Proceedings of the ASME 2023
International Mechanical Engineering Congress and Exposition
IMECE2023
October 29–November 2, 2023, New Orleans, Louisiana

IMECE2023-XXXX

MACHINE LEARNING BASED SEARCH FOR ACCESS POINTS IN ANOMALY DETECTION MODEL

Vishnu Gangadhara Naik[2,*] Tagir Fabarisov[1], Andrey Morozov[1],

[1]University of Stuttgart, Germany
{first.last}@ias.uni-stuttgart.de
[2]st176200@stud.uni-stuttgart.de

ABSTRACT

Heterogeneous systems are characterized by a high degree of complexity that poses challenges to conventional anomaly detection methods. One of the challenges is the production of large amounts of data where Deep Learning based Anomaly Detection Models (DLAD) have been demonstrated to outperform conventional techniques in detecting anomalies. However, DLAD models require the manual selection of Access Points, which are specific points in a system from which the training data is recorded. The selection of Access Points is a critical task that can significantly affect the performance of the anomaly detection models. It requires domain knowledge and expertise in the intricacies of the system, which is difficult to acquire and prone to human errors. As the size of the system grows, selecting Access Points becomes increasingly challenging.

In this paper, we propose a new machine learning-based algorithm called the Access Point Search Algorithm (APSA). The aim of APSA is to automate the task of finding the optimal set of Access Points that can aid DLAD models in detecting anomalies in a particular system. The algorithm utilizes a special error detector that dynamically takes in multiple signals and forecasts the signal values. The objective of finding the optimal set of Access Points is formulated as a feature selection problem, which is supervised using a binary variant of the Grasshopper Optimization Algorithm (GOA). We demonstrate the feasibility and effectiveness of the proposed algorithm by deploying it in a Simulink model.

The proposed algorithm was able to reduce 80% of the Access Points required for further anomaly detection using DLAD models. The Access Points selected by APSA showed a high probability of detecting anomalies over the Access Points that were not selected. We also illustrate the reliability of the proposed algorithm by feeding the signals from all the Access Points in the set provided by APSA into the DLAD model one by one. The reliability is further discussed by carrying out a fault injection experiment on the Simulink model which further proved the reliability of the proposed algorithm. Finally, we discuss the advantages and limitations of the proposed algorithm.

The results suggest that the proposed algorithm can efficiently and effectively select the optimal set of Access Points for DLAD models to detect anomalies in component signals. It offers a promising solution to automate the tedious and error-prone task of selecting Access Points, thereby reducing the domain knowledge and expertise required. However, the proposed algorithm also has certain limitations, such as its dependence on the quality and quantity of data and the assumptions made in formulating the feature selection problem. These limitations require further investigation to enhance the robustness and applicability of the proposed algorithm.

Keywords: Deep Learning, Anamoly Detector, cyber-physical system, machine learning, meta-heuristic optimization, fault detection, forecasting, feature selection

NOMENCLATURE

CPS	Cyber-physical systems
ICS	Industrial Control Systems
ITS	Intelligent Transportation Systems
ML	Machine Learning
DLAD	Deep Learning-based Anomaly Detection
NN	Neural Network
AP	Access Point
APSA	Access Point Search Algorithm
FS	Feature Selection
GOA	Grasshopper Optimization Algorithm
BGOA-S	Sigmoid variant of Binary GOA
ED	Error Detector
MSE	Mean Squared Error
LSTM	Long Short-Term Memory networks
MAE	Mean Absolute Error
RMSE	Root Mean Square Error

*Corresponding author: vishnugnaik@gmail.com

MAPE Mean Absolute Percentage Error
PI Proportional plus Integral
RPM Revolutions Per Minute

1. INTRODUCTION

Cyber-physical systems (CPS) are being deployed in most critical infrastructure applications where safety, reliability, and robustness are critical. They are used in a variety of applications, including Industrial Control Systems (ICS), Intelligent Transportation Systems (ITS), smart grids, and aerial systems. To provide rich functionality CPS are evolving into complex heterogeneous systems which are composed of various subsystems. To model, analyze, and program these subsystems, a formal model for each of them must be developed. Each of these formal models includes a computational mechanism that uses various components to realize the characteristics of the subsystem. These components have to communicate with one another and execute in a specific order. Thus the complexity of the control system increases as the functionality and size of sub-systems grow.

1.1 Threats in CPS

CPS face a wide range of threats [1], from network to physical, as a result of their heterogeneity and complexity, hence security in CPS should be emphasized. In addition to this if the data which is circulated inside the system gets corrupted due to fault it can cause the system to deviate from its expected behavior. Some of the outliers might resemble the correct data and can put the system at further risk. For instance, consider a case where sensor readings consist of anomalies, which resemble a particular status of a system. Now the system activates some set of actions based on the wrongful status of the system. Considering all these reasons, it is of utmost importance to mitigate these outliers and threats. The first step to achieving it is by detecting them using a novel method known as anomaly detection. The process of detecting an outlier in a dataset or stream of data is known as anomaly detection. It has gained traction in a wide range of applications [2], including intrusion detection and fraud detection. This work focuses on data science methods for anomaly detection in heterogeneous systems such as CPS.

1.2 Evolution of anomaly detection

Some of the earliest anomaly detection techniques relied on various methods for learning the normal status of the CPS [3]. Some of these include state and rule estimation, such as the Kalman filter. The more advanced model relied on statistical models such as Gaussian and histogram-based models. Any statistical model makes the general assumption that normal data always falls in the high probability region and anomaly data falls in the low probability region. To use this model, first, there is a need to determine or assume the underlying probability distribution, and then apply a statistical inference test to determine whether the unknown data falls into the low or high probability region. As can be seen, this method necessitates expert knowledge, where experts derive specific mathematical rules, or one must be aware of or assume the underlying distribution of normal data.

