An ensemble learning algorithm for optimization of spark ignition engine performance fuelled with methane/hydrogen blends

Mohammad-H. Tayarani-N.*[a], Amin Paykani[b]

[a] *University of Hertfordshire, College Lane, Hatfield, AL10 9AB, United Kingdom*
[b] *School of Engineering and Materials Science, Queen Mary University of London, London, E1 4NS, United Kingdom*

Abstract

The increasing global demand for sustainable and cleaner transportation has led to extensive research on alternative fuels for Internal Combustion (IC) engines. One promising option is the utilization of methane/hydrogen blends in Spark-Ignition (SI) engines due to their potential to reduce Green House Gas (GHG) emissions and improve engine performance. However, the optimal operation of such an engine is challenging due to the interdependence of multiple conflicting objectives, including Brake Mean Effective Pressure (BMEP), Brake Specific Fuel Consumption (BSFC), and nitrogen oxide (NO_x) emissions. This paper proposes an evolutionary optimization algorithm that employs a surrogate model as a fitness function to optimize methane/hydrogen SI engine performance and emissions. To create the surrogate model, we propose a novel ensemble learning algorithm that consists of several base learners. This paper employs ten different learning algorithms diversified via the Wagging method to create a pool of base-learner algorithms. This paper proposes a combinatorial evolutionary pruning algorithm to select an optimal subset of learning algorithms from a pool of base learners for the final ensemble algorithm. Once the base learners are designed, they are incorporated into an ensemble, where their outputs are aggregated using a weighted voting scheme. The weights of these base learners are optimized through a gradient descent algorithm. However, when optimizing a problem using surrogate models, the fitness function is subject to approximation uncertainty. To address this issue, this paper introduces an uncertainty reduction algorithm that performs averaging within a sphere around each solution. Experiments are performed to compare the proposed ensemble learning

algorithm to the classical learning algorithms and state-of-the-art ensemble algorithms. Also, the proposed smoothing algorithm is compared with the state-of-the-art evolutionary algorithms. Experimental studies suggest that the proposed algorithms outperform the existing algorithms.

Keywords: Spark Ignition Engine, Methane, Hydrogen, Evolutionary Algorithms, Surrogate Models, Ensemble Learning.

---

1. Introduction

Passenger cars, motorcycles, and small engines rely on Spark Ignition (SI) combustion mode, but because of the low compression ratio and stoichiometric operation, their thermal efficiency is limited. Increasing per capita energy demand and stringent CO_2 emissions regulations motivate the use of low-carbon fuels in the transport sector. Natural gas has a crucial impact on reducing CO_2 emissions from combustion engines thanks to their favourable H/C ratio [1, 2]. Additionally, the high octane number and high knock resistance of methane allow running the engine to have higher compression ratios [3, 4]. Moreover, lean natural gas combustion has shown the potential to improve efficiency compared to stoichiometric gasoline engines, but suffers from unstable and poor ignitability of the fuel-air mixture, leading to incomplete combustion or misfire [5]. The reduction of flame speed at lean operation results in significant cycle-to-cycle variations (CCV) [6]. Hydrogen is considered a suitable candidate as an additive for lean-burn natural gas-fueled SI engines, due to its higher laminar flame speed, wider flammability limits and small quenching distance [7, 8].

The optimization of performance and emissions of low-carbon fueled SI engines is of paramount importance due to increasing environmental concerns and the demand for efficient and clean combustion systems. Achieving optimal engine performance involves finding the best combination of operating parameters, such as spark timing, air-fuel ratio, and fuel blend, while simultaneously minimizing emissions of pollutants such as Nitrogen oxide (NO_x). Traditional optimization methods often rely on manual adjustments and trial-and-error processes, which are time-consuming and may not guarantee optimal results [9, 10, 11]. In recent years, the combined use of machine learning and optimization techniques has emerged as a powerful approach to tackle this challenge [12, 13]. Machine learning algorithms, such as neural networks have the capability to learn from data and extract valuable insights.

By training on large datasets comprising engine performance measurements and design parameters, these algorithms can uncover complex relationships and patterns that may not be readily apparent through traditional analytical methods. Optimization techniques, on the other hand, provide a systematic way to search for the best set of design parameters that optimize specific objectives, such as fuel efficiency, power output, or emissions reduction. By defining appropriate fitness functions and constraints, optimization algorithms can explore the vast design space to identify optimal solutions.

Traditionally, statistical and mathematical methods including response surface methodology were used for modelling. Recently, with the development of machine learning algorithms, it has been shown that these algorithms are more capable of finding an accurate model for the problem in this paper. This is especially true when the problem is complex and few data records are available. In these cases, the traditional methods fail to provide accurate results [14, 15].

Many works have been done in the literature to optimize the performance of hydrogen mixture-fueled engines via machine learning algorithms. Performance and emission parameters of wheat germ oil hydrogen dual fuel were investigated in [16], in which machine learning algorithms are employed to predict the parameters of the engine. In [17], an engine with non-edible rubber seed oil biodiesel with hydrogen is studied, where machine learning algorithms are employed to build a model of the engine. Genetic algorithms and machine learning algorithms are employed in [18, 19] to optimize a hydrogen-fueled Wankel rotary engine. In this work, to train the model, the Latin hypercube sampling was performed to collect data. To perform a regression study of a Hydrogen-enriched Compressed Natural Gas (HCNG) fueled spark ignition engine, ML algorithms are employed in [20], where the engine torque and NOx emissions have been used to create the model. To study the efficacy of experimental results for a dual fuel compression engine, on hydrogen and diesel, 29 regression algorithms were employed in [21] to model NOx, CO2, and smoke of the engine. In [22], an enhanced automated Machine Learning model is presented for the optimization of cycle-to-cycle variation in hydrogen-enriched methanol engines. The authors show that the ML approaches provide a higher quality Pareto front. In another work [23], an engine fueled with different hydrogen to Compressed Natural Gas (CNG) blends was investigated under different fuel rations. The work uses support vector machines to study engine behaviour.

In many real-world optimization problems, the calculation of the fitness

function is so expensive that optimization becomes too arduous and time-consuming, if not impossible [24]. These problems, which are known as Expensive Multi-Modal Optimization Problems (EMMOPs) [25] are usually optimized via surrogate models, that is a surrogate model of these systems is built, and the optimization is performed based on the model. In some problems, using the real problem in the optimization is not practical. For example, in [26], a deep neural network is used to build a surrogate model of groundwater level. To solve the partial differential equation-constrained optimization problem, a deep neural network is presented in [27], in which an iteration selection of training data through a feedback loop between the neural network and the optimization problem is adopted. In [28], the optimization of the proton exchange membrane fuel cell is performed via a three-dimensional steady-state model. Another approach is to use multi-surrogate models, in which multiple surrogates are used to create models of different aspects of the system [29]. When a surrogate model of a system is created, it is usually difficult to make a match between the surrogate model and the problem modalities. To manage this, a dual surrogate-assisted algorithm is presented in [30], to detect new modalities. A literature review on the applications of surrogate models in evolutionary optimization can be found in [31].

It is shown in the literature that a combination of learning algorithms, also known as ensemble learning outperforms individual learning algorithms [32, 33, 34]. This is because the combination provides diversity [35] among the decision-makers. Diversity in ensemble learning is achieved via Heterogeneous and Homogeneous approaches. To achieve diversity, Heterogeneous methods [36] use different classifiers and Homogeneous [37] methods inject randomness into the training phase of the learning algorithm (example of which include bagging [37] that changes the distribution of training data to reach different training sets and random subspace [38] that creates randomness by randomly selecting a subset of features). Because ensemble learning algorithms have shown promising results, in this paper we design an ensemble learning algorithm to build a surrogate model of the hydrogen mixture SI engine fueled with methane/hydrogen blends.

The ensemble learning algorithm in this paper is formed of a set of learning algorithms including Learning Vector Quantization Neural network (LVQ) [39], Probabilistic Neural Networks (PNN) [40], Feed-forward Neural Networks (FNN) [41], Cascade-Forward Neural Networks (CFNN) [42], Radial Basis Networks (RBN), Function Fitting Neural Network (FFNN) [43],

K-Nearest Neighbor (KNN) [44], Pattern Recognition Network (PRN) [45], Generalized Regression Neural Network (GRNN) [46], and Exact Radial Basis Network (RBE) [47]. These algorithms can be categorized into two main groups. On the one hand, we have discriminative learning algorithms which include KNN, LVQ, FNN, CFNN, RBN, FFNN, PRN, and GRNN. On the other hand, we have a generative learning algorithm that includes PNN.

Finding the best set of engine parameters that results in the optimal engine performance is a crucial task [48]. It often requires data collection and numerical simulation of the engine which are often time-consuming and costly. The use of machine learning methods in combination with optimization techniques has gained significant attention in recent years due to their potential to improve the performance of internal combustion engines. While traditional optimization methods, such as genetic algorithms [49, 50], have been extensively applied in engine optimization, machine learning approaches offer several advantages. Machine learning methods can effectively model complex relationships between input parameters and engine performance, allowing for more accurate and efficient optimization [51]. These methods can discover hidden patterns and nonlinear dependencies, enabling the identification of optimal operating conditions that may be overlooked by traditional methods. Additionally, machine learning algorithms can adapt and learn from data, enabling the optimization process to continuously improve over time. By leveraging the power of machine learning and optimization, researchers can uncover optimal engine configurations that maximize performance, minimize emissions, and enhance fuel efficiency more effectively than traditional genetic algorithm methods. Therefore, incorporating machine learning methods into engine optimization research holds great promise for achieving advanced levels of performance and efficiency in internal combustion engines.

The contribution of this paper can be summarized as follows

- A novel approach for optimizing the performance and emissions of an SI engine running on methane/hydrogen blends is presented which leverages evolutionary optimization techniques to achieve the desired engine performance.

- A surrogate model of the engine is developed and an evolutionary algorithm is employed to optimize its performance. To construct the surrogate model, we introduce an ensemble learning algorithm comprising multiple base learners.

- It is crucial to ensure diversity among these base learners to enhance overall performance. To address this, we propose an evolutionary algorithm that optimizes the diversity within the ensemble learning algorithm.

- An evolutionary pruning algorithm is introduced to identify the optimal subset of base learners, thus maximizing the performance of the ensemble learning algorithm.

- When surrogate models are employed to evaluate the fitness of solutions, uncertainties arise due to approximation errors. To mitigate this issue, we propose an uncertainty reduction algorithm that performs averaging within a sphere surrounding the current solution. This algorithm effectively reduces uncertainty and enhances the accuracy of fitness evaluations.

- To validate our proposed approach, we compare it against several state-of-the-art algorithms, and our experimental results demonstrate its superior performance. Our method offers significant advancements in optimizing the performance and emissions of SI engines fueled with methane/hydrogen blends. By integrating evolutionary optimization, ensemble learning, uncertainty reduction, and surrogate modelling techniques, we pave the way for the optimal design of more efficient and cleaner combustion systems.

The rest of this paper is organized as follows. Section 2 explains how the data are acquired and prepared, in section 3 we identify and describe the optimization problem. The proposed method is presented in section 4, section 5 presents the experimental studies, and section 6 concludes the paper.

2. Engine specifications and experimental facility

A series of measurements were carried out on a four-stroke single-cylinder SI engine on a test bench. The engine specifications are given in Table 1 [50]. To calculate the output torque and control the speed, the engine is mounted on a water-cooled eddy current dyno. A gas mixing system consisting of one flow sensor for CH_4 and three flow controls for the other gases is mounted before the gas valve to change the desired fuel mixture of CH_4 and H_2. For this study, no synthetic Exhaust Gas Recirculation (EGR)

has been taken into account. For correct model parameter calibration, a venturi mixer homogeneously mixes the intake air and the fuel until it reaches the engine.

Table 1: Engine Specifications.

Engine name	Swissauto
Number of cylinders	1
Bore (mm)	75
Stroke (mm)	56.5
Connecting rod length (mm)	95
Total displacement (cc)	250
Compression ratio	12.5:1
Inlet valve closing (CA deg ATDC)	-112
Exhaust valve closing (CA deg ATDC)	109
Fuel supply	Venturi gas mixer, naturally aspirated

The measurement matrix comprises variable methane-hydrogen ratio (f_{H_2}), air-fuel equivalence ratio (λ), and spark timing (ST) at a constant engine speed of 3000 rpm and fully unthrottled operation. Some of the experimental data points are listed in Table 2.

3. Engine Optimization

The brake mean effective pressure (BMEP) is the external shaft work per unit volume done by the engine. It measures the engine's ability to achieve high airflow and use that air effectively to generate torque. BMEP is a measure of the engine's efficiency in converting fuel into mechanical work. Higher BMEP values indicate a more powerful engine, which is crucial for applications where performance and power are priorities. It allows for the comparison of engines of different sizes and configurations on a common basis, making it easier to evaluate performance improvements.

The brake-specific fuel consumption (BSFC) is the fuel flow rate divided by the brake power. BSFC measures the amount of fuel consumed by the engine to produce a specific amount of power. It is a direct indicator of an engine's fuel efficiency. Lower BSFC values signify that the engine is consuming less fuel to produce a given amount of power, which is essential for reducing operating costs and environmental impact. There is a trade-off between maximum power output (BMEP) and fuel efficiency (BSFC). During

Table 2: Validation cases and operating parameters.

