IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 21, 2019, accepted September 18, 2019, date of publication September 23, 2019, date of current version October 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2942962

Self-Tune Linear Adaptive-Genetic Algorithm for Feature Selection

CHING SHENG OOI, MENG HEE LIM, AND MOHD SALMAN LEONG
Institute of Noise and Vibration, Universiti Teknologi Malaysia Kuala Lumpur, Kuala Lumpur 54100, Malaysia

Corresponding author: Ching Sheng Ooi (chingshengooi@gmail.com)

This work was supported in part by the Institute of Noise and Vibration UTM through the Higher Institution Centre of Excellence (HICoE) under Grant R.K130000.7809.4J226, Grant R.K130000.7809.4J227, and Grant R.K130000.7809.4J228, and in part by the UTM Research University under Grant Q.K130000.2543.11H36.

ABSTRACT Genetic algorithm (GA) is an established machine learning technique used for heuristic optimisation purposes. However, this natural selection-based technique is prone to premature convergence, especially of the local optimum event. The presence of stagnant performance is due to low population diversity and fixed genetic operator setting. Therefore, an adaptive algorithm, the Self-Tune Linear Adaptive-GA (STLA-GA), is presented in order to avoid suboptimal solutions in feature selection case studies. STLA-GA performs parameter tuning for mutation probability rate, population size, maximum generation number and novel convergence threshold while simultaneously updating the stopping criteria by adopting an exploration-exploitation cycle. The exploration-exploitation cycle embedded in STLA-GA is a function of the latest classifier performance. Compared to standard feature selection practice, the proposed STLA-GA delivers multi-fold benefits, including overcoming local optimum solutions, yielding higher feature subset reduction rates, removing manual parameter tuning, eliminating premature convergence and preventing excessive computational cost, which is due to unstable parameter tuning feedback.

INDEX TERMS Classification, exploration-exploitation cycle, feature selection, parameter tuning, STLA-GA.

I. INTRODUCTION

Genetic Algorithm (GA) is a superior machine learning technique inspired by Darwinism. Recent GA development has covered a wide range of applications, such as system identification [1], optimisation [2], [3], prognosis [4], data classification [5], feature selection [6], and image processing [7]. Nevertheless, the evolutionary searching nature of GA provides a double-edged sword: it is proven to be useful in executing heuristic optimisation but at the risk of premature convergence [8]. The tendency of local optimum occurrence is likely caused by a lack of diversity in candidate selection and static genetic operator setting. The genetic operator consists of selection, crossover, and mutation, together with GA stopping criteria. Moreover, the significance of tedious parameter tuning activity is unknown not only because a universal optimisation solution is unlikely but also due to the exponential increases in combination number with respect to the genetic parameters involved [9].

The associate editor coordinating the review of this manuscript and approving it for publication was Alberto Cano.

Given the local optimum likelihood, a GA parameter tuning methodology with reasonable computational cost is necessary. Thus, this case study introduces an adaptive GA feature selection technique, the Self-Tune Linear Adaptive-GA (STLA-GA) for classification purposes. Compared to existing parameter tuning algorithms, the uniqueness of STLA-GA is attributed to its fitness-performance-based adaptive nature. The ability to gradually prioritise search space exploration or exploitation activity within the allowable parameter range theoretically gives STLA-GA the upper hand in terms of flexibility, optimality and stability.

In the next section, recent research developments on GA parameter tuning are discussed. Then, a detailed explanation for a list of feature selection methodologies and classifiers are provided. Lastly, the performance results and optimisation techniques on four classification datasets are examined.

II. LITERATURE REVIEW: DEVELOPMENT OF MACHINE LEARNING PARAMETER TUNING ALGORITHM

Numerous machine learning improvement efforts have favoured the parameter tuning algorithm over conventional

manual tuning practice. For GA in particular, the main target of parameter tuning is to avoid premature convergence and slow finishing. In fact, a novel Dynamic Crossover GA (DCGA) employs multiple crossover operators instead of one in order to generate quality offspring [10]. The proposed adaptive mechanism reproduces the population ratio that corresponds to the fitness performance of respected crossover operators. The aforementioned methodology has been further extended into crossover and mutation probability rates with respect to targeted search space [11]. Furthermore, the Adaptive Operators Evolutionary Algorithm (AOEA) has substituted static genetic operator structure with the self-adaptive Genetic Programming (GP) tree [12]. The co-evolutionary technique manages the selection probabilities of a group of operators in a stepwise design until no changes arise (null operation). In addition, the Self-Configuring GA (SCGA) manipulates the probability value for genetic operators concurrently with reference to fitness function [13]. The Intuitionistic Fuzzy Logic (IFL) has been applied to the Multi-population GA (MpGA) in a top-down fashion in order to cross-validate the effectiveness of the genetic operators' sequence shuffling [14].

Meanwhile, the self-tuning GA advancement can eliminate GA parameter input tuning altogether. The removal of manual parameter tuning is achieved by introducing chromosome embedded evolving indicators in a closed-loop self-adaptive system. For example, the Self-Adaptive GA (SAGA) applies an individual-level encoder alternative to ensure that the desired crossover and mutation probability parameters are tuned to the uniqueness of the subjected problem statement [15]. A self-tuning GA process has been developed in order to regulate heuristic function parameters by performing problem characterization with mathematical equations [16]. The suggested technique has produced competent results without requiring an understanding of problem domains. A new aggregated voting parameter approach has been presented as an extra gene in the chromosome to realize a self-adaptive tournament size for population scale [17]. Moreover, the Bayesian belief network has been customized as an implicit probabilistic parameter value feedback mechanism based on fitness function performance [18]. The Bayesian Effect Assessment (BEA) adaptively controls automotive-related parameters by forecasting the usefulness of candidates' solutions using a novel success threshold.

On the other hand, the mutation probability rate, which numerically indicates the Population Movement Intensity (PMI), has been a major tuning target to accommodate the fitness landscape and multimodality [19]. To maintain genetic diversity while increasing feature selection searching efficiency, a genetic mutation operator, immigration, is invented to allow candidates to roam between island-based subpopulations [20]. Similarly, Adaptive Genetic Algorithms (AGAs) enable desirable initial weights for neural network learning by ensuring population variety with a simultaneous revision of crossover and mutation values [21]. Reference [22] has explained that a diverse initial population leads to an optimal solution for the Resource-Constrained Assembly Line Balancing Problem (RCALBP). The initialization of diverse population can be achieved by employing The Latest Workstation (TLW) rule-based method initialized population combined with an additional self-tuned anti-backlogging algorithm between crossover and mutation.

In recent years, appropriate self-tuning procedures has been implemented into other machine learning techniques. For instance, a unique Self-Adaptive Local Search (SALS) has been introduced for multi-objective optimisation purposes [23]. The SALS performs an automatic heuristic variable learning process to eliminate complicated parameter fine-tuning and to deliver performance enhancement. A fuzzy logic-tuned algorithm was executed to manipulate parameter setting iteratively, based on Particle Swarm Optimisation (PSO) performance [24]. As a result, the automatic customization of inertia and velocity ranges as well as cognitive and social values for an individual particle yields viable solutions with a faster convergence rate than standard evolutionary algorithms. An improved firefly algorithm was assigned to resolve multivariate non-linear equations with a real and complex number [25]. By benchmarking against GA and differential evolution, the firefly algorithm obtained precise roots with adaptive light intensity modification by referring to the finest firefly collections without a *priori* knowledge.

In conclusion, various efforts from previous research in parameter tuning algorithm, especially for GA heuristic function, has shed light on performance upgrade in an adaptive manner. The resolutions include control of genetic operator values and functions during simulation and the introduction of new heuristic evaluation tasks, operators and constraints. In general, the proposed parameter tuning methodology can be organized into three main categories: deterministic approach, adaptive approach and self-adaptive approach [26]. Nonetheless, a parameter tuning technique has yet to be developed to consider genetic operators and stopping criteria, even though the interdependency of both GA customized rules is vital to the stability of the parameter tuning mechanism, especially multi-objective optimisation.

We also observed that as STLA-GA parameter tuning requires additional computation process, the trade-off between computational effort and machine learning algorithm performance is non-trivial. Reference [27] has suggested that the population size is maintained at a constant value regardless of the number of operators because computational time is proportional to population size [28]. On the other hand, [15] has argued that a population size above 15 is more robust in preserving genetic variation. Furthermore, the robustness of the parameter tuning algorithm is doubtful since the fitness response is not only problem-specific but also dependent on adopted if-else conditions, criteria and search strategies. In addition, [29], [30] has noted that tuning the value of the crossover operator and small-scale population sizing is advantageous for exploitation, while a relatively large mutation rate and population pool are preferable when exploration is required.

TABLE 1. Confusion matrix: description.

		Actual Label 1	Actual Label 0	
Prediction	1	TP	FP	$\hat{P}$ = TP+FP
	0	FN	TN	$\hat{N}$ = FN+TN
		P = TP+FN	N = FP+TN	

TP = Actual label is Positive and is predicted correctly as Positive

TN = Actual label is Negative and is predicted correctly as Negative

FP = Actual label is Negative and is mislabelled as Positive

FN = Actual label is Positive and is mislabelled as Negative

TP+FN = Total Positive label P

FP+TN = Total Negative label N

TP+FP = Total predicted samples as Positive $\hat{P}$

TN+FN = Total predicted samples as Negative $\hat{N}$

III. METHODOLOGY

Supervised classification learning is generally designed in two phases: the classifier training stage and followed by classifier testing. During the training period, the designated classifier applies embedded learning function f to correlate *feat* training input features $x \in \mathbb{R}^{feat}$ and K unique output target labels $y \in \{1, 2, \cdots, K\}$. The classifier learning process $f : \mathbb{R}^{feat} \rightarrow \{1, 2, \cdots, K\}$ is conducted by supplying training feature vector sampling x_s to represent target label y_s at the instant of sampling time $s = \{1, 2, \cdots, N-1, N\}$ for N samples.

Upon satisfactory learning results, trained classifiers are given the testing feature vector to perform prediction according to the instant of sampling time. Prediction accuracy refers to the similarity between classifier-predicted target labels and testing-output target labels. By labelling the classifier-predicted output as $\hat{y}$, the misclassification rate *mis* for testing-fold dataset size N is measured as:

$$error,\ e = l(\hat{y}, y) \begin{cases} 1, & if\ \hat{y} \neq y \\ 0, & if\ otherwise \end{cases}$$
$$mis = \frac{1}{N} \sum e \tag{1}$$

Further comparison between y and $\hat{y}$ is possible with the confusion matrix. Assuming binary classification, two prediction outcomes are possible: Positive (1) and Negative (0). The confusion matrix provides an overview of the tabulation of N instance numbers for both decisions by arranging classifier prediction outcomes in rows and having actual output target labels correspond to columns. Table 1 outlines the confusion matrix structure and elements. The diagonal elements in a 2×2 confusion matrix present the number of correct predictions: True Positive (TP) and True Negative (TN). Off-diagonal elements in a confusion matrix produce two types of prediction error: False Positive (FP) and False Negative (FN). Classifier performance is then measured quantitatively with confusion matrix statistical analysis.

Nonetheless, because the feature quality is unknown, redundant input data may contribute to overfitting, which decreases prediction accuracy. Moreover, the feature subset combination number is an exponential function of the total feature number, 2^{feat}. This study's objective is to demonstrate the effectiveness of heuristic feature selection optimisation by resolving 'the curse of dimensionality'. Typically, the Genetic Algorithm (GA) heuristic function applies fitness function as the optimum target. Evidently, GA feature selection optimisation is multi-objective, which include but are not limited to optimal fitness function response with the most relevant feature subset. Thus, it is feasible to embed unique classifiers into GA to search for relevant feature subset $\mathbb{R}^{GA\,feat} \subset \mathbb{R}^{feat}$ which returns prediction error cost function $\hat{e}$.

$$\hat{e} = \underset{e}{\operatorname{argmin}} f(\mathbb{R}^{GA\,feat}) \tag{2}$$

The following subsections introduce the GA feature selection methodology and propose modifications in Self-Tune Linear Adaptive-GA (STLA-GA). Next, it briefly explains the theory of Particle Swarm Optimisation (PSO) and discusses a list of classifiers, including the Artificial Neural Network (ANN), k-Nearest Neighbors (k-NN) and Support Vector Machine (SVM).

A. STANDARD GA

The GA is designed to discover the optimum solutions within the *feat*-dimensional search space using the theory of evolution through natural selection. By means of natural selection, the GA repeatedly evaluates the current candidate population and revises the upcoming population based on the chosen candidates. The chosen candidates are those parents with the best score among the current population or are randomly selected candidates. For each successive generation, one or more chosen candidates are responsible for creating a pool of offspring with similar genes. The chosen candidate remains available within the population until it is replaced by offspring with a better fitness score.

Three standard genetic operators and stopping criteria are available to modulate the population. First, the selection operators choose the parents from the candidate population. Second, crossover operators combine the chromosomes of the parents. Third, mutation operators induce random perturbation on the chromosomes before passing them to the offspring. The GA simulation stops when the pre-set stopping criteria are met. The general stopping criteria are maximum generation numbers, maximum generation for static best score, and target score.

In terms of the feature selection problem, the candidate is represented by the chromosome CH as a binary string of numbers 0 and 1 ($CH \in \{0, 1\}$), with the exact length of the number of features *feat* ($CH \in \mathbb{R}^{feat}$). The number

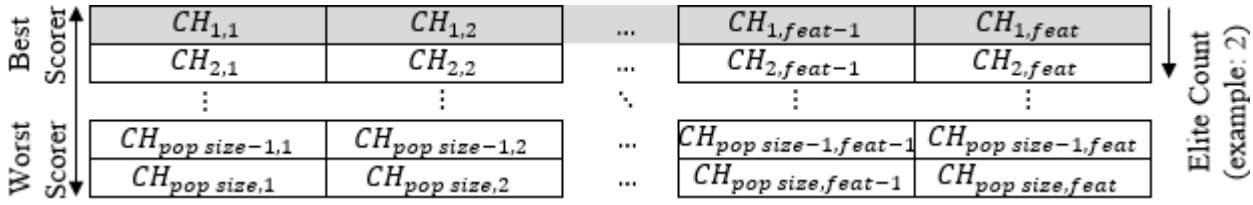

FIGURE 1. Tournament selection and elite count function.