FIGURE 1: FAULT DETECTION PROCESS USING DLAD

Machine Learning (ML) based methods were introduced to detect anomalies that do not require mathematical expertise and instead, they learn the underlying distribution from labeled data. However, these techniques necessitate a large amount of labeled data and the feature must be manually extracted from the raw data [4]. Furthermore, they do not account for CPS characteristics such as spatial-temporal correlation, immutability, and program execution semantics. Various independent techniques should be employed to capture individual characteristics, which again demands high domain knowledge. As CPS become more complex, these individual overhead techniques become more complicated and, eventually, infeasible. Even after overcoming all of these disadvantages, we still face the challenge of adapting our detection techniques to new CPS characteristics.

To avoid these complications, Deep Learning-based Anomaly Detection (DLAD) methods for detecting anomalies in CPS have been introduced [2]. DLAD is an extension of the ML technique that eliminates the need for feature extraction. The data is fed directly into the Neural Network (NN), which automatically learns the underlying probability distribution. In some cases, this technique manages to outperform conventional Anomaly Detection methods, but it is highly dependent on the data that it is trained on. If the data is highly biased, the DLAD models will become ineffective as they are only trained on certain types of data, either normal or anomalous data. This reduces the performance of the DLAD models and forces us to find data that represents all of the system's information.

1.3 Identifying the problem

In this work, we will be referring to the point from which data is recorded in the system as Access Point (AP) shown in the dotted line in Fig.1. The general way of finding faults using DLAD is to train DLAD models on the system data recorded during normal operation. The trained model is then deployed in the system to detect faults in real-time by reading the system's operational data and the sensor signal as shown in Fig. 1. As the system gets more complex the footprint also increases due to which a high number of AP should be identified to reliably detect faults in the system. This generates an enormous amount of data and to keep up with this data size the complexity of the DLAD should be increased which leads to high inference time. To tackle this problem the number of APs required to efficiently detect a fault in a complex system should be decreased.

This is not a straightforward problem because the quality of data determines the efficiency and performance of the DLAD [5].

As a result, finding the right set of APs to detect system faults is critical. This activity necessitates extensive domain knowledge and experience, which is difficult to obtain and prone to human error. Hence an algorithmic method should be developed that can automate the process of identifying a set of optimal APs.

1.4 Our contribution

In this paper, an Access Point Search Algorithm (APSA) is introduced, which finds an effective set of APs that promote fault detection. The algorithm takes in a set of APs that are manually identified based on basic intuition and provides a set of APs that ensures better performance during fault detection.

2. STATE OF THE ART

A feature is an independent measurable property of a process under observation [6]. Using these features ML models can solve classification or regression problems. In the past few years, the number of features used for a particular application is expanding from tens to hundreds. In some cases for a fixed sample size, the performance of the models increases and slowly decreases as the number of features grows [7]. Hence finding an optimal set of features is a crucial task in order to obtain desired model performance. Feature Selection (FS) is a process where a subset of features is selected from an input feature set that can describe the input data efficiently and provide good prediction results [8].

To remove irrelevant features a selection criterion is required to measure the relevance of the feature to the output of the model. Based on the selection criteria used FS can be broadly divided into two types [9] as listed below:

1. **Filter method**: In this method, the relevance of the feature is determined using variable ranking techniques which are independent of the learning algorithm [6].

2. **Wrapper method**: This method utilizes the learning method to measure the relevance of the feature [10]. Various search algorithms are proposed in the literature to find a sub-optimal subset of features.

In general, the FS technique tries to reduce the feature set dimension to reduce the complexity of the prediction model and to promote performance. Even in the AP search problem, we are trying to reduce the AP set dimension to reduce the complexity of the anomaly detector thereby promoting error detection. Due to these similarities, FS techniques can be considered to solve the AP search problem.

3. FUNDAMENTAL RESEARCH OR PRELIMINARY CONCEPTS

3.1 Proposed Solution

In this paper, a wrapper-based FS process will be used to find the effective set of APs. Some common steps involved in FS and how they are directly adopted to solve our problem are shown in Fig. 2. Suitable methods are chosen to implement each of the steps involved in the FS process and an algorithm called AP Search Algorithm (APSA) is formulated. APSA is a search algorithm used to find an effective set of APs in a particular system. This algorithm is designed as a wrapper-based FS problem but here instead of features, we have APs. This problem has a very high-dimensional search space as we will be dealing with multiple APs. More details about this algorithm will be discussed in Section 4. According to [11], in order to find an optimal feature subset using the wrapper method, three factors listed below must be determined:

TABLE 1: CHOSEN METHODS FOR APSA

Component	Chosen methods
Selection technique	BGOA-S
Evaluator	ED
Selection criterion	Performance better than Global solution

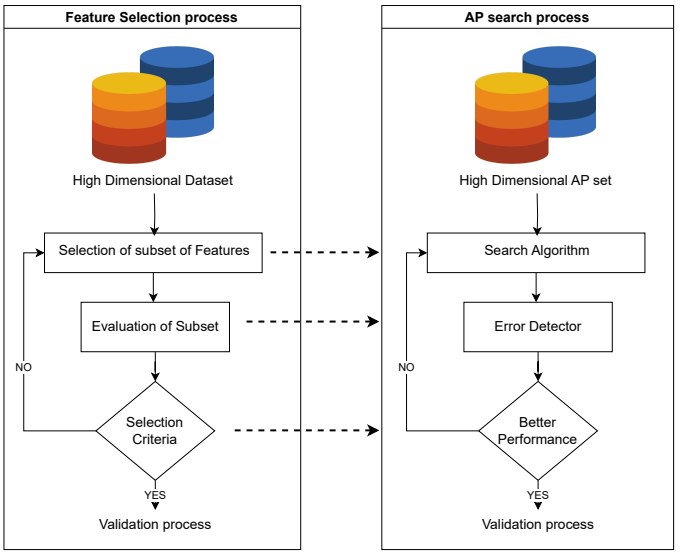


FIGURE 2: COMPARISION BETWEEN COMMON STEPS INVOLVED IN FS AND AP SEARCH PROCESS

1. **Searching technique**: To find a subset of features we will employ a meta-heuristic search algorithm; more information on searching techniques is provided in Section 3.2.

2. **Evaluator**: The evaluator evaluates the provided subset and provides the fitness factor. In this paper, we used the performance of the ED on the test dataset as the fitness factor, more details about ED will be discussed in Section 3.3.

