Journal of Theoretical and Applied Information Technology
30th April 2026. Vol.104. No.8

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A DATA-DRIVEN APPROACH FOR EQUIPMENT FAILURE PREDICTION IN DATACENTRE NETWORKS

TAIWO AMOO[1,2] , VICTOR ODUMUYIWA[1], OLADIPUPO SENNAIKE[1]

[1]Department of Computer Science, University of Lagos, Lagos, Nigeria

[2]Department of Computer Science, Pan-Atlantic University, Lagos, Nigeria

E-mail: [1,2]tamoo@pau.edu.ng, [1]vodumuyiwa@unilag.edu.ng, [1]osennaike@unilag.edu.ng

ABSTRACT

A datacenter network comprises intermediary devices such as servers, routers, switches, etc., which are interconnected to allow for seamless exchange of information. These intermediary devices in the network are configured to record events occurring at any point in time, thereby depicting the current condition of these devices. However, these event logs, being massive datasets, are characterized by high volume, velocity and variety, making the manual-based analysis difficult. Recently, machine learning algorithms have been proposed in predicting failures using time-based sequential data such as log data, with Support Vector Machine (SVM) achieving remarkable success. Despite the robustness and high accuracy of SVM, it is not suitable for classifying large datasets due to the complexity of training relative to the volume of data and the high cost of manually labelling large datasets. To this end, a recent study employed SVM with a weakly supervised learning approach called Multi Instance Learning (MIL) to reduce the labelling effort required for large datasets in classification tasks. However, these MIL-SVM-based algorithms do not account for temporal information when addressing classification tasks. They also fail to account for the consistent shift in data distribution associated with big data, particularly in event logs, when selecting SVM hyperparameters during training for classification tasks. This work therefore proposed a Multi-Instance Selector-SVM (MIS-SVM) based on a multi-objective hyper-heuristics framework for failure prediction in a datacenter network. The Multi-Instance Selector (MIS) algorithm was developed to select likely positive and negative instance representatives per group of instances (bag) to reduce noise in both positive and negative bags. Subsequently, a hyper-heuristic framework using the lexicographic multi-objective genetic algorithm was developed to select competitive MIS-SVM hyperparameter values to improve generalization performance on time-based sequence data, such as logs. The proposed model was evaluated using three conflicting objectives —False Positive Rate (FPR), False Negative Rate (FNR), and Number of Support Vectors (NSV)—on three publicly available log datasets. The results showed that the proposed Hyper-heuristic-based MIS-SVM yielded the maximum p-value of 0.008 on M1 dataset, 0.001 on M2 dataset, and 0.005 on M3 dataset, with a p-value of 0.05 as the significance level. This demonstrated that the hyper-heuristic-based MIS-SVM is statistically superior to existing methods in the literature.

Keywords: *Machine Learning; Event Logs; Hyper-parameter Optimization; Multi-Instance Learning*

1.INTRODUCTION

Datacenter networks are infrastructure that comprises applications and various hardware components, such as servers, routers, switches, and storage devices, interconnected to facilitate the exchange of information. These networks are typically characterized by a large number of hardware components that enable them to carry data or offer services seamlessly. For example, Microsoft's Datacenter network deploys tens of thousands of switches to connect several servers, ranging from hundreds of thousands to millions [1]. The impact of failures in these networks can be costly, as diagnosing, identifying, and repairing them can take considerable time and resources, resulting in irreversible consequences such as service provider revenue loss and user service disruption. An example is Facebook's 6 hours of downtime in 2021, which cost roughly $60 million in lost revenue. Predicting failures is therefore essential, even when failure is inevitable; recovery actions can be done in advance to minimize the impact on users. However, as networks become increasingly complex, scalable, and heterogeneous, ensuring system reliability can be daunting for domain experts.

System logs have proven to be an essential source of data for failure prediction in datacenter networks. They record events at runtime within devices or applications to help understand their state. These log files document the operational status of network components by linking recorded events with their corresponding times of occurrence. However, one increasing challenge related to log-based failure prediction is its big-data nature, characterized by high volume, velocity, and variety of data per unit time. Therefore, existing works have not sufficiently addressed the significant costs of labelling these logs by domain experts [2], [3], [4], [5]. Additionally, the role of temporal information in logs for effective classification tasks has often been overlooked [6], [7], [4], [8]. For example, the sequence in which event log messages arrive can serve as a vital pattern for predicting failures. Lastly, the ongoing shift in log data distribution, mainly caused by device manufacturers' continuous updates to log messages, has not been adequately considered [5][9][10].

To address these shortcomings, a Multi-Instance Selector-SVM (MIS-SVM) based on a multi-objective hyper-heuristics framework for failure prediction in datacenter networks is proposed. The Multi-Instance Selector (MIS) is necessary to reduce the inherent noise in a weakly supervised setting. The use of hyper-heuristics over the other optimization methods will be considered using multi-objectives, such as False Positive Rate (FPR), False Negative Rate (FNR) and Number of Support Vectors (NSV) model complexity, to select competitive SVM configuration values. This is because when obtaining SVM configuration, hyper-heuristics can perform independently of the tasks at hand thus making them generalizable and applicable for different problem instances. Instead of single kernel type, multiple kernel types and their parameters will be explored by the hyper-heuristics before reaching the most competitive SVM configuration values. This study made the following contributions:

- This study formulated the SVM configuration as a multiple objective optimization problem in the presence of scarce labels (weakly supervised setting) for classification tasks.
- This study proposed an algorithm for selecting positive and negative instance representatives in a weakly supervised setting for classification tasks.
- The development of a hyper-heuristic framework that addresses the shift in distribution of data that can be associated with time-based sequence data.

This work is organized as follows. Section II presents reviews of existing work in system management, the state-of-the-art in multi-instance SVM methods, and hyper-heuristics. Section III formalizes the multi-instance learning, describes the proposed multi-instance selector support vector machine, followed by the formulation of the MIS-SVM process for failure prediction. Then, the configuration of the multi-objective MIS-SVM based on hyper-heuristics is explained. Section IV presents the proposed hyper-heuristic framework for selecting optimal configurations and the proposed MIS-SVM for positive and negative instance selection. Section V presents the experimental setup comprising of dataset description, results and discussion. Finally, Section VI presents the conclusion of this work.

