Tree (VHT) [24] and VFDT-Hadoop [25] focused on reducing computational complexity.

These progressive developments have been eclipsed by the emergence of the Hoeffding Anytime Tree (HATT) category, marking a significant leap in the evolution of Hoeffding Trees.

In 2018, the HATT category was introduced to optimize the performance of Hoeffding Trees in the context of evolving data streams. The EFDT [5], as the practical instantiation of HATT, prioritizes the selection and deployment of splits based on immediate utility, with the flexibility to reverse decisions if a higher purity score is achieved by another attribute. Recent studies [26] have begun to recognize EFDT as the state-of-the-art approach for incremental decision tree surveys, effectively displacing the traditional VFDT.

2.3 Hoeffding Trees: Energy Efficiency

While previous advancements have focused on improving the ability to handle both stationary and evolving data streams, the energy efficiency of these processes remains a critical issue due to the substantial storage and computational resources required [27]. This section highlights the efforts aimed at enhancing the hardware and energy efficiency of online learning decision trees. The VFDT utilizes the n_{min} parameter to determine the adequacy of instances for performing splits at leaf

nodes. A fixed n_{min} value can lead to inefficiencies, causing unnecessary computations and energy consumption. To mitigate this, authors in [6] proposed the Hoeffding Trees with n_{min} Adaptation (VFDTn_{min}) approach, where each leaf node has its own n_{min} value, ensuring that computations are performed only when splits are imminent. This adaptation resulted in an 89% reduction in energy consumption, with a minimal 3% decrease in accuracy. In a similar vein, the Strict Very Fast Decision Tree (SVFDT) [7] was introduced to optimize hardware usage and reduce memory costs. SVFDT maintains leaves as such unless a split is deemed necessary, achieving comparable predictive performance to VFDT while significantly reducing processing time and memory usage.

The present section aimed at tracing the methodological evolution of the HT algorithm, revealing a dual trajectory: Subsection 2.2 outlined fundamental advances focused on improving the core of the HT classifier in terms of predictive robustness when addressing the complexity of data streams, whereas the implementations illustrated in Subsection 2.3 targeted the contemporary improvements that began targeting the HT's computational efficiency. However, it should be noted that the two energy-efficient incremental trees relied on the original VFDT. An HT implementation that was subsequently deemed outdated following the introduction of the EFDT suggests an improvement of the latter through a faster and more flexible model construction strategy, as highlighted in Subsection 2.2. In light of its statistical superiority, the authors of [8] chose to rely upon the EFDT algorithm to implement the Green Accelerated Hoeffding Trees (GAHT), an energy-efficient online decision tree. With the aim of enhancing the model's stability and prediction accuracy when applied to real-world datasets, the current work extends the GAHT implementation by combining it with methodological refinements of its node-splitting component.

3 Green Accelerated Hoeffding Trees

3.1 GAHT Basic idea

Inspired by the VFDT n_{min} adaptation, the Green Accelerated Hoeffding Trees (GAHT) algorithm was introduced in [8], as an approach that can obtain competitive accuracy to the EFDT but at reduced energy cost.

The idea behind GAHT is to use a per-node splitting process that varies according to the observed data distribution per leaf node. Based on a calculated fraction, either EFDT or basic Hoeffding tree growth process will be applied for a given leaf node, meaning that the EFDT re-evaluation process will only be applied for some nodes, rather than for all nodes in the tree, as in the case of a basic EFDT, which will significantly help reduce the calculations and the consumed energy.

3.2 GAHT Algorithm

This section presents a summary of the GAHT growth process, highlighting the most important aspects.

Upon receiving an instance, the algorithm processes the tree branches until reaching a leaf node, at which point it calculates the fraction of observed instances at this leaf:

$$fraction = \frac{n_l}{n_{sinceCreation}/n_{leaves}}, \tag{4}$$

where n_l denotes the number of instances observed at this particular leaf, $n_{sinceCreation}$ designates the number of instances observed in the tree since the creation of this leaf, and n_{leaves} – the overall number of leaves in the tree. Having determined the value of the fraction using equation 4, the result gets compared with the $deactivateThreshhold$ and the $growFastThreshhold$, where three cases may arise:

1. If fraction $<$ $deactivateThreshhold$, the node is deactivated, meaning no further splitting is done.

2. If the leaf node is active and the fraction $\leq$ $growFastThreshhold$,

the leaf is transformed into a node by employing the same algorithmic steps that are used in the construction of a basic HT.

3. If the leaf node is active and the fraction $> growFastThreshhold$, the leaf is transformed into a node by employing the same algorithmic steps that are used in the construction of an EFDT.