3. **Selection criterion**: This factor determines whether or not the feature subset should be chosen. In practice, the evaluator's performance on the subset is evaluated, and a subset is chosen based on the decided selection criterion.

Table 1 shows the methods we chose in this paper to implement all the factors of the wrapper-based FS problem which in turn implements APSA.

3.2 Grasshopper Optimization Algorithm (GOA)

GOA [12] is a nature-inspired meta-heuristic optimization algorithm that mimics the Grasshopper behavior in a swarm.

The main characteristic of swarming grasshoppers is to search for food. This search algorithm can be used to find a subset of AP in our AP search problem. One of the distinguishing characteristics of any nature-inspired algorithm is that it has two main phases: exploration and exploitation. These phases assist the algorithm in increasing its convergence rate and avoiding local optima while searching for a target. During the exploitation process, search agents tend to move locally in the search space. However, they are encouraged to move quickly during the exploration process. Because grasshoppers naturally go through these two stages when looking for food, their swarm behavior can be modeled as an optimization algorithm. Equation (1) represents a mathematical model of grasshopper swarm behavior.

$$X_i^d = c_1 \left(\sum_{j=1, j\neq i}^{N} c_2 \frac{ub_d - lb_d}{2} s\left(|x_j^d - x_i^d|\right) \frac{x_j - x_i}{2} \right) + \widehat{T_d} \quad (1)$$

Here X_i^d is the position of the i$^{\text{th}}$ search agent or grasshopper in a search space. N is the number of grasshoppers, ub_d and lb_d are the lower and upper bounds of the positions in the search. $\widehat{T_d}$ is the target value or global best solution discovered so far in d$^{\text{th}}$ dimension. s is the social strength between search agents and it is given by Eq. (2). Where f is attraction intensity, l is attractive length scale and the author in [12] suggests $l = 1.5$ and $f = 0.5$ for optimal results.

$$s(r) = f e^{\frac{-r}{l}} - e^{-r} \quad (2)$$

The attraction between search agents will be controlled by the coefficient c as shown in Eq. (1) which is updated with the number of iterations. The c coefficient is calculated using Eq. (3). Where c_{max} and c_{min} are maximum and minimum values allowed for c, l is the current iteration number and L is the maximum iteration number.

$$c = c_{max} - l \left(\frac{c_{max} - c_{min}}{L} \right) \quad (3)$$

However, due to the nature of the AP search problem, where the search space is represented using binary values [0, 1], this algorithm cannot be used directly. To solve our problem, we must modify this algorithm to either a binary or a chaotic variant. In [13], the author has introduced 3 binary variants of GOA but we will be using the Sigmoid variant of Binary GOA (BGOA-S). The equation for calculating grasshopper position is modified as shown in Eq. (4). This equation is similar to Eq. (1) but the target vector is eliminated because the target vector in the AP search problem will be a set of binary values.

$$\Delta X = c_1 \left(\sum_{j=1, j\neq i}^{N} c_2 \frac{ub_d - lb_d}{2} s\left(|x_j^d - x_i^d|\right) \frac{x_j - x_i}{2} \right) \quad (4)$$

Equation (4) will be used to calculate the probability of updating the binary solution's element from 0 to 1. The probability is calculated using a sigmoid function as shown in Eq. (5)

$$T(\Delta X_t) = \frac{1}{1 + e^{-\Delta X_t}} \quad (5)$$

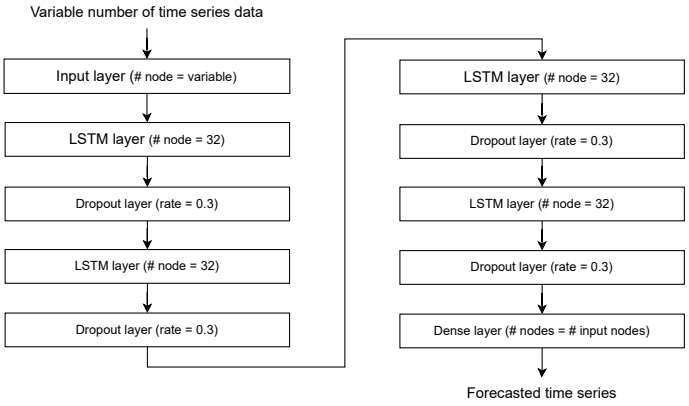


FIGURE 3: ARCHITECTURE OF THE IMPLEMENTED VARIABLE INPUT AND OUTPUT ED

The position of the current search agent or grasshopper will be updated using the criteria shown in Eq. (6).

$$X_{t+1}^d = \begin{cases} 1 & if\ r < T(\Delta X_{t+1}^d) \\ 0 & if\ r \geq T(\Delta X_{t+1}^d) \end{cases} \quad (6)$$

3.3 Error Detector (ED)

The prediction type anomaly detector's performance is dependent on the forecasting precision, hence an AP is considered effective when the data collected from that AP will provide better forecasting capability. To evaluate the subset of AP provided by the search algorithm a new ED is introduced in this paper. It is designed to be a single objective optimization problem and implemented as a multi-input/multi-output NN. ED is a predictive type of anomaly detector that takes in multiple input signals and forecasts all the input signals. As the size of the AP subset will be varying for every iteration we have designed this ED to change its input size dynamically on demand. The NN architecture of the proposed ED is shown in Fig. 3. The NN structure of the ED is intentionally kept complex to tackle the problem of under-training when a large number of signals are fed. The hyper-parameters used in this NN are shown in Table 2.

The proposed ED is written in Python, a widely used high-level general-purpose programming language. TensorFlow and Keras are Python libraries used to build the NN. The ED is implemented as a sequential model, which is the most common and basic type. The ED is implemented in Python as a class. A forecaster object with the desired input size can be created using this class. This aids in maintaining ED's dynamic size input features. The Python class also exposes a method that calculates and returns forecaster or ED performance metrics. PerMetrics is used internally by this method to calculate performance metrics. PerMetrics is a Python library for measuring the performance of machine learning models. More information about PerMetrics can be found in [14].