Case	Speed	Air–fuel equivalence ratio (λ)[1]	Hydrogen fraction (f_{H_2})[2]	Spark timing (ST)
	(rpm)	(-)	(%vol)	(CA BTDC)[3]
1	3000	1.4	0	45
2	3000	1.4	10	45
3	3000	1.4	25	45
4	3000	1.4	25	60
5	3000	1.6	10	45
6	3000	1.6	25	45
7	3000	1.6	50	45
8	3000	1.6	25	70
9	3000	1.8	50	45
10	3000	1.8	50	60

[1] λ is the air-fuel equivalence ratio which is the ratio of actual air-fuel ratio to stoichiometry for a given mixture. $\lambda = \frac{\text{actual air-fuel ratio}}{\text{stoichiometric air-fuel ratio}}$.
[2] $f_{\mathrm{H_2}} = \frac{\text{Volume of hydrogen}}{\text{Total volume of the mixture}}$. [3] stands for Crank Angle Before Top Dead Center.

the design and development phases, BMEP is often used to set targets for power output, while BSFC is used to set targets for fuel efficiency. Optimising engine performance involves finding a balance between these two metrics to meet specific application requirements, whether it is for high power output or fuel economy. Achieving high BMEP with low BSFC is the goal, but this often involves compromises and trade-offs that must be carefully managed. BMEP and BSFC often have an inverse relationship; increasing power (BMEP) can lead to higher fuel consumption (BSFC). By considering both BMEP and BSFC, engine designers can achieve a holistic optimisation of the SI engine's performance [52].

This research paper addresses the problem of multi-objective optimization for an SI engine fueled with methane/hydrogen blends. The primary objective is to find an optimal operating point that simultaneously maximizes Brake Mean Effective Pressure (BMEP) and minimizes Brake Specific Fuel Consumption (BSFC) and Nitrogen Oxides (NO_x) emissions. Achieving this balance requires considering various engine parameters, including spark

timing, air-fuel ratio, and hydrogen content in the fuel blend. We develop a surrogate optimization algorithm to explore the design space to identify the optimal Pareto solutions, representing the trade-offs between BMEP, BSFC, and NO_x emissions. The obtained Pareto front enables engine designers to make informed decisions based on the desired trade-offs between BMEP, BSFC, and NO_x emissions. Furthermore, the results provide insights into the impact of various engine parameters on the performance and emissions characteristics of methane/hydrogen-fueled SI engines.

4. An Ensemble Algorithm as the Surrogate Model

In this research paper, our primary objective is to construct a surrogate model of the engine and subsequently utilize optimization algorithms on the model to identify the optimal set of parameters that optimize engine performance and emissions across multiple objectives. However, conducting engine experiments and generating the necessary data can be both time-consuming and costly. Extensive experimentation, data collection, and equipment maintenance impose significant financial burdens. Moreover, acquiring a substantial amount of data can lead to significant delays in developing and deploying machine learning models. In this particular experiment, the dataset is limited to only 68 data points, which adversely affects the performance of the learning algorithms and subsequently impacts the optimization process through the surrogate model. To address this challenge, we propose the use of an ensemble learning algorithm specifically tailored to mitigate these limitations. It is widely acknowledged in the literature that for ensemble learning algorithms to exhibit optimal performance, the base learners comprising the ensemble should possess diverse characteristics [36]. When designing ensemble learning algorithms, two distinct types of diversity can enhance the resulting ensemble's performance [53]. The first type is inherent diversity, which arises from the inherent differences in the structure and nature of the base learners themselves. For instance, the inherent diversity can be observed in the contrasting characteristics of Support Vector Machines (SVM) and Artificial Neural Networks (ANNs). On the other hand, behavioural diversity is achieved through data manipulation during training, aiming to create diversity among the base learners. By leveraging these diverse characteristics, our proposed ensemble learning algorithm aims to overcome the challenges associated with limited data availability and improve the overall performance of the optimization process.

Many works have tried to improve the diversity among the base learners in order to improve the performance of the ensemble learning algorithms [53]. This is usually performed by training the base learners with different subsets of training data. Building the surrogate model for the engine, this task cannot be performed in this paper because the number of training data is limited. Another approach used in the literature, which is called Wagging [54], gives different weights to each of the data records in the training phase. Thus, none of the training data is removed from the training phase of the base learners. In the original version of Wagging [55], the weights of the data records are determined via the Gaussian distribution. This results in the weight of some of the instances being zero which effectively removes them from the training phase [56]. This might not be an issue when the set of training data is large; however, in our case, it is not a suitable approach. To manage this, in [57] Poisson distribution is used to assign the random weights.

Wagging, or "Weighted Average of Model Predictions" is a technique primarily used in ensemble learning, where multiple models are combined to make predictions. The idea behind wagging is to reduce variance and improve generalization by leveraging the diversity of multiple models [58]. In this algorithm, a set of diverse base models is trained on the same dataset using different subsets of the data or by employing different algorithms. The diversity among these models can be achieved through various means such as using different features, different training algorithms, or different hyperparameters [59]. In this scheme, instead of directly averaging the predictions of all models, wagging assigns weights to each model's predictions based on their performance or reliability. Typically, models with higher accuracy or lower error rates are assigned higher weights [60]. To generate the output, a weighted average serves as the final prediction of the ensemble model [61]. One advantage of the wagging algorithm is that by combining multiple models, wagging reduces overfitting and improves the model's ability to generalize to unseen data [62]. Another advantage is that ensemble methods like wagging are less sensitive to noisy data or outliers since they aggregate predictions from multiple sources [63]. Also, the diversity among base models allows the ensemble to capture different aspects of the underlying data distribution, leading to better overall performance [64]. Wagging can also provide insights into the relative importance of different models in the ensemble, offering some degree of interpretability [65].

The idea here is that in Wagging algorithms, assigning the weights to the training data is part of the training process and a random assignment

of the weights is not necessarily the optimal way of diversifying the base learners [66]. Assigning the weights is an optimization process and a search space can be defined on the weight space, where the objective is to find the set of classifiers that are as diverse as possible. In this research, we propose an evolutionary algorithm that gets as input a learning algorithm and a set of training data and finds and returns a set of weights for the Wagging algorithm that maximizes the performance of the resulting ensemble algorithm.

In this study, we try to develop an ensemble learning algorithm that benefits from both inherent and behavioural diversity [67]. To achieve inherent diversity, the ensemble learning algorithm comprises a set of different algorithms including Learning Vector Quantization Neural network (LVQ), Probabilistic Neural Networks (PNN), Feed-forward Neural Networks (FNN), Cascade-Forward Neural Networks (CFNN), Radial Basis Networks (RBN), Function Fitting Neural Network (FFNN), K-Nearest Neighbor (KNN), Pattern Recognition Network (PRN), Generalized Regression Neural Network (GRNN), and Exact Radial Basis Network (RBE). To choose these algorithms, a preliminary study was performed on different learning algorithms and it was found that these algorithms are most suited for the problem targeted in this paper. To decide which algorithms to choose in our ensembles, we used several learning algorithms and trained them on the data set that we studied in this paper. The learning algorithms were trained and tested, and the ones with the highest performances were used in the ensemble. We used the Matlab implementation of these algorithms and the Matlab default parameters were used to train the models.

The behavioural diversity is achieved via applying an evolutionary Wagging algorithm on each of these learning algorithms. The proposed algorithm trains each of the base learners with the set of training data, where each of the data records has different weights. These weights must be set in a way that these base learners achieve the best performance and diversity at the same time. The resulting ensemble learning algorithm then consists of a large number of base learners, because each of the learning algorithms has been diversified into several base learners. In the next step, the proposed algorithm applies a pruning [68] algorithm on the data to reduce the number of base learners and at the same time retain the performance of the ensemble algorithm. The pruning task is an optimization process that is performed in this research via a multi-objective optimization algorithm.

4.1. The Proposed Diversifier Algorithm

The proposed ensemble learning algorithm is presented in Figure 1. The algorithm consists of three phases, which are the diversifier, the pruning, and the ensemble learning algorithms. The diversifier gets as input the learning algorithms and using an evolutionary Wagging algorithm diversifies the learning algorithms to create some new set of learning algorithms. Then, in the next phase, the pruning algorithm removes some of the base learners and generates the ensemble algorithm.

The diversifier algorithm gets m learning algorithms L_k, $k = 1 \dots m$ as input. Then, it diversifies each of the learning algorithms L_k into n base learners denoted by L_k^l, $l = 1 \dots n$ that are diversified via the evolutionary Wagging algorithm. The algorithm generates $m \times n$ base learners that have both behavioural and inherent diversity. Some of this large number of base learners may be redundant, in the sense that removing some of them should decrease computational time without affecting the performance of the ensemble learning algorithm. Therefore, the pruning stage searches through the set of base learners and chooses a subset that maximizes the performance of the ensemble learning algorithm while minimizing the computational complexity of the algorithm. The output of the ensemble learning algorithm is found via a weighted sum of the base learners. The optimal value of the weights is found via a gradient descent algorithm.

The parameter n determines the number of learning algorithms in the ensemble. A preliminary study on this parameter shows that small n ($n < 5$) does not create a large enough ensemble, so the performance of the ensemble is not much improved compared to the base learners (compared to the individual base learners). A too-large n, on the other hand, ($n > 15$) creates a too-large ensemble which results in a high computational cost without any improvement in the performance. This is because the pruning algorithm chooses a subset of ensembles that reach the optimal performance. Thus, if a large n is chosen, the pruning algorithm removes the redundant learning algorithms to achieve the optimal set of base learning algorithms. We found $n = 10$ to be a reasonable number for the experiments.

Algorithm 1 presents the proposed evolutionary diversifier algorithm. The diversifier gets as input a learning algorithm, L_k, and the data and by assigning different weights to each data record in the Wagging process creates a set of base learners. The weights of the data records are assigned to maximize the base learner's individual accuracy and the diversity among them. In this evolutionary optimization of the weights of the data records, each individual

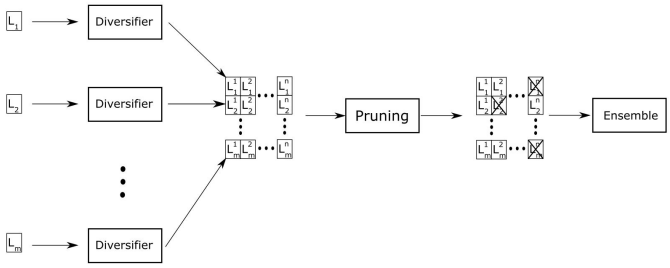

Figure 1: The proposed diversification and pruning algorithm structure.

is a matrix of size $d \times n$, where d is the number of data records and n is the number of base learners that are generated by the diversifier algorithm. The individuals are represented by x, where $x_{ij} \in (0, 1)$ represents the weight of the i-th data record in j-th base learner.

Algorithm 1: The proposed evolutionary diversifier algorithm.

1 $\tau = 0$;
2 input the learning algorithm L_k and the data;
3 initialize the population P^0 ;
4 **while** *not termination condition* **do**
5 $\quad \tau = \tau + 1$;
6 $\quad$ evaluate P^τ;
7 $\quad$ perform selection on $P^{\tau-1}$;
8 $\quad$ perform crossover and mutation on $P^{\tau-1}$ and generate P^τ;
9 **end**
10 RETURN the best-observed solution.;

In step 3 of the algorithm, the population is initialized randomly. The individuals are initialized randomly via the Poisson distribution. It is suggested in [57] that a Poisson random assignment of the weights offers the best performance. Initializing the individuals this way, creates solutions that are already good solutions and the evolutionary algorithm starts the search from a region in the search space that is more likely to include optimal solutions.

In step 6, the individuals are evaluated. This algorithm aims to get a learning algorithm and the data records and produce n base learners that are diverse. Here the only objective is not to increase diversity, because the performance of the base learners is also important. A set of base learners that

are diverse but are not individually accurate is not a good choice. Therefore, we have a two-objective optimization problem and propose an optimization algorithm that maximizes the performance of each base learner and the diversity among the base learners at the same time. To evaluate each individual x, the base learners L_k^l, $j = l \dots n$ are generated based on the weights suggested by x. Then, the base learners are trained and tested to measure their performance and diversity.

To measure the performance of a base learner, it is trained on the training data and tested on the validation data. Because the size of the data set is small here, dividing the data into training and validation data would result in a very small training set which affects the performance of the algorithm. To manage this, we use the *leave-one-out* with the 1-fold scheme in training and validation. That is, if there are d data records, the base learners are trained using $d-1$ data records and their performance is measured based on the one remaining data record. This whole process is performed d times until all the data records have been left out of the training phase and the performance of the base learner is measured based on the left-out data records. Then the mean squared error between the estimated and the true target is used as the performance and is found as,

$$E(L_k^l, x) = \frac{1}{3d} \sum_{u=1}^{d} \sum_{o=1}^{3} \left(L_k^l(y_u, x, o) - r_o^u \right)^2 \tag{1}$$

where $E(L_k^l, x)$ is the error of the base learner L_k^l according to the weights suggested by the individual x, r_o^u is the target for the data record y_u (r_1 is NO_x, r_2 is BSFC, and r_3 is BMEP), and $L_k^l(y_u, x, o)$ is the output of the base learner L_k^l for the o-th objective of the data record y_u when it is trained on the training data based on the weights suggested by x with the data record y_u left out. Note that the learning system has three outputs here, so the error is measured as the mean squared error over all three outputs. Also, note that the targets r_o are normalized between (0,1) so all have the same weight in the calculations.

The individual x produces n base learners denoted by L_k^l, $l = 1 \dots n$ and the first objective of the evolutionary algorithm is defined as the sum of the errors over all these base learners as,

$$\mathcal{E}(L_k, x) = \sum_{l=1}^{n} E(L_k^l, x). \tag{2}$$

The second objective is diversity among the base learners. Diversity is achieved when different classifiers miss-classify different data records. In modelling problems, diversity is achieved when there is a difference between the output of base learners for different data records. The diversity between two base learners is defined in this paper as,

$$D(L_k^l, L_k^h, x) = \frac{1}{3d} \sum_{u=1}^{d} \sum_{o=1}^{3} \left(L_k^l(y_u, x, o) - L_k^h(y_u, x, o) \right)^2, \tag{3}$$

where $L_k^l(y_u, x, o)$ and $L_k^h(y_u, x, o)$ are the o-th output of the base learner L_k^l and L_k^h for the data record y_u when it is trained on the training data based on the weights suggested by x with the data record y_u left out respectively. The diversity among all the base learners is found as

$$\mathcal{D}(L_k, x) = \frac{1}{n(n-1)} \sum_{l=1}^{n} \sum_{h=l+1}^{n} D(L_k^l, L_k^h, x). \tag{4}$$

Note that we have two types of diversity measures here. The function $\mathcal{D}(L_k, x)$ calculated the diversity among all the base learners, and the function $D(L_k^l, L_k^h, x)$ measures the diversity between two base learners.

