Learning Boosts Optimisation: Surrogate-Assisted Real Engine Calibration

Hao Tong*†, Jiyuan Pei*†, Qingquan Zhang*†, Jialin Liu*†
*Research Institute of Trustworthy Autonomous System
Southern University of Science and Technology
†Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation
Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, China
htong6@outlook.com, liujl@sustech.edu.cn

Xudong Feng‡¶, Feng Wu§¶
‡Xidian University
Xi'an, China
§School of Power and Energy
Northwestern Polytechnical University
Xi'an, China
¶AECC
Chengdu, China

***Abstract*—Computational intelligence methods have been widely applied to model-based engine calibration. Engine calibration based on computational fluid dynamics (CFD) calculations is time-consuming and constrained. In this paper, we model a real-world aero-engine calibration problem with many parameters as an expensive optimisation problem with hidden constraints. Two surrogate-assisted meta-heuristic frameworks using offline and online strategies are proposed in this paper for efficient aero-engine calibration. A surrogate model is trained on engine parameter settings, that lead to valid and invalid CFD calculations, to predict the feasibility of new parameter settings. Parameter settings that are predicted as infeasible by the surrogate model will be eliminated for evaluation during search to reduce the time wasted on infeasible solutions. To validate our approaches, instantiation of the offline and online frameworks are implemented with a neural network model and a self-adaptive particle swarm optimisation and verified on calibrating a real aero-engine model. Both the proposed offline and online frameworks significantly speed up the calibration in terms of real-time performance compared with the approach without using a surrogate model. The surrogate model not only improves the calibration efficiency but also is capable of indicating the importance of parameters to guide the calibration order.**

***Index Terms*—Engine calibration, Hidden constrained optimisation, Expensive optimisation, Surrogate model.**

I. Introduction

Engine calibration is an essential problem in the engine design. It aims to adjust a group of parameters to ensure the performance of an engine [1] with the help of an engine simulator. An engine simulator is used which takes a parameter setting as input and outputs a number of values to indicate the quality and feasibility[1] of the input setting. Typically, the calibration process can be modelled as a constrained black-box optimisation problem. The meta-heuristic algorithms, such as evolutionary algorithms [2], are very suitable for such kind of problems. However, the simulation process of performing the parameters with a given engine model to evaluate parameters' quality is usually time-consuming. Therefore, the engine calibration is a typical expensive constrained optimisation problem.

Plenty of works have been published in the literature to solve expensive constrained problems. To obtain a better solution given limited computational resources, surrogate models are usually used to approximate the actual fitness function and assist meta-heuristics for optimising expensive problems [3]–[5]. The expensive evaluation process usually outputs both the objective value and the value for penalising constraints violation for one simulation [6]. Building one or two models to approximate the objective function and constraints penalty function together or separately, respectively, have been investigated in the literature. For example, Wang et al. [7] used a regression neural network to approximate the objective and constraints penalty simultaneously. Runarsson [8] constructed two surrogate models to approximate separately the objective and constraints penalty.

However, most researches focused on explicitly constrained problems in which the actual constraints violation values are provided explicitly [9]. Existing approaches for explicitly constrained problems, such as penalty functions [10], stochastic ranking [8] and feasibility rules [11], have been very mature and shown to be effective. Few works considered hidden constrained problems, in which only a binary value is provided to indicate the feasibility of a solution [12]. Lee et al. [13] combined expected improvements strategy and the predicted probability of being a valid solution for expensive hidden constrained problem. They chose one sample solution,

This work was supported by the AECC, the National Natural Science Foundation of China (Grant No. 61906083), the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (Grant No. 2017ZT07X386), the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2021A1515011830), the Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531) and the Shenzhen Fundamental Research Program (Grant No. JCYJ20190809121403553).

Corresponding author: Jialin Liu (liujl@sustech.edu.cn).

[1]"Feasibility" here means if the computational fluid dynamics (CFD) calculations performed by the engine simulator converge given the input engine parameter values. If not, then the parameter values are considered infeasible and cannot be used in the real engine.

which maximises the value of expected improvements and the predicted probability of being a valid solution, for actual simulation [13]. It is efficient when handling low-dimensional problems but hard to handle well median- or high-dimensional problems. Müller and Day [14] constructed an approximate model for the objective function and the hidden constraints using a radial basis function (RBF) surrogate model. The frequent updates of both the objective's surrogate model and the constraints' model also lead to high computational cost. Hence, it is more suitable for tackling highly costly problems of which each simulation takes hours or even longer.