4. ACCESS POINT SEARCH ALGORITHM (APSA)

4.1 Methodology

Figure 4 depicts the proposed APSA in this paper. This algorithm is a minor modification of the BGOA-S optimization

TABLE 2: HYPER-PARAMETER VALUES FOR ED

Component	Hyper-parameter	Value
Hidden Layers	Layers	4 consecutive LSTM layers
	# of LSTM units	32
	Dropout rate	0.3
Output layer	Layers	1 dense layer
	Activation function	linear
	Dense units	variable
Optimizer	Type	Adam
	Learning rate	0.001
Objective function	Loss function	Mean Squared Error (MSE)
	Objective type	Minimize

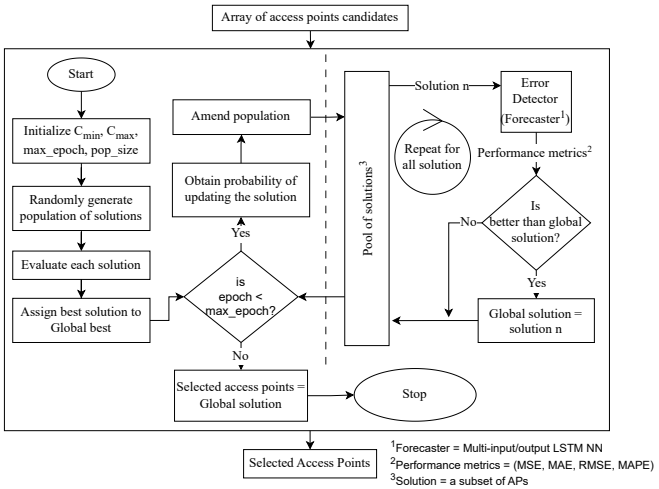


FIGURE 4: FLOWCHART OF APSA

algorithm introduced in [13]. The algorithm in BGOA-S was designed to solve a classification problem. However, the APSA introduced in this thesis work will be used for anomaly or fault detection. We should also keep in mind that we are dealing with time series data. The APSA will optimize the forecaster's objective function because we are using prediction-based anomaly detection. On a high level, the algorithm can be divided into two parts (shown in Figure 4 by a dotted line), each of which is responsible for:

1. Amending the search agent's position in a search space.
2. Evaluating the solutions against the error detector's objective function and updating the global solution

4.1.1 Description of algorithm. In the first part of the APSA, the algorithm starts by initializing hyperparameters, and these are chosen by the user of the APSA. Some of the hyper-parameters are related to the search algorithm GOA. In [12] the author presents value ranges and some suggested values for these hyper-parameters based on the experiments. In APSA, we will

TABLE 3: SUGGESTED VALUES FOR HYPER-PARAMETERS OF GOA

Hyper-parameter	Author suggested value	Suggested range
C_{min}	0.00001	[0.00001 - 0.2]
C_{max}	1	[0.2 - 5.0]
Maximum epoch	Not Suggested	[16 – 32]
Population size	Not Suggested	[10 – 10000]

TABLE 4: OPTIMIZATION AND OBJECTIVE OPTIONS SUPPORTED BY APSA

Optimization metrics	Objective option
MSE	Min or Max
Mean Absolute Error (MAE)	Min or Max
Root Mean Square Error (RMSE)	Min or Max
Mean Absolute Percentage Error (MAPE)	Min or Max

try to start with those values and fine-tune if the performance is not as expected. The exact values used for the experiment will be discussed in Section 6. Table 3 records the hyper-parameters range and suggested values by the author.

In addition to these GOA's hyper-parameters, the optimization metric and its corresponding objective should also be chosen. Table 4 shows the optimization metrics supported by the APSA, along with the respective objective options. APSA only supports regression metrics because as mentioned earlier we are trying to optimize the forecaster objective function and as we know forecasting is a regression problem.

Next, the algorithm moves to generate an initial population set which is a collection of random solutions. Each member of a population is a solution, and each solution is a set of APs in various combinations. The initial population set is then evaluated using the objective function that is ED. After each solution has been evaluated, the best solution is selected as the Global best. The global best solution is one that has a better objective measure, which means the lowest or highest of the optimization metric chosen. The next course action of this algorithm will be repeated until the current epoch number equals the maximum epoch number specified in the algorithm's initialization part. When the algorithm enters a loop, it computes the likelihood of updating the position of the search agent or solution. Equation (3) is used to calculate the value of parameter c based on the current epoch. The probability is then calculated using the transfer function discussed in Section 3.2 using Eq. (4) and (5) one after the other. This calculation is done for each element in the solution set and they are amended based on the criteria mentioned in Eq. (6).

At this point, the algorithm enters into the second part of the algorithm, which is critical because it assesses the quality of each solution. As previously stated, these solutions are a collection of APs, and each solution must be evaluated. First, the ED is trained by feeding all the AP's train datasets of a solution set. Then the ED is evaluated on the test datasets of the corresponding solution set and the performance metrics as

mentioned in Table 4 are recorded. As every solution set is a random combination of APs, the size of the solution set cannot be determined. At each epoch and population, the solution set will be of varying size. This necessitates the need for ED to be adaptable enough to accommodate various size solutions at various points. The proposed ED in this paper is useful in this situation. As mentioned in Section 3.3, the ED is a dynamic input forecaster. As a result, the ED can handle any number of APs by adjusting the number of inputs based on the size of the respective solution set at that point. Now the quality of each solution is determined based on the objective option chosen in the initial part of APSA for particular optimization metrics.

If a solution set outperforms the global solution, then the global solution will be replaced by that solution set; otherwise, the algorithm will skip this step. Better performance is determined by taking into account the objective, which is whether the solution under consideration has a lower or higher optimization metric than the global solution. In the first part of the algorithm, both the optimization metric and the objective are chosen. Whether or not the solution set is better than the global solution, the algorithm moves on to the next solution set in the pool. The cycle is then repeated until the evaluation of each solution set is completed. When it is finished, the algorithm breaks the cycle and moves on to the next step. The algorithm now compares the current epoch number to the maximum epoch chosen. If the current epoch exceeds the maximum epoch, the algorithm stops and returns the global solution set. By now it must be clear that this global solution set represents the most optimal APs available in the provided AP set. As a result, the AP set which is provided at the end by the APSA can be regarded as the most important APs in the system under consideration.