Because we have a multi-objective optimization problem, we propose a fitness function in which the fitness of a solution x is measured according to the diversity and performance of its base learners with respect to those of other solutions in the population. The fitness of a solution x^i is measured as the number of solutions that are dominated by the solution as,

$$\mathcal{F}(x^i) = \sum_{j=1}^{p} [\![\mathcal{E}_{x^i}(L_k) < \mathcal{E}_{x^j}(L_k)]\!] + \sum_{j=1}^{p} [\![\mathcal{D}_{x^i}(L_k) > \mathcal{D}_{x^j}(L_k)]\!]$$

where $[\![statement]\!]$ returns 1 if the *statement* is true and returns 0 otherwise. This formula measures the fitness of the solution x^i by counting the number of solutions in the population that is dominated (outperformed in terms of performance in equation 2 and diversity in equation 4) by the solution x^i.

In step 7, the selection process is applied and the parents for the next generation are chosen. This paper uses the tournament selection method. In step 8, the crossover and mutation operators are applied.

4.2. The Proposed Pruning Algorithm

In the proposed algorithm, the diversifier algorithm gets each of the m learning algorithms and generates n base learners from each, so combined, $m \times n$ base learners are generated that are diverse in terms of behavioural and inherent diversity. Because the diversifier algorithm performs independently on the learning algorithms, the combined behaviour of the $m \times n$ learning algorithms is not considered in the optimization process. Also, an ensemble of $m \times n$ base learners is computationally expensive to employ for many applications. To optimize the performance of the ensemble learning algorithm and reduce its computational complexity, a pruning algorithm is employed in this paper to select from the $m \times n$ base learners a subset that offers the best performance for the ensemble learning algorithm.

Algorithm 2: The proposed pruning QEA algorithm.

```
1  τ = 0;
2  gets as input the the set of base-learners L_k^l, l = 1...n and
     k = 1...m;
3  initialize the population Q^0 based on equation 11;
4  observe Q^0 to generate Z^0;
5  evaluate Z^0;
6  store Z^0 into B^0;
7  while not termination condition do
8  |  τ = τ + 1;
9  |  observe Q^τ to generate Z^τ;
10 |  evaluate Z^τ ;
11 |  find the best neighbor of each q-individual and store in b^i if it is
   |    better than b^i ;
12 |  update Q^τ using Q-gate;
13 end
14 RETURN the best-observed solution.;
```

Algorithm 2 presents the pruning algorithm. In this paper, we adopt Quantum Evolutionary Algorithm QEA [69] for the pruning task. QEA was originally designed for the class of binary combinatorial optimization problems and because the pruning problem belongs to this category, QEA is, in nature more suitable for the problem. In our experiments, we show that

the performance of QEA in solving this problem is better than the other existing algorithms. A detailed description of the algorithm is as follows.

QEA uses a probabilistic representation for individuals, where the individuals are represented by a unit of information called q-bit which determines the probability of the bits in the individual being zero or one. Thus, in this representation, the solutions are each a string probability density functions that can represent binary strings and are represented as,

$$q = \begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_j & \dots & \alpha_u \\ \beta_1 & \beta_2 & \dots & \beta_j & \dots & \beta_u \end{bmatrix}, \tag{5}$$

where $|\alpha_j|^2 + |\beta_j|^2 = 1$, $|\alpha_j|^2$ is the probability of the j-th q-bit being zero, $|\beta_j|^2$ is the probability being one, and u is the problem dimension. The evolutionary process in QEA is performed via the "update" operator which in each step of the algorithm adjusts the values of α and β such that better solutions are represented with higher probability. The update operator in step 12 is defined as,

$$\begin{bmatrix} \alpha_j^{\tau+1} \\ \beta_j^{\tau+1} \end{bmatrix} = \begin{bmatrix} \cos(\delta\theta) & -\sin(\delta\theta) \\ \sin(\delta\theta) & \cos(\delta\theta) \end{bmatrix} \begin{bmatrix} \alpha_j^{\tau} \\ \beta_j^{\tau} \end{bmatrix}, \tag{6}$$

where $\delta\theta$ is the rotation angle that controls the convergence speed of the algorithm and τ is the iteration of the algorithm. A more detailed description of QEA can be found in [69]. We used the value $\delta\theta = 0.01$ as it was shown in the literature to provide the optimal results for most of the problems [69].

A description of the pruning algorithm in algorithm 2 is as follows. The population is initialized in step 3. The solutions in this algorithm, represented by z, are each an $m \times n$ matrix of zeros and ones where $z_{kl} = 1$ means that the base learner l_k^l is selected to participate in the ensemble and $z_{kl} = 0$ means the base learner l_k^l is pruned from the ensemble. In evolutionary algorithms, usually the initialization is performed randomly, in a way that the solutions are uniformly scattered around the search space. In QEA, this is performed by setting,

$$\begin{bmatrix} \alpha_{lf}^{i0} \\ \beta_{lf}^{i0} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}. \tag{7}$$

In the pruning problem, there is a prior knowledge that the base learners that perform better and incorporate more in the diversity are better to keep

and it is better to remove solutions that incorporate less into the diversity and are poorer in terms of performance. Thus, in this paper, we propose a method in the initialization step that generates quantum individuals that with more probability select the high-performance base learners and prune the inferior ones.

The performance of a base-learner is measured in a *leave-one-out* scheme, as explained in steps 5 and 10 of algorithm 2. The performance of the algorithm is measured as equation 2 and the diversity brought to the ensemble by the base learner L_k^l is measured by computing the pairwise diversity between the classifier L_k^l and all other base learners in the ensemble,

$$\mathbb{D}(L_k^l) = \sum_{g=1}^{m} \sum_{h=1}^{n} D(L_k^l, L_g^h), \tag{8}$$

where $D(.,.)$ is the diversity between two base learners and is measured as equation 3. The quality of a base learner then is calculated as,

$$\begin{aligned} \mathcal{G}(L_k^l) = &\sum_{g=1}^{m} \sum_{h=1}^{n} \left[\!\left[E(L_k^l) < E(L_h^g) \right]\!\right] + \\ &\sum_{g=1}^{m} \sum_{h=1}^{n} \left[\!\left[\mathbb{D}(L_k^l) > \mathbb{D}(L_h^g) \right]\!\right], \end{aligned} \tag{9}$$

where $[\![statement]\!] = 1$ if *statement* is true and it is $[\![statement]\!] = 0$ otherwise. Then the base learners are sorted based on their quality, and the rank of the solutions is used to determine the probability with which the base learner is pruned or left in the ensemble. Note that here, we are measuring the performance of the individual base learners, so we use equation 1. In the initialization step of QEA, the q-individuals are initialized to represent binary solutions that with higher probability contain the higher-quality base learners. The rank of a base learner L_k^l is found as,

$$\mathcal{R}(L_k^l) = \sum_{g=1}^{m} \sum_{h=1}^{n} \left[\!\left[\mathcal{G}(L_k^l) < \mathcal{G}(L_h^g) \right]\!\right]. \tag{10}$$

The rank varies between $[0, m \times n - 1]$ where the rank of the highest quality base learner is zero and the rank of the worst one is $m \times n - 1$. We propose

the following formula for initializing the q-individuals,

$$\begin{bmatrix} \alpha_{kl}^{i0} \\ \beta_{kl}^{i0} \end{bmatrix} = \begin{bmatrix} \sqrt{\frac{\mathcal{R}(L_k^l)}{m \times n - 1}} \\ \sqrt{\frac{m \times n - \mathcal{R}(L_k^l) - 1}{m \times n - 1}} \end{bmatrix}. \tag{11}$$

Using this initialization scheme, the q-individuals will represent the solutions in which the probability of the highest-quality base learner remaining in the ensemble is one and that of the lowest-quality base learner is zero. The probability of other base learners participating in the ensemble is set according to their quality.

The observed solutions are evaluated in steps 5 and 10 of the algorithm. The binary solution z is evaluated by training the ensemble learning algorithm based on the *leave-one-out* scheme and measuring its performance. In this scheme, all but one of the training data are used to train the learning algorithm, and the left-out data record is used to measure the performance of the algorithm. This process is performed for all the training data and the fitness of the solution is measured as the average performance of the ensemble learning algorithm over all the training data.

The evaluation process involves the training of the base learners based on the weights suggested by the diversifier algorithm (algorithm 1), selecting the subset of base learners suggested by the solution z_{kl}, and aggregating the selected base learners via a weighted voting system. For the data record y_u, the output of the ensemble is calculated as,

$$L(y_u, z, o) = \sum_{l=1}^{n} \sum_{k=1}^{m} z_{kl} w_{kl} L_k^l(y_u, o), \tag{12}$$

where z_{kl} determines if the base-learner L_k^l participates in the in the ensemble, $L(y_u, z, o)$ is the o-th output of the ensemble learning algorithm for the data record y_u, $L_k^l(y_u, o)$ is the o-th output of the base learner L_k^l for the data record y_u, and w_{kl} is the weight of the base learner L_k^l in the voting scheme.

The weight of each base learner in the voting process is optimized via a gradient descent algorithm. The cost function is defined as the mean square error between the predicted value via the ensemble and the target value and is measured as,

$$\mathcal{C}(L, z) = \frac{1}{3|T|} \sum_{o=1}^{3} \sum_{\forall y_u \in T} (L(y_u, z, o) - r_o^u)^2, \tag{13}$$

where $|T|$ is the number of records in the training data and T is the set of training data. The gradient of the error function with respect to the voting weights is found as,

$$\frac{\partial \mathcal{C}}{\partial w} = \frac{1}{3|T|} \frac{\partial \sum_{o=1}^{3} \sum_{\forall y_u \in T} (L(y_u, z, o) - r_o^u)^2}{\partial w} = \tag{14}$$

$$\frac{1}{3|T|} \frac{\partial \sum_{o=1}^{3} \sum_{\forall y_u \in T} (L(y_u, z, o) - r_o^u)^2}{\partial L(y_u, z, o)} \frac{\partial L(y_u, z, o)}{\partial w} = \tag{15}$$

$$\frac{2}{T} \sum_{o=1}^{3} \sum_{\forall y_u \in T} \left((L(y_u, z, o) - r_o^u) \sum_{l=1}^{n} \sum_{k=1}^{m} z_{kl} L_k^l(y_u, o) \right). \tag{16}$$

In the gradient descent process, the weights are updated as,

$$\delta w = \frac{2\eta}{T} \sum_{o=1}^{3} \sum_{\forall y_u \in T} \left((L(y_u, z, o) - r_o^u) \sum_{l=1}^{n} \sum_{k=1}^{m} z_{kl} L_k^l(y_u, o) \right), \tag{17}$$

where η is the learning rate in the gradient descent algorithm. When the ensemble algorithm is trained and the weights w_{kl} in the voting scheme are optimized, the performance of the ensemble algorithm is measured to find the fitness of the binary solution z. The fitness here is measured as the mean square error between the predicted value and the target value according to equation 1. The evolutionary process in algorithm 2 performs to find the best subset of base learners. After the pruning algorithm is finished and the optimal subset of base learners is found, the gradient descent algorithm in equation 17 runs to optimize the weight of the base learners in the voting scheme.

Note that in finding the fitness of a solution in the pruning process, the training of the ensemble learning algorithm only involves performing equation 17. Training the base learners L_k^l on the training data is performed once. The output of the base learners on each data record is stored in a lookup table and when it is needed in the fitness evaluation, it is fetched from the table.

Similar to PSO, in the QEA algorithm, the particles look at the fitness of the particles around them and move toward them. In step 11, each q-individual checks the value of its neighbouring solutions, chooses the best one and applies the update operator based on this value.

4.3. Evolutionary Optimization of Engine Performance

This paper proposes an ensemble learning algorithm to build a surrogate model of the engine performance under different conditions. Then, the surrogate model is used in an evolutionary algorithm to estimate the performance of the engine for a given set of input parameters. This is used as the fitness in the optimization process. To design the optimization algorithm some notes should be taken into account. First, the problem is a multi-objective optimization problem, where different conflicting objectives are to be optimized. Thus, a multi-objective optimization algorithm should be used. Second, because a surrogate model is used in the optimization process, the fitness of the solutions is affected by a type of uncertainty known as approximation uncertainty [70, 15]. Although the ensemble learning algorithm adopted in this paper acts as an uncertainty reduction process (as it performs averaging over a number of base learners' outputs), there still remains uncertainty in the fitness evaluation process. It is crucial to manage the uncertainty as it misleads evolutionary algorithms in their search process. There are many works that have targeted the uncertainty problem in evolutionary algorithms and the main approach has been to a way of averaging to reduce the uncertainty [71]. In these methods, the fitness function is re-sampled and a way of averaging is adopted. For the problem in this paper, however, resampling cannot provide a solution, because the surrogate model is deterministic and generates the same output for a given individual. To reduce approximation uncertainty, this paper proposes an averaging operator on the fitness of a number of solutions in a sphere around the current solution. The proposed evolutionary optimization of engine performance is presented in algorithm 3.

In algorithm 3, the set $\mathcal{H}$ is used to contain all the solutions that have been found through the search process. The uncertainty reduction algorithm uses this set in its process. In step 5 of the algorithm, the surrogate model is used to estimate the performance of the engine for the solution x_i. In step 8, algorithm 4 is used to reduce the uncertainty of the estimated performance via the surrogate model. The resulting value is denoted by $\hat{L}(x_i)$.