2.RELATED WORKS

2.1 Network Management Approaches Based On Logs

Logs are files which contain data of system events at runtime for any production environment [13]. These events can range from a user logging in to the network to a failure experienced in the system/component. Typically, an event constitutes set of fields such as raw log message which depicts the state of the system at that time, date and timestamp, the device information issuing the event, time zone, and severity level. Typically, these logs are generated as unstructured texts by the system source codes using logging statements such as printf() and Console.Writeline() written by developers as shown in Figure 1 below. Then, they are usually stored locally or remotely (usually flat-file) as an event log. The log messages can prove to be very useful in reliability assurance tasks in large distributed systems since it represents the particular systems' behaviour over time. An example of system log event is shown below:

2017-06-20 08:46:59.936431 12.0.0.100 <164>24970: LC/0/0/CPU0:Jun 26 08:43:12.208 WAT: vic_0[370]: %PLATFORM-VIC-4-RX_LOS:InterfaceGigabitEthernet0/0/0/45, Detected Rx Loss of Signal

However, these event logs, being massive datasets, are characterized by high volume, velocity

and variety, making the manual-based analysis difficult. One approach to log management is using big data frameworks to speed up log processing by breaking large datasets into subsets to support system management. [14] proposed a cloud-based framework for large-scale log mining using Elasticsearch and Apache Spark. They leveraged Elasticsearch and Apache Spark to speed up the log mining process due to being compute-intensive and time-intensive. However, only basic analytics based on message frequency counts were applied to the log mining process.

Another significant method is event correlation, which uses predetermined heuristics to identify sequences of events that frequently occur (temporal patterns). These sequences of events (patterns) are subsequently used as predictive rules to predict failures. However, effective rules depend on the predefined time window used and require the entire configuration to be determined by domain experts. [11] proposed a heuristic algorithm named Timeslot Coverage Model (TiCom) for detecting multiple periods and their respective periodic patterns. [12] proposed an algorithm for mining periodic patterns in event logs using the minimum description length (MDL) criterion in evaluating the candidate patterns. However, this work is restricted to periodic patterns, sacrificing other patterns that could prove detrimental to the system.

Recently, machine learning methods have been utilized in automatic detection or prediction of failures in event logs. Machine Learning based methods aim to predict failures in systems ahead of time thereby proactively providing early warnings ahead for potential failures and reducing time to repair by system administrators. [13] developed a failure prediction approach based on log files using Random Indexing and Support Vector Machines. Random Indexing is applied to represent the sequences and sequences are associated to class of failures and non-failures. However, manual labelling was assumed from domain experts and performance measure was based only on accuracy. [2] proposed a failure prediction model by a supervised learning algorithm, Random Forest, using monitoring logs of switches in a datacenter network.

[3] developed a proactive failure management approach for cloud systems by comparing a number of supervised learning systems for failure prediction such as Support Vector Machines (SVM), Random Forest (RF), K-Nearest Neighbours (KNN), Classification and Regression Trees (CART) and Linear Discriminant Analysis (LDA). SVM was adjudged to be the best performing method for failure prediction. [14] proposed a novel deep-learning based prediction scheme for system-level hardware failure prediction. A novel temporal convolutional neural network was utilized by normalizing the distribution of attributes of the samples from different vendors which made it not sensitive to noise in the time dimension.

[15] proposed log-based anomaly detection approach by employing the mini-hash algorithm and the multi-vantage-point (MVP) tree for selecting the k clusters for the log messages. It only considered individual log messages and not the sequences of log messages in determining presence and absence of failure. They leveraged only system logs which assumed a manually labelled log data by domain experts. Additionally, performance was based on a single objective such as accuracy. [16] proposed a failure prediction algorithm based on multi-layer Bidirectional Long Short Term Memory (Bi-LSTM) to identify task and job failures in the cloud.

[17] proposed a failure detection system based on BERT and attention-based OLSTM (Optimized Long Short-Term Memory Networks) classifier for detecting failures in the IT infrastructure. The proposed algorithm is evaluated on five different infrastructures. [18] proposed a log analysis system based on natural language processing techniques for analyzing irregular system logs. The study proposed natural language processing techniques such as polarity score, Word2Vec, TF-IDF as vectorization techniques and conventional classifiers to analyze logs. The models' performance was evaluated using infrastructure logs.

[19] proposed a comparative study of the performance of six machine learning models for predicting job failures in a HPC system based on scheduler logs. The study showed that tree-based models performed better in terms of accuracy and computational cost. However, existing works have not considered the significant costs associated with labelling big data, such as logs, by domain experts.

2.2 Multi-instance Learning

A multi-instance learning approach is a form of weakly supervised learning that requires fewer labels. Formally, the task is to learn $f: X \rightarrow Y$ from a training dataset $D = \{(X_1, y_1), \ldots, (X_m, y_m)\}, where\ X_i = \{x_{i1}, \ldots, x_{im}\} \subseteq X$ is called a bag, $x_{ij} \in X_i (j \in \{1, \ldots, m\})$ is an instance of the bag, m is the number of instances in is an instance of the bag, m is the number of instances in X_i, and $y_i \in Y = \{+1, -1\}$. X_i is a positive bag if there exists an xij that is positive. The goal is to predict labels for unseen bags, a task known as Multi-Instance Learning (MIL). In the MIL setting, each training sample contains multiple instances, but only a single label, called the bag label, covers all the multiple instances. The actual label for each instance is unknown. As a result, each bag represents multiple instances, and each instance is described by a feature vector, but an associated label is never reported for each instance. The only information available about an instance, aside from its feature vector, is its membership relationship to a bag. MIL approach is adaptable to big data because it doesn't require training labels for each instance in the bag. [24] proposed a Standard Multi-Instance (SMI) hypothesis for drug activity prediction, which aims to build a classifier without the knowledge of individual instance labels for classification tasks. The SMI hypothesis, as shown in equation (1), states that a bag is labelled positive if and only if at least one instance in the bag is positive, and negative otherwise.

$$f(x) = \begin{cases} +1, & if\ \exists\ y_i = +1,\ \ \forall_i \in I \\ -1, & otherwise \end{cases} \quad (1)$$

In addition to the SMI hypothesis, some other existing works have proposed different types of hypothesis[20]. Due to the unknown true label of each instance in the bag, it is necessary to develop algorithms to select the true instance representative(s) of the bag label. Existing works can be majorly divided into four groups: Categories that leverage mapping functions to convert a bag into an instance (new vector space) to represent the bag, Categories that leverage building models directly on the instances in the bag, Categories that leverage lazy learners, Categories that leverage sequence data.

i. Categories that leverage mapping functions to convert a bag into a standard supervised learning task [21] [22] [23] [24] [25] [26] [27]. The instance (new vector space) is utilized to represent the bag. Therefore, the corresponding bag label becomes the instance label. However, the assumptions associated with this MIL category are not suited for sequence data where the temporal structure of the instances has to be considered.
ii. Categories that leverage building models directly on the instances in the bag. These involve the instances selected in each bag, and their respective bag labels are assigned to them. Altogether, these methods leverage iterations to select likely positive instances, as there is no certainty about the instance labels [18][19] [20] [21]. However, these works are not suitable for big data due to the number of iterations required to reach convergence.
iii. Categories that leverage lazy learners in solving multi-instance learning problems [28]. However, this MIL method performs poorly when the negative bags used for training differ from those used for testing. It is also not adaptable to time-based sequence data.
iv. Categories that leverage sequence data - [39] proposed two MIL approaches, ABClass and ABSim, for sequence data classification. The work aimed to address dependencies in sequence data across bags. However, it is not adaptable to time-based sequence data, as it doesn't account for the ordering of events associated with timestamps.