5. CASE STUDY

In this section, we will go over some theoretical details about the Engine Timing Simulation model that we will be using in this paper. The feasibility and effectiveness of the proposed algorithm, APSA, are assessed by carrying out some experiments on this simulation model.

5.1 Description

Engine Timing Model is a standard Simulink model available on the MATLAB official website. It simulates a four-cylinder internal combustion engine with spark ignition [15]. Using well-defined physical principles, this model will comprehensively model everything from the throttle to the crankshaft output. The dynamic behavior of the system was modeled using appropriate empirical relationships without adding unnecessary complexity to the model. In this paper, we will be using the closed-loop control variant of this model, which is an improved version of the engine timing model. Given the scope of this paper, we will not discuss the theoretical details of this model but in [16] more detailed description can be found.

This model is called `sldemo_enginewc`, and it includes a closed loop that makes use of the Simulink model's flexibility and extensibility. This model consists of an engine speed controller and a fast throttle actuator so that changes in load torque have little effect. This is easily accomplished in Simulink by including a discrete-time Proportional plus Integral (PI) controller in the engine model. As the operating point shifts, the integrator must adjust the steady-state throttle, and the proportional term compensates for the integrator's phase lag. Equation (7) is used to express this concept mathematically. Where, N_{set} and N are set-point and current speed in Revolutions Per Minute (RPM), K_P and K_I are proportional and integral gains.

$$\theta = K_P(N_{set} - N) + K_I \int (N_{set} - N)dt \tag{7}$$

TABLE 5: ENGINE SPEED SETPOINTS USED FOR THE EXPERIMENT

Time range (seconds)	Engine speed (RPM)
0 – 2	2000
2 – 4	3000
4 – 8	4000
8 – 12	5000
12 – 16	6000
16 – 20	5000
20 – 24	4000
24 – 30	2000

5.2 Modeling

Figure 5 depicts the Simulink model for a closed-loop engine timing system. Several subsystems are shown here, each representing a different aspect of an engine timing system. To simulate the microprocessor implementation, a discrete-time controller is used. As a result, the integral term in Equation (7) must be approximated using a discrete-time method. As is common in the industry, the controller execution is synchronized with the crankshaft rotation of the engine. The controller is embedded in a triggered subsystem that is triggered by the valve timing signal. It is a trigger subsystem that is activated when either the rising or falling edge of an actual engine speed is detected. The controller receives two inputs, actual engine speed and desired engine speed in RPM, and generates a throttle angle signal to compensate for the difference caused by the load torque.

The original model from the MATLAB official website had a step set point that incrementally gave desired RPM ranging from 2000 to 3000 RPM in 5-time unit intervals. This generated a simulation data set of 2,451 sample points in about 10 seconds. However, this is insufficient for training our ED model as it leads to under-training due to the high complexity of the model. To circumvent this issue, we need to collect more data. A dynamic set-point generator subsystem is designed to generate speed set-points as depicted in Table 5. After plugging this subsystem into the Engine timing simulation model and executing the simulation for about 30 seconds, 11,696 sample points were recorded.

6. EVALUATION AND RESULTS DISCUSSION

This chapter summarizes all of the findings from the experiments carried out during the evaluation of APSA. The evaluation is conducted in two phases as listed below:

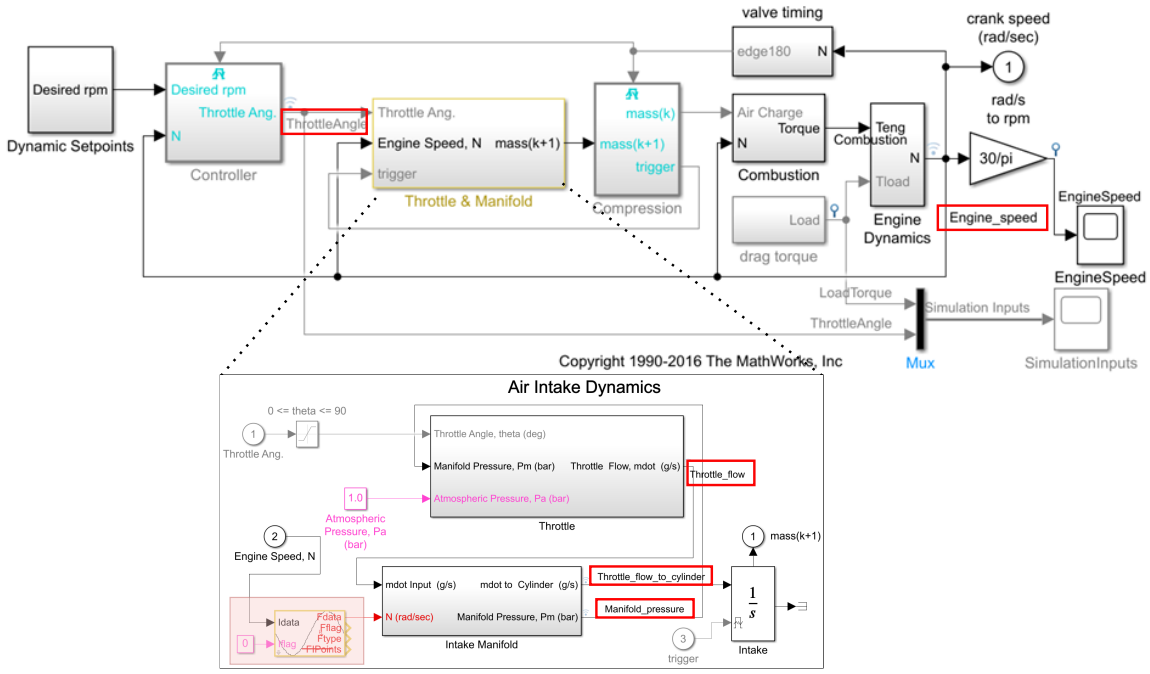


FIGURE 5: CLOSED-LOOP ENGINE TIMING SIMULATION MODEL WITH IDENTIFIED AP(S) (MARKED IN RED BOXES) AND FAULT INJECTOR FI-BLOCK (MARKED IN RED SHADED BOX)

```
[INFO] - >Problem: P, Epoch: 16, Current best: 0.0245, Global best: 0.0245, Runtime: 1263.13237 seconds
INFO:FeatureSelection.BinaryGOA.specialized_optimizer.BGOA_S.OriginalBGOAS:>
Problem: P, Epoch: 16, Current best: 0.0245, Global best: 0.0245, Runtime: 1263.13237 seconds
==========Results==========
Best features: ['Engine_speed']
Best performance result of selected metric: 0.0245
```

FIGURE 6: RESULTS OBTAINED FROM APSA FOR ENGINE TIMING MODEL (SELECTED AP(S) MARKED IN RED BOX)

Phase 1: Deploy APSA in the Simulation model and record results.