Following the above-outlined cases, the GAHT algorithm dynamically constructs the tree, generating three different types of nodes during its execution:

- **Standard nodes:** these nodes evolve according to the same growth criteria as standard Hoeffding trees.
- **Fast-growing nodes:** these nodes follow the EFDT splitting process by creating less restrictive splits, which results in faster tree growth on these branches.
- **Inactive nodes:** when the $fraction$ value within a given node is less than the $deactivateThreshhold$, this node will be inactive.

GAHT was tested using synthetic and real-world benchmark datasets, the metrics being energy efficiency and accuracy; the results showed that its performance was similar to that of EFDT while using 70% less energy [8]. Since the GAHT algorithm is a passive online tree designed for stationary data streams, the authors have not attempted to judge its effectiveness in evolving stream scenarios, as this is not its primary objective. Given that conceptual drift is a common challenge in real-world data streams, this study will also conduct experiments on GAHT with datasets exhibiting concept drift, thus addressing its behaviour and stability in drifting situations.

The following section examines the node separation limits implemented by GAHT, as well as the introduced separation component variants to be used.

4 Contribution

The current state-of-the-art implementation of the Green Adaptive Hoeffding Tree (GAHT) uses the Hoeffding Bound and a basic split-

ting mechanism derived from the Hoeffding tree. These mechanisms are found to exhibit poor performance based on the findings in [28]. Several alternative proposals to enhance these mechanisms have been made [29] [28] [30], and when applied to other implementations of Hoeffding trees [31], these alternatives have demonstrated favourable results. Our research aims to explore these methods within the GAHT framework. This section outlines the challenges and proposed solutions to address them.

4.1 Correcting the usage of the Hoeffding Bound

In [29], the authors argue that the Hoeffding Bound is too restrictive, as it only applies to numeric variables and requires an input that can be expressed as a sum of independent variables.

Consequently, multiple efforts have been made to improve the Hoeffding inequality's usage within the tree growth process, suggesting various alternatives such as McDiarmid inequality in [29], Doubled Hoeffding bound, and Accuracy-Gain from [28], in addition to the Half-Hoeffding Bound [30]. Previous research applying Hoeffding Bound variants to the EFDT hierarchical tree [31] revealed promising results. This study builds upon this foundation by exploring the effects of these same variants on the GAHT algorithm. Specifically, we investigate:

- **Half-Hoeffding bound**: As shown in [30], the half-Hoeffding bound allows the Hoeffding Tree to grow more rapidly using four times fewer data elements to make a split, compared to what is needed in the case of using the original Hoeffding bound. In addition, the authors state that using the Half-Hoefding bound gave higher accuracy rates.

- **Doubled Hoeffding bound**: This variant of the Hoeffding bound is designed to enhance the correctness of splitting a leaf node [28], making it particularly beneficial for Hoeffding trees that do not revise their splits, thereby aligning well with VFDT. However, its effectiveness in the EFDT, which performs revisions for each split, has not yet been evaluated. Given that the GAHT extends both VFDT and EFDT [8], we investigate in this study

whether the Doubled Hoeffding bound would be advantageous in improving the GAHT's accuracy.

4.2 Node Splitting Criteria

Most of the Hoeffding Trees algorithms tend to use information gain as splitting criteria in the tree growth process.

However, as noted in [32], the use of information gain cannot be combined with Hoeffding's inequality. The authors advocate for the initial use of Misclassification error (MC) as a measure of impurity, particularly at the lower levels of tree construction, and for higher levels, more traditional measures like information gain (IG) or the Gini index (Gini) are more effective. Consequently, a new criterion has been proposed in [32], which hybridizes Misclassification error and the Gini index to leverage their combined advantages. When applied to the EFDT in [31], this novel algorithm showed improved results.

This study contributes novel GAHT algorithm variants aimed at improving accuracy while maintaining this latter energy efficiency. These variants are based on a systematic exploration of Hoeffding Bound configurations (standard, half, and double), 3 split criteria (information gain, Gini index, and misclassification error), and a hybrid impurity measure combining misclassification error with both Gini index and information gain, exploiting the strengths of each criterion at different stages of tree growth. The complete set of resulting variants is shown in Table 1 (see p. 13).

The details of the various experiments and the execution environment will be presented in the following sections of this paper.