In this paper, we adopt SPEA2 [72] as the multi-objective algorithm and genetic algorithm as its evolutionary algorithm which calculates the fitness of the solution $\mathcal{F}'(x^i)$ as,

$$\mathcal{F}'(x^i) = \sum_{j=1}^{n} \left(\prod_{o=1}^{2} \left[\!\left[\hat{L}(x_i, o) < \hat{L}(x_j, o) \right]\!\right] \left[\!\left[\hat{L}(x_i, 3) > \hat{L}(x_j, 3) \right]\!\right] \right) \tag{18}$$

Algorithm 3: The proposed optimization.

```
1  initialize the population X;
2  H = ∅;
3  while not termination condition do
4  |  for i = 1 → n do
5  |  |  calculate the model's output L(x_i)
6  |  end
7  |  for i = 1 → n do
8  |  |  calculate L̂(x_i) via algorithm 4
9  |  end
10 |  for i = 1 → n do
11 |  |  calculate the fitness of x_i, F'(x_i) via equation 18
12 |  end
13 |  perform selection;
14 |  apply crossover and selection;
15 end
```

where $[\![statement]\!] = 1$ if *statement* is true and it is $[\![statement]\!] = 0$ otherwise. The fitness function $\mathcal{F}'(x^i)$ counts the number of solutions in the population that is dominated by the solution x^i. That is, it counts the number of solutions that their NO_x and BSFC are larger and BMEP is smaller than those of the solution x^i.

In the proposed algorithm, to reduce uncertainty in measuring the performance of the engine for a solution, a number of solutions are found in a radius ρ around the current solution, and a weighted average over the output of the surrogate model for these solutions is calculated [73]. In this scheme, an adaptive method is employed that estimates the uncertainty in the vicinity of the current solution, and based on the level of uncertainty, sets the size of the averaging pool. To measure the uncertainty, some solutions around the current solution are generated and the output of the surrogate model for them is found. Then the standard deviation of these values is found and used as a measure of the level of uncertainty. The solutions with higher uncertainty receive a larger averaging pool.

The proposed uncertainty reduction algorithm 4 gets the population of solutions X and the set $\mathcal{H}$ which contains all the solutions that have been evaluated in the search process. In step 5 of the algorithm, the performance

Algorithm 4: The proposed uncertainty reduction algorithm.

$\tau = 0$;
input: the population X and the set of solutions $\mathcal{H}$;
set the parameters ρ and λ;
for $i = 1 \rightarrow n$ **do**
    estimate for x_i the surrogate model's output $L(x_i)$;
    $\mathcal{N}_i = \emptyset$;
    for *all solutions* $y \in \mathcal{H}$ **do**
        if $\delta(x_i, y) < \rho$ **then**
            $\mathcal{N}_i = \mathcal{N}_i \cup \{y\}$
        end
    end
    while $|\mathcal{N}| < \frac{T}{2}$ **do**
        generate a solution y randomly in a sphere of radius ρ around x_i;
        $\mathcal{N}_i = \mathcal{N}_i \cup \{y\}$;
        $\mathcal{H} = \mathcal{H} \cup \{y\}$;
    end
end
for $i = 1 \rightarrow n$ **do**
    $s_i = \sigma(\{L(z_i)|z \in \mathcal{N}_i\})$
end
for $i = 1 \rightarrow n$ **do**
    $t_i = \frac{R(s_i)}{\sum_{j=1}^{n} R(s_j)} \times \frac{T}{2}$
end
for $i = 1 \rightarrow n$ **do**
    for $j = 1 \rightarrow t_i$ **do**
        generate a solution y randomly in a sphere of radius ρ around x_i;
        $\mathcal{N}_i = \mathcal{N}_i \cup \{y\}$;
        $\mathcal{H} = \mathcal{H} \cup \{y\}$;
    end
end
for $i = 1 \rightarrow n$ **do**
    $\hat{L}(x_i) = \frac{\sum_{\forall y \in \mathcal{N}_i} L(y) e^{-\lambda \delta(x_i, y)}}{2 \sum_{\forall y \in \mathcal{N}_i} e^{-\lambda \delta(x_i, y)}} + \frac{L(x_i)}{2}$
end

of the engine for the solutions in the population is estimated via the surrogate model.

In the proposed algorithm, to reduce uncertainty, on average T solutions are evaluated around the current solution, where a larger number of evaluations are given to solutions with higher uncertainty. In step 6, the set $\mathcal{N}_\rangle$ is created in which the neighbour solutions in the radius ρ around the current solution x_i are stored. The averaging process to reduce uncertainty is then performed over the solutions in this set. The algorithm creates the set $\mathcal{H}$ to save computations. By storing all the solutions that have been evaluated, the algorithm can use the previous evaluations in the uncertainty reduction process. In step 7 of the algorithm, the solutions in the set $\mathcal{H}$ that are in the radius ρ around the current solution are found and added to the set of neighbour solutions $\mathcal{N}_i$. This way, the algorithm uses the previously evaluated solutions for the uncertainty reduction process. At first, in step 12, $T/2$ function evaluations are given to each solution, and then the algorithm measures the level of uncertainty around each solution to assign more evaluations to the solutions with higher uncertainty. In this step, if the number of solutions in $\mathcal{N}_i$ is less than $T/2$, some random solutions in the radius ρ are generated and added to the set. These are also added to $\mathcal{H}$ for future computations. In step 19, the uncertainty around the current solution is measured by calculating the standard deviation of the output of the surrogate model of the solutions in $\mathcal{N}_i$.

The proposed algorithm assigns more function evaluations to the solutions with higher uncertainty (standard deviation, s_i). In step 22, the solutions in the population are ranked based on their standard deviation, and more function evaluations are assigned to the solutions with larger standard deviation. The operations in these steps distribute the remaining $T/2$ evaluation budgets to the solutions. The process is performed as,

$$t_i = \frac{R(s_i)}{\sum_{j=1}^{n} R(s_j)} \times \frac{T}{2}, \tag{19}$$

where t_i is the number of evaluations assigned to the solution x_i and $R(s_i)$ is the rank of the solution x_i based on its standard deviation,

$$R(s_i) = |\{j | s_j < s_i\}|. \tag{20}$$

In brief, $R(s_i)$ is the number of solutions in the population that have a smaller standard deviation in their neighbourhood than x_i.

In step 24, the neighbourhood sets $\mathcal{N}_i$ is filled with randomly generated solutions. In step 31 the proposed uncertainty reduction is calculated. Here, the uncertainty reduced value, $\hat{L}(x_i)$ is measured as

$$\hat{L}(x_i) = \frac{\sum_{\forall y \in \mathcal{N}_i} L(y) e^{-\lambda \delta(x_i, y)}}{2 \sum_{\forall y \in \mathcal{N}_i} e^{-\lambda \delta(x_i, y)}} + \frac{L(x_i)}{2} \tag{21}$$

where $\delta(.,.)$ measures the Euclidean distance between two vectors. The formula calculates the weighted average of the output of the surrogate model among the current solution and the solutions in $\mathcal{N}$. The weights in this process decay exponentially with the distance between a solution and the current solution. In this process, λ controls the decay rate. A study on the effect of the values of ρ and λ on the performance of the algorithm is presented later in this paper and the best value for these parameters can be found.

5. Experimental Results

We first start our experiments by studying the parameters of the base learner algorithms. The parameters of the learning algorithms are as follows, RBN, has two parameters, *Spread* of radial basis functions and *DF*, neurons between displays (Matlab default is S=1 and DF=25), LVQ has two parameters, learning rate LR and HS hidden layer size (Matlab default is HS=10 and LR=0.01), PNN has one parameter *Spread* (Matlab default is S=1), RBE has one parameter *Spread* (Matlab default is S=1), CFNN has one parameter, HS hidden layer size (Matlab default is HS=10), PRN has one parameter, HS hidden layer size (Matlab default is HS=10), FFNN has one parameter, HS hidden layer size (Matlab default is HS=10), GRNN has one parameter *Spread* (Matlab default is S=1), KNN has one parameter k (Matlab default is k=1), FFN has one parameter, HS hidden layer size (Matlab default is HS=10).

To find the best performance of the learning algorithms, a study is performed on the parameters of the learning algorithms in this paper. A grid approach is used by testing the algorithms for the following set of parameters: $S = \{0.1, 0.2, \ldots 1.4, 1.5\}$, $HS = \{3, 4, 5, \ldots, 19, 20\}$, and $k = 1, 3, \ldots, 9, 11$. The best parameters for each learning algorithm are presented in table 3. Interestingly, the optimal number of hidden layers for all the learning algorithms is smaller than Matlab default values. This could be because of the small training data size. It seems that a few numbers of the learning parameters are enough to capture the behaviour of the system.

	First Parameter	Second Parameter
RBN	$S = 0.7$	$DF = 19$
LVQ	$HS = 8$	0.02
PNN	$S = 0.6$	-
RBE	$S = 0.5$	-
CFNN	$HS = 7$	-
PRN	$HS = 6$	-
FFNN	$HS = 8$	-
GRNN	$S = 0.5$	-
KNN	$k = 5$	-
FNN	$HS = 7$	-

Table 3: The best parameters for each of the learning algorithms. This is found by averaging over 30 independent runs.

This section presents the experimental studies on the proposed algorithm. As shown in Figure 1, the proposed algorithm consists of three main components, which are the base learners, the diversifier, and the pruning algorithm. In this section, we conduct an ablation study to examine the impact of individual components of the algorithm on its performance. The ablation study involves systematically removing each component of the algorithm one by one and analyzing the effect on performance.

The proposed diversifier algorithm gets a learning algorithm L_k and diversifies them by generating n learning algorithms that are different in terms of the subset of features on which they have been trained. Table 4 presents the mean square error of experiments on the learning algorithms when a single learning algorithm is used versus when the learning algorithm is diversified and an ensemble of 10 learning algorithms is generated. The outputs of these base learners are aggregated via an averaging scheme. This shows the effect of the proposed diversification algorithm on the performance of the learning algorithms. The data suggest that the proposed diversifier creates an ensemble of learning algorithms that outperforms the single algorithms.

In this table, the column "single" refers to the experiments performed on the learning algorithm without applying the diversification algorithm, and the column "diversified" represents the results when the algorithm 1 is employed to create 10 base-learners. The best results in each column are bolded and presented in the row labelled as "Best".

In Table 4, the last row which is denoted by "Ens.' represents the performance of the ensemble learning algorithm that is generated by combining the

	NO_x		BMEP		BSFC	
	Single	Diversified	Single	Diversified	Single	Diversified
RBN	3.80e+06	3.01e+06	1.38e+00	1.01e+00	1.71e+05	1.17e+05
LVQ	2.96e+06	2.30e+06	8.66e+00	6.42e+00	1.49e+05	1.18e+05
PNN	2.96e+06	1.83e+06	8.67e+00	5.61e+00	1.49e+05	1.16e+05
RBE	4.98e+06	3.05e+06	4.54e+01	2.97e+01	2.04e+05	1.24e+05
CFNN	2.79e+05	2.02e+05	4.50e-01	3.29e-01	7.55e+04	5.21e+04
PRN	4.35e+06	3.33e+06	4.63e+00	3.17e+00	7.37e+05	5.36e+05
FFNN	**2.44e+05**	**1.92e+05**	**4.24e-01**	3.35e-01	8.35e+04	6.38e+04
GRNN	4.30e+05	2.97e+05	5.27e-01	**3.22e-01**	5.27e+04	**3.34e+04**
KNN	4.10e+05	2.75e+05	5.41e-01	4.04e-01	**5.27e+04**	3.82e+04
FNN	4.89e+05	3.29e+05	7.53e-01	5.19e-01	1.18e+05	8.87e+04
Best	2.44e+05	1.92e+05	4.24e-01	3.22e-01	5.27e+04	3.34e+04
Ens.	1.45e+05	8.31e+04	2.02e-01	1.04e-01	2.93e+04	1.51e+04

Table 4: The mean square error of experiments on the learning algorithms, when a single learning algorithm is used versus when the diversification algorithm (algorithm 1) is employed to build an ensemble of 10 learning algorithms. The results are averaged over 30 runs.

learning algorithm at the corresponding column. That is, the value for the row "Ens." and column "Single" presents the performance of an ensemble learning algorithm that is formed of the single learning algorithms $L_1 \dots L_m$ without applying the diversification algorithm. The output of this ensemble is found as the weighted sum of the output of these base learners,

$$L(y_u, z, o) = \sum_{k=1}^{m} w_k L_k(y_u, o), \tag{22}$$

where $L_k(y_u, o)$ is the o-th output of the base-learner L_k for y_u.

The value for the row "Ens." and the column "Diversified", presents the performance when an ensemble of the diversified learning algorithms is used. That is, first the algorithm 1 is used to diversify the learning algorithms $L_1 \dots L_m$. The output of the ensemble here is found as the weighted sum of these base-learners as

$$L(y_u, z, o) = \sum_{k=1}^{m} \sum_{l=1}^{n} w_{kl} L_k^l(y_u, o). \tag{23}$$

The weights of this voting scheme are optimized via a gradient descent algorithm.

For NO_x and BMEP, the best performance among the learning algorithms is achieved by FFNN, while for BFFC, the best performance is achieved via GRNN. The fact that the FFNN does not offer a good performance for BSFC, suggests that some algorithms are better at predicting for some metrics while they perform worse for other metrics. This is one reason why only one learning algorithm is not enough to reach an optimal performance and there is a need for an ensemble so the algorithms cover one another's weaknesses. Some algorithms like CFNN, produce the worst performances among all the algorithms in all the metrics. However, this does not mean that they should be removed from the ensemble. Because an ensemble can benefit from a relatively inferior learning algorithm. This is because there are data records for which the relatively inferior learning algorithm performs better than others so can improve the overall accuracy.

The data in Table 4 suggest that the proposed diversifier algorithm can generate an ensemble algorithm that outperforms the base learners with a good margin. The data also suggest that an ensemble of diversified classifiers performs better than the ensemble of a single learning algorithm.