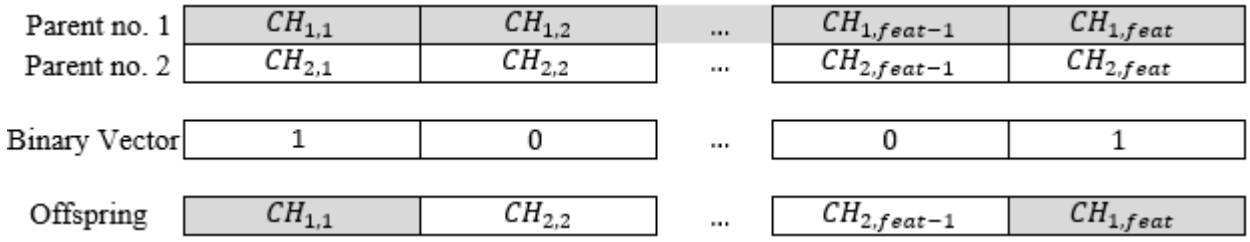

FIGURE 2. Scattered crossover function.

of candidates is labelled as *pop size*. The *tournament* selection function is employed to choose the best scorer as the parent from a candidate pool. The number of parents is determined using the *elite* count function value (see Fig. 1). The *scattered* crossover function is the most favorable to deal with a bit string population without linear constraints (see Fig. 2). Lastly, the mutation probability value designates the chance that a particular gene flips from its initial binary state. The probability of a mutation event occurring is denoted as $0 \leq P(mu) \leq 1$.

The product of the genetic operators is a group of updated binary sequence responsible for feature selection in next GA generation. Each and every feature to be considered will be assign to a binary element in the sequence. For ease of practice, the column number of a particular feature in the input matrix is equivalent to the element index number in the sequence. The feature is selected as part of feature subset if the corresponding element of a sequence is '1'. By contrast, element '0' indicates the feature is excluded as a member of feature subset. With feature subset as input, the target of the fitness function in GA is to minimise prediction error to reach as close to zero as possible (i.e., global optimal solution).

Standard GA practice applies a static genetic parameter setting over generations. The initial populated candidates are randomly generated to enable equal distribution within the sparse search space, as the location of the global best score is unknown. The subsequent populated candidates generated with low probability mutation rate is expected to search within the local surrounding of the best score. During the occurrence of local convergence, it is considered to be redundant when mostly identical populated candidates repeatedly exploit a narrow search space for generations. On the other hand, relatively large population size and high mutation probability rate are advantageous for search space exploration. Nonetheless, the result of a large evaluation number is comparatively computation expensive and elitism tracking that will be ignored if the mutation event happens frequently.

B. STLA-GA

As the name implies, STLA-GA alters typical genetic operator parameters and unique thresholds in an adaptive fashion in order to acquire multi-objective optimisation. The parameter tuning flow sequence begins with the initialization of the conservative parameter for mutation probability rate (*mu*), population size (*pop size*), generation limit (*opt gen*), and convergence threshold values. Conservative initialization at generation $iter = 0$ is an attempt to exploit the initial elite child to achieves the global optimum value with minimal computation cost.

After allowing convergence for a predetermined candidate evaluation number, defined as the *threshold*, STLA-GA re-evaluates a list of related parameters upon static global optimal detection or new global best value update. In other words, classifier performance over simulation time serves as the STLA-GA parameter-setting indicator. Assuming local convergence occurs with the presence of stagnant performance over 2 GA generations, STLA-GA shifts to an exploration setting via a set of linear additive gain increment control equations (3) - (5). By means of an exploration setting, STLA-GA aims to improve population diversity and extend the search space by gradually increasing the population size and mutation probability rate.

$$gain = \frac{mu_{max} - mu_{iter}}{opt\ gen - iter + 1}$$
$$mu_{iter+1} = mu_{iter} + gain \quad (3)$$
$$pop\ size_{iter+1} = pop\ size_{iter} + gain \times ga_{iter} \quad (4)$$
$$threshold = pop\ size + threshold \quad (5)$$

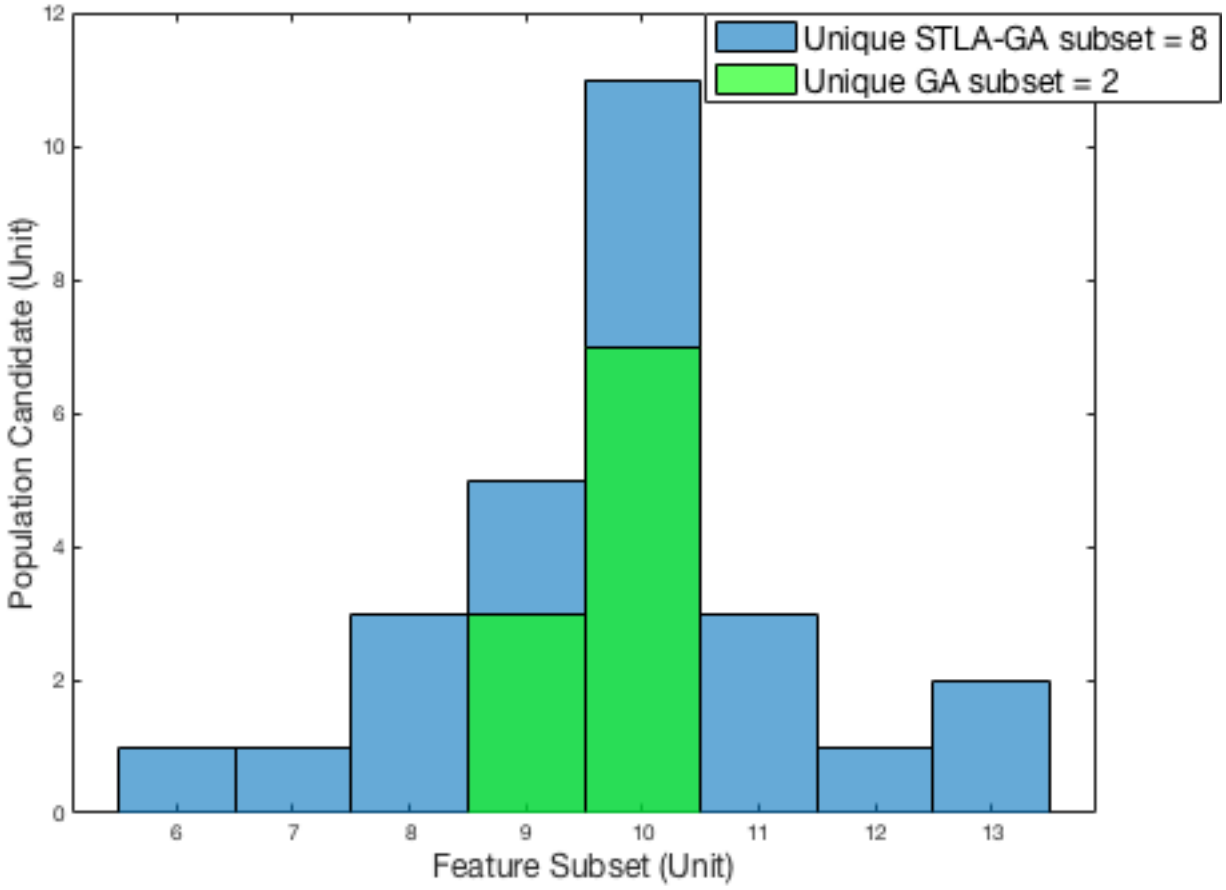

FIGURE 3. Feature subset number from the population of the last generation.

The significance of linear additive gain increment control equations is illustrated by comparing the candidate population of a 17-feature dataset using two GA methods. Standard GA adopts static conservative parameter setting, while STLA-GA uses exploration-exploitation cycle-embedded parameter tuning framework. When convergence occurs at the end of the simulation, STLA-GA opts for the exploration setting (higher number of population candidates and mutation probability) rather than settling for identical GA parameters over generations. Fig. 3 demonstrates the histogram for the feature subset size available from the population generated by the GA methods at the end of simulation. The histogram's bar colours are set as blue and green to symbolise the feature subset size produced by STLA-GA and Standard GA, respectively. As Fig. 3 shows, feature subset size extracted from the last STLA-GA population varies in contrast to the last Standard GA population. The STLA-GA exploration setting creates diverse populated candidates, ranging from 6 to 13 in feature subset size, while Standard GA population members are either 9 or 10 features. The unique feature subset size is 8 for STLA-GA and 2 for Standard GA. Population diversity involving STLA-GA is maintained at an acceptable level, which is superior in global best solution discovery to the Standard GA population.

Next, the exploitation setting is initiated upon an update of the global best score. Exploitation settings engage initial conservative GA parameters, and they aim to capitalize on the potential new global best solutions available in new neighborhoods during the convergence period before returning to the exploration setting.

In summary, the STLA-GA parameter tuning mechanism focuses on supplying an advantageous background based on the recent classifier performance. In contrast to the static GA genetic parameter setting, the dynamic STLA-GA genetic parameters allow the heuristic search method to choose exploring or exploiting the targeted search space as a feasible means to search for global best scores. While attempting to avoid local convergence, the exploration-exploitation cycle allows for an analytical trade-off between computation cost and information-tracking. Fig. 4 provides the summarized STLA-GA exploration-exploitation cycle.

To observe STLA-GA-tuned parameters, revisions to the current stopping criteria are essential. More specifically, a set of hybrid criteria comprising new parameter tracking and typical GA stopping criteria are necessary to monitor heuristic optimisation activity. As such, a floating value range limit is recommended to each STLA-GA tuned parameter in order to ensure stable feedback and prevent inconsistent

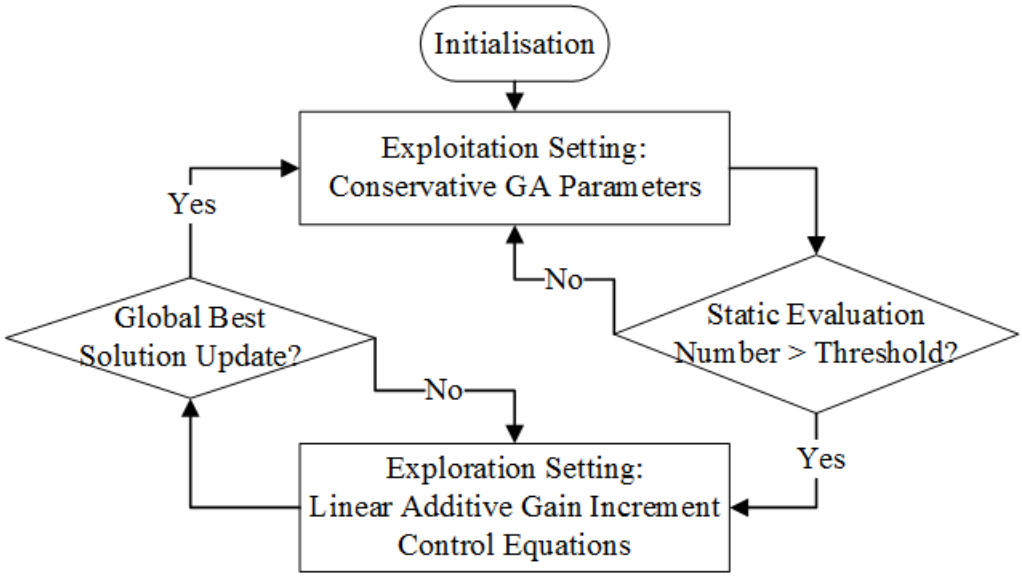

FIGURE 4. STLA-GA exploration-exploitation cycle.

performance as a result of overaggressive parameter variation. For instance, a STLA-GA tuned *mu* higher than 0.5 (*max mu*), pop size above three times the initial values (*max pop*) or *threshold* value higher than the expected STLA-GA evaluation number remaining (*gen iter*) results in exhaustive simulation and lost information.

To allow room for convergence, STLA-GA also modifies the maximum generation limit, *opt gen*. Thus, the full exploitation-exploration cycle activation is engaged to avoid undesirable premature termination. Moreover, the convergence limit, *Dist*, is applied in order to review the performance gap between iterative global best and population average. The *Dist* update represents a numerical local convergence indicator when the difference between the best performer and population average reach an unprecedented low. In other words, the current population lacks diverse candidates as convergence occurs.

For every STLA-GA generation, the aforementioned values are measured, compared and updated accordingly. The adaptive exploitation-exploration iterations are terminated only when the stopping criteria has been met. A thorough STLA-GA parameter tuning execution flow is depicted in Fig. 5 and Algorithm pseudo code 1 and 2.

C. PARTICLE SWARM OPTIMISATION

Particle Swarm Optimisation (PSO) is regarded as a branch of Swarm Intelligence (SI) attentive to the collective patterns of group-based particles. Deemed the opposite of individualistic optimisation methods, which have limited capabilities, PSO has an origin motivated by the sociality of animals in a high-pressure environment (e.g., searching for food). Analogously, PSO tracks the interaction of a swarm of homogeneous particles as a candidate population. Every particle performs similar fitness function individually. Subsequently, PSO encourages environment-related information-exchange among peers to promote collaboration and competitiveness in the quest for the global best solution.