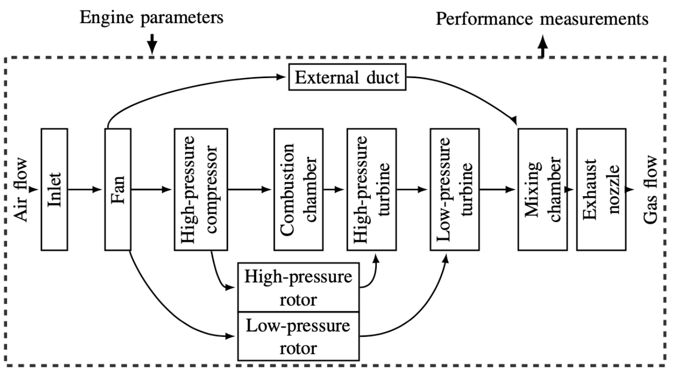

Fig. 1: Schematic diagram of the engine studied in this work. Image edited from Fig. 2 of [15] with authors' permission.

In this paper, we model a real aero-engine calibration problem [15] (illustrated in Fig. 1) as a computationally expensive problem. This real aero-engine has 27 input parameters to be calibrated simultaneously, and the simulation time varies between several or dozens of seconds for different parameter settings due to the CFD calculations. Given a parameter setting, the engine simulation outputs a binary value to indicate its feasibility and the performance metrics returned by CFD calculations. A general formulation of such hidden constrained optimisation problem is as follows:

$$\begin{aligned} \text{Min} \quad & f(\boldsymbol{x}) \\ \text{Subject to} \quad & g(\boldsymbol{x}) = 1, \quad g(\boldsymbol{x}) \in \{0, 1\}. \end{aligned} \tag{1}$$

Since only the violation of constraints is determined by $g(\boldsymbol{x})$ and a binary value is returned, it is impossible to build a regression model for predicting the penalty functions. Thus, determining a solution's feasibility is intuitively regarded as a binary classification problem. We employ a surrogate model, i.e., classification model, to help pre-select potentially feasible solutions for the expensive aero-engine calibration problem studied in this paper. The contributions of this paper are as follows.

- We model a real aero-engine calibration problem as a computationally expensive problem with hidden constraints that the costly simulation process only provides a binary value to indicate parameters' feasibility rather than the quantified penalty value of constraints violation.
- For a such expensive problem with hidden constraints, we propose two efficient online and offline frameworks combining surrogate model and meta-heuristics to tackle the expensive constraints and reduce the computational cost. More specifically, a classification model is constructed either offline or online to predict and pre-select possibly feasible solutions as candidate solutions for a meta-heuristic algorithm.
  – In the offline framework, an offline surrogate is trained before the optimisation process with more sampling data and then used during the whole optimisation process without being further updated.
  – The online framework maintains an archive of evaluated samples during the optimisation process and updates the surrogate model once its accuracy is not satisfied.
- The surrogate model enhances the performance of the meta-heuristic algorithm for automated calibration, which is demonstrated in our empirical studies on calibrating a real aero-engine model. Moreover, it helps human engineers further analyse the importance of each parameter to the performance of aero-engine.

The remainder of this paper is organised as follows. Section II introduces the real aero-engine calibration problem considered in this work. Section III details our proposed frameworks. Section IV presents the empirical studies and Section V concludes.

II. A Real Aero-engine Calibration Problem

In this paper, we study a real aero-engine calibration problem [15]. Given an engine model, the goal is to minimise the difference between its performance and its theoretically optimal performance at several given operation points $s_1, s_2, \ldots, s_k$ by adjusting a set of parameters, i.e., a physical setting in the engine model.

Human engineers usually repeatedly and manually tune a parameter setting at several operation points in a sequential manner as illustrated in Fig. 2, while computational methods allow to consider the performance at several given operation points simultaneously. Therefore, given d parameters to be adjusted simultaneously, donated as $\boldsymbol{x} = \{x_1, .., x_d\}$, a number of m numerical measurements of its performance at a operation point s, denoted as $\boldsymbol{y} = \{y_{1,s}, ..., y_{m,s}\}$, can be obtained by a *computational model* which performs the computational fluid dynamics (CFD) calculations, provided by human experts. The desired values of the measurements are denoted as $\boldsymbol{y}^* = \{y^*_{1,s}, ..., y^*_{m,s}\}$. The quality of an engine parameter setting $\boldsymbol{x}$ is usually evaluated with the root-mean-square error (RMSE) between the actual values of measurements to their desired values. The objective function $f(\boldsymbol{x})$ of this aero-engine calibration problem, i.e. RMSE, is formulated as

$$f(\boldsymbol{x}) = \sqrt{\frac{1}{k*m}\sum_{j=1}^{k}\sum_{i=1}^{m}\left(\frac{Simulate(\boldsymbol{x}, s_j) - y^*_{i,s_j}}{y^*_{i,s_j}}\right)^2} \tag{2}$$

where y^*_{i,s_j} denotes to the target value of the i^{th} measurement of the optimal engine at operation point s_j and $k = 33$, $m =$