To date, studies identified as using SVM with the MIL approach for classification tasks have not considered the temporal structure of log event sequences. This can be critical to identifying proper sequence of disruptive events (failure signatures) in devices or applications in the network.

2.3 Hyperheuristics

Hyper-heuristics are approaches that leverage on search methods for selecting or designing heuristics for solving computational search problems [29]. To this end, they can perform independently of the tasks at hand, making them generalizable and applicable to different problem instances. Despite its qualities, in literature, works based on selection hyper-heuristics have been scarcely considered as a

multi-objective problem. Therefore, to date, [30] proposed the only work that has defined hyper-heuristics based SVM as a multi-objective problem. However, the proposed hyper-heuristics in their work only considered the traditional SVM approach, which is a supervised learning approach. This makes it unsuitable for Multi-Instance Learning SVM methods, as it violates their assumptions. In [31], the lexicographic multi-objective genetic algorithm is proposed for Fair Feature Selection (FFS), implemented in the proposed approach known as the Lexicographic multi-objective Genetic Algorithm for Fair Feature Selection (LGAFFS). The primary goal of LGAFFS is to select a subset of relevant features optimized for a given classification algorithm. This is achieved by simultaneously optimizing five measures. In another research [32] which involves Hierarchical Classification (HC) rule induction, specifically for classifying Transposable Elements (TEs) in Bioinformatics, using the Hierarchical Classification with a Lexicographic Genetic Algorithm (HC-LGA). HC-LGA is a novel Global method designed to address the challenging trade-off between accuracy and interpretability. HC-LGA utilizes the Multi-Objective Lexicographic approach to leverage defined priority orders for its objectives. When comparing two solutions (rules), the highest priority objective is checked first; if one solution is significantly better regarding that objective (exceeding a small threshold ϵ), it is chosen as the winner. If no significant difference is found, the next objective is evaluated. [33] addresses a Multi-Objective Multi-Port Fabric Dyeing Machine Planning Problem inspired by a real-life textile manufacturing challenge. The objective of the proposed Lexicographic Multi-Objective Genetic Algorithm (Lex-MOGA) is to minimize three conflicting objectives: Total tardiness (Obj1), total number of washes (Obj2), and total machine fixed cost (Obj3). However, these works were not utilised as hyper-heuristics.

3.PROBLEM DESCRIPTION

In this section, a formal representation of Multi-Instance Learning is described. The proposed Multi-Instance Selector Support Vector Machine is formulated for failure prediction. Next, the configuration of the proposed multi-objective MIS-SVM, based on hyper-heuristics, is explained.

3.1 Multi-instance Learning

Let D represent a training data set, $D=\{(X_1, y_1), ..., (X_m, y_m)\}$, where $Xi = \{x_{i1}, \ldots, x_{i,m}\} \subseteq X$ is called a bag,

$x_{ij} \in Xi$ ($j \in \{1, ..., m\}$) is an instance of the bag,

m is the number of instances in Xi, and

$y_i \in Y = \{+1,-1\}$ is binary label for bag i .

X_i is a positive bag if there exists an x_{ij} that is positive.

The objective is to learn a model f whose decision function $sgn(f(X_i))$ accurately predicts the label of a bag.

3.2 Multi-Instance Selector SVM (MIS-SVM)

The main idea of MIL-SVM is to select instance representatives that yield an optimal separating margin between the positive and negative bags. In the multi-instance learning approach, classifiers are optimized over bag labels instead of instance labels. In relation to the standard multi-instance assumption, only one instance out of all instances in a bag is required to be positive before an unknown bag label is classified as positive. The steps proposed to select the positive and negative instance representatives in each positive and negative bags are depicted in Algorithm 1. These selected instance representatives were used to train the MIS-SVM. In defining the notion of a bag margin, the MIL version of SVM is formulated as follows [34]:

$$\min_{w,b,\xi} \frac{1}{2} \| w \|^2 + C \sum_I \xi_I$$
$$s.t\ \forall I: Y_I \max_{i \in I} (\langle w, x_{sI} \rangle + b) \geq 1 - \xi_I, \xi_I \geq 0 \quad (2)$$

where,

$w = (w1, w2, w3, \ldots, wn)^T$ is the weight vector;
C is the margin parameter (or penalty);
x_{s_I} is instance representative of bag B_I;
S_I is a set of indices for the bag representatives;
ε is the insensitive loss coefficient, which controls the number of support vectors and optimizes over bag errors;
ξ and ξ_i^* are two slack variables, which take nonnegative values;
max() denotes that the margin of the positive bag is defined by the margin of the "most positive pattern" On the other hand, the margin of the negative bag is defined the "least negative" pattern.
Therefore, the samples are kernelized by replacing the inner product of paired samples with their

corresponding kernel values $K(x_I, x_K)$as shown in equation below:

$$K(x_I, x_K) = \langle \Phi(x), \Phi(x_i) \rangle \quad (3)$$

where

Φ(.) is the mapping function to the feature space. The kernel function is employed to calculate the dot product of two data points in the high-dimensional space. Some of the kernel functions used widely in SVM are shown in Table 1.

Table I: Kernel Functions

	Kernel Functions
	Formula
Radial	$K(x, x_i) = \exp(-\alpha\|x, x_i\|^2)$
Sigmoidal	$K(x, x_i) = (\alpha(x.x_i) + \beta)^d$
Polynomial	$K(x, x_i) = (\tanh(\alpha(x.x_i) + \beta)$

here are two types of kernel functions: local and global. An example of a local kernel function is the radial kernel function, which is known to have a good learning ability but poor generalization. A global kernel function example is the polynomial kernel, which is known to have good generalization but poor learning ability. A general problem in kernel selection is determining which kernel is suitable for a given problem instance. Therefore, this work considered different kernel functions in Table I to optimize over bag errors.

3.3 MIS-SVM Configuration Formulation

MIS-SVM setup defines the optimal values (configuration) for C, the kernel type, and its parameters. The goal is to identify SVM configurations from the set of all possible options that minimize the expected error when evaluated on entirely new data. This can be formulated as a black-box optimization problem aimed at minimizing the cross-validation error (I) and can be represented as a tuple in the form <MIS-SVM, Θ, D, C, S>, where

MIS-SVM – represents a learning algorithm

Θ - represents the range of possible SVM configurations, including C, kernel type, and kernel parameters.