The collective behavior of particles operating simultaneously as multi-agents in the *feat*-dimensional search space is characterized as particle movement constituting both cognitive component and social component. In other words, the trajectory of a particle across a search space is navigated by both personal best position P and global best position P_{best}. Assuming the initialized *pop size* particles are equally distributed throughout the search space and particle i is a member of the candidate population ($i \in \{1, 2, \cdots, pop\ size\}$). The coordinate vector $x_i(j = 0) = [x_{i1}, x_{i2}, \cdots, x_{ifeat-1}, x_{ifeat}]$ describes the starting location of particle i. The updated position $x_i(j + 1)$ (6) is a function of the previous position and newly revised velocity vector $v_i(j + 1) = [v_{i1}, v_{i2}, \cdots, v_{ifeat-1}, v_{ifeat}]$ (7).

$$x_i(j+1) = x_i(j) + v_i(j+1) \tag{6}$$

$$v_i(j+1) = \underbrace{v_i(j)}_{inertia} + \underbrace{c_1(P_i - x_i(j))R_1}_{cognitive} + \underbrace{c_2(P_{best} - x_i(j))R_2}_{social} \tag{7}$$

Velocity vector v is responsible for controlling the trajectory and momentum of a particle by considering three influencing factors: inertia, cognitive component, and social component. Inertia inherits $v(j)$ from the previous iteration to secure path stability and enable personal information-sharing over iterations. The cognitive component compares current position $x(j)$ and P while allowing a particular particle incline towards P during next iteration. Concurrently, the social component stimulates the tendency of approaching P_{best} obtained by the swarm through calculating the distance between $x(j)$

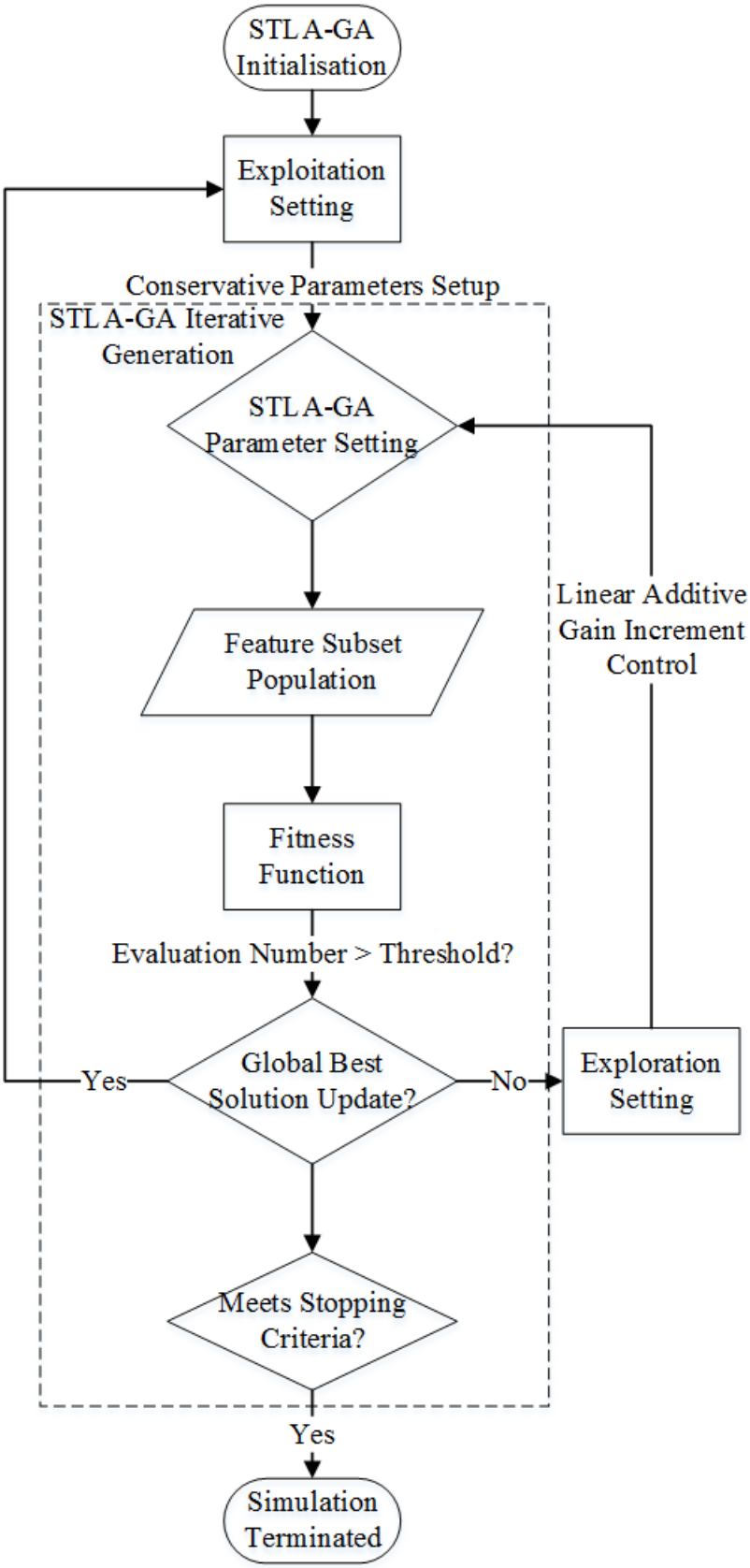

FIGURE 5. STLA-GA parameter tuning process flow.

Algorithm 1 Algorithm for STLA-GA Feature Selection Evaluation

Input: STLA-GA operating parameters, training and testing datasets

Output: Feature subset

Initialisation : Load STLA-GA operating parameters, training and testing datasets

LOOP Process

for *Generation* $= 0$ to *opt gen* **do**
  for *candidate* $= 1$ to *pop size* **do**
    Classifier training with training dataset contains feature subset only
    Perform prediction and validation with testing dataset
    if (iterative best classification score, $g_{iter} >$ existing global best classification score, g_{best}) $\vee$ ($Generation = 0 \wedge candidate = 1$) **then**
      update $g_{best} = g_{iter}$, $ga_{iter} = 0$, classifier, and best feature subset
      reset initial STLA-GA and classifier setting
    else
      update static iteration: $ga_{iter} = ga_{iter} + 1$
    end if
  end for
  Run STLA-GA parameter tuning function
end for
return Best feature subset

Algorithm 2 Algorithm for STLA-GA Parameter Tuning

Input: STLA-GA operating parameters

Output: STLA-GA operating parameters

Initialisation : Load STLA-GA operating parameters

Check convergence status for current *Generation*: *Dist* = current average Score - current Best Score
if (*Dist* $<$ *Dist* g_{best}) **then**
  update *Dist* g_{best} = *Dist*, *opt gen*, and convergence generation count, *con iter*
  if convergence generation count equals convergence generation limit: (*con iter* $\Longleftrightarrow$ *con max*) **then**
    Convergence obtained: STLA-GA simulation stopped
  end if
end if
if static iteration, ($ga_{iter} > threshold$) **then**
  update STLA-GA parameters exploration setting: *gain*, *pop size*, *mu*, *threshold*, and *score*
  update expected evaluation number remaining, *gen iter*
end if
Check STLA-GA customized Stopping Criteria:
if ($mu > maxmu$) $\vee$ ($popsize > maxpop$) $\vee$ ($threshold > gen\ iter$) **then**
  Convergence detected: STLA-GA simulation stopped
end if
if ($mu = 0.1$) $\wedge$ (*opt gen* - current *Generation* $<$ *Minimum Convergence Gen*) **then**
  Allow room for convergence: *opt gen* = *opt gen* + *Minimum Convergence Gen*
end if
return Updated STLA-GA operating parameters $\wedge$ Stopping Criteria message

and P_{best}. The iterative update on position x and velocity v for the candidate population ends upon meeting predetermined stopping criteria. The standard PSO stopping criteria can be found similarly to those of GA: PSO simulation ends when maximum iteration (*opt gen*), maximum convergence iteration of global best score, or the pre-set fitness function target is achieved.

Because PSO implementation is parameter-dependent, numerous parameter-wise considerations have been suggested in pursuit of PSO performance improvement. The particle velocity involving several elements is an integral part for PSO algorithm. During initialization, starting velocity $V(j = 0) = \{v_1(j = 0), v_2(j = 0), \cdots, v_{pop\ size}(j = 0)\}$ is advised to set small random numbers as exploration ability is assured by evenly distributed particle positioning. The combination of acceleration constant $C = \{c_1, c_2\}$, $0 \leq c_1, c_2 \leq 4$ defines the iterative pace to reach to P and P_{best}. Based on case study results, C is initialized to 2 to form a compatible pair with randomly generated diagonal matrix $R \in \{R_1, R_2\}, R \sim U(0, 1)$ to induce a stochastic variable into the revised velocity value [31]. Meanwhile, velocity damping and inertia weights are recommended to deter unstable velocity and out-of-range particles. Velocity damping is proposed in the form of velocity-allowable range, which varies in accordance with the search space area. As for inertia weight allocation, the alternatives include fixed limit range, stochastic weight assignment, and linearly weight increment/decrement. Hence, (7) with additional inertia weight ω is rewritten as:

$$v_i(j+1) = \omega(j+1)v_i(j) + c_1(P_i - x_i(j))R_1 + c_2(g_{best} - x_i(j))R_2$$

In relation to feature selection complication, additional conversion is needed to adapt the velocity vector $v_i \in \mathbb{R}^{feat}$ to binary sequence. The swarm-based heuristic optimisation method is coined specifically as Binary PSO (BPSO) in response to transformed v' (8) and binary sequence x_i (9). It is worth mentioning that $0 < v' < 1$. Consequently, v'_{im} functions as the perturbation probability for index number m in the x_i vector. A simplified BPSO feature selection sequence is outlined in Algorithm pseudo code 3.

$$v'_{im} = \frac{1}{1 + e^{-v_{im}}}, \quad m = \{1, 2, \cdots, feat\} \tag{8}$$

$$x_{im}(j+1) = \begin{cases} |x_{im}(j) - 1|, & if\ v'_{im} > \mu \\ x_{im}(j), & if\ v'_{im} < \mu \end{cases} \mu \sim U(0, 1) \tag{9}$$

Algorithm 3 Algorithm for BPSO Feature Selection Evaluation

Input: BPSO operating parameters, training and testing datasets

Output: Feature subset

Initialisation : Load BPSO operating parameters, training and testing datasets

LOOP Process

for *iteration* $j = 0$ to *opt gen* **do**
  for *particle* $i = 1$ to *pop size* **do**
    Assign *particle* position/binary sequence, $x_i(j)$
    Classifier training with training dataset contains feature subset only
    Perform prediction and validation with testing dataset, $f(x_i(j))$
    if (iterative classification score, $f(x_i(j)) >$ existing best classification score for *particle* $i, f(P_i)) \vee (j = 0)$ **then**
      update $P_i = x_i(j)$
      if (iterative classification score, $f(x_i(j)) >$ existing global best classification score, $f(P_{best})) \vee (j = 0 \wedge i = 1)$ **then**
        update $P_{best} = x_i(j)$
      end if
    end if
  end for
  Update *particle* velocity, $v_i(j+1)$ via (7)
  Transform $v_i(j+1)$ to $v_i'(j+1)$ via (8)
  Update *particle* position/binary sequence, $x_i(j+1)$ via (9)
  Check BPSO Stopping Criteria
end for
return Best feature subset P_{best}

Significantly, a high level of similarity between BPSO and GA heuristic optimisation is observed. Notably, both methods deploy candidate populations for the global optimal solution iterative search, derive benefits from elite child/fittest swarm, apply mutation/perturbation to create population diversity, and obey similar stopping criteria. Therefore, different heuristic feature selection concepts will be compared.

D. ARTIFICIAL NEURAL NETWORK

Artificial Neural Networks (ANNs) are multilayered model structures on human brain functionality. In essence, ANN consists of the input layer, hidden layer, and output layer which are formed by one or more neurons in parallel and are interconnected in series via neurons (see Fig. 6). Signals flows into the ANN model as input x through the leftmost input layer to be reduced to neuron weight w and output signal y. Thus, $x \cdot w = y$. w is assessed with a predefined training function to symbolise the connection strength between x and y. The output signal eventually serves as the input for the adjacent layer to the right. The evaluation is conducted for the subsequent layers until the output layer located at the rightmost position to allow detailed data abstraction in multitier. Throughout the training phase, w for every neuron is tuned until the output performance is optimized.

It is observed that the quantity of input and output neurons are fixed in consistent with the feature size and the number of output label. The dimension of the hidden layer and training function, however, can be customized to determine the model complexity.

E. K-NEAREST NEIGHBORS

k-Nearest Neighbors (k-NN) classification seeks the shortest k-interval between feature x and the targeted points of class y. The pre-set value k ranges between $1 \leq k < N$ and refers to the number of the nearest neighbor points to be identified from N member x. Two main types of distance cost function generally apply to locate k-NN: an exhaustive search and the k-d tree method. Exhaustive search measures and sorts the computed point distance in ascending order while k-d tree segregates point distance according to *BucketSize* per group. Distance metrics selection depends on the feature number. Exhaustive search is suitable for classification involving more than 10 features, and k-d tree performs better with fewer features.

Fig. 7 presents a simplified example of k-NN classification. k-NN classifier explores the nearest points to the origin from input feature x_1 and x_2 with k-d tree search strategy. By setting $k = 3$, the target is comparatively closer to x_2 as the 3 nearest points are identified as being from x_2.

F. SUPPORT VECTOR MACHINE

The theory behind Support Vector Machine (SVM) is best explained with binary classification (see Fig. 8). The function of a SVM is to customize the decision boundary using a fitted line known as the hyperplane. The hyperplane plot is guided by a selection of data points (support vectors) from classes affected by the decision boundary. The fitness of a hyperplane is influenced by two factors: the separation of data points corresponding to associated class and the distance between hyperplane and support vectors. Hence, the hyperplane mapping (10) search for optimizing the vacant area adjacent to the hyperplane and the nearest support vector by solving $\|\beta\|$ cost function.

$$f(x) = x'\beta + b = 0 \tag{10}$$

$$y_i f(x_i) \geq 1, \quad i = \{1, 2, \cdots, N-1, N\} \tag{11}$$

A hyperplane is considered acceptable when (11) is satisfied. x and y refer to N training input data points and respective class labels, where $x, \beta \in \mathbb{R}^N, y \in \pm 1$, and $b = \mathbb{R}$. The support vectors located on the boundary of the vacant area are defined as $y_i f(x_i) = 1$. Notice the value for $y_i f(x_i)$ is directly proportional to the gap between data point x_i and the hyperplane.