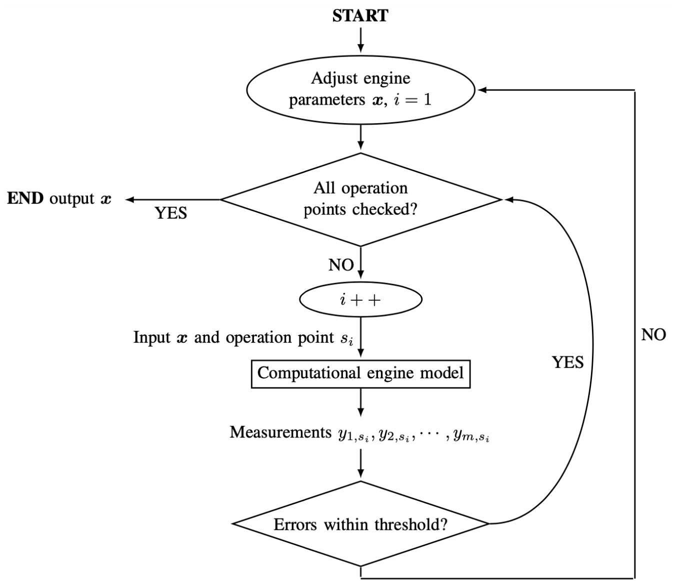
START
Adjust engine parameters $\boldsymbol{x}$, $i = 1$
All operation points checked?
END output $\boldsymbol{x}$
YES
NO
$i++$
Input $\boldsymbol{x}$ and operation point s_i
Computational engine model
Measurements $y_{1,s_i}, y_{2,s_i}, \cdots, y_{m,s_i}$
Errors within threshold?
YES
NO

Fig. 2: Manual engine calibration process. Image reproduced from Fig. 1 of [15] with authors' permission. An acceptable parameter setting $\boldsymbol{x}$ should ensure measurement errors within certain threshold at several given operation points $s_1, s_2, \ldots, s_k$.

15 in this aero-engine calibration problem. $Simulate(\boldsymbol{x}, s_j)$ represents the simulation process of computational engine model on input $\boldsymbol{x}$ and operation point s_j. On the other hand, the feasibility of an input $\boldsymbol{x}$, i.e. $g(\boldsymbol{x})$, is also determined by $Simulate(\boldsymbol{x}, s_j)$.

Based on human knowledge, there are some phenomena in engine design that are unformulated but significantly affect an engine's performance. The computational model includes those phenomena as an unformulated constraint. The output of this model is a set of numerical measurements that describe the engine's performance, $\boldsymbol{y}$ together with a binary indicator of the feasibility of the input parameter setting $\boldsymbol{x}$. The feasibility can be obtained only when a simulation is over. A simulation of the computational model is usually time-consuming. Moreover, determining an infeasible input parameter setting sometimes costs much more time than determining a feasible one. The main reason is that a parameter is determined infeasible by the computational model if the CFD calculations performed to simulate the setting diverge or do not converge to a certain level after certain time or a number of iterations (i.e., maximum budget of CFD calculations), while simulating a feasible parameter setting will lead to a convergence before using up the budget. Depending on the engine, the number of parameters to be calibrated and the maximum budget or desired precision of CFD calculations, the time cost of determining an infeasible parameters varies from several seconds to hours. For reference, according to our experimental tests, it costs on average 0.73 second in our experimental platform (Windows10 with Intel Core i7-8700CPU@3.2GHz) to obtain the performance measurements on our aero-engine model if the input parameter setting is feasible. However, the simulation time varies from 1 second to 6 seconds if the input parameter setting is infeasible.

Additionally, an engine should be calibrated under several different operation points. In other words, an identical parameter setting should satisfy different performance requirements under different testing scenarios, thus different $\boldsymbol{y}^*$'s, at the same time. Therefore, this engine calibration problem can be viewed as an expensive optimisation problem when considering dozens of operation points.

III. Surrogate-assisted Engine Calibration Framework

Meta-heuristics have been applied to automatically calibrate engines, however the feasibility of parameter settings were rarely taken into account [1], [16]–[22]. As mentioned in Section II, sampling infeasible solutions (i.e., parameter settings) is likely to consume much more computational cost which significantly reduces the calibration efficacy. The optimisation cannot benefit from infeasible solutions because hidden constraints cannot provide an explicit degree of constraints violation. The calibration efficacy would be improved by eliminating infeasible solutions before the actual simulations. Therefore, we propose two surrogate-assisted engine calibration frameworks, including an offline framework and an online one, in which a classification surrogate model is constructed based on sampled feasible and infeasible solutions to predict whether a new input solution is feasible or not. The offline framework only constructs one surrogate model before the calibration starts, while the online framework updates the surrogate model during the calibration process. In both frameworks, we adopt an extreme barrier approach [23], which sets a penalty value to an infeasible solution as its fitness. The details of the offline and online frameworks are described in Sections III-A and III-B, respectively.

A. *Offline surrogate-assisted engine calibration framework*

The offline framework (Algorithm 1) uses a meta-heuristic algorithm $\mathcal{A}$, such as an evolutionary algorithm (EA) to search for optimal engine parameter setting and a surrogate model, $\hat{g}$, constructed with a set of offline collected samples to predict the feasibility of solutions generated during optimisation. The surrogate model remains unchanged during the whole optimisation process.