D – distribution of the bags containing multiple instances

C – Cost function

S – Statistical information

Each Θ ∈ Θ represents a potential configuration of the SVM. The cost function C corresponds to a single execution of the MIS-SVM using Θ to solve a problem instance π ∈ D. The statistical information S (e.g., a mean value) summarizes the outcomes of C obtained when evaluating the SVM across multiple instances. The primary objective of the proposed hyper-heuristic framework is to identify a Θ ∈ Θ that optimizes C(Θ).

3.4 Multi-Objective Formulation

In a multi-objective optimization problem, two or more objectives' functions are defined that need to be simultaneously optimized. In the MIL-SVM, its accuracy can represent the trade-off between complexity (number of support vectors) and margin (C). Accuracy may not be suitable for imbalanced data such as logs since a large value for C to increase model's generalization ability may lead to an increase in misclassification (errors). In contrast, a large number of support vectors may lead to overfitting. In relation to failure prediction, there are usually rare positive instances that depict failures. Therefore, depending on accuracy as a measure of performance can be misleading. Therefore, in order to reduce classification errors (false alarms) while minimizing the model complexity of the SVM, a multi-objective formulation of the MIL SVM is proposed. The following objectives: false positives rate (FPR), false negative rate (FNR) and number of vectors (NSV) are formulated as conflicting objectives to optimize over bag errors for failure prediction in IT networks:

False Positive Rate (FPR): This is a measure of performance of a classifier which is calculated as the ratio between negative instances that were classified incorrectly as positive (false positives) and the total number of actual negative instances. This is also termed as probability of false alarm. This False Positive Rate criterion is more suitable for classification tasks when imbalanced data is being encountered. For the reason that it gives us true depiction of the performance of our classifier when negative instances are correctly classified. Therefore, a system with a high false positive rate will generate numerous false alarms for network operators, which can render the system impracticable.

The false positive rate, $FPR = \frac{FP}{FP+TN}$ where FP is the number of false positives, TN is the number of true negatives

False Negative Rate (FNR): This is a measure of performance of a classifier which is calculated as

the ratio between positive instances that were classified incorrectly as negatives (false negatives) and the total number of actual positive instances. In relation failure prediction, where there are usually more negative examples, classifier may be skewed towards negative examples which may not accurately classify positive examples.

The false negative rate, $FNR = \frac{FN}{FN+TP}$
where FN is the number of false negative, FN is the number of false negatives
Number of Support Vectors (NSV): represents the complexity or the upper bound of the expected number of errors.
A general multi-objective optimization problem of the minimization type can be represented as follows:

$$\min F(X) = [f_1(x), f_2(x), \dots, f_m(x)]$$
$$s.t. x_i^{(L)} \leq x_i \leq x_i^{(U)}$$
$$g_j(x) \geq 0 \; j = 1,2 \dots J \qquad (4)$$

where
X= (x1, x2, . . ., xN) is a set of N decision variables,
m is the number of objectives fi,
g is the number of constraints(X),
$x_i^{(L)}$ is the lower bound on the ith decision variable,
$x_i^{(U)}$ is the upper bound on the ith decision variable.
f1(x) is False positive rate
f2(x) is False negative rate
f3(x) is Number of support vectors

4.METHODOLOGY

The methodology consists of two parts: MIS-SVM and a Hyper-heuristic framework. The hyper-heuristic framework selects a competitive configuration (Margin parameter, kernel type, and kernel type parameters) that is passed to the MIS-SVM. On the other hand, the MIS-SVM selects a likely positive and negative instance from the training examples in the positive bag. Then, MIS-SVM receives the configuration from the hyper-heuristic framework and checks the magnitudes of error (True Positive, False Negative, and Number of support Vectors) over the training examples. The MIS-SVM output will be the objective function's values (True Positive Rate, False Negative Rate and Number of Support Vectors). This process is continued in a certain number of iterations. The proposed hyper-heuristic framework is discussed in the following subsections below:

4.1 Proposed Hyper-heuristics Framework

Hyper-heuristics are approaches that leverage on search methods for selecting or designing heuristics for solving computational search problems. Therefore, this work employed a choice function (CF) selection hyper-heuristic approach which selects a combination of pre-existing heuristics. The CF hyper-heuristic is employed to select evolutionary operators of a lexicographic multi-objective genetic algorithm for the evolution of competitive hyper-parameter values of the MIS-SVM. This is important in order to improve generalization performance and cater for consistent shift in data distribution. The following steps are shown below:

4.1.1 The Lexicographic Multi-Objective Genetic Algorithm

The Lexicographic Multi-Objective Genetic Algorithm (LMOGA) adapts the standard genetic algorithm (GA) framework by incorporating the idea of lexicographic ordering. Solutions are evolved by prioritizing conflicting objectives. It is a multi-objective optimization algorithm that seeks to focus on searching for solutions to optimization problems by hierarchically considering conflicting objectives, unlike Pareto-based methods that search for trade-off solutions. In LMOGA, objectives are given priorities, such that the objective with the highest priority is used to rank solutions to the optimization problem. In case of a tie, the objective with the next highest priority is considered, up until the tie is broken or the solutions cannot be separated in terms of all the considered objectives. Evaluation of solutions in this research is based on three conflicting objectives, False Positive Rate (FPR), False Negative Rate (FNR) and Number of Support Vectors (NSV).
This study proposes a hyper-heuristic framework that integrates a Lexicographic Multi-Objective Genetic Algorithm (LMOGA) to optimize hyper-parameter selection for Multiple-Instance Selector Support Vector Machines (MIS-SVM) applied to time-based sequence data, such as logs. The LMOGA adapts the standard genetic algorithm by employing lexicographic ordering to hierarchically prioritize three conflicting objectives: minimizing False Positive Rate (FPR), minimizing False Negative Rate (FNR), and reducing the Number of Support Vectors (NSV). Unlike Pareto-based methods that explore trade-offs, LMOGA ranks solutions based on the highest-priority objective, resolving ties by considering subsequent objectives

in order, ensuring focused optimization aligned with user-defined priorities. To navigate the solution space effectively, a choice function heuristic guides the selection of low-level heuristics at each search step. This function evaluates heuristics based on their individual impact (f1), pairwise sequential impact (f2), and time since last use (f3), enhancing the search's efficiency. The low-level heuristics include Parametrized Gaussian Mutation, three variants of Differential Mutation, Arithmetic Crossover, and Polynomial Mutation. The framework was evaluated on three publicly available log datasets, demonstrating improved generalization performance by balancing the competing objectives effectively

Solution Representation: In order to represent each solution, a one-dimensional array is employed as shown below:

$$[C, KF, k_1, k_2, \dots, k_F] \quad (5)$$

where,
C is the penalty parameter (or margin parameter),
KF is the index of the selected kernel function in Table I, and
k1, k2,..., kF are the parameters of that kernel function.
This single-dimensional array represents each solution/array as a single configuration of the SVM.