There are two class label assignment options available for SVM classification purposes. For multiclass classification

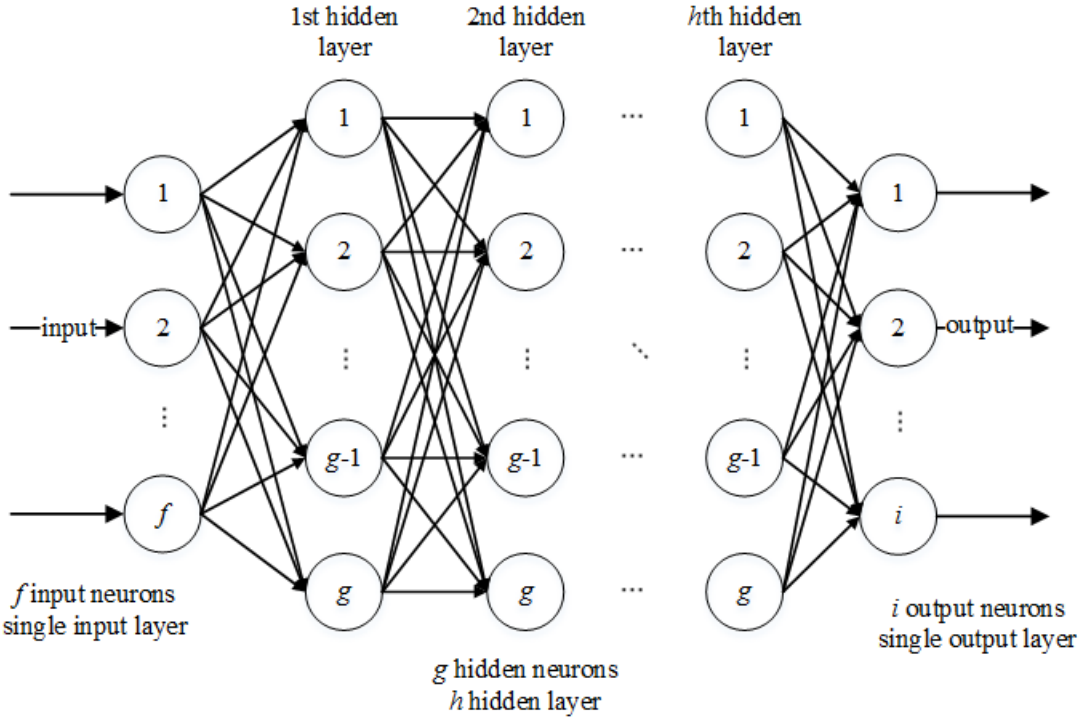

FIGURE 6. Artificial neural network: overview.

with a finite number of class labels K, one-versus-one setup implements the max-wins voting strategy for each pair of classes. The one-versus-one strategy breaks down one multiclass classification problem into $K(K-1)/2$ binary classification problems. Meanwhile, the one-versus-all configuration exercise uses a ''winner-takes-all'' tactic.

IV. MATERIAL AND METHODS

The newly developed STLA-GA algorithm and two feature selection benchmark methodologies (Standard GA and BPSO) are subjected to four types of multiclass machinery dataset for validation. To reflect the state of the targeted machines from different perspectives, the dataset was obtained using various sensor types. The input measurement identifies as feature ranging from temperature (°C), speed (m/s), vibration (mm/s), pressure (Bar), volume flow (l/min), torque (N/m) to sensor locations and directions.

The first dataset involves condition monitoring of a hydraulic system [32]. For a standard 60-second cycle, 17 elementary component process indicators focused on the cooler, valve, pump, and accumulator are recorded under constant load. The cyclic measurement is repeated 2,205 times. For every measurement cycle, the health status of the components is denoted as the equipment health percentage (%), leakage level (Unit), internal component pressure reading (Bar), and a stability unit scale. The resulting output target is a numeric matrix combining five unique health indication vectors ($y_{Hydraulic} \in \mathbb{R}^{2205\times5}$). Table 2 outlines the hydraulic system dataset information.

The second dataset presents a mechanical pump analysis for diagnosis purpose [33]. For 209 sampling iterations, seven features are registered together with the corresponding equipment state. The supplied input features are pump speed, measurement frequency, number of measured component and support, and measurement values along with measurement type and direction for the instant measurement event. Six equipment states are labelled as class numbers in a range of between one to six. Table 3 shows the details for pump dataset.

Third dataset relates to classifying robot execution failure events [34]. A robot execution failure dataset studies the force and torque sampling measurement in triaxial force sensor direction and triaxial torque sensor direction. Eight statistical features are extracted from a given time series sampling measurement at a particular sensor direction. A total of 48 features (8 features $\times$ 6 sensor directions) are generated to categorise 463 occurrences. Fifteen unique class labels are observed after combining all five failure execution datasets. Table 4 itemizes the feature inputs and target labels for robot execution failure dataset.

Nonlinear model parity-equations residual generation for centrifugal pumps fault diagnosis [35] is organized as the fourth classification dataset. A centrifugal pumps dataset containing pump rotational speed (RPM), motor torque (Nm), pressure differential (Δp), and flow rate (m^3/s) is recorded at a sampling frequency of 10 Hz . Four interconnected centrifugal pumps system models are acquired to simulate the nonlinear function under the influence of pump operating

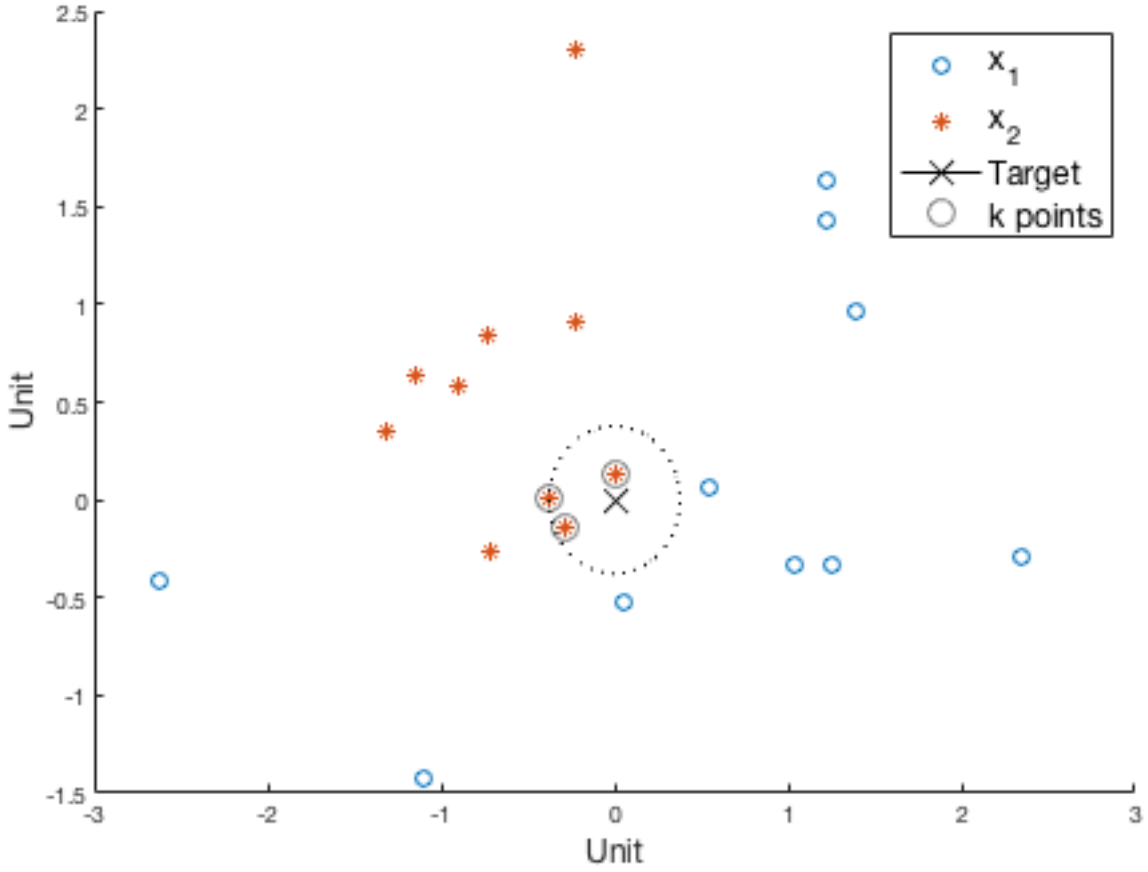

FIGURE 7. ***k*-nearest neighbors classification: overview.**

speed (see Fig. 9). The estimate of model parameters θ_d, $d = \{1, 2, \cdots, 9, 10\}$ utilizes static pump system identification, piecewise linear approximation, and the nonlinear autoregressive with exogenous terms (ARX) model to construct the black box function in linear form for three speed intervals (Table 5). As a result, intermediate outputs are simulated with Pseudorandom Binary Signal (PRBS) as reference input initialization setup.

Consequently, fault diagnosis is examined using residual analysis under various running conditions with different operating speeds. The residues $r = \{r_1, r_2, r_3, r_4\}$ for 10 pump operation classes (1 healthy pump condition and 9 faulty modes) are the variation between measured outputs and simulated outputs. For each mode, 20 residue-generated statistical features (5 features $\times$ 4 models) are extracted for 50 repetitions. Table 6 summaries the list of feature inputs and target labels for the respective dataset. Additionally, Table 7 provides the statistical features implemented in the datasets.

To ensure a consistent benchmark across feature selection methodologies, standardisation is introduced to the selection of initial heuristic optimisation parameters and machine learning classifiers. The parameter initialization for the conservative GA genetic operator (Table. 8) and BPSO (Table. 9) are coupled separately with five types of classifiers. The aforementioned standard classifier setting is selected due to its ease of implementation and highest prediction accuracy acquired from trial and error.

A recap of the hydraulic system condition monitoring dataset indicates that the output target label consists of a matrix with five unique numeric vectors of equal length ($N = 2205$). Each vector represents a particular equipment health indicator. The k-NN and SVM classifiers address prediction in vector form. In consequence, five k-NN classifiers and five SVM classifiers are needed to examine the output label matrix. Furthermore, k-NN and SVM generates different feature subset vectors for each hydraulic system equipment health indicator. Due to the dataset complexity, an ANN classifier is generated for a hydraulic system dataset feature selection benchmark. The ANN model returns a feature subset vector and performs numeric prediction in matrix form, which are both attributable to its multi-layer model structure. A classic graphical ANN model (Levenberg-Marquardt function feedforward model with one hidden layer and a size of 10 parallel neurons) is applied to the hydraulic system dataset to observe the output labels.

For the multiclass classification dataset (pump, robot execution failure, and centrifugal pumps residual analysis), five unique classifiers are implemented to determine the

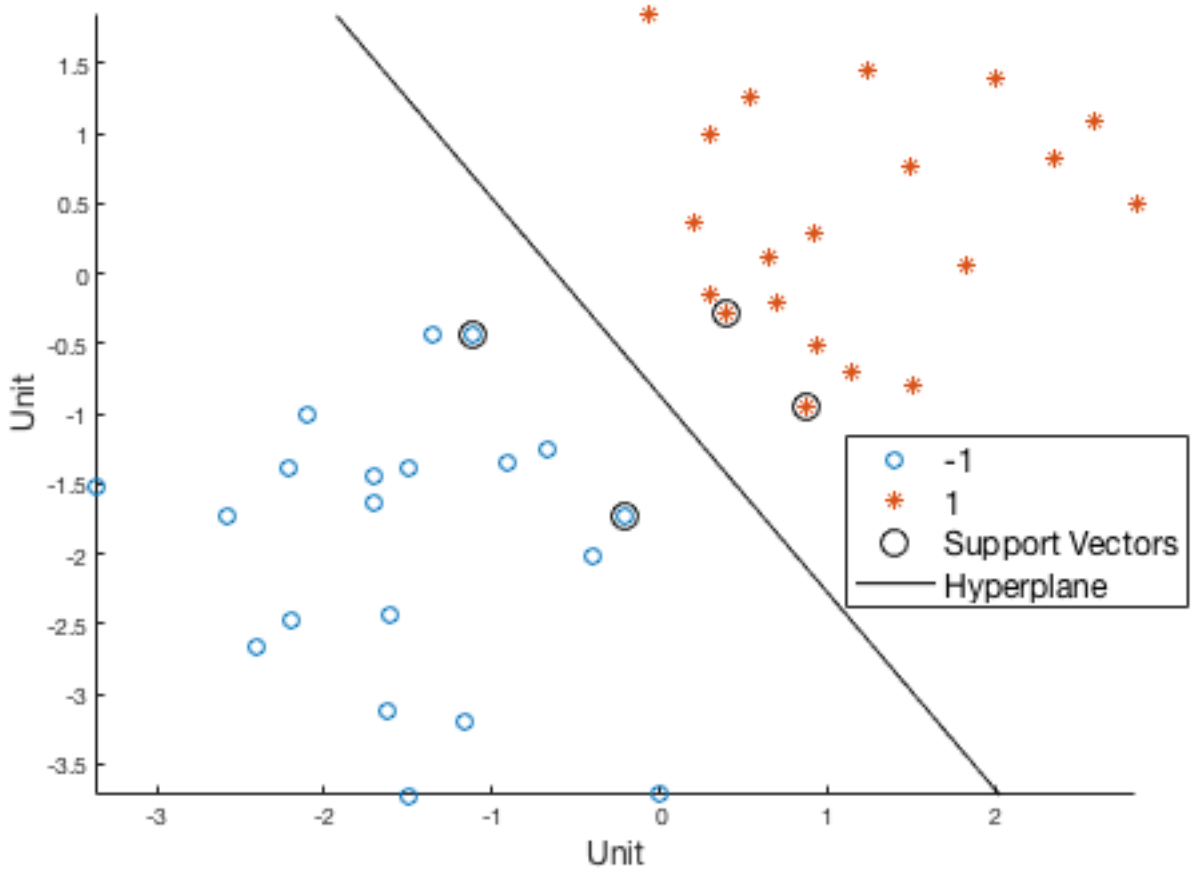

FIGURE 8. Support vector machine binary classification: overview.