5 Experiments and Results

5.1 Experimental Setup

To carry out the experiments, we used an open-source framework for real-time stream mining, called Massive Online Analysis (MOA) [33]. The use of MOA is justified by the fact that it provides a wide range of stream mining algorithms, and it can be easily extended with new algorithms. The MOA's framework ease of extensibility enabled us

Table 1. GAHT Algorithm Variants

Algorithm Variant	Splitting Criterion	Hoeffding Bound Variant
GAHT IG	Information Gain	Standard HB
GAHT Gini	Gini index	Standard HB
GAHT MC	Misclassification error	Standard HB
GAHT IG Half	Information Gain	Half HB
GAHT Gini	Gini index	Half HB
GAHT MC Half	Misclassification error	Half HB
GAHT IG Double	Information Gain	Doubled HB
GAHT Gini Double	Gini index	Doubled HB
GAHT MC Double	Misclassification error	Doubled HB
GAHT MC+Gini Hybrid	Misclassification error + Gini index	Standard HB
GAHT MC+Gini Hybrid Double	Misclassification error + Gini index	Doubled HB
GAHT MC+Gini Hybrid Half	Misclassification error + Gini index	Half HB
GAHT MC+IG Hybrid	Misclassification error + Information Gain	Standard HB
GAHT MC+IG Hybrid Double	Misclassification error + Information Gain	Doubled HB
GAHT MC+IG Hybrid Half	Misclassification error + Information Gain	Half HB

to initially add the GAHT implementation provided in [8], and our improved GAHT implementations.

For each algorithm evaluation, we employ the Test-Then-Train evaluation method (also known as EvaluatePrequential) provided in the Massive Online Analysis (MOA) framework. In this methodology, the learning process follows the following steps:

1. Initial Phase: The model begins with an empty tree and builds its initial structure using the first few instances of the data stream.

2. Continuous Learning Process: For each subsequent instance in the stream:

    - First, the current model makes a prediction on the incoming instance (Test phase).
    - The true label is then revealed and used to evaluate the model's performance.
    - Finally, the same instance is used to update the model (Train phase)

This approach enables continuous evaluation of the model's performance on unseen data while simultaneously allowing the model to adapt to evolving patterns in the data stream. The incremental nature of this method makes it particularly suitable for streaming scenarios where both immediate performance assessment and continuous learning are required [34].

Each experiment was evaluated using 3 key metrics: accuracy, CPU time, and energy consumption (measured in kilowatt-hours).

Accuracy is defined as the proportion of correctly classified instances out of the total number of instances in the data stream. CPU time (T) represents the total execution time in seconds. To quantify energy consumption, an appropriate energy measurement tool is required. Accordingly, this research will use the Python package Codecarbon [35].

Codecarbon can be used to evaluate the amount of hardware power (expressed in Kilowatt per hour) required to run a given code sequence. This tool monitors the power supply to the underlying hardware at

regular intervals, with the default interval set at 15 seconds. Complementary details pertaining to this software can be found in [35].

To ensure a fair and robust evaluation, the experiments were carried out on 6 real-world benchmark datasets frequently used in the literature to assess the performance of incremental classification algorithms, along with 3 imbalanced concept drift scenarios, namely abrupt, gradual, and incremental.

Table 2. Properties of used datasets

Datasets	Features	Features Types	Label Type	Instances	Classes
Airlines	7	Nominal Numeric	Nominal	539,383	2
Electricity	8	Nominal Numeric	Nominal	45,312	2
HT sensor	11	Nominal Numeric	Nominal	919 438	3
Poker	10	Numeric	Numeric	1,025,010	10
KDD	41	Nominal Numeric	Nominal	1,000,000	23
Forest Covertype	54	Numeric	Numeric	581,012	7
Insects Abrupt Imb	33	Numeric	Nominal	355,275	6
Insects Gradual Imb	33	Numeric	Nominal	143,323	6
Insects Incremental Imb	33	Numeric	Nominal	452,044	6

As highlighted in [36], an optical sensor made to measure the flight properties of insects was used to gather data. Over the course of around three months, data was collected in a controlled, non-stationary set-

ting simulating various concept drift types. As for the remaining data, they were obtained from the UCI Machine Learning Repository [37], consisting of real datasets with no missing values requiring any preprocessing step, allowing a direct comparison of the algorithm's performance. Thus, combining these datasets will allow us to get a collection that encompasses a wide range of feature dimensions, instance counts, class imbalanced distributions, and concept drift scenarios, providing a robust testbed for the GAHT algorithm variants.

5.2 Results and Discussion

Tables 3, 4 and 5 report the performance of GAHT when used with simple and hybrid decision tree splitting criteria, namely: information gain (IG), Gini index (Gini) and classification error (McError), the hybrid criteria McError + IG and McError + Gini, together with the three Hoeffding Bound (HB) variants described (Half, original, and double HB).