Phase 2: Test reliability of APSA

Both phases require data collected from the MATLAB Simulink software to determine whether the proposed algorithm is feasible and reliable.

6.1 Data Collection

Before we get into experiment setup and results it is crucial to understand the data collection process, as there are overwhelming multiple ways to do it. This is a critical step because the credibility of APSA is heavily reliant on the quality of the data we collect. MATLAB provides a variety of methods for collecting simulation data, each tailored to a specific application.

In this paper, we will be collecting data using the Signal logging method. In this method, signals are logged during the simulation and once the simulation is terminated the signals are exported to either an Excel sheet or a MAT file. This is the easiest of all other methods as it is very intuitive to the user. This method is well suited for our work as we need full simulation data and it is needed in some data structures.

6.2 Phase 1 Results

In this phase of evaluation, the APSA is deployed in the simulation model, and check the compatibility of the APSA, with the simulation model. The experiment is conducted following the below steps:

1. Identify potential APs in a Simulink model based on basic intuition.
2. Collect data from identified APs using the method discussed in Section 6.1.
3. Execute APSA on the collected data in step 2.
4. Record results provided by APSA in step 3.

The simulation model shown in Fig. 5 is used for this phase of the experiment and it is carried out without injecting faults (by disabling fault injector). We identified 5 APs on basic intuition that is supported by the theoretical explanation provided in [16]. Table 6 lists the identified APs and provides an explanation for each one. The APs chosen are marked in red boxes in Figure 5, and they are named as shown in Table 6. We will use the same nomenclature throughout the experiment because consistency in these names is critical because the APSA ultimately provides a set consisting of AP names. At that point, we should be able to correctly identify the APs to which the APSA refers.

The data is collected from the identified APs and as mentioned in Section 5.2 there are 11,696 sample points per AP. As we have identified 5 APs in total we are dealing with 58,480 data points. As discussed in Section 4.1.1, there are some hyperparameters of APSA that should be chosen before executing APSA. These parameters are chosen upon experimentation as shown in

TABLE 6: LIST OF IDENTIFIED APS IN ENGINE TIMING MODEL

AP name	Description
ThrottleAngle	The actuating signal coming out of controller
Engine_speed	The engine speed recorded out of Engine dynamics subsystem
Throttle_flow_to_cylinder	The throttle flow from the intake manifold to intake cylinder
Manifold_pressure	The pressure calculated by the intake manifold subsystem
Throttle_flow	The throttle flow calculated by throttle subsystem

TABLE 7: HYPER-PARAMETERS OF APSA

Hyper-parameter	Value
Cmin	0.00004
Cmax	1
Maximum epoch	16
Population size	10
Min or max problem	Min
Optimization metric	MSE
Input sample timestep	24
Output or forecasted sample timestep	24
Timeshift between input and output sample	24

Table 7. The APSA is executed on the data collected from all the APs to find an effective set of APs. The APSA ran for approximately 4 hours and 29 minutes and produced the results shown in Fig 6. As can be inferred from Fig 6, APSA provided a set with only 'Engine_speed' as an effective AP. This means that according to APSA, only the 'Engine_speed' signal is sufficient to detect a fault in the Engine timing model. This is almost an 80% reduction in the size of an AP set. The optimal MSE value recorded when an error detector is trained and tested on the data collected from 'Engine_speed' AP is 0.024. Engine speed is one such signal that is directly connected to all subsystems, so any fault appearing in any part of the system should eventually affect the engine speed signal. Thus, the results provided by APSA already look promising.

6.3 Phase 2 Results

In this phase of evaluation, the reliability of the results obtained from APSA is studied. Data collected from all APs will be fed into the DLAD one by one to evaluate and compare the results obtained on each AP. The experiment is conducted following the below steps:

1. The dataset is divided into individual set having only one AP's data.
2. This individual dataset is divided into train and test datasets.
3. Each individual AP's train set is used to train DLAD.
4. Once the training is completed, trained DLAD is tested on the respective test datasets.

TABLE 8: PERFORMANCE OF DLAD

AP name	R^2 Score	MSE
Engine_speed	**0.9876**	**0.0266**
Throttle_flow_to_cylinder	0.9005	0.0483
Throttle_flow	0.8913	0.0443
Manifold_pressure	0.8308	0.0558
ThrottleAngle	0.7635	0.0761

TABLE 9: FIBLOCK SPECIFICATION FOR FAULT INJECTION EXPERIMENT

Parameter	Value
Fault type	Stuck-at
Fault injection method	Deterministic
Event value (s)	8
Fault injection effect	Constant
Effect value (s)	1

5. The performance of these individual DLADs is recorded for further study.

We will not discuss technical detail about how DLAD is implemented as it falls out of this paper's scope. As we had earlier identified 5 APs, hence 5 DLADs were trained, and performance metrics like R^2 score and MSE were recorded as shown in Table 8 which is sorted in descending order based on R^2 scores. We can see that DLAD's performance on data from Engine_speed AP is far ahead of any other APs. In a prediction-type anomaly detector, the effectiveness of its fault detection directly relies on forecasting capabilities. Hence we can conclude that Engine_speed is an effective AP among others in this model due to its superior performance. In the previous phase, we saw that APSA also provided Engine_speed as an effective AP and thus the results obtained in this phase prove the reliability of APSA empirically.