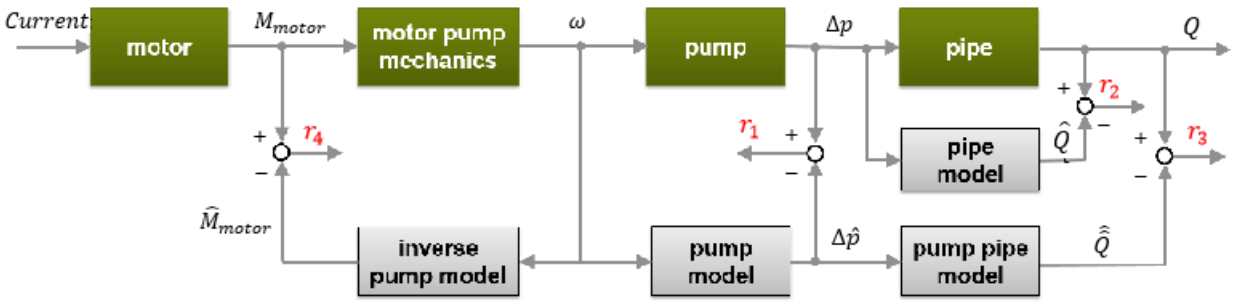

FIGURE 9. Centrifugal pumps modeling and residual analysis: system overview.

best predictor. A scaled conjugate gradient function (trainscg) feedforward model with one hidden layer the size of 10 parallel neurons is assigned as the standard ANN pattern recognition classifier. Distance based k-NN with $k = 5$ is integrated with two operating frameworks: an exhaustive search and the k-d tree. The SVM classifier is engaged with both the one-versus-all and one-versus-one approaches. The best predictor of classification performance is further embedded into feature selection methods for comparison.

Different prediction appraisal rules are required, as the available target label varies between datasets. The standard system identification regression efficacy indicator is applied to analyses numeric prediction of hydraulic system states. For the hydraulic system, the numerical prediction accuracy is represented by the average percentage of coefficient of determination, R_T^2 regression analysis (12):

$$R_T^2 = \frac{\sum(\hat{y} - y)^2}{\sum(y - \bar{y})^2} \tag{12}$$

where $\bar{y}$ is the average output.

For multiclass classification dataset, the classification prediction accuracy ($(1 - mis) \times 100\%$) is measured when the

TABLE 2. Dataset 1 description: hydraulic system.

Label	No.	Description
Feature	1	$PS1$ Pressure in Bar, 100 Hz
	2	$PS2$ Pressure in Bar, 100 Hz
	3	$PS3$ Pressure in Bar, 100 Hz
	4	$PS4$ Pressure in Bar, 100 Hz
	5	$PS5$ Pressure in Bar, 100 Hz
	6	$PS6$ Pressure in Bar, 100 Hz
	7	$EPS1$ Motor power in W, 100 Hz
	8	$FS1$ Volume flow in l/min, 10 Hz
	9	$FS2$ Volume flow in l/min, 10 Hz
	10	$TS1$ Temperature in °C, 1 Hz
	11	$TS2$ Temperature in °C, 1 Hz
	12	$TS3$ Temperature in °C, 1 Hz
	13	$TS4$ Temperature in °C, 1 Hz
	14	VS Vibration in mm/s, 1 Hz
	15	CE Cooling efficiency (virtual) in %, 1 Hz
	16	CP Cooling power (virtual) in kW, 1 Hz
	17	SE Efficiency factor in %, 1 Hz
Class	1	Cooler Condition in %
	2	Valve Condition in %
	3	Internal pump leakage in Unit
	4	Hydraulic accumulator in Bar
	5	Stable flag in Unit

TABLE 3. Dataset 2 description: pump.

Label	No.	Description
Feature	1	# Component Number in Unit
	2	sup Machine Support in Unit
	3	cpm Measurement Frequency in Hz
	4	mis Measurement in Unit
	5	$misr$ Earlier Measurement in Unit
	6	dir Filter type of the measurement and direction in Unit
	7	$omega$ Machine Speed in RPM
Class	1-6	Machine state conditions in Unit

TABLE 4. Dataset 3 description: robot execution failure.

Label	No.	Description
Feature	1-6	Average
	7-12	Standard Deviation
	13-18	Skewness
	19-24	Kurtosis
	25-30	Crest Factor
	31-36	Impulse Factor
	37-42	Shape Factor
	43-48	Margin Factor
Class	1	Normal/ OK
	2	Unique robot learning problem: collision
	3	Unique robot learning problem: fr collision
	4	Unique robot learning problem: obstruction
	5	Unique robot learning problem: back collision
	6	Unique robot learning problem: front collision
	7	Unique robot learning problem: left collision
	8	Unique robot learning problem: right collision
	9	Unique robot learning problem: lost
	10	Unique robot learning problem: moved
	11	Unique robot learning problem: slightly moved
	12	Unique robot learning problem: bottom collision
	13	Unique robot learning problem: bottom obstruction
	14	Unique robot learning problem: collision in part
	15	Unique robot learning problem: collision in tool

prediction outcome and testing output label subset are available. Next, the classifier performance is evaluated with confusion matrix statistical analysis. For multiclass classification, we split the confusion matrix for K unique class labels into K binary confusion matrices and perform the macro-averaging method [36]. To obtain the average confusion matrix statistical indicator value, every class label takes a turn as the target label (Positive label).

Six confusion matrix statistical indicators are applied according to Table 10:

1) The classifier sensitivity is introduced as the ratio of True Positive (TP) samples to total Positive (P) samples.
2) The classifier specificity is the ratio of True Negative (TN) samples out of the total Negative (N) samples.
3) The classifier precision is the proportion of TP samples with respect to total predicted Positive samples (TP+FP).
4) Undesirable False Positive Rate (FPR) is the fraction of FP samples in total predicted Negative samples (FP+TN).
5) F-measure is a balancing function to calculate the harmonic mean of sensitivity and precision.
6) Matthews Correlation Coefficient (MCC) measures the strength of the statistical relationship between y and $\hat{y}$, especially when an imbalanced dataset is involved. It is worth noting that the numerator of MCC multiplies the inner diagonal elements of the confusion matrix in a criss-cross pattern. By contrast, the denominator of MCC is the multiplication of the outer element of the confusion matrix.

A classifier is considered perfect when there is a total agreement between y and $\hat{y}$ (F-measure $\wedge$ MCC $= 1$). A completely wrong classifier is obtained when total disagreement happens between y and $\hat{y}$ (F-measure $= 0 \wedge$ MCC $= -1$).

In addition, ten-fold cross-validation is imposed on the simulation datasets to evaluate the reproducibility of feature subset-trained classifier performance when subjected to untrained datasets. In cross-validation, the loss function *kfoldLoss* (13) is applied to calibrate the average misclassification proportion of testing-fold target output. The robustness of a cross-validated classifier is inversely proportional to

TABLE 5. Centrifugal pumps modeling and residual analysis: description.

Model	Function	Residual
Static Pump	$\Delta\hat{p}(t) = \theta_1\omega^2(t) + \theta_2\omega(t)$	$r_1 = \Delta p - \Delta\hat{p}$
Dynamic Pipe	$\widehat{Q}(t) = \theta_3 + \theta_4\sqrt{\Delta p(t)} + \theta_5\widehat{Q}(t-1)$	$r_2 = Q - \widehat{Q}$
Dynamic Pump-Pipe	$\widehat{\widehat{Q}}(t) = \theta_3 + \theta_4\sqrt{\Delta\hat{p}(t)} + \theta_5\widehat{\widehat{Q}}(t-1)$	$r_3 = Q - \widehat{\widehat{Q}}$
Dynamic Inverse Pump	$\widehat{M}_{motor}(t) = \theta_6 + \theta_7\omega(t) + \theta_8\omega(t-1) + \theta_9\omega^2(t) + \theta_{10}\widehat{M}_{motor}(t-1)$	$r_4 = M_{motor} - \widehat{M}_{motor}$

TABLE 6. Dataset 4 description: centrifugal pumps fault diagnosis.

Label	No.	Description
Feature	1-4	Average
	5-8	Maximum Amplitude
	9-12	Variance
	13-16	Kurtosis
	17-20	1-norm
Class	1	Healthy
	2	Fault 1: Wear at clearance gap
	3	Fault 2: Small deposits at impeller outlet
	4	Fault 3: Deposits at impeller inlet
	5	Fault 4: Abrasive wear at impeller outlet
	6	Fault 5: Broken blade
	7	Fault 6: Cavitation
	8	Fault 7: Speed sensor bias
	9	Fault 8: Flowmeter bias
	10	Fault 9: Pressure sensor bias

loss value.

$$kfoldLoss = \frac{1}{10}\sum_{kfold=1}^{10} mis_{kfold} \tag{13}$$

The prediction outcome for 10 simulation cycles-including the number of feature subset, computation time, prediction accuracy, confusion matrix statistical analysis, and classification loss for the ten-fold cross-validated classifier-are tabulated in the following section. The simulation is performed using the 2019a version of Matlab software as the Integrated Development Environment (IDE). Matlab software is run on a 64-bit Microsoft Windows 7 operating system installed with 2.20 GHz Intel Xeon CPU processor and 32 GB random access memory.

TABLE 7. Statistical features: description.

No.	Feature	Description	Equation
1	Mean	Mean value for N elements in vector X	$\mu = \frac{1}{N}\sum_{i=1}^{N} X_i$
2	Standard Deviation	Distribution of N elements in vector X	$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} \lvert X_i - \mu\rvert^2}$
3	Skewness	Asymmetricity of N elements in vector X relative to mean value	$S = \frac{\frac{1}{N}\sum_{i=1}^{N}(X_i-\mu)^3}{\sigma^3}$
4	Kurtosis	Measure the tendency of having anomaly in vector X	$k = \frac{\frac{1}{N}\sum_{i=1}^{N}(X_i-\mu)^4}{\sigma^4}$
5	Crest Factor	The ratio of absolute maximum value over root mean square	$CF = \frac{\lVert X\rVert_\infty}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\lvert X_i\rvert^2}}$
6	Impulse Factor	The ratio of absolute maximum value over absolute mean value	$IF = \frac{\lVert X\rVert_\infty}{\lvert\mu\rvert}$
7	Shape Factor	The ratio of root mean square over absolute mean value	$SF = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\lvert X_i\rvert^2}}{\lvert\mu\rvert}$
8	Margin Factor	The ratio of absolute maximum over the power of two of the average of square root's sum	$MF = \frac{\lVert X\rVert_\infty}{(\frac{1}{N}\sum_{i=1}^{N}\sqrt{\lvert X_i\rvert})^2}$
9	Maximum Amplitude	Maximum value in vector X	X_{max}
10	Variance	Squared of standard deviation	$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}\lvert X_i - \mu\rvert^2$
11	1-norm	Summation of absolute values for N elements in vector X	$1-norm = \frac{1}{N}\sum_{i=1}^{N}\lvert X_i\rvert$

V. RESULTS AND DISCUSSION

This section describes the prediction outcome for four machinery datasets. The result tabulation begins with a classifier prediction performance review when subjected to multiclass classification datasets, according to Table 11, 12, and 13. In general, the objective of a classifier is to obtain optimal prediction accuracy by minimising undesirable prediction error. For each dataset, the classifier with the highest prediction accuracy is embedded into feature selection mechanisms. The prediction outcome with and without the feature selection technique is compared, followed by a feature selection performance evaluation. Table 14 to 17 summaries the average value for classifier prediction performance, all of which correspond to feature selection methods and datasets.

A. CLASSIFIER PREDICTION PERFORMANCE WITHOUT FEATURE SELECTION

Table 11 indicates the pump dataset prediction outcomes using five unique classifier settings. The ANN pattern

TABLE 8. Genetic algorithm parameter initialization selection.

Parameter Name	Value
Mutation probability rate, mu	0.1
Population size, $pop\ size$	10
Generations, $opt\ gen$	10
Maximum converged generation count, con max	$opt\ gen$ / 2
Threshold	$pop\ size \times 2$
Static candidate evaluation Count, ga_{iter}	0
Converged generation count, $con\ iter$	0
Maximum allowable mutation probability rate, $max\ mu$	0.5
Maximum allowable population size, $max\ pop$	$pop\ size \times 3$

TABLE 9. Binary particle swarm optimisation parameter initialization selection.

Parameter Name	Value
Population size, $pop\ size$	10
Iterations, $opt\ gen$	10
Cognitive acceleration constant, c_1	2
Social acceleration constant, c_2	2
Maximum Velocity, V_{max}	6
Maximum inertia weight, ω_{max}	0.9
Minimum inertia weight, ω_{min}	0.4

TABLE 10. Confusion matrix statistical analysis.

No.	Indicator	Equation
1	Sensitivity	$\frac{TP}{P}$
2	Specificity	$\frac{TN}{N}$
3	Precision	$\frac{TP}{TP+FP}$
4	False Positive Rate, FPR	$\frac{FP}{FP+TN}$
5	F-Measure	$2 \times \frac{Sensitivity \times Precision}{Sensitivity+Precision}$
6	Matthews Correlation Coefficient, MCC	$\frac{(TP \times TN)+(FP \times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

TABLE 11. Classifier prediction performance without feature selection for pump dataset.

Prediction Accuracy	Classifier Method: ANN trainscg	k-NN exhaustive	k-NN k-d tree	SVM one-vs-all	SVM one-vs-one
Mean (%)	47.57	42.17	42.20	22.26	22.63
Best (%)	47.97	42.47	42.47	23.10	23.24
Difference (%)	0.00	5.50	5.50	24.87	24.73
$kfoldLoss$ (Unit)	0.6570	0.5769	0.5670	0.6274	0.6246
Evaluation Time (s)	10	0.4	0.3	135.1	179.0

TABLE 12. Classifier prediction performance without feature selection for robot execution failure dataset.