The state-of-the-art GAHT implementing the standard HB combined with the IG serves as a baseline for comparison in this study. Moreover, when determining the effectiveness of the GAHT implemented variants, the measured energy consumption will be considered as a non-negligible factor in evaluating the solidity of these variants. This is necessary because GAHT is firmly rooted in the Green AI paradigm, which considers energy efficiency to be just as important as the model's predictive performance. Accordingly, while some variants might reach an outstanding accuracy, if they waste more energy than GAHT IG, they will be considered to be an inefficient GAHT variant in our research. Table 6 will consolidate and highlight for each dataset and drift type, the most outstanding GAHT variants when compared to baseline GAHT using the aforementioned measures. We shall mention that previous studies recommending the use of these HB or split criterion variants did not adopt this criterion when evaluating any of them. A preliminary analysis of the results reveals that most of the suggested GAHT variants perform competitively and frequently outperform the baseline implementation. Moreover, many of these combinations offer higher accuracy levels while preserving the GAHT's crucial energy efficiency.

5.2.1 Impact of the Node Splitting Criteria Variants

We will start by discussing the effect of both single and hybrid node splitting criteria.

Impact of Single Split Criterion Variants

The results support the hypotheses of the authors in [29], who claim that the basic combination of HB with IG can work quite well only when applied to exclusively numerical stream instances (this is exemplified on the forest fully numerical dataset). Additionally, the results of the GAHT HB+Gini index confirm the authors' statement in [38] that, unlike IG, Gini can be applied to both numerical and nominal data sets, thereby outperforming the HB+IG variants of GAHT.

In [39], both information gain(IG) and gini index (GI) provided approximate accuracy values in batch decision trees when applied to static data, irrespective of the binary/multiclass balance/imbalance scenarios. However, the current results demonstrated that such distributions do produce varying accuracy levels when the split criterion is applied within the HB for incremental online decision tree construction. Nevertheless, results in Table 3 indicate that when these two division criteria were combined with HB and applied (separately) to extremely unbalanced multiclass numerical data sets, namely poker, the Gini index displayed more stable divisions, which highlight the shortcomings of the HB+IG combination in managing class imbalance, even for numerical streams. Indeed, according to [32] and [39], the Gini index frequently generates a division that produces purer, better organized data subsets, a feature that would help mitigate class imbalance.

Results for the Electricity, HT-Sensor, and kdd datasets support the authors' hypothesis in [29] that when HB is applied with IG or Gini for tree construction on stream scenarios that encounter instances with highly nominal features (whether they be binary or multiclass severely imbalanced datasets), performances might fall off. Additionally, among single split criteria, preliminary trials demonstrated the superiority of the GAHT HB+McError with streams having an increased presence of nominal features, thus resolving the problem HB encounters with IG and Gini when classifying such streams. This is because, rather

than raising the criteria value as in the Gini, the McError is utilized as the basis for a split-measure function with the goal of improving the accuracy gain during the splitting process, as the authors note in [32].

Impact of Hybrid Split Criterion Variants

The hybrid GAHT MC+GINI node splitting criteria outperforms the exclusive use of McError or Gini. This confirms the hypothesis formulated in the article [32], which states that, combined with a second criterion, misclassification guarantees more optimal splits and prevents the drawback of McError described in [32], which consists in random splits if no attribute that maximizes accuracy levels is chosen. Consequently, the use of the Gini index avoids these random splits, which would then reduce the performance of the online decision tree.

5.2.2 Impact of Hoeffding Bound variants

According to the authors of [32] and [30], the use of the Hoeffding bound (HB) can still be considered a heuristic method that produces satisfactory practical results. An assumption verified by the results in the above Table 3. In the current subsection, we focus on the analysis of GAHT implementing the above node separation criteria with a set of HB variants, which have been shown in the literature to outperform HB. The performance of these variants is illustrated in Tables 4 and 5.

Impact of Half Hoeffding Bound

As mentioned earlier in the upper sections, real-world stream classification scenarios face the unique challenges of concept drift, which requires frequent model updates. The authors' statements in [30], pointing out that the usage of half HB tends to grow nodes rapidly, lead us to believe that this will help models to adapt quickly to this so-called concept drift. Consequently, the performance of the GAHT Half HB variants will be mainly evaluated by their results on non-stationary (evolving) data.