As shown in Algorithm 1, first, an initial sample set (e.g., a population in EA) $\mathcal{P}$ is collected by Latin Hypercube Sampling (LHS), which aims at collecting samples distributed as uniformly as possible. These samples (individuals) are then evaluated by the actual simulator to obtain their quality and feasibility. Based on those evaluated samples, a classification surrogate model $\hat{g}$ is constructed. In the main loop of Algorithm 1, the algorithm $\mathcal{A}$ generates a new sample set $\mathcal{P}'$, the feasibility of which will be predicted by the surrogate model $\hat{g}$. For each sample in $\mathcal{P}'$, if it is predicted as feasible by $\hat{g}$, then it will be evaluated by the actual simulator to get its actual quality (or fitness) and actual feasibility; otherwise its fitness is set as an infinite value. The actually infeasible solution will also be assigned an infinite value as fitness. The algorithm $\mathcal{A}$ then selects promising solutions between $\mathcal{P}$ and $\mathcal{P}'$ to form the new $\mathcal{P}$. The framework will repeat the above loop and terminate until running out of a given number of actual simulations, i.e., calls of the engine's computational model.

Algorithm 1: Offline surrogate-assisted engine calibration framework

```
   Input: $\mathcal{A}$: meta-heuristic (e.g., EA)
   Input: $\hat{g}$: classification surrogate model constructed
          offline with a number of evaluated samples
1  $\mathcal{P} \leftarrow$ Sampling an initial set of solutions with LHS;
2  for Each solution $\boldsymbol{x} \in \mathcal{P}$ do
3      Simulate $\boldsymbol{x}$ to get its actual evaluation $f(\boldsymbol{x})$ and
        feasibility $g(\boldsymbol{x})$;
4  while stopping condition is not satisfied do
5      Sample a new set $\mathcal{P}'$ from $\mathcal{P}$ by $\mathcal{A}$;
6      for Each $\boldsymbol{x}'$ in $\mathcal{P}'$ do
7          $quality = \infty$;
8          $predictedFeasible = \hat{g}(\boldsymbol{x}')$; // Predict
               the feasibility of $\boldsymbol{x}'$
9          if $predictedFeasible$ then
10             $quality, isFeasible = f(\boldsymbol{x}'), g(\boldsymbol{x}')$;
                   // Simulate $\boldsymbol{x}'$ with the
                   actual computational model
                   of engine, return its
                   evaluation (quality) and
                   feasibility
11     Select promising solutions between $\mathcal{P}$ and $\mathcal{P}'$
        according to their evaluations ($quality$ values) to
        form a new set $\mathcal{P}$.
   Output: Best solution in $\mathcal{P}$
```

B. Online surrogate-assisted engine calibration framework

The online framework constructs a surrogate model before the optimisation starts and updates it during the optimisation progress. The samples generated during the optimisation are collected as new training data for updating the surrogate model. The online framework is described in Algorithm 2.

A number of initial solutions are sampled and forms $\mathcal{P}$. The actual simulator, i.e., an engine's computational model, evaluates them and returns the actual evaluation value and actual feasibility of each solution. In this framework, we maintain two archives to save all the evaluated solutions during the optimisation, XP for saving feasible solutions and XN for saving infeasible ones. An initial classification surrogate model $\hat{g}$ is then constructed using all the samples in $X = XP \cup XN$. After that, the framework starts the main optimisation loop. At each iteration, the algorithm $\mathcal{A}$ will generate a new set $\mathcal{P}'$, the feasibility of each solution in which will be predicted by the surrogate model $\hat{g}$. Determining whether a solution should be actually evaluated or not is similar to the offline framework. In addition, the evaluated solution will be added to XP or XN according to its actual feasibility. If the predicted feasibility of an evaluated solution is incorrect (i.e., false positive), it will be recorded by a counter n_{FP}. The number of all solutions which are predicted to be feasible in $\mathcal{P}'$ (i.e., positive predictions) is also counted and denoted as n_P. At the end of each iteration, the surrogate model is re-constructed if its precision is lower than a threshold ϵ. The algorithm $\mathcal{A}$ then selects promising solutions between $\mathcal{P}$ and $\mathcal{P}'$ to form a new $\mathcal{P}$. Finally, the whole algorithm terminates after running out of a given number of actual simulations.

During optimisation, we limit the size of XP and XN. The oldest recorded samples will be removed if XP or XN exceeds its capacity after adding newly evaluated samples. It can not only limit the amount of training data to avoid super long training time but also help the newly constructed model focus more on the current search area by removing former samples.