Prediction Accuracy	Classifier Method: ANN trainscg	k-NN exhaustive	k-NN k-d tree	SVM one-vs-all	SVM one-vs-one
Mean (%)	58.26	59.57	59.57	59.42	62.65
Best (%)	61.30	60.87	60.87	60.87	66.52
Difference (%)	5.22	5.65	5.65	5.65	0.00
$kfoldLoss$ (Unit)	0.3457	0.3762	0.3819	0.2854	0.2772
Evaluation Time (s)	6.5	5.7	5.9	5.4	5.0

TABLE 13. Classifier prediction performance without feature selection for centrifugal pumps residual analysis for fault diagnosis dataset.

Prediction Accuracy	Classifier Method: ANN trainscg	k-NN exhaustive	k-NN k-d tree	SVM one-vs-all	SVM one-vs-one
Mean (%)	77.96	64.40	64.80	68.29	80.28
Best (%)	79.20	67.20	67.20	70.40	82.80
Difference (%)	3.60	15.60	15.60	12.40	0.00
$kfoldLoss$ (Unit)	0.2221	0.3750	0.3798	0.1580	0.1514
Evaluation Time (s)	2.2	3.7	3.8	4.8	5.0

TABLE 14. ANN numerical prediction result: Hyrdraulic system.

Dataset: Hydraulic System Prediction Criteria		Feature Selection Method: Without FS	Standard GA	STLA-GA	BPSO
Accuracy (%)	Mean	80.68	86.51	92.52	86.40
	Std Dev	0	0.25	1.37	2.71
	Best	80.68	89.70	94.20	87.75
	Difference	13.52	4.50	0	6.45
	$kfoldLoss$	0.3611	0.1767	0.1054	0.1468
Feature Subset	Mean	17	9.1	9.1	11.1
	Std Dev	0	1.10	1.37	0.78
	Best	17	8	8	10
	Reduction (%)	0	52.94	52.94	41.18
Evaluation	Number (Unit)	1	110	200	110
	Time (s)	1695	20352	45106	37792
	Time (%)	3.76	45.12	100.00	83.79

TABLE 15. k-NN multiclass classification prediction result: pump.

Dataset: Pump Prediction Criteria		Feature Selection Method: Without FS	Standard GA	STLA-GA	BPSO
Accuracy (%)	Mean	42.47	46.87	47.64	47.97
	Std Dev	0	0.24	0.47	0.48
	Best	42.47	47.52	48.60	48.49
	Difference	6.13	1.08	0	0.11
	$kfoldLoss$	0.5830	0.5472	0.5177	0.5249
Feature Subset	Mean	7	5.1	5.5	4.6
	Std Dev	0	0.32	0.85	1.15
	Best	7	5	4	4
	Reduction (%)	0	28.57	42.86	42.86
Confusion Matrix Statistical Analysis	Sensitivity	0.2910	0.2940	0.3344	0.3144
	Specificity	0.8731	0.8753	0.8866	0.8794
	Precision	0.3415	0.3524	0.5371	0.3569
	FPR	0.1269	0.1247	0.1134	0.1206
	MCC	0.1858	0.1939	0.2913	0.2132
	F-Measure	0.2963	0.2982	0.3459	0.3210
Evaluation	Number (Unit)	1	110	178	110
	Time (s)	0.3	14.0	20.0	6.0
	Time (%)	1.65	70.00	100.00	30.00

recognition model documented the highest prediction accuracy, followed by k-NN classifiers and SVM classifiers. However, the ANN model also recorded one of the highest *kfoldLoss* values. During model training, the ANN model is likely to experience bias in datasets, which causes decrements in prediction accuracy upon ten-fold cross-validation. We selected the k-NN classifier with k-d tree framework specifically for pump dataset feature selection due to the second highest prediction accuracy (42.47%) together

TABLE 16. SVM multiclass classification prediction result: robot execution failure.

Dataset: Robot Failure Prediction Criteria		Feature Selection Method			
		Without FS	Standard GA	STLA-GA	BPSO
Accuracy (%)	Mean	62.65	69.87	71.30	70.18
	Std Dev	1.50	1.12	0.92	1.85
	Best	66.52	70.87	72.61	72.29
	Difference	6.09	1.74	0	0.32
	$kfoldLoss$	0.2772	0.2622	0.2454	0.2632
Feature Subset	Mean	48	25.90	23.00	26.00
	Std Dev	0	2.33	1.49	3.46
	Best	48	26	21	29
	Reduction (%)	0	45.83	56.25	39.58
Confusion Matrix Statistical Analysis	Sensitivity	0.3411	0.4543	0.4576	0.4300
	Specificity	0.9724	0.9788	0.9800	0.9797
	Precision	0.3056	0.4330	0.4398	0.4016
	FPR	0.0276	0.0212	0.0200	0.0203
	MCC	0.3013	0.4162	0.4263	0.3979
	F-Measure	0.3189	0.4214	0.4359	0.4062
Evaluation	Number (Unit)	1	110	171	110
	Time (s)	5.0	1388.2	2103.2	1411.7
	Time (%)	0.24	66.00	100.00	67.12

TABLE 17. SVM multiclass classification prediction result: Centrifugal pumps residual analysis for fault diagnosis.

Dataset: Centrifugal Pumps Prediction Criteria		Feature Selection Method			
		Without FS	Standard GA	STLA-GA	BPSO
Accuracy (%)	Mean	80.28	81.40	82.44	81.38
	Std Dev	1.14	2.38	2.15	2.54
	Best	82.80	84.00	85.60	84.40
	Difference	2.80	1.60	0.00	1.20
	$kfoldLoss$	0.1514	0.1594	0.1474	0.1627
Feature Subset	Mean	20	13.80	12.80	13.56
	Std Dev	0	1.32	1.81	1.81
	Best	20	13	11	13
	Reduction (%)	0.00	35.00	45.00	35.00
Confusion Matrix Statistical Analysis	Sensitivity	0.8115	0.8266	0.8489	0.8233
	Specificity	0.9766	0.9784	0.9846	0.9816
	Precision	0.8105	0.8265	0.8475	0.8294
	FPR	0.0234	0.0216	0.0154	0.0184
	MCC	0.7859	0.8031	0.8311	0.8015
	F-Measure	0.8062	0.8211	0.8441	0.8188
Evaluation	Number (Unit)	1	110	196	110
	Time (s)	5.0	82.1	128.0	50.7
	Time (%)	3.91	64.14	100.00	39.58

with the lowest *kfoldLoss* value (0.5670) and a total evaluation time of 0.3 s.

With a similar classifier setting, Table 12 displays the robot execution failure dataset prediction results. Of five options, the SVM classifier in conjunction with the one-versus-one approach achieved the best prediction accuracy (66.52%), the least *kfoldLoss* value (0.2772) and a total evaluation time of 5.0 s. The one-versus-one SVM classifier is the most suitable alternative for robot execution failure dataset feature selection.

Table 13 shows the centrifugal pumps residual analysis for fault diagnosis dataset prediction performance with five different classifier setups. The one-versus-one approach embedded SVM classifier obtained the highest prediction accuracy, at 82.80%. The robustness of the SVM classifier is assured by the lowest *kfoldLoss* values gained in cross-validation, although SVM classifier required the longest evaluation time (4.8 s and 5.0 s). SVM classifier combined with the one-versus-one approach is the preferable option for feature selection comparison.

B. BENCHMARKING OF FEATURE SELECTION TECHNIQUES

By establishing prediction without feature selection as the baseline condition, the mean prediction accuracy percentage increases at least 5.84%, 4.40%, 7.22%, and 1.10%, respectively, with standard feature selection practice. The corresponding *kfoldLoss* value for feature subset-trained classifiers reduces in response to the increment of prediction accuracy. As the results suggest, the main contributing factor to prediction performance is likely the reduction of the overfitting issue in feature selection. With a combination of relevant features, prediction performance appears directly proportional to the feature subset reduction percentage.

It should be highlighted that additional computation effort is required to perform heuristic feature selection as displayed in total evaluation number and time. The effectiveness of feature selection methodologies based on dataset prediction outcomes is reviewed in subsequent sections, followed by a thorough conclusion.

1) DATASET 1: HYDRAULIC SYSTEM

Table 14 tabulates the numerical prediction performance for the hydraulic system condition monitoring dataset. The STLA-GA method attained the highest average prediction accuracy (92.52%) and feature subset reduction percentage (52.94%). We noticed that both Standard GA and STLA-GA obtained an identical best feature subset reduction percentage. Nevertheless, both feature selection methods deliver unique feature subsets: 8 of 17 features selected by STLA-GA are (relatively speaking) more relevant in reflecting the conditions of the hydraulic system with the maximum prediction accuracy (94.20%) and minimum *kfoldLoss* value (0.1054).

The advantage of STLA-GA can be verified with the hydraulic system prediction performance presented in Fig. 10 and 11. Plotting classifier performance over generations reveals that convergence occurs as prediction error remains stagnant prior to tenth generation. Eventually, STLA-GA acknowledges that convergence occurred, switches to the exploration setting, and overcomes the local optimum at tenth generation. Since global best score improved near the end of the simulation, generation limit *opt gen* is extended accordingly to accommodate the full exploration-exploitation cycle. In contrast to Standard GA, low population candidate variation and static genetic operator parameters are less likely to help in a similar situation.

2) DATASET 2: PUMP

Table 15 outlines the pump multiclass classification prediction results, showing the STLA-GA selected feature subset performed slightly better than the other two feature selection choices. A noticeable classifier performance gap

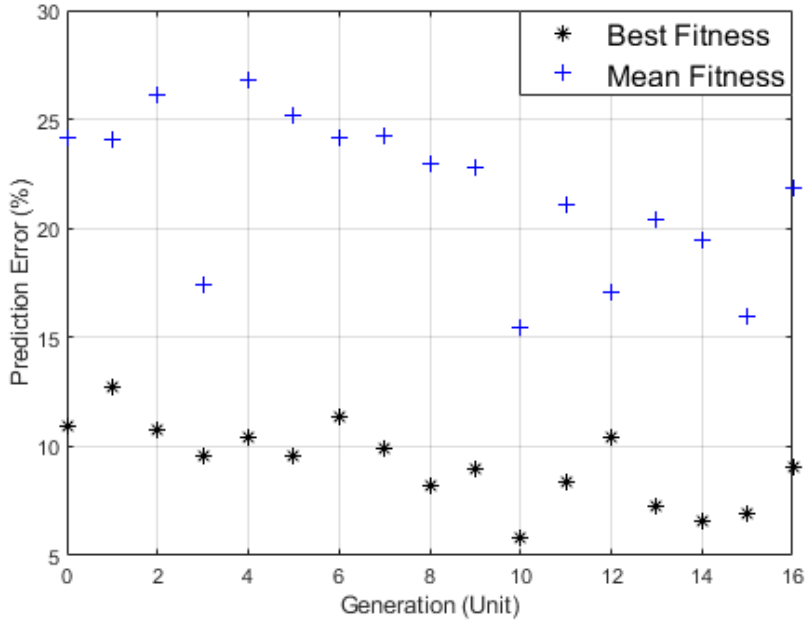

FIGURE 10. Hydraulic system dataset: STLA-GA fitness function performance over generations.

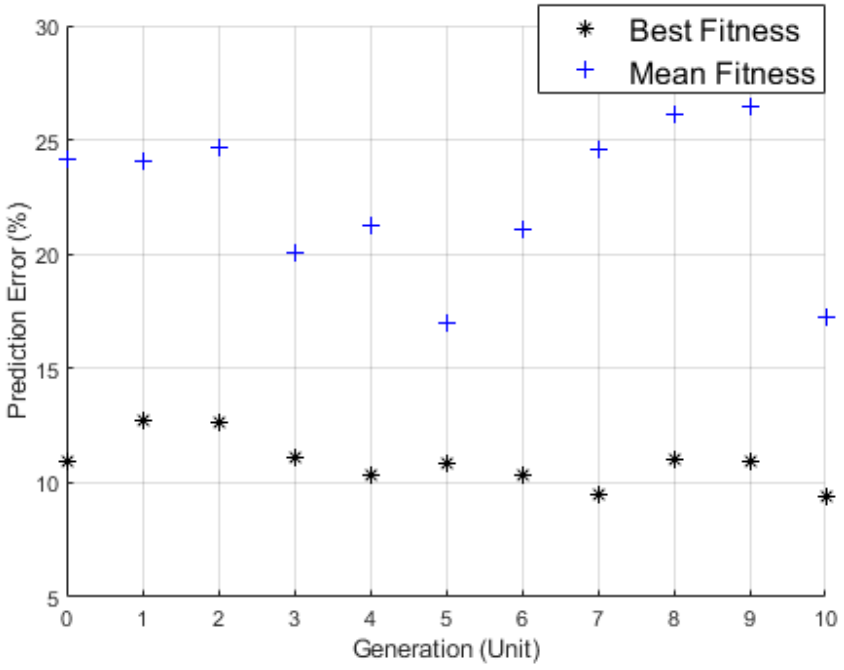

FIGURE 11. Hydraulic system dataset: standard GA fitness function performance over generations.

is observed in confusion matrix statistical analysis, even though the dissimilarity in prediction accuracy and *kfoldLoss* is marginal. The STLA-GA feature subset-trained classifier improved classification precision as $\hat{y}$ accommodates lower FP value, leading to considerably higher MCC and F-measure indicators. The main reason behind the classification performance difference is likely STLA-GA's distinctive feature subset. Similar to hydraulic system dataset, BPSO and STLA-GA are on par in terms of feature reduction percentage (42.86%). However, both feature selection techniques selected non-identical feature subset, which contribute to classifier performance variation.