Table 3. Performances of GAHT Split Criterion Variants

Dataset	GAHT IG	GAHT Gini	GAHT Mc	GAHT Mc+IG	GAHT Mc+GI
Airlines	**61.14%**	63.68%	59.68%	61.24%	59.98%
	13.47s	16.53s	12.58s	15.50s	34.56s
	0.000196	0.000241	0.000183	0.000226	0.000505
Forest	**88.36%**	87.34%	87.68%	88.16%	87.92%
	39.85s	36.78s	37.65s	40.71s	37.86s
	0.000581	0.000536	0.000548	0.000593	0.000552
kdd	99.54%	99.60%	**99.83%**	99.69%	99.72%
	51.64s	52.51s	**47.53s**	51.54s	55.52s
	0.000753	0.000766	**0.000693**	0.000751	0.000809
poker	54.30%	58.28%	55.92%	54.57%	**59.19%**
	19.60s	14.45s	15.65s	23.49s	**18.46s**
	0.000286	0.000211	0.000228	0.000342	**0.000269**
HT sensor	99.64%	99.82%	**99.90%**	99.54%	99.83%
	16.50s	15.50s	**15.69s**	15.71s	15.48s
	0.000241	0.000226	**0.000229**	0.000229	0.000226
Electricity	86.10%	86.90%	**87.30%**	84.80%	86.80%
	03.64s	02.39s	**02.41s**	02.43s	03.40s
	0.000052	0.000035	**0.000035**	0.000035	0.000049
Insect Abrupt Imbal.	74.85%	75.00%	73.52%	74.75%	**75.52%**
	23.49s	22.55s	21.50s	22.63s	**21.50s**
	0.000342	0.000329	0.000313	0.000330	**0.000313**
Insect Gradual Imbal.	69.55%	70.55%	**71.60%**	69.15%	70.60%
	15.74s	13.59s	**12.67s**	13.79s	12.77s
	0.000229	0.000211	**0.000184**	0.000201	0.000186
Insect Incremental Imbal.	**76.64%**	76.06%	76.04%	76.44%	76.40%
	34.97s	30.54s	26.59s	32.66s	31.62s
	0.000509	0.000445	0.000387	0.000476	0.000461

Table 4. Performances of GAHT Half Hoeffding Bound Variants

Dataset	GAHT IG Half	GAHT Gini Half	GAHT Mc Half	GAHT Mc+IG Half	GAHT Mc+GI Half
Airlines	60.14% 13.46s 0.000196	61.18% 23.59s 0.000344	60.12% 17.60s 0.000256	62.74% 15.55s 0.000226	60.34% 27.51s 0.000401
Forest	88.32% 41.80s 0.000609	**89.12%** **37.80s** **0.000551**	87.90% 35.71s 0.000520	89.20% 38.94s 0.000567	89.06% 37.84s 0.000551
kdd	98.48% 47.50s 0.000692	99.68% 54.50s 0.000794	99.78% 50.62s 0.000738	**99.82%** **47.59s** **0.000693**	99.72% 55.52s 0.000809
poker	53.35% 18.85s 0.000272	51.29% 13.46s 0.000196	54.51% 17.48s 0.000255	55.33% 22.64s 0.000330	**57.57%** **15.70s** **0.000229**
HT sensor	99.73% 16.52s 0.000243	99.78% 16.64s 0.000242	99.94% 16.60s 0.000242	99.55% 15.48s 0.000225	**99.85%** **15.51s** **0.000226**
Electricity	86.10% 02.45s 0.000036	88.00% 03.47s 0.000053	85.70% 02.61s 0.000038	84.80% 02.59s 0.000038	**86.80%** **02.44s** **0.000035**
Insect Abrupt Imbal.	74.65% 24.62s 0.000359	73.17% 19.62s 0.000286	73.30% 17.47s 0.000255	75.15% 21.60s 0.000315	**75.50%** **20.64s** **0.000301**
Insect Gradual Imbal.	69.45% 13.49s 0.000197	**71.00%** **12.67s** **0.000185**	69.35% 12.56s 0.000183	69.15% 13.49s 0.000197	69.55% 13.51s 0.000197
Insect Incremental Imbal.	**76.90%** **34.79s** **0.000507**	76.54% 30.87s 0.000450	76.72% 29.60s 0.000431	76.70% 33.68s 0.000491	76.78% 28.60s 0.000417

Table 5. Performances of GAHT Double Hoeffding Bound Variants

Dataset	GAHT IG Double	GAHT Gini Double	GAHT Mc Double	GAHT Mc+IG Double	GAHT Mc+GI Double
Airlines	62.74% 13.47s 0.000196	**63.22%** **13.46s** **0.000196**	58.75% 12.24s 0.000179	62.74% 15.46s 0.000225	60.34% 27.48s 0.000401
Forest	**89.50%** **38.92s** **0.000567**	88.70% 37.82s 0.000551	88.14% 34.85s 0.000508	89.20% 38.91s 0.000567	89.06% 37.91s 0.000552
kdd	99.74% 49.51s 0.000723	**99.89%** **54.54s** **0.000795**	99.89% 51.61s 0.000752	99.82% 51.54s 0.000751	99.72% 56.50s 0.000823
poker	55.63% 18.53s 0.000270	55.49% 14.44s 0.000210	55.50% 15.70s 0.000229	55.33% 22.49s 0.000328	**57.57%** **15.46s** **0.000225**
HT sensor	99.65% 14.66s 0.000214	99.81% 15.47s 0.000225	**99.90%** **15.51s** **0.000226**	99.55% 15.70s 0.000229	99.85% 15.49s 0.000226
Electricity	85.80% 02.40s 0.000035	83.60% 02.41s 0.000035	**87.90%** **02.39s** **0.000035**	84.80% 02.66s 0.000039	86.80% 03.39s 0.000049
Insect Abrupt Imbal.	74.35% 21.50s 0.000313	74.67% 21.47s 0.000313	73.10% 20.92s 0.000305	75.15% 24.50s 0.000357	**75.50%** **19.49s** **0.000284**
Insect Gradual Imbal.	69.45% 13.71s 0.000200	67.90% 11.53s 0.000168	67.90% 10.66s 0.000155	69.15% 13.74s 0.000200	**69.55%** **11.65s** **0.000170**
Insect Incremental Imbal.	76.52% 30.62s 0.000446	73.84% 25.65s 0.000374	56.24% 18.57s 0.000271	76.70% 33.65s 0.000490	**76.78%** **30.64s** **0.000446**