C. Discussion

Both the offline and online frameworks use a surrogate model to pre-select the possibly feasible solutions, therefore their performance highly depends on the model's accuracy. Since the offline framework only trains the model before the optimisation process, it requires certain amount of offline samples to obtain a surrogate model of high accuracy. In this paper, 500 initial samples, including 250 feasible ones and 250 infeasible ones collected from the whole search space are used for training an offline model. On the other hand, the online framework re-constructs the surrogate model during the optimisation process, which might also be computationally expensive. Hence, our online framework uses two archives of fixed size for saving training samples to bound the training time. According to our empirical studies, it only costs 0.1680 seconds on average for training a neural network model on our experimental platform (Windows10 with Intel Core i7-8700CPU@3.2GHz), which is still cheaper than the average time consumed to simulate a solution. Besides, we only re-construct the prediction model when the true positive prediction accuracy is lower than a threshold, which also helps save computational resources by decreasing the frequency of re-construction. The following section provides more details of comparing the offline and online frameworks.

IV. Experiments

In this paper, we implement two instantiations of the proposed frameworks. The performance of the two instantiations is also compared and analysed on calibrating a real aero-engine described in [15].

A. Experimental setting

For the sake of simplicity, we applied a self-adaptive Particle Swarm Optimisation (saPSO) algorithm which was also proposed for the aero-engine calibration problem in our previous work [15]. The parameter setting in saPSO is same as the setting in [15]. The maximum number of fitness evaluations

Algorithm 2: Online surrogate-assisted engine calibration framework

```
Input: $\mathcal{A}$: meta-heuristic (e.g., EA)
Input: $\epsilon$: threshold of precision
1  $\mathcal{P} \leftarrow$ Sampling an initial set of solutions with LHS;
2  $XP = \emptyset$ // Archive for feasible samples
3  $XN = \emptyset$ // Archive for infeasible ones
4  for Each solution $\boldsymbol{x} \in \mathcal{P}$ do
5      Simulate $\boldsymbol{x}$ to get its actual evaluation $f(\boldsymbol{x})$ and
         feasibility $g(\boldsymbol{x})$;
6      if $g(\boldsymbol{x})$ then
7          Save $\boldsymbol{x}$ to $XP$;
8      else
9          Save $\boldsymbol{x}$ to $XN$;
10 Train a classification surrogate model $\hat{g}$ on $XP \cup XN$;
11 while stopping condition is not satisfied do
12     Sample a new set $\mathcal{P}'$ from $\mathcal{P}$ by $\mathcal{A}$;
13     $n_P = 0$; // Count feasible predictions
14     $n_{FP} = 0$; // Count false positives
15     for Each $\boldsymbol{x}'$ in $\mathcal{P}'$ do
16         $quality = \infty$;
17         $predictedFeasible = \hat{g}(\boldsymbol{x}')$; // Predict
               the feasibility of $\boldsymbol{x}'$
18         if $predictedFeasible$ then
19             $quality, isFeasible = f(\boldsymbol{x}'), g(\boldsymbol{x}')$;
                   // Simulate $\boldsymbol{x}'$ by the
                   actual simulation process,
                   obtain its fitness and
                   feasibility
20             $n_P++$;
21             if $isFeasible$ then
22                 Save $\boldsymbol{x}'$ to $XP$;
23             else
24                 Save $\boldsymbol{x}'$ to $XN$;
25                 $n_{FP}++$;
26             Update $X$: $X = XP \cup XN$;
27     if $\frac{n_P - n_{FP}}{n_P} < \epsilon$ then
28         Re-construct $\hat{g}$ using updated $X$;
29     Select promising solutions between $\mathcal{P}$ and $\mathcal{P}'$
         according to their evaluations ($quality$ values) to
         form a new set $\mathcal{P}$.
Output: Best solution in $\mathcal{P}$
```

is set as a relatively large value: $100d$, where d is the number of decision variables, i.e., number of parameters to be calibrated, because we would like to investigate how much the computational time will cost to reach the different calibration requirements. The threshold for precision ϵ in online framework set as 0.9 finally after parameter configuration by ourselves. All the experiments are programmed in MATLAB R2020b and executed on a Windows10 with Intel Core i7-8700CPU@3.2GHz.

The two generated algorithm instances are abbreviated as Off-saPSO and On-saPSO. Both employed a neural network, using deep learning toolbox in MATLAB[2], to construct the surrogate model. The network in our experiments has one hidden layer with 50 neurons, and the input layer has 27 neurons for inputting the 27 different aero-engine parameters, and one neuron for the output layer to output the probability of being feasible. Moreover, 500 samples are used to train the surrogate model in the offline framework, including 250 feasible and 250 infeasible samples. In the online framework, all the data in the archives are used as training data during the optimisation process.

B. Influence of classification model on optimisation algorithm

TABLE I: Best solution (Mean $\pm$ Standard Deviation) and the average time used by saPSO, Off-saPSO and On-saPSO over 25 independent runs. The symbol in Off-saPSO/On-saPSO represents the Off-saPSO/On-saPSO significantly win ($>$) saPSO or no significant difference ($\approx$).

Algorithm	saPSO	Off-saPSO	On-saPSO
MEAN$\pm$STD (%)	0.3423$\pm$0.1268	0.3262$\pm$0.0848 ($\approx$)	0.2990$\pm$0.1026 ($\approx$)
TIME (s)	2,475	1,993 ($>$)	2,290 ($\approx$)

TABLE II: Average and standard deviation of number of actual simulations for reaching the different RSME requirements. The symbol in Off-saPSO/On-saPSO represents the Off-saPSO/On-saPSO significantly win ($>$) saPSO or no significant difference ($\approx$).