Fig. 12 and 13 portray the difference between STLA-GA and Standard GA when local convergence was encountered

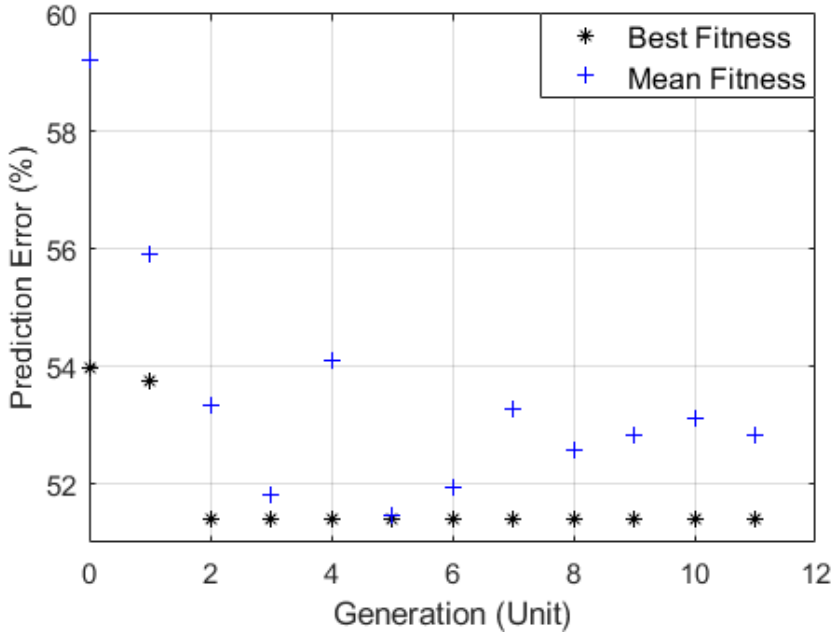

FIGURE 12. Pump dataset: STLA-GA fitness function performance over generations.

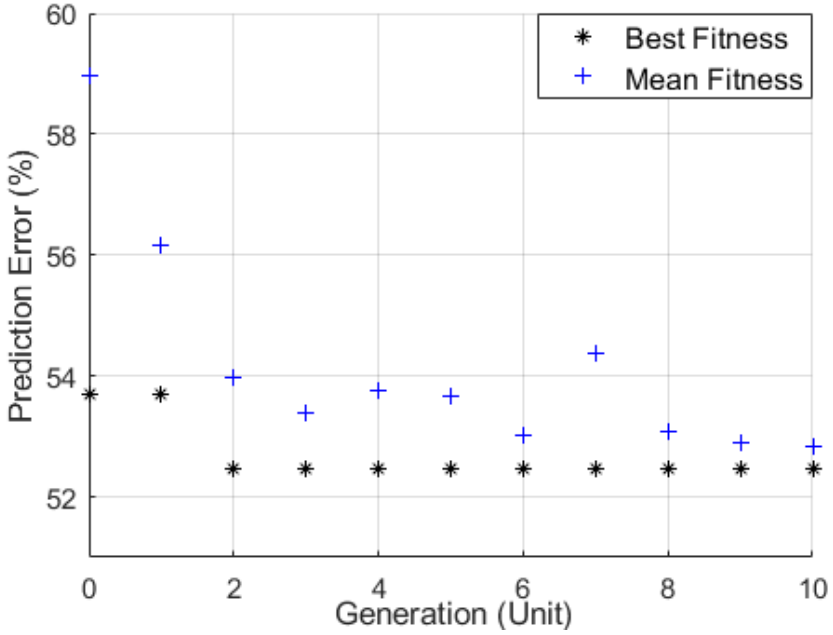

FIGURE 13. Pump dataset: standard GA fitness function performance over generations.

at the beginning of heuristic simulation. STLA-GA and Standard GA managed to overcome local convergence at the second generation before remaining stagnant in performance for the rest of the generations. Further investigation demonstrates STLA-GA achieved lower prediction error after switched to exploration setting, in contrast to the static genetic operators setting.

3) DATASET 3: ROBOT EXECUTION FAILURE

Table 16 demonstrates the results on robot execution failure multiclass classification prediction performance. STLA-GA secured the best prediction accuracy (72.61%) and least *kfoldLoss* value (0.2454), with the highest feature reduction rate (56.25%). More in-depth research saw the STLA-GA feature subset-trained classifier outperformed

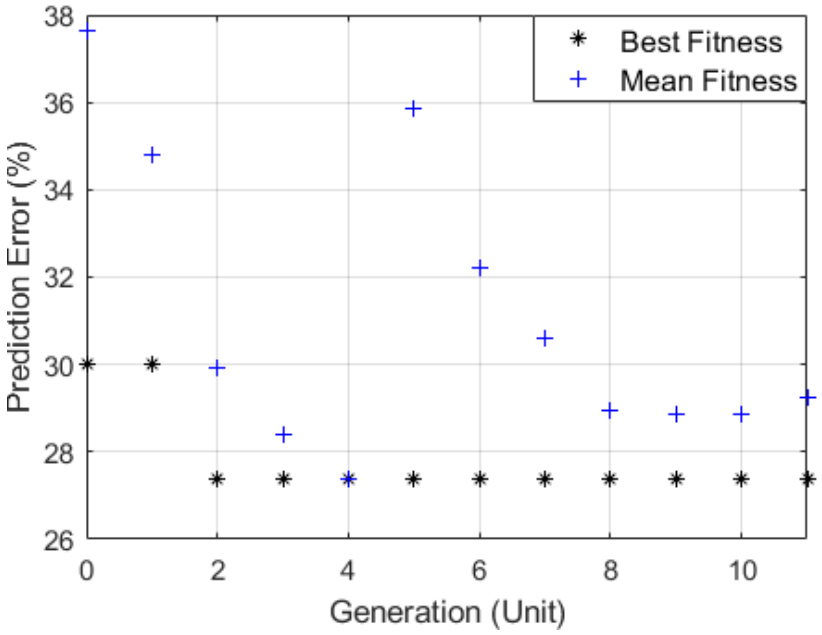

FIGURE 14. Robot execution failure dataset: STLA-GA fitness function performance over generations.

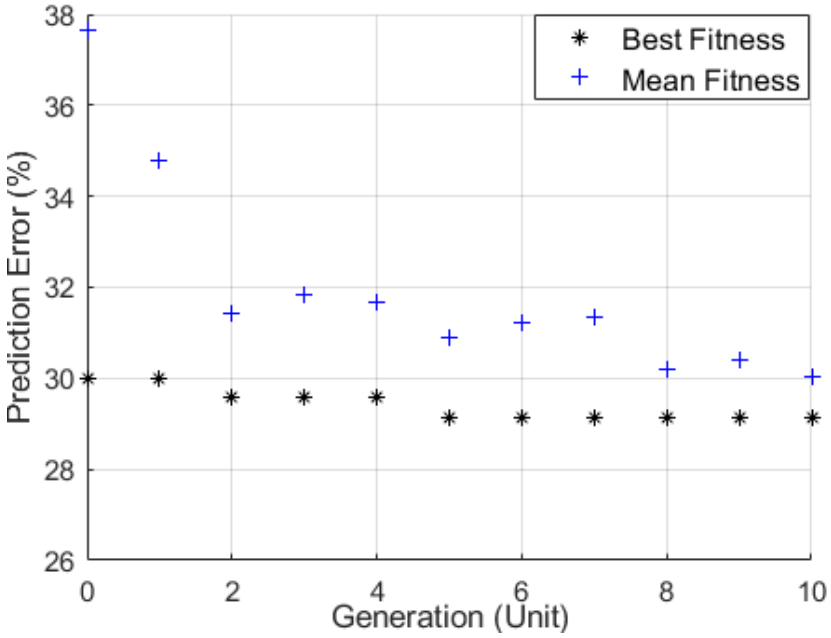

FIGURE 15. Robot execution failure dataset: standard GA fitness function performance over generations.

other feature selection methods by small margin in all six confusion matrix statistical features. We have concluded the STLA-GA selected the least feature subset (21 out of 48) amongst the three feature selection options and yet these features are the most relevant ones.

Fig. 14 and 15 also illustrate the utility of the exploration-exploitation cycle in adapting to updated classifier performance. Both STLA-GA and Standard GA experienced local convergence from the beginning of simulation. Eventually, STLA-GA activated the exploration setting and overcame local convergence at second generation. Likewise, Standard GA solved local convergence at second and fifth generation, but the unwanted prediction error was still greater than STLA-GA (29.13% versus 27.39%).

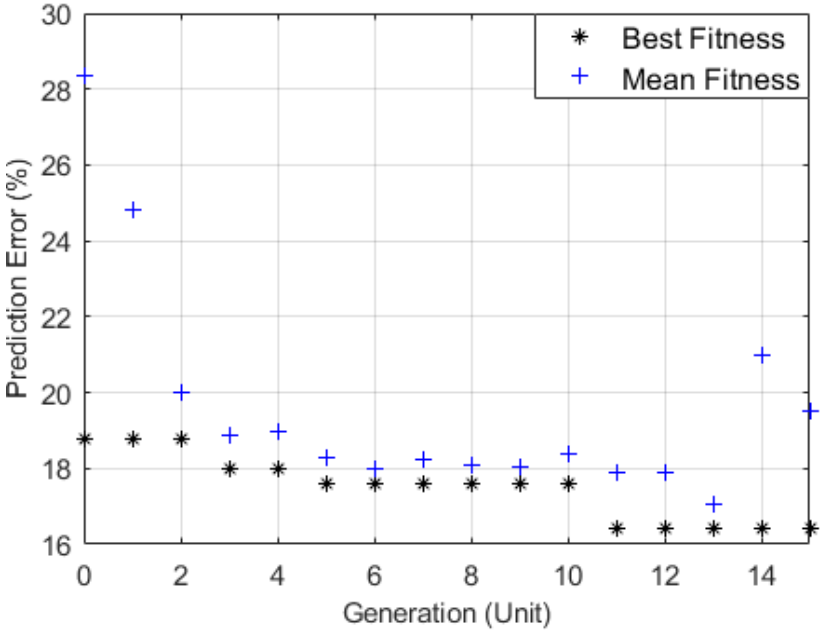

FIGURE 16. Centrifugal pumps residual analysis dataset: STLA-GA fitness function performance over generations.

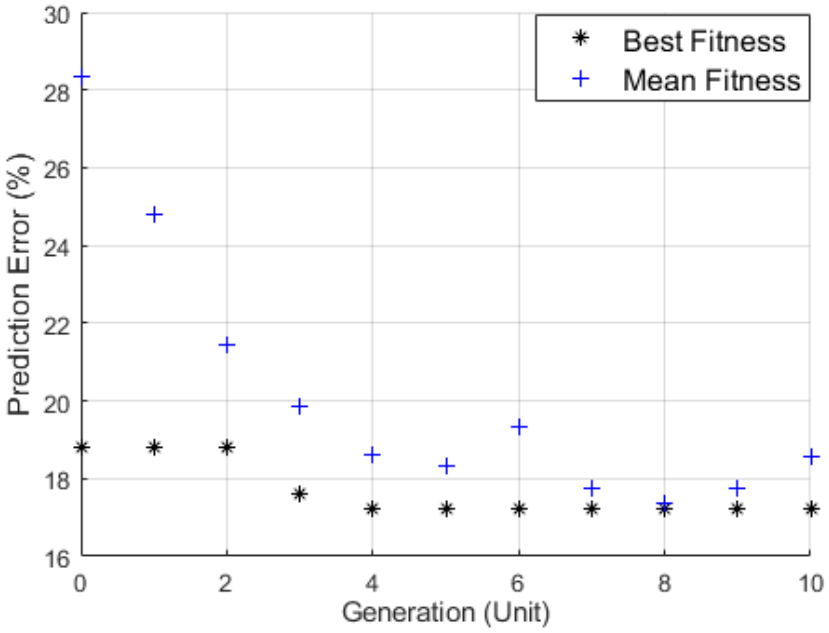

FIGURE 17. Centrifugal pumps residual analysis dataset: standard GA fitness function performance over generations.

4) DATASET 4: CENTRIFUGAL PUMPS RESIDUAL ANALYSIS FOR FAULT DIAGNOSIS

Table 17 details the centrifugal pumps fault diagnosis multi-class classification prediction, with 11 of 20 unique residue features acquired from the nonlinear parity-equations deemed to be of STLA-GA preference. The STLA-GA feature subset-trained classifier delivered optimal performance with the highest prediction accuracy (85.60%) and least *kfoldLoss* value (0.1474), as well as having the best feature reduction percentage at 45%. The feature subset chosen by STLA-GA are also excelled in the confusion matrix statistical analysis. The superiority of STLA-GA feature subset-trained classifier is best shown with the highest MCC and F-Measure values amongst feature selection alternatives.

Fig. 16 and 17 also demonstrate the ability of STLA-GA to obtain a lower prediction error than Standard GA. Local convergence is observed in both Standard GA and STLA-GA from the beginning until the second generation. Over the generations, STLA-GA gradually shifted to the exploration setting instead of the static genetic operator setting in an effort to reduce prediction error and again restored to the exploitation setting immediately upon the performance update. Ultimately, STLA-GA managed to outperform Standard GA with further prediction error reduction in the eleventh generation before the simulation was terminated. Additionally, revision of the maximum generation limit *opt gen* proven effective when STLA-GA overcame local convergence multiple times, including after tenth generation in this case study.

VI. CONCLUSION

The novel parameter tuning method, STLA-GA, has been introduced to solve the local optimum problem in heuristic optimisation. STLA-GA parameter tuning is performed by embedding a unique exploration-exploitation cycle as a function of the latest fitness performance. The parameter tuning strategy can be categorised using two main adaptive characteristics. First, STLA-GA tunes mutation probability rate, population size, and novel convergence threshold to target a desirable search space, avoid local convergence, and eliminate manual parameter tuning. Second, maximum generation number and customized stopping criteria are altered to enable stable parameter tuning feedback while keep distance from premature termination and excessive computational effort.

Sufficient machinery-related dataset prediction simulation and the feature selection methodology benchmark have explicitly emphasised STLA-GA's optimal performance regardless of classifier choice. Relative to Standard GA and BPSO as reputable feature selection choices, STLA-GA returns more relevant feature subset or higher feature reduction percentages. The significance of the STLA-GA feature subset is expressed as better classifier prediction accuracy and minimal *kfoldLoss*. The quality of the STLA-GA feature subset-trained classifier has been further validated by outstanding confusion matrix statistical measures, in particular of MCC and F-Measure. The main difference between STLA-GA and standard practice is that the adaptive exploration-exploitation cycle embedded in STLA-GA allows feasible parameter tuning for advantageous heuristic optimisation behavior. The influence of the exploration-exploitation cycle has been verified by comparing fitness performance over generations between STLA-GA and Standard GA (Fig. 10 to Fig. 17).