Population Initialization: In this step, population of solutions (PS) are randomly initialized. This is achieved using the equation (6) below to generate a random value to each decision variable in a given solution(x):

$$x_i^p = l_i^p + Rand_i^p(0,1) \times \left(u_i^p - l_i^p\right)$$
$$p = 1,2,\dots,|PS|,\ i=1,2,\dots,d \quad (6)$$

where
i is the index of the decision variable,
d is the total number of decision variables,
p is the index of the solution,
|PS| is the population size,

$Rand_i^p(0,1)$ returns a random value in the range [0,1] for the ith decision variable,

l_i^p is the lower bound on the value of that decision variable,
u_i^p is the upper bound

Solution Selection: This step entails the selection of solutions that will make it to the mating pool. In order to have diversity in the selection of solutions(parent) for mating, this step is achieved using the tournament selection method. This is achieved such that a defined number of individuals, k is selected randomly from the larger population and the best individual is selected from a set of randomly selected solutions in the parent population.

Fitness Calculation: This framework uses a lexicographic multi-objective approach for evaluating the fitness of the solution in the classification task. This is achieved by prioritizing the objective values, in this work, False Positive Rate is prioritized as the 1st objective, False Negative Rate as the second objective and NSV as the third objective. Therefore, the optimization procedure is solved one at a time to indicate the fitter individuals which is represented below:

$$in\ F_i\ (x)$$
$$subject\ to:$$
$$F_j\ (x)$$
$$\leq F_j\left(x_j^*\right); j=1,2,\dots,i-1; i>1$$
$$i = 1, 2, \dots, m, \quad (7)$$

i represents the order of the function, and $f_j(x_j^*)$ represents the optimal solution for the jth objective function from the jth iteration.

4.1.2 Choice Function Hyper-heuristics:

This serves to automate the heuristic selection process. The proposed high-level strategy is described below:

4.1.2.1 Heuristic Selection

In order to guide the solution to good areas of the solution space, a choice function heuristic method is utilised to select the low-level heuristics at each point in the search process. The choice function is an intelligent method for selecting heuristics that guide the search process to a good solution. It is composed of three terms: f1, which measures the individual impact of each low-level heuristic; f2, which measures the combined impact of using two heuristics successively; and f3, which measures the time taken since the heuristic was selected. The low-level heuristic with the highest score is chosen at each decision point using the choice function, F. Hence, the low-level heuristic is now applied to the solution. The exploitation phase of the search space is decided based on the

performance of the low-level heuristics using f1 and f2. In contrast, the exploration phase is defined based on f3, which determines the heuristics that have not been used recently.

The parameters (α,β, and γ) determines the weight of importance of each component,f1,f2 and f3 respectively which is dependent on the recent performance when the heuristic was applied [35] :

$$f_1(h_j) = \sum_n \alpha^{n-1} \frac{I_n(h_j)}{T_n(h_j)} \quad (8)$$

$$f_2(h_k,h_j) = \sum_n \beta^{n-1} \frac{I_n(h_k,h_j)}{T_n(h_k,h_j)} \quad (9)$$

$$f_3(h_j) = \tau(h_j) \quad (10)$$

Where

$I_n(h_j)$ represents the changes in the objective function;

low-level heuristics is based on the selection, recombination and mutation operators inherent in Genetic Algorithm (GA). Selection is the identification of the individuals(parents) that will be responsible for reproducing the next offspring. Tournament selection method is proposed.

Recombination is the process by which new solutions are generated by combining genes from parents. It is also known as crossover. In this work, three crossover operators are proposed: BLX-crossover, Arithmetic crossover, Linear Crossover and Unfair Average Crossover, Simulated binary crossover (SBX

Four types of mutation are proposed: Parametrized Gaussian Mutation, Differential Mutation 1, Differential Mutation 2, Polynomial Mutation. The following genetic operators are defined below:

1) Parametrized Gaussian Mutation

$$x = x + N(Mean, \sigma^2) \quad (14)$$

Where

Mean = 0 and σ^2 is the standard deviation.

2) Differential Mutation 1

$$X = x_1 + F \times (x_2 - x_3) \quad (15)$$

3) Differential Mutation 2

$$x = x_1 + F \times (x_2 - x_3) + F \times (x_4 - x_5) \quad (16)$$

4) Differential Mutation 3

$$x = x_1 + F \times (x_1 - x_2) + F \times (x_3 - x_4) \quad (17)$$

where

x1, x2, x3, x4 and x5 are five different solutions selected from the mating pool with respect to the solution selection process above. F is a scaling factor

5) Arithmetic Crossover

$$X = \lambda \times x_1 + (1 - \lambda) \times x_2 \quad (18)$$

where

λ is a randomly generated number, within the range

$\lambda \in [0,1]$

x1 is the first selected parent,

x2 is the second selected parent

6) Polynomial Mutation

$$x = \begin{cases} x_1 + \sigma \times (b - a), \text{if Rand} \leq 0.5 \\ x_1, \text{otherwise} \end{cases} \quad (20)$$

and

$$\sigma = \begin{cases} (2 \times \text{Rand})^{\frac{1}{(\eta+1)}} - 1, \text{if Rand} \leq 0.8 \\ 1 - (2 - 2 \times \text{Rand})^{\frac{1}{(\eta+1)}}, \text{otherwise} \end{cases} \quad (21)$$

where

η is a constant

a and b are the lower and upper bounds, respectively, on the value of the ith decision variable.

Based on the selection, recombination and mutation procedures of the GA, a total of six (5) low-level heuristics are proposed for the hyper-heuristics as shown in Table II.