The results show that such a bound will not significantly enhance model performances when processing drifting data. More specifically, when examining the behavior of each GAHT variant when applying the original HB method versus using the Half HB method, a considerable drop in accuracy was observed for both abrupt and progressive drift scenarios. Meanwhile, some GAHT variants, namely GAHT Gini Half, IG Half, and McError Half, demonstrated an improvement over using HB for stationary data namely electricity, forest, and HT_sensor, as well as incremental concept drift, which is thought to be a virtual drift in [40] (not as severe as the so-called real drifts). This leads us to call for systematic model growth and updates to handle real-world concept drift scenarios, thereby suggesting that stable splits are better suited to handling non-stationary data than fast, random splits.

Another key observation regarding the GAHT Half HB variants is the increase in CPU time, which is accompanied by an increase in energy consumption. This is explained by the fact that non-stable splits will cause the GAHT to revise splits more frequently due to the use of the EFDT paradigm within this latter; obviously, such revisions will include repeatedly calculating the split measures for all of the sets and seeking the optimum split attribute.

Impact of Double Hoeffding Bound

Doubling the Hoeffding bound confirms the need to perform splits with confidence when growing and updating an incremental decision tree, which is also confirmed by the results of the single and hybrid use of the GI split criteria with McError on stationary and evolving data streams. Not only would this HB variant ensure stable online trees, but it would also avoid GAHT executing several split revisions, as most of the executed splits will be more stable.

Table 6 summarizes the GAHT variants that outperform the basic GAHT IG HB implementation while maintaining the latter's energy efficiency by selecting the best of all the aforementioned GAHT variants (displayed in Tables 3, 4, and 5).

Table 6. Performant and Energy-Efficient GAHT Variants

Dataset	Performant Algorithm	Algorithm Performance	GAHT IG Performance
Airlines	GAHT Gini Double	63.22% 13.46s 0.000196kwh	61.14% 13.47s 0.000196kwh
Forest	GAHT IG Double	89.50% 38.92s 0.000567kwh	88.36% 39.85s 0.000581kwh
kdd	GAHT Mc Double	99.89% 51.61s 0.000752kwh	99.54% 51.64s 0.000753kwh
HT sensor	GAHT Mc Double	99.90% 15.51s 0.000226kwh	99.64% 16.50s 0.000241kwh
Electricity	GAHT Mc Double	87.90% 02.39s 0.000035kwh	86.10% 03.46s 0.000052kwh
poker	GAHT MC+Gini Hybrid	59.19% 18.46s 0.000269kwh	54.30% 19.60s 0.000286kwh
Insect Abrupt Imbalanced	GAHT Gini	75.00% 22.55s 0.000329kwh	74.85% 23.49s 0.000342kwh
Insect Gradual Imbalanced	GAHT Mc	71.60% 12.67s 0.000184kwh	69.55% 15.74s 0.000229kwh
Insect Incremental Imbalanced	GAHT IG Half	76.90% 34.79s 0.000507kwh	76.64% 34.97s 0.000509kwh

6 Conclusion

The present work focused on improving an energy-efficient incremental decision tree, namely the Green Accelerated Hoeffding Trees (GAHT), to effectively classify real-world data streams.

Standard GAHT achieves good performance only when used to process balanced, fully numerical data streams, a condition that is not always met in real-time applications. Consequently, our study introduces over 6 variants of GAHT, targeting its limitation induced by combining Hoeffding Bounds with the information gain (IG) criterion for incremental tree growth. These variants were obtained using different combinations of Hoeffding bounds and node splitting criteria, thus overcoming the initial limitations of the algorithm.