However, STLA-GA performance and feature subset recorded amongst the highest standard deviation values and computational effort in the form of total evaluation number and time for all cases. The uncertainty related to performance is inevitable since STLA-GA has been given an additional parameter tuning task in order to maintain a certain level of population diversity. Yet, reasonable population diversity and computation costs are rewarded with a superior classifier and feature subset combination.

The STLA-GA stopping criteria analysis also indicates insightful parameter tracking information at the end of simulation. Based on the 10 observations, the termination of STLA-GA was triggered by various customized stopping criteria, while standard GA and BPSO ends when the maximum generation number is achieved. Indeed, an extensive range of simulation likelihoods are considered through STLA-GA hybrid stopping criteria, despite dealing with multi-objective optimisation. The hybrid STLA-GA stopping criteria not only enable a parameter stability feedback mechanism but also perform excessive computation costs and unrealistic GA information generation detection.

The existing algorithm will be further improved with an integrated classifier in order to reduce performance discrepancy due to classifier selection. It is possible to describe, in theoretical terms, the Multiple Input Multiple Output (MIMO) mathematical relationship with the application of system identification modeling. Furthermore, the algorithm will also be tested on different dataset types, such as images and text recognition.

REFERENCES

[1] J. G. Monroe, J. Ducoste, and E. Z. Berglund, "Genetic algorithm–genetic programming approach to identify hierarchical models for ultraviolet disinfection reactors," *J. Environ. Eng.*, vol. 145, no. 2, 2019. doi: 10.1061/(ASCE)EE.1943-7870.0001492.

[2] V. Mytilinou and A. J. Kolios, "Techno-economic optimisation of offshore wind farms based on life cycle cost analysis on the UK," *Renew. Energy*, vol. 132, no. 2, pp. 439–454, Mar. 2019. doi: 10.1016/j.renene.2018.07.146.

[3] Y.-L. Lai, P.-C. Shen, C.-C. Liao, and T.-L. Luo, "Methodology to optimize dead yarn and tufting time for a high performance CNC by heuristic and genetic approach," *Robot. Comput.-Integr. Manuf.*, vol. 56, pp. 157–177, Apr. 2019. doi: 10.1016/j.rcim.2018.09.006.

[4] H. Lu, H. Wang, and S. W. Yoon, "A dynamic gradient boosting machine using genetic optimizer for practical breast cancer prognosis," *Expert Syst. Appl.*, vol. 116, pp. 340–350, Feb. 2019. doi: 10.1016/j.eswa.2018.08.040.

[5] A. Cano and B. Krawczyk, "Evolving rule-based classifiers with genetic programming on GPUs for drifting data streams," *Pattern Recognit.*, vol. 87, pp. 248–268, Mar. 2019. doi: 10.1016/j.patcog.2018.10.024.

[6] R. Hou, Y. Xia, Q. Xia, and X. Zhou, "Genetic algorithm based optimal sensor placement for L_1-regularized damage detection," *Struct. Control Health Monit.*, vol. 26, no. 1, p. e2274, 2019. doi: 10.1002/stc.2274.

[7] J. Zheng, D. Zhang, K. Huang, and Y. Sun, "Image segmentation framework based on optimal multi-method fusion," *IET Image Process.*, vol. 14, no. 1, pp. 186–195, 2019. doi: 10.1049/iet-ipr.2018.5338.

[8] H. M. Pandey, A. Chaudhary, and D. Mehrotra, "A comparative review of approaches to prevent premature convergence in GA," *Appl. Soft Comput.*, vol. 24, no. 1, pp. 1047–1077, Nov. 2014. doi: 10.1016/j.asoc.2014.08.025.

[9] A. Eiben, Z. Michalewicz, M. Schoenauer, and J. Smith, "Parameter control in evolutionary algorithms," in *Studies in Computational Intelligence*. Berlin, Germany: Springer-Verlag, 2007, pp. 19–46. [Online]. Available: https://hal.inria.fr/inria-00140549/document

[10] T.-P. Hong and H.-S. Wang, "Automatically adjusting crossover ratios of multiple Crossover Operators," *J. Inf. Sci. Eng.*, vol. 14, no. 2, pp. 369–390, 1998.

[11] C. L. Karr and E. Wilson, "A self-tuning evolutionary algorithm applied to an inverse partial differential equation," *Appl. Intell.*, vol. 19, no. 3, pp. 147–155, Nov. 2003.

[12] A. F. Cruz-Salinas and J. G. Perdomo, "Self-adaptation of genetic operators through genetic programming techniques," in *Proc. Genetic Evol. Comput. Conf. (GECCO)*. New York, NY, USA: ACM, 2017, pp. 913–920. doi: 10.1145/3071178.3071214.

[13] A. Koromyslova, M. Semenkina, and R. Sergienko, "Feature selection for natural language call routing based on self-adaptive genetic algorithm," in *Proc. IOP Conf. Ser., Mater. Sci. Eng.*, vol. 173, 2017, Art. no. 012008. doi: 10.1088/1757-899X/173/1/012008.
[14] M. Angelova and T. Pencheva, "How to assess multi-population genetic algorithms performance using intuitionistic fuzzy logic," in *Advanced Computing in Industrial Mathematics. BGSIAM* (Studies in Computational Intelligence), vol. 793, K. Georgiev, M. Todorov, I. Georgiev, Eds. Cham, Switzerland: Springer, 2018, pp. 23–35. doi: 10.1007/978-3-319-97277-0_3.
[15] J. Kivijärvi, P. Fränti, and O. Nevalainen, "Self-adaptive genetic algorithm for clustering," *J. Heuristics*, vol. 9, no. 2, pp. 113–129, 2003. doi: 10.1023/A:1022521428870.
[16] J. Pérez, R. A. Pazos, J. Frausto, G. Rodríguez, L. Cruz, G. Mora, and H. Fraire, "Self-tuning mechanism for genetic algorithms parameters, an application to data-object allocation in the Web," in *Computational Science and Its Applications—ICCSA* (Lecture Notes in Computer Science), vol. 3046, A. Laganá, M. L. Gavrilova, V. Kumar, Y. Mun, C. J. K. Tan, and O. Gervasi, Eds. Berlin, Germany: Springer, 2004, pp. 77–86. doi: 10.1007/978-3-540-24768-5_9.
[17] A. E. Eiben, M. C. Schut, and A. R. de Wilde, "Boosting genetic algorithms with self-adaptive selection," in *Proc. IEEE Int. Conf. Evol. Comput.*, Vancouver, BC, Canada, Jul. 2006, pp. 477–482. doi: 10.1109/CEC.2006.1688348.
[18] A. Aleti, "An adaptive approach to controlling parameters of evolutionary algorithms," Ph.D. dissertation, Swinburne Univ. Technol., London, U.K., 2012. [Online]. Available: http://users.monash.edu.au/~aldeidaa/papers/thesis.pdf
[19] S. Blum, R. Puisa, J. Reidel, and M. Wintermantel, "Adaptive mutation strategies for evolutionary algorithms," in *Proc. EVEN Weimarer Optimierungsund Stochastiktage 2.0*, 2005, pp. 1–13. [Online]. Available: https://www.dynardo.de/fileadmin/Material_Dynardo/bibliothek/WOST_2.0/WOST_2_AdaptiveMutation_En.pdf
[20] D. Popovic, C. Moschopoulos, R. Sakai, A. Sifrim, J. Aerts, Y. Moreau, and B. De Moor, "A self-tuning genetic algorithm with applications in biomarker discovery," in *Proc. IEEE 27th Int. Symp. Comput.-Based Med. Syst.*, New York, NY, USA, May 2014, pp. 233–238. doi: 10.1109/CBMS.2014.10.
[21] L. A. Wulandhari, A. Wibowo, and M. I. Desa, "Improvement of adaptive GAs and back propagation ANNs performance in condition diagnosis of multiple bearing system using grey relational analysis," *Comput. Intell. Neurosci.*, vol. 2014, 2014, Art. no. 419743. doi: 10.1155/2014/419743.
[22] N. T. P. Quyen, J. C. Chen, and C.-L. Yang, "Hybrid genetic algorithm to solve resource constrained assembly line balancing problem in footwear manufacturing," *Soft Comput.*, vol. 21, no. 21, pp. 6279–6295, Nov. 2017. doi: 10.1007/s00500-016-2181-3.
[23] C. Alabas-Uslu and B. Dengiz, "A self-adaptive heuristic algorithm for combinatorial optimization problems," *Int. J. Comput. Intell. Syst.*, vol. 7, no. 5, pp. 827–852, 2014. doi: 10.1080/18756891.2014.966992.
[24] M. S. Nobile, P. Cazzaniga, D. Besozzi, R. Colombo, G. Mauri, and G. Pasi, "Fuzzy Self-Tuning PSO: A settings-free algorithm for global optimization," *Swarm Evol. Comput.*, vol. 39, pp. 70–85, 2018. doi: 10.1016/j.swevo.2017.09.001.
[25] M. K. A. Ariyaratne, T. G. I. Fernando, and S. Weerakoon, "A self-tuning modified firefly algorithm to solve univariate nonlinear equations with complex roots," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Vancouver, BC, Canada, Jul. 2016, pp. 1477–1484. doi: 10.1109/CEC.2016.7743964.
[26] B. Gloger, "Self adaptive evolutionary algorithms," Univ. Paderborn, Paderborn, Germany, Tech. Rep. Fachbereich 17—Mathematik/Informatik, 2004. [Online]. Available: https://pdfs.semanticscholar.org/0e61/76ae5ba91c6147b1a33574412488cba18bb8.pdf
[27] C.-C. Hsu, S.-I. Yamada, H. Fujikawa, and K. Shida, "A fuzzy self-tuning parallel genetic algorithm for optimization," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 883–893, Sep. 1996. doi: 10.1016/0360-8352(96)00039-3.
[28] M. Burtscher and P. Ratanaworabhan, "gFPC: A self-tuning compression algorithm," in *Proc. Data Compress. Conf.*, Snowbird, UT, USA, Mar. 2010, pp. 396–405. doi: 10.1109/DCC.2010.42.
[29] M. Castelli, L. Manzoni, L. Vanneschi, S. Silva, and A. Popovič, "Self-tuning geometric semantic genetic programming," in *Genetic Programming and Evolvable Machines*, vol. 17, no. 1, pp. 55–74, Mar. 2016. doi: 10.1007/s10710-015-9251-7.
[30] F. G. Lobo and C. F. Lima, "Adaptive population sizing schemes in genetic algorithms," in *Parameter Setting in Evolutionary Algorithms* (Studies in Computational Intelligence), vol. 54, F. G. Lobo, C. F. Lima, and Z. Michalewicz, Eds. Berlin, Germany: Springer, 2007. doi: 10.1007/978-3-540-69432-8_9.
[31] F. Marini and B. Walczak, "Particle swarm optimization (PSO). A tutorial," *Chemometrics Intell. Lab. Syst.*, vol. 149, pp. 153–165, Dec. 2015. doi: 10.1016/j.chemolab.2015.08.020.
[32] N. Helwig, E. Pignanelli, and A. Schütze, "Condition monitoring of a complex hydraulic system using multivariate statistics," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, Pisa, Italy, 2015, pp. 210–215. doi: 10.1109/I2MTC.2015.7151267.
[33] F. Bergadano, A. Giordana, L. Saitta, F. Brancadori, and D. De Marchi, "Integrated learning in a real domain," in *Proc. 7th ML Conf.*, Austin, TX, USA, 1990, pp. 322–329. doi: 10.1016/B978-1-55860-141-3.50042-0.
[34] L. S. Lopes and L. M. Camarinha-Matos, "Feature transformation strategies for a robot learning problem," in *Feature Extraction, Construction and Selection* (The Springer International Series in Engineering and Computer Science), vol. 453, H. Liu and H. Motoda, Eds. Boston, MA, USA: Springer, 1998. doi: 10.1007/978-1-4615-5725-8_23.
[35] R. Isermann, *Fault-Diagnosis Applications. Model-Based Condition Monitoring: Actuators, Drives, Machinery, Plants, Sensors, and Fault-tolerant Systems*. Berlin, Germany: Springer-Verlag, 2011. doi: 10.1007/978-3-642-12767-0.
[36] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manage.*, vol. 45, no. 4, pp. 427–437, Jul. 2009. doi: 10.1016/j.ipm.2009.03.002.

CHING SHENG OOI received the B.Eng. degree in automation and manufacturing system engineering from Sheffield Hallam University, U.K., in 2010, and the M.Sc. degree in control engineering from Coventry University, U.K., in 2014. He is currently pursuing the Ph.D. degree with the Institute of Noise and Vibration, Universiti Teknologi Malaysia (UTM) Kuala Lumpur, Kuala Lumpur, Malaysia. His research interests include machinery condition monitoring, and fault diagnosis using signal processing and artificial Intelligence.

MENG HEE LIM received the B.Eng., M.Eng., and Ph.D. degrees in mechanical engineering from Universiti Teknologi Malaysia (UTM) Skudai, Malaysia, in 2000, 2003, and 2010, respectively, where he is currently a Senior Lecturer and also the Senior Consultant of the Institute of Noise and Vibration, with more than 15 years of experience in acoustics and vibration. He is specialized in power generation gas and steam turbines, blade failures diagnostics, and experimental modal analysis.

MOHD SALMAN LEONG received the B.Sc. and Ph.D. degrees in mechanical engineering from Heriot-Watt University, U.K., in 1978 and 1983, respectively. He is currently a Professor with Universiti Teknologi Malaysia (UTM), where he is also a Principal Consultant and the Founder of the Institute of Noise and Vibration, with more than 40 years professional engineering consulting experience. He is acknowledged by the industry and government agencies as the leading authority in acoustics and noise vibration in the country. He has been involved in many of the mega-projects and high impact consulting and investigation projects in oil gas, power generation, infrastructure, and construction industries.