The proposed diversified algorithm can improve the performance of the ensemble learning algorithm because it creates base learners that are different in performance. An ensemble learning algorithm that consists of similar base learners does not significantly improve the performance as compared to the individual base learners because it aggregates the same decision over some decision-makers. An ensemble learning consisting of diverse base learners outperforms the individual base learners because they can cover each other for their mistakes. When one algorithm makes a mistake, the other algorithms can fix the decision, only if they are capable of making different decisions.

To present more statistical study on the data in this table, table 9 summarizes more detailed results on the data in Table 4. Table 9 includes the standard deviation and the p-value of the Wilcoxon rank-sum test between the data. We use the non-parametric Wilcoxon rank-sum test because we could not assume that the data follow a normal distribution.

In this paper, we propose QEA as the diversifier algorithm because QEA is specifically suitable for binary coded problems and can manage diversity among solutions [69]. To compare the proposed QEA diversifier algorithm to existing evolutionary algorithms, Table 9 performs a comparison between different optimization algorithms when they are used as the diversifier. The proposed algorithm is compared to some existing evolutionary algorithms which include Genetic Algorithms, Rotating Crossover Operator Differential Evo-

lution (RCODE) [74], Level-Based Learning Swarm Optimizer (LLSO) [75], Ma-ssw-chains (MASSW) [76], Differential Evolution (DE), PSO, Fast Evolutionary Strategy (FES) [77], and Real-coded Compact Genetic Algorithms (RCGA) [78]. In all the experiments the population size is set to 10 and the termination condition is set to 100 iterations. These algorithms each have some parameters that affect their performance. Tuning these parameters for the problems in this paper is very computationally expensive as they involve training and testing machine learning algorithms. Therefore, in this paper, we use the best parameters for these algorithms as studied and suggested in [79]. The ANOVA test is also applied to the data and the results are presented in the last column of the table. The data in table 9 suggest that the best performance is achieved by QEA. Hereafter in this paper, diversification in all the experiments is performed using the QEA.

The next stage in the proposed algorithm is the pruning algorithm which is designed to prune the set of $m \times n$ base-learners generated in the diversification stage and find the optimal subset of classifiers that minimize the computational cost without highly affecting the performance of the resulting ensemble. We employ QEA to prune the ensemble learning algorithm. An ablation scheme is presented here to study the performance of the ensemble before and after pruning. Table 5 shows the experimental results for the pruning process when different optimization algorithms are used to prune the ensemble learning algorithm. The results of the ensemble learning algorithm before pruning are presented in the last column of this table, which means that all the $m \times n$ base learners are used in the ensemble. Interestingly, after the pruning is applied, although the number of base learners decreases, the performance of the algorithm improves. This can be attributed to the fact that the algorithm prunes the redundant base learners. This process not only reduces the computational cost of the algorithm but also improves its performance.

The proposed pruning algorithm improves the ensemble's performance because a large set of diverse base learners is not always an optimal set. The way the base learners complement one another in terms of the decisions they make is a complicated behaviour as diversity and the difference between the decisions the algorithms make are not the only contributing factor to the final performance. Finding the optimal subset of base learners is a combinatorial optimization problem that requires its own tools to solve. This explains why the proposed pruning algorithm improves the performance of the ensemble further.

	NO_x	BMEP	BSFC
QEA (1)	5.189e+04	6.848e-02	9.598e+03
FEP (2)	6.649e+04	8.053e-02	1.208e+04
GA (6)	7.327e+04	9.642e-02	1.370e+04
PSO (7)	7.747e+04	1.004e-01	1.479e+04
DE (3)	6.743e+04	8.350e-02	1.345e+04
FES (5)	7.248e+04	9.103e-02	1.289e+04
MASW (9)	8.564e+04	9.936e-02	1.550e+04
LLSO (8)	8.062e+04	1.068e-01	1.462e+04
RCGA (4)	7.061e+04	8.401e-02	1.310e+04
no pruning	8.311e+04	1.040e-01	1.513e+04

Table 5: The performance of the ensemble learning algorithm with and without pruning. The data are averaged over 30 runs. The numbers in the brackets after the name of each algorithm show the Friedman rank of the algorithm.

The proposed ensemble learning algorithm is compared with some existing ensemble learning algorithms including Stacking LR (STLR) [80], Random Forest (RF) [80], AdaBoost J48 (ABJ48) [80], Oblique Random Forest (oRF) [81], (Multisurface Proximal Random Forest) MPRoF-P [82], Majority Voting (MV) [80], RoF [82] and Classification by Cluster Analysis (CBCA) [80]. Table 6 shows the experiments on these algorithms where the data are averaged over 30 runs. All experiments in this paper adopt the leave-one-out scheme in which in each experiment, all the data records except one are used to train the models, and the remaining one is used to test the model. This process is performed over all the data records and the results are averaged over all the records. This results in a more rigorous approach that makes sure that the training phase of the models is not contaminated with the test data. As the data in this table suggest, the proposed algorithm performs better than the existing algorithms.

Figure 2 shows the Pareto front of the solutions found via the proposed uncertainty reduction algorithm. As the graph suggests, the proposed algorithm finds a wide range of solutions that are optimized with respect to the given objectives. Note that NO_x and BSFC should be minimized and BMEP should be maximized. The solutions in the Pareto front are the non-dominated solutions, meaning these are the solutions that are optimized with respect to at least one objective, in a way that no other solution has reached a better value for the objective(s). For example, the solution at the left of the graph has low BMEP (BMEP should be maximized) and high BSFC (BSFC should be minimized), but it appears in the Pareto front because it

	NO_x	BMEP	BSFC
MPRoF-P (8)	8.099e+04	1.143e-01	1.530e+04
STLR (6)	8.189e+04	1.120e-01	1.412e+04
CBCA (3)	6.812e+04	9.373e-02	1.239e+04
MV (5)	7.773e+04	9.757e-02	1.522e+04
ABJ48 (4)	7.600e+04	1.005e-01	1.502e+04
RF (7)	8.452e+04	1.065e-01	1.465e+04
oRF (2)	6.622e+04	9.109e-02	1.281e+04
RoF (9)	8.529e+04	1.045e-01	1.603e+04
EVEL (1)	5.192e+04	6.851e-02	9.606e+03

Table 6: A comparison between the proposed ensemble learning algorithm and existing ensemble learning algorithms. The numbers in the brackets show the Friedman rank of the algorithms.

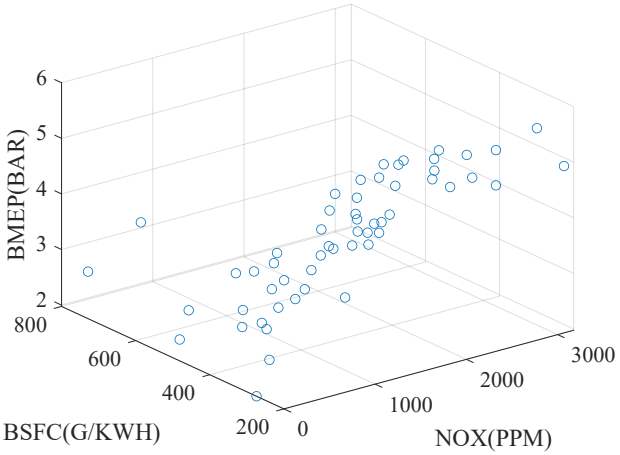

Figure 2: The Pareto front of the results found via the proposed uncertainty reduction algorithm.

minimizes NO_x to a degree that no other solution has reached. Similarly, the solutions at the top right of the graph have high NO_x (NO_x should be minimized) but they appear in the Pareto front because they achieve a BSFC and BMEP to a degree that no other solution has reached. Table VI shows the values for the solutions in the Pareto front.

Table 7 compares the performance of different optimization algorithms in solving the problem in this paper in terms of the hypervolume of the Pareto front that is found by each of the algorithms Hypervolume is a measure that is usually used to compare different Pareto fronts. In this paper we compare our algorithm with a set of existing algorithms which include GA [83] (mu-

Optimiser	Mean MSE	STD	Rank
GA	8.88e+06	5.45e+04	5.3
GA+Smoothing	9.49e+06	9.10e+04	1
PSO	7.57e+06	4.53e+04	24
PSO+Smoothing	7.96e+06	1.63e+04	20.5
DE	7.80e+06	5.49e+04	22.9
DE+Smoothing	8.34e+06	3.47e+04	15.1
ES	7.92e+06	3.25e+04	21.3
ES+Smoothing	8.46e+06	3.85e+04	12.6
FES	7.91e+06	8.59e+04	21.4
FES+Smoothing	8.29e+06	1.81e+04	16.2
EP	8.40e+06	4.03e+04	14
EP+Smoothing	8.85e+06	9.44e+04	5.8
FEP	8.04e+06	1.88e+03	19
FEP+Smoothing	8.52e+06	2.81e+04	10.9
MASW	8.15e+06	4.65e+04	18.1
MASW+Smoothing	8.52e+06	2.70e+04	10.8
EDA	8.63e+06	9.49e+04	8.6
EDA+Smoothing	9.21e+06	3.70e+04	2.1
RCODE	8.52e+06	8.22e+04	10.9
RCODE+Smoothing	8.99e+06	9.73e+04	4.1
LLSO	8.27e+06	4.31e+04	16.7
LLSO+Smoothing	8.76e+06	3.10e+04	6.9
RCGA	8.57e+06	3.73e+04	9.3
RCGA+Smoothing	9.10e+06	5.63e+04	3.2

Table 7: A comparison between the hypervolume of the Pareto front found via different optimization algorithms. The data are averaged over 30 runs. The last column shows the Friedman rank of the algorithm.

tation rate=0.05, crossover rate 0.7), Evolutionary Programming (EP, parameters set st [84]), Fast Evolutionary Programming (FEP, parameters set as [85]), Evolutionary Strategy (ES, parameters set as [86]), Fast Evolutionary Strategy (FES, L=1, S=1.1) [77], Ma-ssw-chains (MASSW, parameters set as [76]) and Differential Evolution (DE, F=0.2, O=0.8). The data in this table suggest that the proposed uncertainty reduction algorithm improves the performance of evolutionary algorithms.

Figure 3 shows the box plot of the data in table 7. In this figure, the algorithms are labelled as follows: 1-GA, 2-GA+Smoothing, 3-PSO, 4-PSO+Smoothing, 5-DE, 6-DE+Smoothing, 7-ES, 8-ES+Smoothing, 9-FES, 10-FES+Smoothing, 11-EP, 12-EP+Smoothing, 13-FEP, 14-FEP+Smoothing, 15-MASW, 16-MASW+Smoothing, 17-EDA, 18-EDA+Smoothing, 19-RCODE, 20-RCODE+Smoothing, 21-LLSO, 22-LLSO+Smoothing, 23-RCGA, and 24-RCGA+Smoothing.

Figure 4 shows the performance of GA+Smoothing (when algorithm 4 is used to smooth the landscape and reduce uncertainty) for different values of

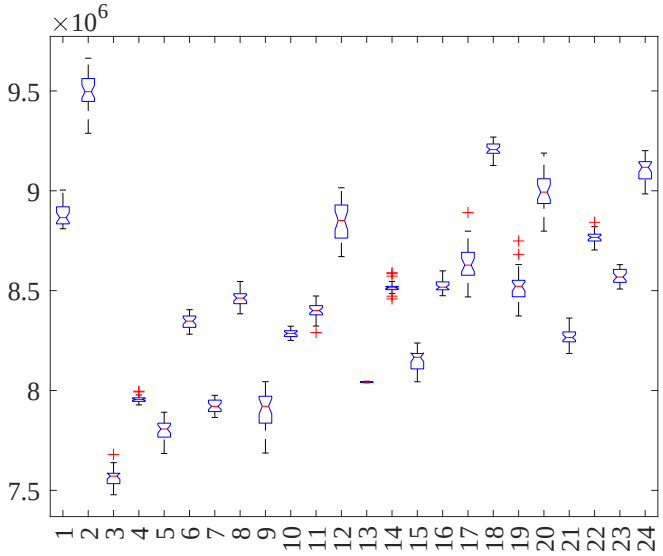

Figure 3: The box-plot of the ANOVA test on the data in table 7. The vertical axis is the performance of the algorithms and the horizontal axis is the algorithms. In this figure, the algorithms are labelled as follows: 1-GA, 2-GA+Smoothing, 3-PSO, 4-PSO+Smoothing, 5-DE, 6-DE+Smoothing, 7-ES, 8-ES+Smoothing, 9-FES, 10-FES+Smoothing, 11-EP, 12-EP+Smoothing, 13-FEP, 14-FEP+Smoothing, 15-MASW, 16-MASW+Smoothing, 17-EDA, 18-EDA+Smoothing, 19-RCODE, 20-RCODE+Smoothing, 21-LLSO, 22-LLSO+Smoothing, 23-RCGA, and 24-RCGA+Smoothing. The data are for 30 runs.

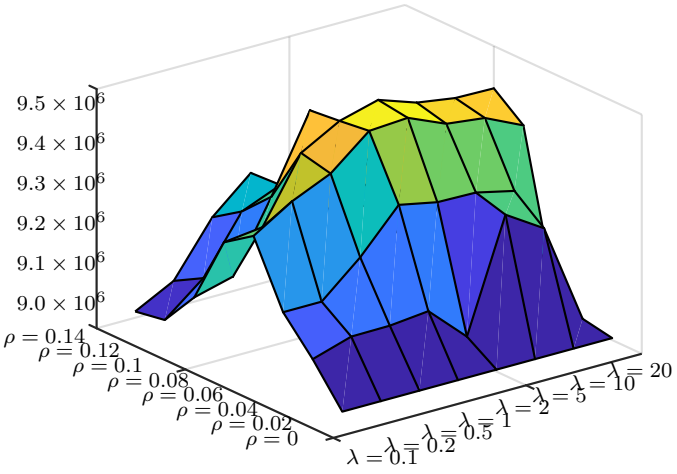

Figure 4: The performance of the proposed uncertainty reduction algorithm with respect to the parameters of the algorithm.

the parameters ρ and λ. The data in this graph are averaged over 30 runs. As the data suggest, the best performance of the algorithm is achieved when $\rho = 0.08$ and $\lambda = 2$. The same experiment is performed on all the algorithms studied in this paper and the results are presented in table 8.