RMSE	saPSO	Off-saPSO	On-saPSO
3.0%	53$\pm$50	31$\pm$30 ($>$)	49$\pm$44 ($\approx$)
2.5%	97$\pm$66	60$\pm$43 ($>$)	79$\pm$59 ($\approx$)
2.0%	179$\pm$66	142$\pm$85 ($>$)	149$\pm$97 ($\approx$)
1.5%	382$\pm$156	306$\pm$182 ($>$)	258$\pm$187 ($>$)
1.0%	807$\pm$295	696$\pm$340 ($\approx$)	598$\pm$245 ($>$)
0.5%	1826$\pm$502	1711$\pm$470 ($\approx$)	1597$\pm$465 ($\approx$)

The results of three different algorithm instances on the engine-calibration problem over 25 independent runs are presented in Table I. The first row represents the average mean error and standard deviation, and the values in the second row represent the average total time consumed by each algorithm instance in the experiments. The signrank test with a 0.05 significance level is performed between saPSO and saPSO with two proposed frameworks in terms of quality of obtained solution (MEAN$\pm$STD) and the time used to obtain the final best solution, respectively. '$>$' denotes that Off-saPSO/On-saPSO performs significantly better than saPSO and '$\approx$' refers to no significant difference. Furthermore, we presented two tables of the number of actual simulations used and running time to reach different RSME requirements ({3.0%, 2.5%, 2.0%, 1.5%, 1.0%, 0.5%}). As shown in Table II, Off-saPSO

[2]*patternnet* function

reaches {3.0%, 2.5%, 2.0%, 1.5%} using the significantly fewer number of actual simulations than saPSO does while On-saPSO performs better than saPSO in reaching the requirements {1.5%, 1.0%}. In Table III, Off-saPSO reaches {3.0%, 2.5%, 2.0%, 1.5%} using significantly shorter time than saPSO and On-saPSO is significantly faster than saPSO in reaching {1.5%, 1.0%}. However, both Off-saPSO and On-saPSO perform similarly with saPSO in reaching requirements 0.5%. Our proposed frameworks significantly increase the algorithm's efficiency and use much less time to obtain a similar result than using the original algorithm without surrogate model in reaching requirements greater than 0.5% in this aero-engine calibration problem.

a) ***Convergence influence analysis****:* From the above results, both the online and offline frameworks enhance the performance of the original saPSO. These two frameworks use a surrogate model to predict the feasibility of a new individual. For the original optimisation algorithm, an infeasible individual is required to be evaluated by an actual simulation to determine its feasibility, which results into expensive cost for each simulation. However, the proposed frameworks use the surrogate model to pre-select feasible individuals. The predicted infeasible individuals will be eliminated directly, and the saved computational resources can be used for running more generations of optimisation. On the other hand, an infeasible individual's fitness is usually inferior with a high probability of this real-world problem. The surrogate model also helps the algorithm eliminate the potential low-quality individuals. Therefore, our proposed frameworks significantly enhanced the original algorithm.

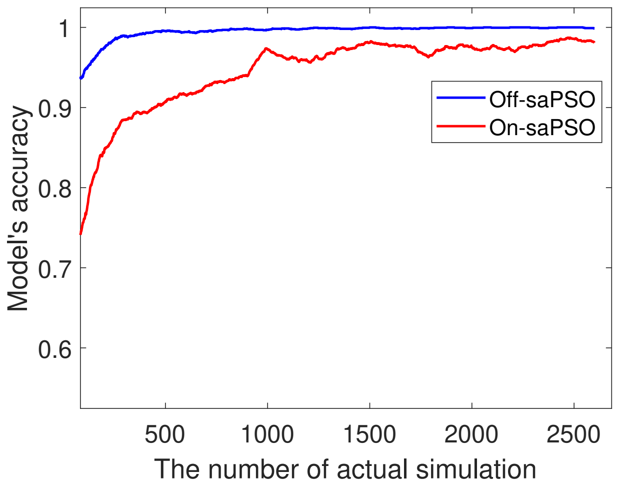

Fig. 3: Dynamic change of surrogate model's accuracy during optimisation.

The Off-saPSO obtains a better performance in reaching a higher RSME requirement and On-saPSO outperforms Off-saPSO when the RSME requirement is lower. It indicates that Off-saPSO convergences faster at the front stage of optimisation but On-saPSO catches up at the latter stage of optimisation (even though Off-saPSO uses less time in reaching 0.5% RSME requirement, it has 4 seconds difference on average and their performance is almost the same in terms of the consumed time). The main reason is the change of the accuracy of surrogate model in current search area. Fig. 3 shows the change of accuracy of the classification model in Off-saPSO and On-saPSO during the optimisation process. The accuracy is calculated on every 90 sampled individuals (same as the population size) during the optimisation process. The blue curve and red curve denote the offline and online frameworks, respectively. The offline model maintains a high accuracy since the beginning of optimisation stage, therefore it uses less computational resource for an easily reached RSME requirement. Therefore, Off-saPSO easily traps in premature convergence. However, the online model's relatively low accuracy increases its search diversity and its increasing accuracy helps it converge to a better result when it has more computational resources.