4.2 Proposed Multi-Instance Selector SVM (MIS-SVM)

The proposed MIL approach employed a bag that contains multiple instances (daily logs sequence of the equipment ranging within a

duration of five days before failure as a rule of thumb) with only a single label being characterized as the bag label for all the instances. In reducing the input of domain experts, the presence or absence of failure, as indicated by the timing of notifications in the trouble ticket database, was used to determine the label of the bag. The following assumptions makes MIL approach suitable for the failure prediction problem in time-based sequence data [25]:

i) It is safer to assume that at least one daily log sequence within a short interval before the final breakdown of the component/system can usually contain the failure patterns that can help the domain experts. However, not all the daily logs within the interval preceding the failure will have failure patterns since the system could be alternating between good and bad states before finally breaking down. Therefore, not all instances within the pre-failure interval should be labelled as positive.

ii) It is assumed that daily logs within the non-failure interval are in normal state

Training: In the context of event logs, when formulating a multi-instance learning approach, an instance can represent the frequency of events within a time window. An example of an instance can be a frequency count of events that happened in one hour, one day, or one minute, depending on the rule of thumb. A bag can be defined as a set of instances over the entire duration, with the number of instances in a bag determined by a domain expert's experience. Although, there may be several of these instances contained in a bag, only one bag label will be attached. A positive bag indicates that one or more instances within it contain failure patterns. A binary label of 1 is assigned as the positive bag label. Meanwhile, a negative bag indicates that there is no instance of a failure pattern in the bag. A binary label of 0 is assigned as the negative bag label. Therefore, the proposed multi-instance selector selects the most likely positive instance representative from the positive bag and randomly selects a corresponding instance from the negative bag to address the problems of noisy instances and class imbalance. The steps are as follows in Algorithm 1 below:

Algorithm 1: Multi-Instance Positive Instance Selector

```
Input: ‘’
    - FailureTime   // Time of recorded failure from trouble ticket
    - LogSequences  // All available log message sequences
    - Interval_Positive = 5 days
    - Interval_SubSplit = 1 day
    - NegativeBags  // Set of candidate negative instances

- LCS //Longest Common Subsequence
Output:
    - PositiveInstance

 - NegativeInstance
Begin
   PositiveBag ← CreateSequences(LogSequences, Interval_Positive)
   SubSplitPeriod ← Interval_SubSplit
   InitialPositiveInstance ← FindClosestInstance(PositiveBag, FailureTime)
   SubSequences ←SplitSequence(InitialPositiveInstance, SubSplitPeriod)

   BestMatch ← NULL
   MaxLength ← -∞
   For each SubSeq in SubSequences do
     For each PosSeq in PositiveBag do
       LCS_Length ← ComputeLCS(SubSeq, PosSeq)
       If LCS_Length > MaxLength then
         MaxLength ← LCS_Length
             BestMatch ← LCS(SubSeq, PosSeq)// Store the actual subsequence
       EndIf
     EndFor
   EndFor
   PositiveInstance ← BestMatch
   NegativeInstance ← RandomSelect(NegativeBags)
   Return PositiveInstance, NegativeInstance
End
```

Table II: Low-Level Heuristics

	Low-Level Heuristics
	Low-level Heuristics (LH)
LH1	LH1: Tournament, Arithmetic Crossover, Parametrized Gaussian Mutation
LH2	LH2: Tournament, BLX Crossover, Polynomial

	Low-Level Heuristics
	Low-level Heuristics (LH)
	Mutation
LH3	LH3: Tournament, Simulated Binary Crossover (SBX), Differential Mutation 2
LH4	LH4: Tournament, Linear Crossover, Differential Mutation 3
LH5	H5: Tournament, Average Crossover, Differential Mutation 3

5.EXPERIMENTAL SETUP

This section provides a summary of the datasets used to evaluate the proposed framework.

5.1 Dataset

In this work, three (3) different datasets were utilized, which comprise system logs and failure tickets of three (3) different switch models over a span of two years. These system logs, collected from switches in several datacentres owned by a top global search engine, contain records of switch hardware failures. M1, M2, and M3 are used to collectively represent the three different switch models and their corresponding system logs and failure tickets collected over two years. In this work, syslog files with sizes less than 1 MB across all M1, M2, and M3 folders were employed. Detailed descriptions of the three different datasets are shown below:

Table III: Dataset Description

Parameter	Final Value
Number of generations	5
Population size	60
Crossover rate	0.5
Mutation rate	0.7

5.2 Parameter Settings

The framework requires specific parameters to be specified beforehand. To establish suitable values, an initial study was carried out. In this process, different options were tested for each parameter while the remaining ones were held constant. Table IV presents the final value of the parameter configurations explored during the study.

Table IV: Parameter Settings

S/N	DATASET DESCRIPTION		
	Datasets	*Total number of messages*	*Dataset Description*
1	M1 Dataset	It contains 1,203,827 log messages	It contains a total of 368 bag instances. (194 positive bags belonging to failure class and 174 negative bags belonging to non-failure class), 142 features in total.
2	M2 Dataset	It contains a total of 6,342,899 log messages	It contains a total of 202 bags. (101 positive bags belonging to failure class and 101 negative bags belonging to non-failure class), 105 features in total.
3	M3 Dataset	It contains a total of 249,439 log messages	It contains a total of 112 bags (64 positive bags belonging to failure class and 48 negative bags belonging to non-failure class), 153 features in total.

6.RESULTS AND DISCUSSION

In this section, the proposed framework (Multi-Instance Selector SVM based on Hyper-Heuristic framework) was evaluated and tested on three different datasets, each comprising system logs of switches from various vendors. Results from the proposed framework were compared in two different dimensions, which demonstrated contribution to the literature: Results of the proposed hyper-heuristic-based Multi-Instance Selector-SVM approach (HH-MIS-SVM) against each low-level heuristic (LH1-LH5) individually; Results of the proposed weakly supervised approach (HH-MIS-SVM) with other existing Multi-Instance Learning methods.

6.1 Comparison of the results of MIPIS-SVM against the results of all low-level heuristics (LH1 to LH5) individually

In this work, three (3) different datasets, M1, M2 and M3, were utilized to evaluate the performance of the proposed MIS-SVM against other multi-instance learning SVMs in the

literature. The following low-level heuristics that were considered are shown in Table II. The results of the proposed HH-MIS-SVM and other individual low-level heuristics were shown in Table V. Each of the individual low-level heuristics was evaluated against the combined HH-MIS-SVM using its average values of FPR, FNR, as shown in Table V. For the basis of comparison for both FPR and FNR metrics, a lower value of FPR or FNR signifies better performance. Additionally, a lower NSV represents a better performance. The algorithm with the best performance for each metric in each dataset is shown in bold in Table V. From the results, the HH-MIS-SVM algorithm outperformed all the individual low-level heuristics across all datasets based on FPR and FNR values. Additionally, as shown in Table V, the proposed HH-MIS-SVM yielded lower NSV values in all datasets, indicating better performance except for the M1 dataset, where it achieved the second-best result. These results justify the proposed hyper-heuristic-based method as an effective algorithm. This was further confirmed by conducting a Wilcoxon test on the performance values of the proposed HH-MIS-SVM relative to each of the individual low-level heuristics (LH1–LH5) in Table VII. In Table VII, a p-value less than 0.05 indicates that the proposed HH-MIS-SVM is statistically superior to the algorithm it is compared against. Whereas, a p-value that is greater than 0.05 indicates that the proposed HH-MIS-SVM is not statistically superior to the algorithm that it is being compared against. The Table VII showed that the proposed HH-MIS-SVM is statistically superior against all low-level heuristics that it was compared against across both M1,M2 and M3 datasets.