Table 9 presents more detailed experiments on the data in table 4. The standard deviation and the Wilcoxon rank-sum test between the data. The non-parametric Wilcoxon rank-sum test is used in this paper because the assumption that the data follow normal distribution could not be made.

Table 10 presents the experimental results on different evolutionary algorithms. In these experiments, the evolutionary algorithms are used to diversify the learning algorithms and generate 10 base learners. The output of these base learners is then aggregated to form an ensemble learning algorithm. The ensemble learning algorithm here uses a gradient descent algorithm to find the optimal value for the weighted voting scheme. The Friedman rank test is performed on these data and the result is presented in the last row of this table. As these data suggest, the proposed algorithm outperforms the existing algorithms. The last column in this table shows the p-value of the ANOVA on the data.

The type of uncertainty that we observe in fitness evaluation, can mislead the search process for evolutionary algorithms significantly. To overcome this,

Optimiser	ρ	λ
GA	0.08	2
PSO	0.06	2
DE	0.08	5
ES	0.08	2
FES	0.1	1
EP	0.06	2
FEP	0.06	2
MASW	0.08	5
EDA	0.08	2
RCODE	0.08	2
LLSO	0.1	1
RCGA	0.08	2

Table 8: The best parameters for the smoothing algorithm applied on different optimization algorithms.

a way should be found to make the fitness evaluations more reliable. The proposed smoothing algorithm can reduce the uncertainty which provides a better evaluation of the fitness function for the optimization algorithm. Such an evaluation can provide a better vision and understanding of the evolutionary algorithm about the fitness landscape which results in a better performance.

To test the significance of the data presented in table 7, the ANOVA test is performed on the data and the results are presented in table 14. In this table, 'SS' is the sum of squares of each source, 'df' is the degree of freedom associated with each source, 'MS' is the mean squared, i.e. the ratio SS/df, and 'Chi-square' is the ratio of mean squares. The small p-value in this table suggests that the null hypothesis that samples are taken from the same mean is rejected with an overwhelming probability. The box plot of the ANOVA test is presented in Figure 3.

Finding the best set of parameters for a mixture-fuel engine is crucial as it can have a huge impact. First, it can minimize the emission of harmful gases, which is very important with the current effects we have on the environment. Second, it can improve the performance of engines in terms of fuel consumption which in turn reduces costs to companies. Performing such optimization requires accurate models of the engines with less uncertainty. To create such models, new ML algorithms should be developed that are capable of modelling the problems with the limited number of data available. The algorithm presented in this paper targets these challenges and as the experimental results suggest, to some degree, achieves the goal.

	NO_x				
	Single		Diversified		
	Mean	STD	Mean	STD	p-value
RBN	3.796e+06	6.263e+06	3.012e+06	5.648e+05	3.021e-01
LVQ	2.964e+06	3.265e+06	2.302e+06	3.567e+05	2.863e-03
PNN	2.962e+06	4.908e+06	1.834e+06	1.958e+05	1.971e-02
RBE	4.983e+06	5.982e+06	3.045e+06	4.313e+05	7.959e-01
CFNN	2.792e+05	3.884e+05	2.024e+05	2.723e+04	1.335e-01
PRN	4.347e+06	5.774e+06	3.328e+06	5.624e+05	2.590e-01
FFNN	2.444e+05	1.894e+05	1.917e+05	3.105e+04	2.220e-01
GRNN	4.302e+05	5.599e+05	2.973e+05	3.453e+04	8.141e-02
KNN	4.095e+05	5.817e+05	2.746e+05	5.409e+04	3.450e-01
FNN	4.887e+05	7.449e+05	3.291e+05	5.292e+04	4.148e-01
	BMEP				
	Single		Diversified		
	Mean	STD	Mean	STD	p-value
RBN	1.376e+00	2.146e+00	1.007e+00	1.434e-01	8.596e-04
LVQ	8.661e+00	7.725e+00	6.420e+00	1.032e+00	3.848e-02
PNN	8.668e+00	1.133e+01	5.610e+00	6.610e-01	8.892e-02
RBE	4.540e+01	5.363e+01	2.972e+01	5.485e+00	7.663e-01
CFNN	4.501e-01	3.575e-01	3.291e-01	5.538e-02	6.926e-02
PRN	4.631e+00	4.289e+00	3.172e+00	4.846e-01	1.101e-01
FFNN	4.241e-01	3.878e-01	3.351e-01	6.637e-02	2.401e-01
GRNN	5.274e-01	8.929e-01	3.221e-01	5.545e-02	1.123e-02
KNN	5.406e-01	8.113e-01	4.041e-01	6.527e-02	4.558e-01
FNN	7.530e-01	1.110e+00	5.190e-01	8.786e-02	1.726e-01
	BSFC				
	Single		Diversified		
	Mean	STD	Mean	STD	p-value
RBN	1.714e+05	1.998e+05	1.175e+05	2.302e+04	3.258e-01
LVQ	1.488e+05	2.180e+05	1.182e+05	1.627e+04	3.281e-01
PNN	1.486e+05	2.152e+05	1.161e+05	1.582e+04	2.839e-03
RBE	2.040e+05	2.894e+05	1.238e+05	2.118e+04	2.165e-02
CFNN	7.548e+04	5.910e+04	5.210e+04	5.905e+03	6.425e-03
PRN	7.374e+05	9.516e+05	5.358e+05	5.663e+04	2.734e-03
FFNN	8.348e+04	6.710e+04	6.380e+04	9.327e+03	1.760e-01
GRNN	5.269e+04	4.678e+04	3.342e+04	5.016e+03	1.340e-01
KNN	5.267e+04	7.869e+04	3.819e+04	7.517e+03	7.305e-02
FNN	1.176e+05	1.997e+05	8.867e+04	1.558e+04	8.499e-01

Table 9: The mean square error of experiments on the learning algorithms, when a single learning algorithm is used versus when the diversification algorithm (algorithm 1) is employed to build an ensemble of 10 learning algorithms. The results are averaged over 30 runs.

		GA	FES	PSO	DE	MASW	LLSO	RCGA	ES	EP	p-value
NOX	RBN	3.012e+06	2.989e+06	2.599e+06	3.041e+06	3.282e+06	2.994e+06	2.898e+06	3.120e+06	3.451e+06	1.8e-08
	LVQ	2.302e+06	2.600e+06	2.061e+06	2.255e+06	2.559e+06	2.386e+06	2.466e+06	2.418e+06	2.723e+06	3.0e-12
	PNN	1.834e+06	2.229e+06	1.965e+06	2.382e+06	2.703e+06	2.739e+06	2.549e+06	2.452e+06	2.367e+06	7.6e-23
	RBE	3.045e+06	4.021e+06	3.735e+06	4.530e+06	4.149e+06	3.760e+06	4.191e+06	4.650e+06	3.802e+06	3.7e-20
	CFNN	2.024e+05	2.395e+05	2.277e+05	2.336e+05	2.605e+05	2.427e+05	2.170e+05	2.188e+05	2.206e+05	1.1e-12
	PRN	3.328e+06	3.652e+06	3.241e+06	4.136e+06	3.464e+06	3.606e+06	3.934e+06	3.655e+06	3.838e+06	5.6e-06
	FFNN	1.917e+05	2.021e+05	1.717e+05	1.857e+05	2.276e+05	2.248e+05	2.066e+05	1.823e+05	1.997e+05	1.3e-13
	GRNN	2.973e+05	3.809e+05	3.638e+05	3.927e+05	3.175e+05	4.017e+05	3.308e+05	3.325e+05	3.366e+05	4.0e-14
	KNN	2.746e+05	3.175e+05	3.348e+05	3.568e+05	3.172e+05	3.752e+05	3.151e+05	3.717e+05	3.562e+05	2.4e-20
	FNN	3.291e+05	4.414e+05	3.778e+05	4.339e+05	4.322e+05	4.055e+05	4.182e+05	4.301e+05	4.161e+05	7.1e-10
BMEP	RBN	1.007e+00	1.048e+00	1.044e+00	1.264e+00	1.240e+00	1.214e+00	1.189e+00	1.090e+00	1.177e+00	1.6e-15
	LVQ	6.420e+00	7.754e+00	6.979e+00	8.252e+00	7.771e+00	7.508e+00	6.444e+00	6.862e+00	8.109e+00	1.6e-08
	PNN	5.610e+00	6.989e+00	6.993e+00	6.725e+00	7.857e+00	6.739e+00	7.496e+00	8.109e+00	8.129e+00	8.9e-19
	RBE	2.972e+01	3.822e+01	3.813e+01	4.291e+01	4.102e+01	3.808e+01	3.463e+01	3.828e+01	4.159e+01	1.3e-17
	CFNN	3.291e-01	3.950e-01	3.781e-01	3.423e-01	3.991e-01	3.733e-01	4.202e-01	3.959e-01	4.002e-01	7.7e-12
	PRN	3.172e+00	4.315e+00	3.426e+00	4.072e+00	3.765e+00	4.146e+00	4.262e+00	4.290e+00	4.234e+00	2.3e-19
	FFNN	3.351e-01	3.664e-01	2.969e-01	3.635e-01	3.789e-01	3.197e-01	3.537e-01	3.912e-01	3.593e-01	2.1e-09
	GRNN	3.221e-01	4.604e-01	3.827e-01	4.400e-01	4.832e-01	4.746e-01	4.732e-01	4.941e-01	4.306e-01	1.5e-36
	KNN	4.041e-01	4.995e-01	4.542e-01	4.600e-01	4.410e-01	4.059e-01	4.641e-01	4.475e-01	4.573e-01	9.8e-11
	FNN	5.190e-01	5.582e-01	6.111e-01	6.700e-01	5.584e-01	5.906e-01	5.993e-01	6.464e-01	5.786e-01	9.1e-12
BSFC	RBN	1.175e+05	1.476e+05	1.313e+05	1.560e+05	1.474e+05	1.296e+05	1.439e+05	1.439e+05	1.445e+05	4.5e-10
	LVQ	1.182e+05	1.396e+05	9.893e+04	1.201e+05	1.117e+05	1.247e+05	1.134e+05	1.161e+05	1.407e+05	5.5e-17
	PNN	1.161e+05	1.140e+05	1.106e+05	1.332e+05	1.244e+05	1.299e+05	1.253e+05	1.361e+05	1.136e+05	2.4e-12
	RBE	1.238e+05	1.900e+05	1.458e+05	1.657e+05	1.539e+05	1.759e+05	1.902e+05	1.719e+05	1.815e+05	2.9e-21
	CFNN	5.210e+04	6.854e+04	5.365e+04	7.199e+04	6.544e+04	5.824e+04	7.001e+04	5.879e+04	5.985e+04	1.3e-30
	PRN	5.358e+05	5.916e+05	5.745e+05	6.927e+05	6.767e+05	5.753e+05	6.542e+05	6.653e+05	6.774e+05	3.1e-17
	FFNN	6.380e+04	7.666e+04	6.575e+04	7.223e+04	7.053e+04	7.335e+04	6.953e+04	6.686e+04	7.635e+04	4.0e-08
	GRNN	3.342e+04	4.545e+04	4.153e+04	4.980e+04	4.338e+04	4.577e+04	4.152e+04	4.334e+04	4.841e+04	6.9e-24
	KNN	3.819e+04	4.705e+04	3.949e+04	4.612e+04	4.291e+04	4.131e+04	4.651e+04	4.299e+04	4.061e+04	1.1e-03
	FNN	8.867e+04	1.049e+05	8.913e+04	1.004e+05	9.350e+04	1.009e+05	1.085e+05	9.705e+04	9.589e+04	3.1e-07
Rank		1	3	6	7	5	8	9	2	4	-

Table 10: A comparison between different evolutionary algorithms used to diversify the learning algorithms. The data are averaged over 30 runs.

	Training		Testing	
Algorithm	Mean	STD	Mean	STD
RBN	6.89	3.40e-02	0.55	2.63e-03
LVQ	8.46	1.90e-01	0.68	1.47e-02
PNN	0.14	1.39e-01	0.01	1.07e-02
RBE	7.16	1.35e-01	0.57	1.05e-02
CFNN	3.96	2.19e-02	0.32	1.69e-03
PRN	4.63	1.00e-02	0.37	7.76e-04
FFNN	2.60	8.79e-03	0.21	6.80e-04
GRNN	0.12	1.47e-01	0.01	1.14e-02
KNN	1.70	8.33e-01	0.14	6.44e-02
FNN	5.29	1.75e-02	0.42	1.35e-03
MPRoF-P	51.46	1.69e+00	2.95	9.91e-02
STLR	56.79	1.68e+00	1.75	6.26e-02
CBCA	56.85	1.88e+00	2.68	8.17e-02
MV	59.05	2.00e+00	2.72	1.05e-01
ABJ48	57.09	1.69e+00	2.07	6.57e-02
RF	54.32	1.61e+00	1.77	7.23e-02
oRF	42.23	1.98e+00	3.14	6.80e-02
RoF	85.22	1.79e+00	2.95	8.10e-02
EVEL	140.04	4.65e+00	3.08	6.63e-02

Table 11: The training and testing time required for each of the algorithms. The data are in seconds.