On the other hand, according to our prior knowledge of this aero-engine calibration problem [15], the neighbour area around the global optimum is the feasible area. Therefore, when the algorithm converges at the latter stage of optimisation, almost all individuals are feasible, making the surrogate model's efficacy decrease. As a result, when On-saPSO finds a high-quality individual, which helps the algorithm quickly converge to the better area, the surrogate model's accuracy will not influence the optimisation a lot. The RMSE requirement of 0.5% is too difficult for these algorithms within such limited number of actual simulations to reach.

TABLE III: Average and standard deviation of running time (*second*) for reaching the different RSME requirements. The symbol in Off-saPSO/On-saPSO represents the Off-saPSO/On-saPSO significantly win (>) saPSO or no significant difference (≈).

RMSE	saPSO	Off-saPSO	On-saPSO
3.0%	48.88±45.98	22.4±22.12 (>)	42.84±40.60 (≈)
2.5%	87.00±58.03	45.16±32.70 (>)	68.84±51.65 (≈)
2.0%	154.60±53.30	105.08±61.95 (>)	127.00±77.64 (≈)
1.5%	307.72±118.68	225.16±131.13 (>)	209.16±141.91 (>)
1.0%	619.32±216.37	507.60±246.31 (≈)	459.92±182.32 (>)
0.5%	1540.52±788.80	1244.08±340.65 (≈)	1248.32±555.11 (≈)

b) ***Time influence analysis****:* The proposed frameworks aim at reducing computational time for the constrained expensive problem. The construction of the surrogate model is also a computationally expensive process to some extent. Therefore, we analysed the time used in the experiments and reported the average total time used by saPSO, Off-saPSO and On-saPSO in Table I and Table III. Both frameworks reduce the total running time for saPSO and the offline framework reduces more time than the online one. The proposed frameworks eliminate the potentially infeasible individuals, whose evaluation time is much more than feasible individuals in this problem. Assuming that saPSO evaluates p percents of

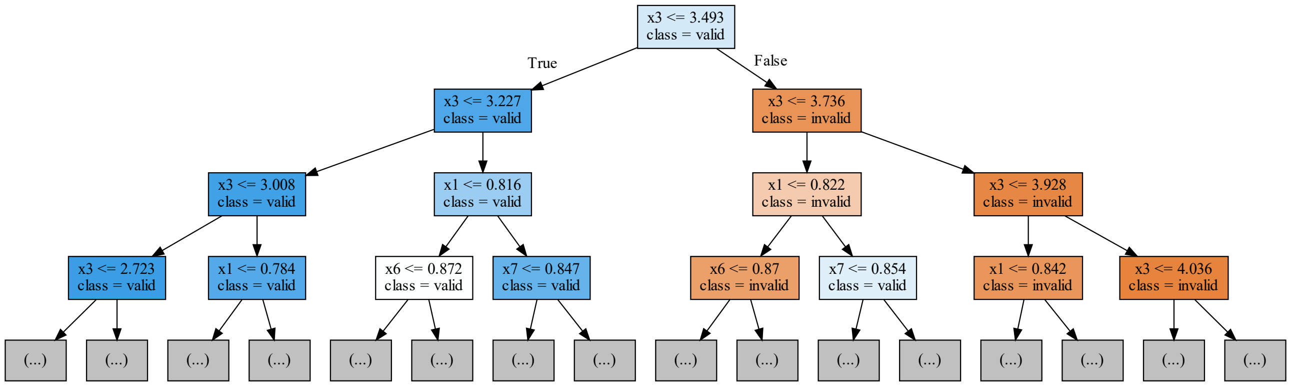

Fig. 4: A decision tree for engine-calibration problem.

feasible individuals among all the evaluated individuals and the surrogate model in Off-saPSO and On-saPSO eliminate all infeasible individuals during optimisation, the total time used by each algorithm excluding the time consumed by other operations can be calculated as:

$$\begin{aligned} T_{saPSO} &= 100d \times p \times t_f + 100d \times (1-p)t_{if} \\ &= 100d \times [t_f + (1-p)(t_{if} - t_f)] \\ &= 100d \times t_f + 100d \times (1-p)(t_{if} - t_f), \\ T_{off-saPSO} &= 100d \times t_f, \\ T_{on-saPSO} &= 100d \times t_f + T_m, \end{aligned}$$

where t_f and t_{if} are the average time for evaluating a feasible individual and an infeasible one, respectively, and $t_{if} > t_f$ in our problem. T_m denotes the time used for constructing surrogate models in the online framework. Therefore, the Off-saPSO should be the algorithm using the shortest time, which is the same as our experimental results. For T_m, we recorded that building a neural network surrogate model only consumed 0.1680 seconds on average. The average ratio of time for constructing surrogate models accounted for 2.78% of total running time in our experiments. Therefore, both offline and online frameworks improved the algorithm's efficiency when solving this expensive constrained problem.