6.2 Comparison of Results of HH-MIPIS-SVM Against the results of Other Multi-Instance SVM Algorithms

The performance of the proposed HH-MIS-SVM against other multi-instance learning SVMs in the literature. The following multi-instance SVM algorithms were considered: Average-based MIL-SVM and Random-based MIL-SVM. The following datasets M1, M2 and M3 were used to demonstrate the performance of the proposed algorithm against each of the existing algorithms. The performance values of the algorithms are depicted in Table VI. Each algorithm was evaluated based on the average values of FPR, FNR, and NSR. For both FPR and FNR metrics, a lower value indicates better performance. Additionally, a lower NSV represents a better performance. The algorithm with the best performance for each metric under each dataset is shown in bold, as depicted in Table VI. From the results, the HH-MIS-SVM algorithm outperformed all other existing multi-instance learning SVM algorithms across all datasets based on FPR and FNR values, except for the FNR value in M3. Additionally, as shown in Table VI, the proposed HH-MIS-SVM yielded lower NSV values in all datasets, indicating better performance.

The experimental results confirmed that both average-based and random-based SVMs are ineffective for time-based sequence data, such as system logs. Three main properties make the proposed algorithm effective in the presence of scarce labels for classification tasks associated with time-based sequence data. First, the inclusion of multiple kernel functions and multi-objective criteria in evaluating the performance of the SVM classifier. Second, the consideration of the longest common subsequence method in removing noise from each positive bag. Third, the development of a hyper-heuristic framework capable of designing different SVM options for various datasets and at different stages of the solution process. This addresses the shift in data distribution that can be associated with time-based sequence data. Ultimately, improving the generalization of the algorithm to unseen examples

7.CONCLUSION

In this work, a hyper-heuristic framework based on multi-instance learning SVM was developed for classification tasks associated with time-based sequence data. A multiple objective optimization problem was formulated for the SVM configuration in which FPR, FNR and NSV were treated as three conflicting objectives. This multiple objective optimization problem was solved using the hyper-heuristic framework. The framework incorporates the lexicographic genetic algorithm in selecting the set of configurations for the SVM. The proposed framework was evaluated on three benchmark datasets collected as system logs from routers and switches in a data centre network. The results of this research showed that the proposed hyper-heuristic framework, based on a multi-instance learning SVM, outperforms other algorithms for classification tasks on time-based sequence data. The study can be continued by proposing an algorithm for handling multiclass classification tasks for time-based sequence data.

Table V: Comparison of Proposed HH-MIS-SVM Results Against the Results of Each Individual Low-Level Heuristics

	M1 dataset			M2 dataset			M3 dataset		
	FPR	***FNR***	***NSV***	***FPR***	***FNR***	***NSV***	***FPR***	***FNR***	***NSV***
LH1	0.68	0.12	147	0.442	0.064	78	0.12	0.08	38
LH2	0.88	0.07	148	0.594	0.024	77	0.192	0.13	39
LH3	0.45	0.262	124	0.498	0.274	75	0.136	0.066	37
LH4	0.92	0.212	164	0.464	0.318	78	1	0.11	41
LH5	0.664	0.186	**110**	0.498	0.22	76	0.166	0.08	38
Proposed Hyperheuristic-based MIS-SVM	**0.42**	**0.064**	120	**0.196**	**0.1**	**70**	**0.08**	**0.026**	**35**

Table VI: Comparison of Proposed HH-MIS-SVM Against Other Multi-Instance Learning SVM Algorithms

	M1 dataset			M2 dataset			M3 dataset		
	FPR	***FNR***	***NSV***	***FPR***	***FNR***	***NSV***	***FPR***	***FNR***	***NSV***
Average-based MIL-SVM	0.754	0.146	127	0.454	0.126	71	0.748	**0.012**	36
Random-based MIL-SVM	0.588	0.232	125	0.378	0.1	76	0.296	0.016	37
Proposed HH-MIS-SVM	**0.42**	**0.064**	**120**	**0.196**	**0.1**	**70**	**0.08**	0.026	**35**

Table VII: P-Values of HH_MIS_SVM Against Other Individual Low Level Heuristics

HH_MIS_SVM vs	M1 dataset	M2 dataset	M3 dataset
	p-value	***p-value***	***p-value***
LH1	0.001	0.001	0.001
LH2	0.001	0.001	0.002
LH3	0.008	0.001	0.005
LH4	0.001	0.001	0.001
LH5	0.001	0.001	0.001

FUNDING: This research was supported by the Tertiary Education Trust Fund (TETFund), Nigeria, under the 2021 TETFUND NATIONAL RESEARCH FUND (NRF) INTERVENTION (GRANT NO. TETFUND/DR&D/CE/NRF/2021/SETI/ICT/00155/VOL.1). The authors acknowledge TETFund's support with gratitude.

REFERENCES:

[1] Y. Zhao *et al.*, "Optical Switching Data Center Networks: Understanding Techniques and Challenges," *Comput. Networks Commun.*, pp. 1–16, 2023, doi: 10.37256/cnc.1220233159.

[2] S. Zhang *et al.*, "PreFix: Switch Failure Prediction in Datacenter Networks," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 2, no. 1, pp. 1–29, Apr. 2018, doi: 10.1145/3179405.

[3] B. Mohammed, I. Awan, H. Ugail, and M. Younas, "Failure prediction using machine learning in a virtualised HPC system and application," *Cluster Comput.*, vol. 1, 2019, doi: 10.1007/s10586-019-02917-1.

[4] T. Tu, "Log2Learn: Intelligent Log Analysis for Real-Time Network Optimization," in *4th International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID)*, Guangzhou: IEEE, 2025, pp. 162–166.

[5] C. Ji and H. Luo, "Leveraging Large Language Model for Intelligent Log Processing and Autonomous Debugging in Cloud AI Platforms," in *8th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*, IEEE, 2025, pp. 348–351.

[6] W. Wu, "Fault Detection and Prediction in Models: Optimizing Resource Usage in Cloud Infrastructure," in *4th International Conference on Intelligent Systems, Communications and Computer Networks*, ACM, 2025, pp. 131–138.

[7] J. Barach, "AI-Driven Causal Inference for Cross-Cloud Threat Detection Using Anonymized CloudTrail Logs," in *Conference on Artificial Intelligence x Multimedia (AIxMM)*, Laguna Hills, CA: IEEE, 2025, pp. 45–50.