6.4 Discussion on reliability of APSA

As stated earlier an AP is called effective if it has good forecasting capabilities and reacts to fault injection as quickly as it is injected. Hence the reliability of APSA can be further proved by conducting fault injection experiments. The fault can be injected in a simulation model using a custom Simulink block called FIBlock, introduced in [17].

The model shown in Figure 5 will be used by enabling the fault injector. The FIBlock is marked using a red transparent box in the "Air Intake Dynamics" subsystem. This subsystem is considered for study as most of the system's activities happen in this subsystem. The fault is injected in Engine_Speed signal which is an input to the "Intake manifold" subsystem. The fault generated by FIBlock will be configured as shown in Table 9.

As the fault is injected at 8^{th} time step, it makes sense to study in and around this time step. The majority of APs reacted to fault at the 8.2-time step, hence cropped images of the simulation output of the APs with and without fault are shown in Fig. 8. We can see from these images that out of all APs Engine_speed showed the most drastic reaction (3 units) to the injected fault.

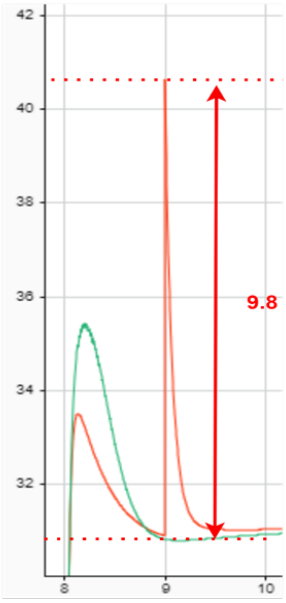


FIGURE 7: COMPARISON BETWEEN FAULT-FREE AND FAULT-INJECTED SIGNAL FROM THROTTLE_FLOW_TO_CYLINDER AP AROUND 9^{th} TIME-STEP

All other APs showed less impact for the injected fault, some APs showed so negligible reaction that would be neglected by the Anomaly detector. Of course, designing a very fine anomaly detector to catch these small interrupts is possible, but it may result in too many false positives.

Some APs showed high disruption in later simulation time steps. For instance, consider Throttle_flow_to_cylinder AP's simulation output at 9^{th} time step which is shown in Fig. 7. Here we can see that the AP showed a significant change in amplitude of about 9.8 units. If this AP is monitored for fault then the anomaly detector detects the fault one full second later which is not desirable in a time-critical system. In addition to this, it poses challenges in determining the fault origination time. Considering all these factors it is clear that Engine_speed is the most effective AP among others due to its strong sensitivity to fault injection and timely reflection of it in its amplitude.

7. LIMITATIONS OF APSA

Though APSA can significantly reduce the number of APs in any system, there are a few facts and limitations to this algorithm that should be considered. Some of these are listed below:

1. It is highly dependent on data quality, if there too many outliers then it might provide non-reliable results.
2. It only provides one set of APs which might be disadvantageous when the fault won't propagate to the selected AP.
3. When APSA provides a set with more than one AP, then individually these APs might not be effective.
4. It highly depends on the learning algorithm chosen.
5. The evaluator used in this work is only applicable to prediction-type anomaly detectors.

8. CONCLUSION

In a complex system the amount of data to be considered for fault detection while using DLAD is enormously high. To circumvent this issue the dimension of the data should be reduced which is done by reducing the number of APs considered for fault detection. To implement this a novel search algorithm called APSA is introduced in this paper. To make this idea feasible an error detector was also introduced. APSA was deployed in a conventional Simulink model and was found that it reduced the AP set by about 80%. The results provided by APSA were also evaluated for reliability using DLAD and Fault injection experiments.

9. SCOPE FOR FUTURE WORK

The approach of using feature selection to reduce the number of APs is only the beginning of a whole new field of research. Because feature selection has been a well-researched topic for a long time, there are numerous ways to incorporate those methods into AP search. Some advanced topics, such as chaotic variants of meta-heuristic techniques for solving feature selection problems can be adopted to get better performance. Various chaos properties, such as quasi-stochasticity, ergodicity, and sensitivity to initial conditions, can aid in boosting the performance of various meta-heuristics techniques [18].

One of the major disadvantages of APSA is that it only provides one set of APs. This can be improved by extending the capability of APSA to provide multiple APs with its effectiveness rating. This helps in choosing the next best effective AP set when one particular AP set doesn't provide the desired outcome. The ensemble method can be implemented to combine multiple feature selection techniques which provide different perspectives. APSA can be modified to support other types of anomaly detectors such as classification type, by simply replacing the suitable evaluator.

ACKNOWLEDGMENTS

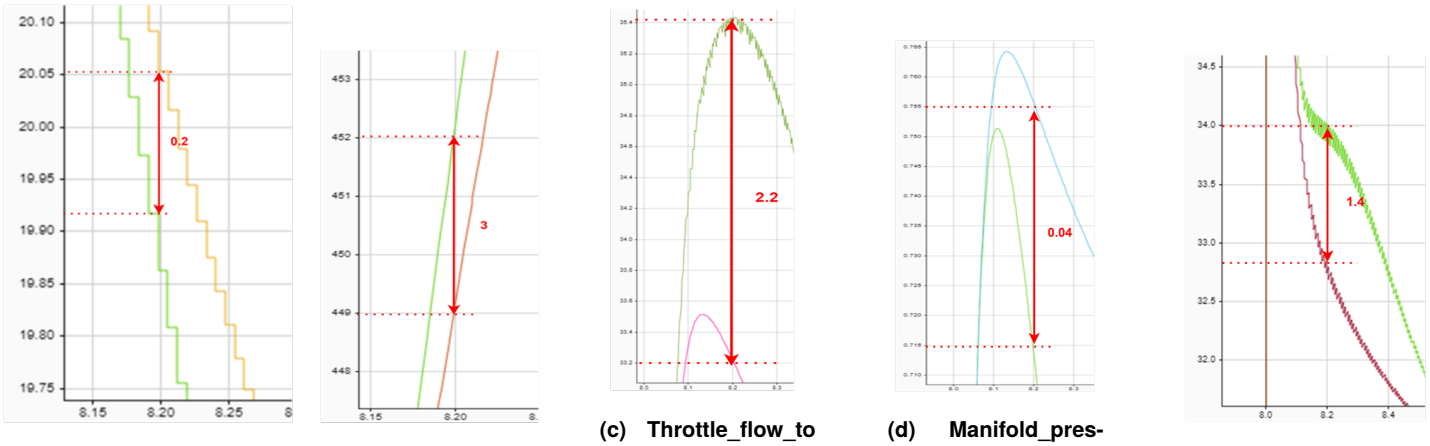
This research is supported by the Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (BAuA, Germany) funds, project F 2497.