This study identifies multiple key insights, starting with the Gini Index's broad applicability and efficiency over the IG across various dataset types. Moreover, it was observed that GAHT misclassification error (McError) showcases better performances when used within nominal stream instances. It was also noted that the usage of a hybrid criterion combining McError with the GI demonstrates superior performance on multi-class datasets of varying dimensions. An additional insight was identified, revealing that GAHT, when combined with the standard HB, the latter establishes a reliable performance for any split criteria other than IG. This is in contrast to the double bound, which greatly enhances IG performances and is especially useful for guaranteeing GAHT stability and energy efficiency in the face of evolving data streams. Lastly, it was observed that half-bound emerges as a strong choice for speed-critical applications with stationary data streams; however, usage of this latter decreases GAHT energy efficiency if too many split revisions are encountered.

Considering the inherent unpredictability of data streams, we expect future work to focus on handling the dimensionality of the stream instances, as this will affect the predictive and energy efficiency of the GAHT variants. Furthermore, we believe that the use of active drift detection techniques will ensure a more reliable and efficient application of the model when dealing with drift situations in real streams, such as recurrent drift, for which the behavior of the GAHT algorithm has not been investigated.

References

[1] M. Naili, "Dynamic data mining based on the stability of dynamic models," Ph.D. dissertation, University of Mohamed Kheider-Biskra, 2019.

[2] W. Zhang and L. Zhao, "Online Decision Trees with Fairness," Oct. 2020, arXiv:2010.08146 [cs]. [Online]. Available: http://arxiv.org/abs/2010.08146

[3] H. I. Bensaoula, M. Lounici, and S. Nait Bahloul, "Improving High Dimensional Data Streams Classification," in *2024 4th International Conference on Embedded & Distributed Systems (EDiS)*, Nov. 2024, pp. 119–124. [Online]. Available: https://ieeexplore.ieee.org/document/10783244

[4] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000, pp. 71–80.

[5] C. Manapragada, G. I. Webb, and M. Salehi, "Extremely fast decision tree," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1953–1962.

[6] E. García-Martín, N. Lavesson, H. Grahn, E. Casalicchio, and V. Boeva, "Hoeffding trees with nmin adaptation," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2018, pp. 70–79.

[7] V. Turrisi da Costa, A. de Carvalho, and S. Barbon Junior, "Strict Very Fast Decision Tree: A memory conservative algorithm for data stream mining," *Pattern Recognition Letters*, vol. 116, pp. 22–28, Sep. 2018.

[8] E. Garcia-Martin, A. Bifet, N. Lavesson, R. König, and H. Linusson, "Green accelerated hoeffding tree," *arXiv preprint arXiv:2205.03184*, 2022.

[9] J. Gao, W. Fan, J. Han, and P. S. Yu, "A general framework for mining concept-drifting data streams with skewed distributions," in *Proceedings of the 7th SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics Publi-

cations, 2007, pp. 3–14.

[10] M. Bahri, A. Bifet, J. Gama, H. M. Gomes, and S. Maniu, "Data stream analysis: Foundations, major tasks and tools," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 11, no. 3, p. e1405, 2021.

[11] S. Kadam, "A survey on classification of concept drift with stream data," 2019. [Online]. Available: https://hal.science/hal-02062610v1

[12] richac, "Data Stream Mining: Techniques and Challenges," Apr. 2014. [Online]. Available: https://datasciencecmu.wordpress.com/2014/04/11/data-stream-mining-techniques-and-challenges/

[13] T. Gisselbrecht, "Bandit algorithms for real-time information gathering in social networks," Ph.D. dissertation, Sorbonne University/Pierre and Marie Curie University-Paris VI, 2017.

[14] B. Khouzam, "Incremental decision trees," Ph.D. dissertation, Master Thesis, Orange Labs, 2009.

[15] C. Salperwyck, V. Lemaire, and D. U. d. P. de Bois, "Arbres en ligne basés sur des statistiques d'ordre," in *Atelier CIDN de la conférence EGC*, 2012.

[16] S. Jia, "A vfdt algorithm optimization and application thereof in data stream classification," in *Journal of Physics: Conference Series*, vol. 1629, no. 1. IOP Publishing, 2020, p. 012027.

[17] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Conference on Knowledge Discovery in Data: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining; 26-29 Aug. 2001.* ACM, 2001, pp. 97–106.

[18] A. Bifet and R. Gavalda, "Adaptive learning from evolving data streams," in *International Symposium on Intelligent Data Analysis.* Springer, 2009, pp. 249–260.

[19] A. Pesaranghader, H. Viktor, and E. Paquet, "Reservoir of diverse adaptive learners and stacking fast hoeffding drift detection methods for evolving data streams," *Machine Learning*,

vol. 107, no. 11, pp. 1711–1743, 2018. [Online]. Available: https://doi.org/10.1007/s10994-018-5719-z

[20] C. Li, Y. Zhang, and X. Li, "Ocvfdt: one-class very fast decision tree for one-class classification of data streams," in *Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data*, 2009, pp. 79–86.