[8] K. Qiu, Y. Zhang, Y. Feng, and F. Chen, "LogAnomEX: An Unsupervised Log Anomaly Detection Method Based on Electra-DP and Gated Bilinear Neural Networks," *J. Netw. Syst. Manag.*, vol. 33, no. 33, 2025.

[9] L. Zhang, T. Jia, M. Jia, Y. Wu, H. Liu, and Y. Li, "ScalaLog: Scalable Log-Based Failure Diagnosis Using LLM," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Hyderabad: IEEE, 2025, pp. 1–5.

[10] S. Taheri, A. Ihalage, Al-Raweshidy, P. Mishra, and S. C. F. M. Hamed, "Domain Tailored Large Language Models for Log Mask Prediction in Cellular Network Diagnostics," *IEEE Trans. Netw. Serv. Manag.*, vol. 22, no. 3, pp. 2370–2381, 2025.

[11] Q. Yuan, J. Shang, X. Cao, C. Zhang, X. Geng, and J. Han, "Detecting multiple periods and periodic patterns in event time sequences," *Int. Conf. Inf. Knowl. Manag. Proc.*, vol. Part F1318, pp. 617–626, 2017, doi: 10.1145/3132847.3133027.

[12] E. Galbrun, P. Cellier, N. Tatti, A. Termier, and B. Crémilleux, "Mining periodic patterns with a mdl criterion," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11052 LNAI, pp. 535–551, 2019, doi: 10.1007/978-3-030-10928-8_32.

[13] I. Fronza, A. Sillitti, G. Succi, M. Terho, and J. Vlasenko, "Failure prediction based on log files using Random Indexing and Support Vector Machines," *J. Syst. Softw.*, vol. 86, no. 1, pp. 2–11, Jan. 2013, doi: 10.1016/j.jss.2012.06.025.

[14] X. Sun *et al.*, "System-level hardware failure prediction using deep learning," *Proc. - Des. Autom. Conf.*, 2019, doi: 10.1145/3316781.3317918.

[15] B. Wang, S. Ying, and Z. Yang, "A Log-Based Anomaly Detection Method with Efficient Neighbor Searching and Automatic K Neighbor Selection," *Sci. Program.*, vol. 2020, 2020, doi: 10.1155/2020/4365356.

[16] J. Gao, H. Wang, and H. Shen, "Task Failure Prediction in Cloud Data Centers Using Deep Learning," *IEEE Trans. Serv. Comput.*, 2020, doi: 10.1109/TSC.2020.2993728.

[17] D. A. Bhanage, A. V. Pawar, K. Kotecha, and A. Abraham, "Failure Detection Using Semantic Analysis and Attention-Based Classifier Model for IT Infrastructure Log Data," *IEEE Access*, vol. 11, 2023.

[18] D. A. Bhanage and A. V. Pawar, "Improving Classification-Based Log Analysis Using Vectorization Techniques," in *Proceedings of Third International Conference on Advances in Computer Engineering and Communication Systems*, Springer, 2023.

[19] J.-W. Park, X. Huang, and C.-H. Lee, "Analyzing and predicting job failures from

HPC system log," *J. Supercomput.*, vol. 80, pp. 435–462, 2024.

[20] J. Foulds and E. Frank, "A review of multi-instance learning assumptions," *Knowl. Eng. Rev.*, vol. 25, no. 1, pp. 1–25, 2010.

[21] J. Wang and Y. Chen, "Image Categorization by Learning and Reasoning with Regions," *J. Mach. Learn. Res.*, vol. 5, pp. 913–939, 2004.

[22] Y. Chen, J. Bi, and J. Z. Wang, "MILES: Multiple-instance learning via embedded instance selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 1931–1947, 2006, doi: 10.1109/TPAMI.2006.248.

[23] W. J. Li and D. Y. Yeung, "MILD: Multiple-instance learning via disambiguation," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 1, pp. 76–89, 2010, doi: 10.1109/TKDE.2009.58.

[24] Z. Fu, A. Robles-Kelly, and J. Zhou, "MILIS: Multiple instance learning with instance selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 958–977, 2011, doi: 10.1109/TPAMI.2010.155.

[25] R. Sipos, D. Fradkin, F. Moerchen, and Z. Wang, "Log-based predictive maintenance," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, 2014, pp. 1867–1876. doi: 10.1145/2623330.2623340.

[26] J. Amores, "MILDE: Multiple instance learning by discriminative embedding," *Knowl. Inf. Syst.*, vol. 42, no. 2, pp. 381–407, 2015, doi: 10.1007/s10115-013-0711-1.

[27] J. Wu, S. Pan, X. Zhu, C. Zhang, and X. Wu, "Multi-instance learning with discriminative bag mapping," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 6, pp. 1065–1080, 2018, doi: 10.1109/TKDE.2017.2788430.

[28] J. Wang and J.-D. Zucker, "Solving Multiple-Instance Problem: A Lazy Learning Approach," in *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., 2000, pp. 1119–1125.

[29] N. Pillay and R. Qu, *Hyper- Heuristics : Theory and Applications*. Springer Nature Switzerland AG, 2018. doi:

https://doi.org/10.1007/978-3-319-96514-7.

[30] N. R. Sabar, X. Yi, and A. Song, "A Bi-objective Hyper-Heuristic Support Vector Machines for Big Data Cyber-Security," *IEEE Access*, vol. 6, pp. 10421–10431, Mar. 2018, doi: 10.1109/ACCESS.2018.2801792.

[31] J. Brookhouse and A. Freitas, "Fair feature selection with a lexicographic multi-objective genetic algorithm," in *International Conference on Parallel Problem Solving from Nature*, Cham: Springer International Publishing, 2022, pp. 151–163.

[32] P. H. Pereira, Gean Trindade Gabriel and R. Cerri, "A lexicographic genetic algorithm for hierarchical classification rule induction," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 846–854.

[33] Y. Demir, "An efficient lexicographic approach to solve multi-objective multi-port fabric dyeing machine planning problem," *Appl. Soft Comput.*, vol. 144, no. 110541, 2023.

[34] G. Melki, A. Cano, and S. Ventura, "MIRSVM: Multi-instance support vector machine with bag representatives," *Pattern Recognit.*, vol. 79, pp. 228–241, 2018, doi: 10.1016/j.patcog.2018.02.007.

[35] M. Maashi, E. Özcan, and G. Kendall, "A multi-objective hyper-heuristic based on choice function," *Expert Syst. Appl.*, vol. 41, no. 9, pp. 4475–4493, 2014, doi: 10.1016/j.eswa.2013.12.050.