REFERENCES

[1] Axelrod, C. Warren. "Managing the risks of cyber-physical systems." *2013 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*: pp. 1–6. 2013. DOI 10.1109/LISAT.2013.6578215.

[2] Luo, Yuan, Xiao, Ya, Cheng, Long, Peng, Guojun and Yao, Danfeng Daphne. "Deep Learning-Based Anomaly Detection in Cyber-Physical Systems: Progress and Opportunities." *CoRR* Vol. abs/2003.13213 (2020). URL 2003.13213, URL https://arxiv.org/abs/2003.13213.

[3] Zacchia Lun, Yuriy, D'Innocenzo, Alessandro, Smarra, Francesco, Malavolta, Ivano and Di Benedetto, Maria Domenica. "State of the art of cyber-physical systems security: An automatic control perspective." *Journal of Systems and Software* Vol. 149 (2019): pp. 174–216. DOI https://doi.org/10.1016/j.jss.2018.12.006.



(a) ThrottleAngle (b) Engine_speed (c) Throttle_flow_to_cylinder (d) Manifold_pressure (e) Throttle_flow

FIGURE 8: COMPARISON BETWEEN FAULT-FREE (GREEN COLOR) AND FAULT-INJECTED SIGNAL (NON-GREEN COLOR) FOR ALL THE AP(S) AROUND 8.2 TIME-STEP

URL https://www.sciencedirect.com/science/article/pii/S0164121218302681.

[4] Chandola, Varun, Banerjee, Arindam and Kumar, Vipin. "Anomaly Detection: A Survey." *ACM Comput. Surv.* Vol. 41 (2009). DOI 10.1145/1541880.1541882.

[5] Pang, Guansong, Shen, Chunhua, Cao, Longbing and Hengel, Anton Van Den. "Deep Learning for Anomaly Detection." *ACM Computing Surveys* Vol. 54 No. 2 (2021): pp. 1–38. DOI 10.1145/3439950. URL https://doi.org/10.1145%2F3439950.

[6] Chandrashekar, Girish and Sahin, Ferat. "A survey on feature selection methods." *Computers Electrical Engineering* Vol. 40 No. 1 (2014): pp. 16–28. DOI https://doi.org/10.1016/j.compeleceng.2013.11.024. URL https://www.sciencedirect.com/science/article/pii/S0045790613003066. 40th-year commemorative issue.

[7] Hua, Jianping, Xiong, Zixiang, Lowey, James, Suh, Edward and Dougherty, Edward R. "Optimal number of features as a function of sample size for various classification rules." *Bioinformatics* Vol. 21 No. 8 (2004): pp. 1509–1515. DOI 10.1093/bioinformatics/bti171. URL https://academic.oup.com/bioinformatics/article-pdf/21/8/1509/48973153/bioinformatics_21_8_1509.pdf, URL https://doi.org/10.1093/bioinformatics/bti171.

[8] Guyon, Isabelle and Elisseeff, André. "An introduction to variable and feature selection." *Journal of machine learning research* Vol. 3 No. Mar (2003): pp. 1157–1182.

[9] Liu, Huan and Motoda, Hiroshi. *Feature extraction, construction and selection: A data mining perspective*. Vol. 453. Springer Science & Business Media (1998).

[10] Kohavi, Ron and John, George H. "Wrappers for feature subset selection." *Artificial intelligence* Vol. 97 No. 1-2 (1997): pp. 273–324.

[11] Zarshenas, Amin and Suzuki, Kenji. "Binary coordinate ascent: An efficient optimization technique for feature subset selection for machine learning." *Knowledge-Based Systems* Vol. 110 (2016): pp. 191–201. DOI https://doi.org/10.1016/j.knosys.2016.07.026. URL https://www.sciencedirect.com/science/article/pii/S0950705116302416.

[12] Saremi, Shahrzad, Mirjalili, Seyedali and Lewis, Andrew. "Grasshopper Optimisation Algorithm: Theory and application." *Advances in Engineering Software* Vol. 105 (2017): pp. 30–47. DOI https://doi.org/10.1016/j.advengsoft.2017.01.004. URL https://www.sciencedirect.com/science/article/pii/S0965997816305646.

[13] Mafarja, Majdi, Aljarah, Ibrahim, Faris, Hossam, Hammouri, Abdelaziz I., Al-Zoubi, Ala' M. and Mirjalili, Seyedali. "Binary grasshopper optimisation algorithm approaches for feature selection problems." *Expert Systems with Applications* Vol. 117 (2019): pp. 267–286. DOI 10.1016/j.eswa.2018.09.015.

[14] Nguyen, Thieu. "A framework of PERformance METRICS (PerMetrics) for artificial intelligence models." (2020). DOI 10.5281/zenodo.3951205. URL https://doi.org/10.5281/zenodo.3951205.

[15] "Modeling engine timing using triggered subsystems." URL https://de.mathworks.com/help/simulink/slref/modeling-engine-timing-using-triggered-subsystems.html.

[16] "Engine timing model with closed loop control." URL https://de.mathworks.com/help/simulink/slref/engine-timing-model-with-closed-loop-control.html.

[17] Fabarisov, Tagir, Mamaev, Ilshat, Morozov, Andrey and Janschek, Klaus. "Model-based Fault Injection Experiments for the Safety Analysis of Exoskeleton System." 2020. DOI 10.3850/978-981-14-8593-0_5770-cd.

[18] Sharma, Manik and Kaur, Prableen. "A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem." *Arch. Comput. Methods Eng.* Vol. 28 No. 3 (2021): pp. 1103–1127.

[19] Chalapathy, Raghavendra and Chawla, Sanjay. "Deep Learning for Anomaly Detection: A Survey." (2019). URL 1901.03407.