Algorithm	GA	FES	PSO	DE	MASW	LLSO	RCGA	ES	EP
RBN	01:55:40.1	01:55:42.7	01:55:13.6	01:55:29.7	01:55:28.5	01:55:18.0	01:55:35.9	01:55:46.4	01:55:58.8
LVQ	02:21:59.8	02:22:03.0	02:21:27.2	02:21:47.1	02:21:45.5	02:21:32.7	02:21:54.6	02:22:07.5	02:22:22.7
PNN	00:02:16.1	00:02:16.1	00:02:15.5	00:02:15.9	00:02:15.8	00:02:15.6	00:02:16.0	00:02:16.2	00:02:16.4
RBE	02:00:08.6	02:00:11.3	01:59:41.0	01:59:57.8	01:59:56.5	01:59:45.6	02:00:04.2	02:00:15.1	02:00:28.0
CFNN	01:06:28.3	01:06:29.8	01:06:13.0	01:06:22.3	01:06:21.6	01:06:15.5	01:06:25.8	01:06:31.9	01:06:39.0
PRN	01:17:40.4	01:17:42.1	01:17:22.5	01:17:33.4	01:17:32.5	01:17:25.5	01:17:37.5	01:17:44.6	01:17:52.9
FFNN	00:43:41.4	00:43:42.4	00:43:31.3	00:43:37.4	00:43:37.0	00:43:33.0	00:43:39.8	00:43:43.8	00:43:48.4
GRNN	00:01:56.1	00:01:56.2	00:01:55.7	00:01:55.9	00:01:55.9	00:01:55.7	00:01:56.0	00:01:56.2	00:01:56.4
KNN	00:28:29.7	00:28:30.4	00:28:23.2	00:28:27.2	00:28:26.9	00:28:24.3	00:28:28.7	00:28:31.3	00:28:34.3
FNN	01:28:48.4	01:28:50.4	01:28:28.0	01:28:40.4	01:28:39.4	01:28:31.4	01:28:45.1	01:28:53.2	01:29:02.7
Total	11:27:09.4	11:27:24.8	11:24:31.4	11:26:07.5	11:26:00.1	11:24:57.9	11:26:44.1	11:27:46.6	11:29:00.0
Pruning	05:43:38.1	05:41:40.7	05:43:24.5	05:42:41.9	05:42:07.5	05:43:06.4	05:42:38.3	05:42:03.2	05:43:54.7

Table 12: Time it takes by the diversifier algorithms in hh:mm:ss.s format.

6. Conclusion

In this paper, we proposed a surrogate method for optimization of the performance and emissions of an SI engine fuelled with methane/hydrogen blends. Experimental data was collected from a research test bench in the lab. The surrogate model is an ensemble learning algorithm that consists of a number of base learner algorithms. It is shown in the literature that diversity among the base learners improves the performance of ensemble learning algorithms. Improving diversity among the base learners is an optimization problem and in this paper, we proposed an evolutionary algorithm to diversify the base learners. Then an evolutionary algorithm is proposed as the pruning algorithm to find among the pool of base learners a subset that creates the best ensemble.

When optimizing a system via surrogate models, the fitness evaluation suffers from approximation uncertainty. Reducing such uncertainty is not possible with existing uncertainty reduction methods. To manage this, in this paper we proposed an uncertainty reduction algorithm that reduces uncertainty via averaging over a set of solutions in a sphere around the current solution. We compared our proposed algorithms with a set of existing algorithms. Experimental studies suggest that the proposed algorithms outperform the existing algorithms.

While the proposed algorithm outperforms the existing ones, some notes should be taken into account. First, in terms of time complexity, because the proposed algorithm spends a large amount of time diversifying the base learners and then pruning them, it requires more time in the design step. However, the time complexity of the resulting learning algorithm is the same

SA	λ	H	NO_x	BMEP	BSFC
44.7	1	0.01	779.06	4.99	268.54
44.33	1.01	0.04	923.95	5.16	289.25
30.48	1	0.09	1214.17	5.27	291.91
43.95	1.02	0.16	1606.24	5.39	286.49
31.2	1.01	0.2	2013.01	5.49	277.49
43.66	1.07	0.01	745.12	4.38	262.36
45.6	1.08	0.05	766.37	4.59	287.7
45.99	1.1	0.25	1846.71	5.12	254.18
30.78	1.16	0.26	1321.93	4.76	240.22
44.72	1.16	0.4	2228.4	5.02	240.57
30.78	1.26	0.24	860.36	4.45	227.59
30.39	1.25	0.3	942.97	4.65	233.06
44.65	1.26	0.35	1045.6	4.74	260
30.87	1.33	0.2	257.65	3.74	232.69
44.93	1.33	0.25	148.9	4.11	236.11
45.91	1.32	0.3	163.29	4.32	267.18
45.44	1.36	0.34	304.2	4.36	293.46
44.43	1.44	0.11	0.18	1.98	273.87
30.34	1.41	0.21	30.53	3.23	254.11
35.15	1.02	0.01	1324.11	5.33	269.16
34.47	1	0.04	1457.33	5.45	289.56
35.79	1.01	0.09	1670.68	5.43	288.46
35.69	1.02	0.14	1952.58	5.37	275.27
36.14	1	0.21	2243.24	5.36	258.98
35.06	1	0.26	2480.81	5.41	238.63
36.21	1.06	0.01	1043.19	4.82	263.44
34.14	1.09	0.06	1141.77	5.01	284.13
34.99	1.07	0.26	1999.08	4.94	243.77
34.97	1.25	0.29	1018.21	4.41	222.5
35.21	1.25	0.36	1050.24	4.58	232.8
35.2	1.24	0.39	1154.33	4.67	240.53
34.27	1.25	0.45	1233.98	4.66	240.39
33.99	1.27	0	1167.66	4.53	230.7
35.54	1.33	0.26	338.64	3.89	227.04
34.72	1.32	0.3	463.55	4.14	240.03
35.4	1.34	0.36	620.45	4.3	253.26
35.18	1.32	0.39	782.21	4.33	259.15
34.43	1.44	0.14	37.19	2.69	248.19
39.1	1.07	0	1489.57	5.12	266.22
39.82	1.08	0.31	2451.64	4.81	231.45
45.99	1	0.21	3069.52	5.43	273.03
43.77	1.02	0.35	3175.59	4.85	226.17
45.94	1.06	0	1920.87	5.2	266.9
46.11	1.35	0.09	139.88	3.86	266.85
44.86	1.42	0.16	50.44	3.35	323
44.92	1.6	0.26	0.41	2.82	728.56
48.88	1.62	0.41	13.96	3.09	316.02
54.73	1.53	0.19	2.7	2.96	457.56
58.8	1.59	0.29	20.92	3.31	264.68
65.08	1.24	0.01	768.15	3.56	224.06
71.1	1.71	0.1	0.36	2.36	481.28
75.42	1.45	0.01	84.02	4.1	301.23
75.66	1.5	0.01	9.6	4.13	588.34
75.02	1.78	0	18.29	4	334.08
70.89	1.44	0.01	61.46	3.68	230.08
69.03	1.59	0.44	784.92	2.42	222.9

Table 13: The input variables for the Pareto front in table 2.

Source	SS	df	MS	F	Prob>F
Columns	1.57e+14	23	6.82e+12	2.15e+03	0
Error	2.21e+12	696	3.18e+09		
Total	1.59e+14	719			

Table 14: The ANOVA test on the data in table 6 where the number of runs is 30.

as the existing ensemble learning algorithms. Because the design phase is performed once, it is acceptable to reach better performance at extra processing time. Note that every small improvement in the performance of a SI engine fueled with methane/hydrogen blends can make a huge difference in terms of fuel consumption, pollutant emissions, and global warming. For such tasks, it is worth creating more accurate systems at the cost of computational complexity. It is also worth mentioning that in the proposed pruning algorithm, we devised a mechanism that uses the time complexity of the resulting ensemble as an objective, so by carefully tuning this parameter, a designer can choose to have a less computationally expensive algorithm (at the cost of reduced performance).

The algorithm presented in this paper has some parameters. We tried to study some of the parameters (see for example figure 4). However, some other parameters require a huge amount of time to tune. In this paper, we used those values for the parameters shown in the literature to provide good results. Given the proposed algorithm has reached the best performance even without fine-tuning indicates its strengths. The algorithm will improve further if a study on the parameters is performed. Performing a thorough analysis of these parameters and studying their effect on the algorithm's performance remains for future work.

The proposed algorithm in this paper uses only the surrogate model to optimize the performance of an SI engine fueled with methane/hydrogen blends. There are still some open questions on how the existing system can be targeted in future work. First, the proposed surrogate model is trained once on the input data, and optimization is performed on the model to find the optimal solution. If it is possible to collect more data from the engine after the model is trained, then online learning can be used to design the surrogate model. In this paper, we did not have the option to receive more data so an offline method was proposed; however, this possibility could be considered for future work.

In the optimization phase of the proposed algorithm, it is only the sur-

rogate model that is employed as the fitness function. In some cases, it might be possible to also use the original system to measure the fitness of the solution. For example, it might be possible to run the engine alongside the surrogate model, and for some solutions, use the engine to estimate the fitness. In these scenarios, the engine can provide more accurate fitness at a higher cost. So the surrogate model is used as a cheap and fast, but less accurate fitness evaluation and the engine can be used as the more accurate but more expensive evaluation. Furthermore, every time the engine is used to measure fitness, the data record can be used to update the surrogate model. Finding the best way of combining the two ways of measuring fitness can be a line of research for future work.

The proposed algorithm in this paper diversifies the base learners by assigning different weights to each data record. If the number of data records is huge, then the number of weights will be large, resulting in a large search space for the diversified algorithm. To manage this, one way is to group the data records and assign different weights to each group of data records. This way the number of weights and thus optimization parameters for the diversified algorithm will be reduced. The challenge here is how to choose these groups. One way could be to assign the data records to the groups randomly. The argument supporting this approach is that randomness itself injects diversity. Another way is to assign similar data records (the data that are closer in the feature space) to the same group. The argument for this approach is that putting similar data records in the same groups increases diversity between groups. Studying the optimal approach remains for future work.

References

[1] T. Korakianitis, A. Namasivayam, R. Crookes, Natural-gas fueled spark-ignition (SI) and compression-ignition (CI) engine performance and emissions, Progress in Energy and Combustion Science 37 (1) (2011) 89–112.

[2] A.-H. Kakaee, A. Paykani, Research and development of natural-gas fueled engines in iran, Renewable and Sustainable Energy Reviews 26 (2013) 805–821.

[3] A.-H. Kakaee, A. Paykani, M. Ghajar, The influence of fuel composition

on the combustion and emission characteristics of natural gas fueled engines, Renewable and Sustainable Energy Reviews 38 (2014) 64–78.

[4] H. M. Cho, B.-Q. He, Spark ignition natural gas engines:A review, Energy Conversion and Management 48 (2) (2007) 608–618.

[5] F. Ma, S. Ding, Y. Wang, Y. Wang, J. Wang, S. Zhao, Study on combustion behaviors and cycle-by-cycle variations in a turbocharged lean burn natural gas si engine with hydrogen enrichment, International journal of hydrogen energy 33 (23) (2008) 7245–7255.

[6] A. Sofianopoulos, D. Assanis, S. Mamalis, Effects of hydrogen addition on automotive lean-burn natural gas engines: critical review, Journal of Energy Engineering 142 (2) (2015) E4015010.

[7] S. O. Akansu, Z. Dulger, N. Kahraman, T. N. Veziroglu, Internal combustion engines fueled by natural gas–hydrogen mixtures, International Journal of Hydrogen Energy 29 (14) (2004) 1527–1539.

[8] R. K. Mehra, H. Duan, R. Juknelevičius, F. Ma, J. Li, Progress in hydrogen enriched compressed natural gas (HCNG) internal combustion engines-A comprehensive review, Renewable and Sustainable Energy Reviews 80 (2017) 1458–1498.

[9] F. Moreno, M. Muñoz, J. Arroyo, O. Magén, C. Monné, I. Suelves, Efficiency and emissions in a vehicle spark ignition engine fueled with hydrogen and methane blends, International Journal of Hydrogen Energy 37 (15) (2012) 11495–11503.

[10] G. Kosmadakis, D. Rakopoulos, C. Rakopoulos, Methane/hydrogen fueling a spark-ignition engine for studying no, co and hc emissions with a research cfd code, Fuel 185 (2016) 903–915.

[11] A.-H. Kakaee, P. Rahnama, A. Paykani, B. Mashadi, Combining artificial neural network and multi-objective optimization to reduce a heavy-duty diesel engine emissions and fuel consumption, Journal of Central South University 22 (2015) 4235–4245.

[12] C. Kavuri, S. L. Kokjohn, Exploring the potential of machine learning in reducing the computational time/expense and improving the reliability

of engine optimization studies, International Journal of Engine Research 21 (7) (2020) 1251–1270.

[13] M. Aliramezani, C. R. Koch, M. Shahbakhti, Modeling, diagnostics, optimization, and control of internal combustion engines via modern machine learning techniques: A review and future directions, Progress in Energy and Combustion Science 88 (2022) 100967.

[14] K. Karunamurthy, A. A. Janvekar, P. Palaniappan, V. Adhitya, T. Lokeswar, J. Harish, Prediction of ic engine performance and emission parameters using machine learning: A review, Journal of Thermal Analysis and Calorimetry 148 (9) (2023) 3155–3177.

[15] M. Tayarani-N., X. Yao, H. Xu, Meta-heuristic algorithms in car engine design: A literature survey, IEEE Transactions on Evolutionary Computation 19 (5) (2015) 609–629. doi:10.1109/TEVC.2014.2355174.

[16] F. J. J. S. Bai, K. Shanmugaiah, A. Sonthalia, Y. Devarajan, E. G. Varuvel, Application of machine learning algorithms for predicting the engine characteristics of a wheat germ oil–hydrogen fuelled dual fuel engine, International Journal of Hydrogen Energy 48 (60) (2023) 23308–23322.

[17] X. Zhang, H. Li, M. Sekar, M. Elgendi, N. Krishnamoorthy, C. Xia, D. P. Matharasi, Machine learning algorithms for a diesel engine fuelled with biodiesel blends and hydrogen using lstm networks, Fuel 333 (2023) 126292.