[21] K. Ramya, R. Priya, P. Sai, and N. Chandrasekhar, "Improved Decision tree algorithm for data streams with Concept-drift adaptation," 2012.

[22] H. Yang and S. Fong, "Incremental optimization mechanism for constructing a decision tree in data stream mining," *Mathematical problems in engineering*, vol. 2013, 2013.

[23] G. Liu, H.-r. Cheng, Z.-g. Qin, Q. Liu, and C.-x. Liu, "E-cvfdt: An improving cvfdt method for concept drift data stream," in *2013 International conference on communications, circuits and systems (ICCCAS)*, vol. 1. IEEE, 2013, pp. 315–318.

[24] N. Kourtellis, G. D. F. Morales, A. Bifet, and A. Murdopo, "Vht: Vertical hoeffding tree," in *2016 ieee international conference on big data (big data)*. IEEE, 2016, pp. 915–922.

[25] S. Desai, S. Roy, B. Patel, S. Purandare, and M. Kucheria, "Very fast decision tree (vfdt) algorithm on hadoop," in *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*. IEEE, 2016, pp. 1–7.

[26] C. Manapragada, M. Salehi, and G. I. Webb, "Extremely fast hoeffding adaptive tree," in *2022 IEEE International Conference on Data Mining (ICDM)*. Los Alamitos, CA, USA: IEEE Computer Society, dec 2022, pp. 319–328. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICDM54844.2022.00042

[27] A. Marascu, "Extracting sequential patterns from data streams," Ph.D. dissertation, Université Nice Sophia Antipolis, 2009.

[28] P. Matuszyk, G. Krempl, and M. Spiliopoulou, "Correcting the usage of the hoeffding inequality in stream mining," in *IDA*, 2013.

[29] L. Rutkowski, L. Pietruczuk, P. Duda, and M. Jaworski, "Decision

trees for mining data streams based on the mcdiarmid's bound," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1272–1279, 2013.

[30] M. Jaworski, P. Duda, and L. Rutkowski, "New splitting criteria for decision trees in stationary data streams," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2516–2529, 2017.

[31] S. Nait Bahloul, O. Abderrahim, A. I. B. Amar, and M. Y. Bouhedadja, "Improvement of data stream decision trees," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 18, no. 1, pp. 1–17, 2022.

[32] L. Rutkowski, M. Jaworski, L. Pietruczuk, and P. Duda, "A new method for data stream mining based on the misclassification error," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1048–1059, 2015.

[33] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "Moa: Massive online analysis, a framework for stream classification and clustering," in *Proceedings of the first workshop on applications of pattern analysis*. PMLR, 2010, pp. 44–50.

[34] J. Gama, R. Rocha, and P. Medas, "Accurate decision trees for mining high-speed data streams," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 523–528.

[35] S. Luccioni, T. Mallet, and S. Friedler, "CodeCarbon.io." [Online]. Available: https://codecarbon.io/

[36] V. M. A. Souza, D. M. d. Reis, A. G. Maletzke, and G. E. A. P. A. Batista, "Challenges in Benchmarking Stream Learning Algorithms with Real-world Data," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1805–1858, Nov. 2020, arXiv:2005.00113 [cs]. [Online]. Available: http://arxiv.org/abs/2005.00113

[37] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[38] L. Rutkowski, M. Jaworski, L. Pietruczuk, and P. Duda,

"Decision Trees for Mining Data Streams Based on the Gaussian Approximation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 108–119, Jan. 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6466324/

[39] S. Tangirala, "Evaluating the Impact of GINI Index and Information Gain on Classification using Decision Tree Classifier Algorithm*," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 2, 2020. [Online]. Available: http://thesai.org/Publications/ViewPaper?Volume=11&Issue=2&Code=IJACSA&SerialNo=77

[40] S. Agrahari and A. K. Singh, "Concept Drift Detection in Data Stream Mining : A literature review," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 10, pp. 9523–9540, Nov. 2022.

Hadjer Imene Bensaoula,
Sarah Nait Bahloul

Received November 10, 2024
Revised April 30, 2025
Accepted May 8, 2025

Hadjer Imene Bensaoula
ORCID: https://orcid.org/0009-0000-0592-733X
SIMPA Laboratory, Computer Science Department, University of Science and Technology of Oran Mohamed Boudiaf USTO-MB, Oran, Algeria
E–mail: hadjerimene.bensaoula@univ-usto.dz

Sarah Nait Bahloul
ORCID: https://orcid.org/0000-0001-9219-8381
LSSD Laboratory, Computer Science Department, University of Science and Technology of Oran Mohamed Boudiaf USTO-MB, Oran, Algeria
E–mail: sarah.naitbahloul@univ-usto.dz