[18] H. Wang, C. Ji, C. Shi, J. Yang, S. Wang, Y. Ge, K. Chang, H. Meng, X. Wang, Multi-objective optimization of a hydrogen-fueled wankel rotary engine based on machine learning and genetic algorithm, Energy 263 (2023) 125961.

[19] H. Wang, C. Ji, T. Su, C. Shi, Y. Ge, J. Yang, S. Wang, Comparison and implementation of machine learning models for predicting the combustion phases of hydrogen-enriched wankel rotary engines, Fuel 310 (2022) 122371.

[20] A. Rao, T. Chen, Y. Liu, F. Ma, Computational analysis of performances for a hydrogen enriched compressed natural gas engine'by advanced machine learning algorithms, Fuel 347 (2023) 128244.

[21] V. Sugumaran, V. Thangavel, M. Vijayaragavan, B. Subramanian, F. J. JS, E. G. Varuvel, et al., Efficacy of machine learning algorithms in estimating emissions in a dual fuel compression ignition engine operating on hydrogen and diesel, International Journal of Hydrogen Energy 48 (99) (2023) 39599–39611.

[22] Y. Zhu, Z. He, T. Xuan, Z. Shao, An enhanced automated machine learning model for optimizing cycle-to-cycle variation in hydrogen-enriched methanol engines, Applied Energy 362 (2024) 123019.

[23] J. Zareei, A. Rohani, Optimization and study of performance parameters in an engine fueled with hydrogen, International journal of hydrogen energy 45 (1) (2020) 322–336.

[24] H. Ma, Z. Li, M. Tayarani, G. Lu, H. Xu, X. Yao, Computational intelligence nonmodel-based calibration approach for internal combustion engines, Journal of Dynamic Systems, Measurement, and Control 140 (4) (2018) 041002.

[25] H. Dong, B. Song, P. Wang, Z. Dong, Surrogate-based optimization with clustering-based space exploration for expensive multimodal problems, Structural and Multidisciplinary Optimization 57 (2018) 1553–1577.

[26] J. Müller, J. Park, R. Sahu, C. Varadharajan, B. Arora, B. Faybishenko, D. Agarwal, Surrogate optimization of deep neural networks for groundwater predictions, Journal of Global Optimization 81 (2021) 203–231.

[27] K. O. Lye, S. Mishra, D. Ray, P. Chandrashekar, Iterative surrogate model optimization (ismo): An active learning algorithm for pde constrained optimization with deep neural networks, Computer Methods in Applied Mechanics and Engineering 374 (2021) 113575. doi:https://doi.org/10.1016/j.cma.2020.113575.
URL https://www.sciencedirect.com/science/article/pii/S004578252030760X

[28] H. Li, B. Xu, G. Lu, C. Du, N. Huang, Multi-objective optimization of pem fuel cell by coupled significant variables recognition, surrogate models and a multi-objective genetic algorithm, Energy Conversion and Management 236 (2021) 114063.

[29] P. Liao, C. Sun, G. Zhang, Y. Jin, Multi-surrogate multi-tasking optimization of expensive problems, Knowledge-Based Systems 205

(2020) 106262. doi:https://doi.org/10.1016/j.knosys.2020.106262.
URL https://www.sciencedirect.com/science/article/pii/S0950705120304536

[30] X. Ji, Y. Zhang, D. Gong, X. Sun, Dual-surrogate-assisted cooperative particle swarm optimization for expensive multimodal problems, IEEE Transactions on Evolutionary Computation 25 (4) (2021) 794–808.

[31] H. Tong, C. Huang, L. L. Minku, X. Yao, Surrogate models in evolutionary single-objective optimization: A new taxonomy and experimental study, Information Sciences 562 (2021) 414–437.

[32] J. A. Benediktsson, J. R. Sveinsson, O. K. Ersoy, P. H. Swain, Parallel consensual neural networks, IEEE Transactions on Neural Networks 8 (1) (1997) 54–64.

[33] L. Breiman, Stacked regressions, Machine learning 24 (1) (1996) 49–64.

[34] L. K. Hansen, P. Salamon, Neural network ensembles, IEEE transactions on pattern analysis and machine intelligence 12 (10) (1990) 993–1001.

[35] L. I. Kuncheva, Diversity in multiple classifier systems (2005).

[36] R. Polikar, Ensemble based systems in decision making, IEEE Circuits and systems magazine 6 (3) (2006) 21–45.

[37] L. Breiman, Bagging predictors, Machine learning 24 (2) (1996) 123–140.

[38] T. K. Ho, The random subspace method for constructing decision forests, IEEE transactions on pattern analysis and machine intelligence 20 (8) (1998) 832–844.

[39] P. Burrascano, et al., Learning vector quantization for the probabilistic neural network, IEEE transactions on Neural Networks 2 (4) (1991) 458–461.

[40] D. F. Specht, Probabilistic neural networks, Neural networks 3 (1) (1990) 109–118.

[41] J. Yang, J. Ma, Feed-forward neural network training using sparse representation, Expert Systems with Applications 116 (2019) 255–264.

[42] B. Warsito, R. Santoso, Suparti, H. Yasin, Cascade forward neural network for time series prediction, in: Journal of Physics: Conference Series, Vol. 1025, IOP Publishing, 2018, p. 012097.

[43] I. Loboda, Y. Feldshteyn, V. Ponomaryov, Neural networks for gas turbine fault identification: Multilayer perceptron or radial basis network? (2012).

[44] M. Steinbach, P.-N. Tan, knn: k-nearest neighbors, in: The top ten algorithms in data mining, Chapman and Hall/CRC, 2009, pp. 165–176.

[45] P. Dolezel, P. Skrabanek, L. Gago, Pattern recognition neural network as a tool for pest birds detection, in: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2016, pp. 1–6.

[46] D. F. Specht, et al., A general regression neural network, IEEE transactions on neural networks 2 (6) (1991) 568–576.

[47] M. R. Sabour, S. M. A. Movahed, Application of radial basis function neural network to predict soil sorption partition coefficient using topological descriptors, Chemosphere 168 (2017) 877–884.

[48] M.-H. Tayarani-N, A. P. Bennett, H. Xu, X. Yao, Improving the performance of evolutionary engine calibration algorithms with principal component analysis, in: 2016 IEEE Congress on Evolutionary Computation (CEC), 2016, pp. 5128–5137. doi:10.1109/CEC.2016.7748340.

[49] A. Paykani, C. E. Frouzakis, K. Boulouchos, Numerical optimization of methane-based fuel blends under engine-relevant conditions using a multi-objective genetic algorithm, Applied Energy 242 (2019) 1712–1724.

[50] A. Paykani, C. E. Frouzakis, C. Schürch, F. Perini, K. Boulouchos, Computational optimization of ch4/h2/co blends in a spark-ignition engine using quasi-dimensional combustion model, Fuel 303 (2021) 121281.

[51] M. Tayarani, A. Esposito, A. Vinciarelli, What an "ehm" leaks about you: Mapping fillers into personality traits with quantum evolutionary feature selection algorithms, IEEE Transactions on Affective Computing 13 (1) (2022) 108–121. doi:10.1109/TAFFC.2019.2930695.

[52] J. B. Heywood, Combustion engine fundamentals, 1ª Edição. Estados Unidos 25 (1988) 1117–1128.

[53] M.-H. Tayarani-N., Evolutionary optimization of policy responses to covid-19 pandemic via surrogate models, Applied Soft Computing 154 (2024) 111359. doi:https://doi.org/10.1016/j.asoc.2024.111359.
URL https://www.sciencedirect.com/science/article/pii/S1568494624001339

[54] B. Krawczyk, M. Woźniak, Wagging for combining weighted one-class support vector machines, Procedia Computer Science 51 (2015) 1565–1573.

[55] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: Bagging, boosting, and variants, Machine learning 36 (1) (1999) 105–139.

[56] G. I. Webb, Multiboosting: A technique for combining boosting and wagging, Machine Learning 40 (2) (2000) 159–196.

[57] J. R. Quinlan, et al., Bagging, boosting, and c4. 5, in: Aaai/Iaai, vol. 1, 1996, pp. 725–730.

[58] G. I. Webb, Z. Zheng, Multistrategy ensemble learning: Reducing error by combining ensemble learning techniques, IEEE transactions on knowledge and data engineering 16 (8) (2004) 980–991.

[59] B. Krawczyk, M. Woźniak, Wagging for combining weighted one-class support vector machines, Procedia Computer Science 51 (2015) 1565–1573.

[60] C. Zhao, D. Wu, J. Huang, Y. Yuan, H.-T. Zhang, R. Peng, Z. Shi, Boosttree and boostforest for ensemble learning, IEEE Transactions on Pattern Analysis and Machine Intelligence (2022).

[61] G. I. Webb, Multiboosting: A technique for combining boosting and wagging, Machine learning 40 (2000) 159–196.

[62] Y. Yang, H. Lv, N. Chen, A survey on ensemble learning under the era of deep learning, Artificial Intelligence Review 56 (6) (2023) 5545–5589.

[63] C. Silva, T. Bouwmans, C. Frélicot, Superpixel-based online wagging one-class ensemble for feature selection in foreground/background separation, Pattern Recognition Letters 100 (2017) 144–151.

[64] O. Sagi, L. Rokach, Ensemble learning: A survey, Wiley interdisciplinary reviews: data mining and knowledge discovery 8 (4) (2018) e1249.

[65] E. Eldesouky, M. Bekhit, A. Fathalla, A. Salah, A. Ali, A robust uwsn handover prediction system using ensemble learning, Sensors 21 (17) (2021) 5777.

[66] M. H. T. Najaran, An evolutionary ensemble convolutional neural network for fault diagnosis problem, Expert Systems with Applications 233 (2023) 120678. doi:https://doi.org/10.1016/j.eswa.2023.120678.
URL https://www.sciencedirect.com/science/article/pii/S0957417423011806

[67] M.-H. Tayarani-Najaran, A novel ensemble machine learning and an evolutionary algorithm in modeling the covid-19 epidemic and optimizing government policies, IEEE Transactions on Systems, Man, and Cybernetics: Systems 52 (10) (2022) 6362–6372. doi:10.1109/TSMC.2022.3143955.

[68] Q. Dai, R. Ye, Z. Liu, Considering diversity and accuracy simultaneously for ensemble pruning, Applied Soft Computing 58 (2017) 75–91.

[69] M. H. Tayarani-N, M. Akbarzadeh-T, Improvement of the performance of the quantum-inspired evolutionary algorithms: structures, population, operators, Evolutionary Intelligence 7 (4) (2014) 219–239.

[70] Yaochu Jin, J. Branke, Evolutionary optimization in uncertain environments-a survey, IEEE Transactions on Evolutionary Computation 9 (3) (2005) 303–317.

[71] P. Rakshit, A. Konar, S. Das, Noisy evolutionary optimization algorithms–a comprehensive survey, Swarm and Evolutionary Computation 33 (2017) 18–45.

[72] E. Zitzler, M. Laumanns, L. Thiele, Spea2: Improving the strength pareto evolutionary algorithm, TIK-report 103 (2001).

[73] M.-H. Tayarani-N., A. Prügel-Bennett, Anatomy of the fitness landscape for dense graph-colouring problem, Swarm and Evolutionary Computation 22 (2015) 47–65. doi:https://doi.org/10.1016/j.swevo.2015.01.005. URL https://www.sciencedirect.com/science/article/pii/S2210650215000152

[74] L. Deng, S. Wang, L. Qiao, B. Zhang, De-rco: Rotating crossover operator with multiangle searching strategy for adaptive differential evolution, IEEE Access 6 (2018) 2970–2983.

[75] Q. Yang, W. Chen, J. D. Deng, Y. Li, T. Gu, J. Zhang, A level-based learning swarm optimizer for large-scale optimization, IEEE Transactions on Evolutionary Computation 22 (4) (2018) 578–594. doi:10.1109/TEVC.2017.2743016.

[76] D. Molina, M. Lozano, A. Sánchez, F. Herrera, Memetic algorithms based on local search chains for large scale continuous optimisation problems: Ma-ssw-chains, Soft Computing 15 (2011) 2201–2220.

[77] X. Yao, Y. Liu, Fast evolution strategies, Control and Cybernetics 26 (1997) 467–496.

[78] E. Mininno, F. Cupertino, D. Naso, Real-valued compact genetic algorithms for embedded microcontroller optimization, IEEE Transactions on Evolutionary Computation 12 (2) (2008) 203–219.

[79] M. H. T. Najaran, M. R. A. Tootounchi, Probabilistic optimization algorithms for real-coded problems and its application in latin hypercube problem, Expert Systems with Applications 160 (2020) 113589.

[80] A. Jurek, Y. Bi, S. Wu, C. D. Nugent, Clustering-based ensembles as an alternative to stacking, IEEE Transactions on Knowledge and Data Engineering 26 (9) (2014) 2120–2137.

[81] B. H. Menze, B. M. Kelm, D. N. Splitthoff, U. Koethe, F. A. Hamprecht, On oblique random forests, in: D. Gunopulos, T. Hofmann, D. Malerba, M. Vazirgiannis (Eds.), Machine Learning and Knowledge Discovery in Databases, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 453–469.

[82] L. Zhang, P. N. Suganthan, Oblique decision tree ensemble via multisurface proximal support vector machine, IEEE Transactions on Cybernetics 45 (10) (2015) 2165–2176.

[83] J. H. Holland, Adaption in Natural and Artificial Systems, 1st Edition, Ann Arbor MI, 1975.

[84] L. J. Fogel, A. J. Owens, M. J. Walsh, Artificial Intelligence through Simulated Evolution, John Wiley, New York, USA, 1966.

[85] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, Evolutionary Computation, IEEE Transactions on 3 (2) (1999) 82–102.

[86] H.-P. Schwefel, Evolution and Optimum Seeking, Wiley, 1995.