C. *Surrogate-assisted problem analysis*

Furthermore, in order to analyse the influence of each input parameter on the aero-engine calibration problem, which is a black-box problem for us, we re-trained a classification model using the data generated during the optimisation process. As the neural network is hard to analyse each input parameter's influence, we trained a decision tree as visualised in Fig. 4. We only draw three depths of the whole tree due to the limited number of pages. According to Fig. 4, the third input parameter, i.e. x_3, is the most crucial parameter for determining the feasibility of an individual. Then, x_1, x_6 and x_7 also play important roles in determining an individual to be feasible or not. The different importance of parameters detected in our experiments could potentially help engineers calibrate a similar engine quickly in the future. Details of this aero-engine and those input parameters are described in [15].

V. Conclusion

In this paper, we modelled a real aero-engine calibration problem as an expensive optimisation problem with hidden constraints aiming to adjust a series of parameters to ensure the performance of an engine. Most of the existing approaches cannot handle the hidden constraints, which only give a binary value to indicate the feasibility. As the evaluation process is time-consuming, we trained a classification-based surrogate model to predict a solution's feasibility. We proposed offline and online frameworks combining the surrogate model with optimisation algorithms to increase algorithms' efficiency for this calibration problem. The offline framework constructed the prediction model before the optimisation process without further updates. The online framework updated the surrogate model during the optimisation process once the current model's prediction for a solution was different from its actual feasibility. In the empirical studies, we employed a self-adaptive particle swarm optimisation algorithm to generate two algorithm instances to solve the calibration problem. The results showed that the surrogate assisted algorithms significantly outperformed the original algorithm in terms of the obtained best solution and the time used for optimisation. Besides, the surrogate model also helped us analyse the different importance of parameters' influence on the feasibility.

In the future, it is valuable to reduce the problem's dimensionality according to the analysis of the classification model so that the optimisation algorithms can benefit from a lower-dimensional problem. The online and offline framework can also be used in an algorithm portfolio to reduce the optimisation risk [24]. Furthermore, we will compare our frameworks with other approaches for aero-engine calibration problem, and it is also worthy to investigate how to transfer the knowledge learnt on calibrated aero-engine models to calibrate new engine models more efficiently.

References

[1] H. Ma, Z. Li, M. Tayarani, G. Lu, H. Xu, and X. Yao, “Model-based computational intelligence multi-objective optimization for gasoline direct injection engine calibration,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 233, no. 6, pp. 1391–1402, 2019.

[2] A. E. Eiben, J. E. Smith *et al.*, *Introduction to Evolutionary Computing*. Springer, 2003, vol. 53.

[3] Y. Jin, “Surrogate-assisted evolutionary computation: Recent advances and future challenges,” *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, Jun. 2011.

[4] H. Tong, C. Huang, J. Liu, and X. Yao, “Voronoi-based efficient surrogate-assisted evolutionary algorithm for very expensive problems,” in *2019 IEEE Congress on Evolutionary Computation*. IEEE, 2019, pp. 1996–2003.

[5] H. Tong, C. Huang, L. L. Minku, and X. Yao, “Surrogate models in evolutionary single-objective optimization: A new taxonomy and experimental study,” *Information Sciences*, vol. 562, pp. 414–437, 2021.

[6] H. Dong, P. Wang, C. Fu, and B. Song, “Kriging-assisted teaching-learning-based optimization (KTLBO) to solve computationally expensive constrained problems,” *Information Sciences*, vol. 556, pp. 404–435, May 2021.

[7] Y. Wang, D.-Q. Yin, S. Yang, and G. Sun, “Global and local surrogate-assisted differential evolution for expensive constrained optimization problems with inequality constraints,” *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1642–1656, May 2019.

[8] T. P. Runarsson, “Constrained evolutionary optimization by approximate ranking and surrogate models,” in *International Conference on Parallel Problem Solving from Nature*. Springer, 2004, pp. 401–410.

[9] E. Mezura-Montes and C. A. C. Coello, “Constraint-handling in nature-inspired numerical optimization: Past, present and future,” *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, Dec. 2011.

[10] R. Farmani and J. A. Wright, “Self-adaptive fitness formulation for constrained optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 5, pp. 445–455, 2003.

[11] K. Deb, “An efficient constraint handling method for genetic algorithms,” *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2-4, pp. 311–338, 2000.

[12] T. D. Choi, O. J. Eslinger, C. T. Kelley, J. W. David, and M. Etheridge, “Optimization of automotive valve train components with implicit filtering,” *Optimization and Engineering*, vol. 1, no. 1, pp. 9–27, 2000.

[13] H. Lee, R. Gramacy, C. Linkletter, and G. Gray, “Optimization subject to hidden constraints via statistical emulation,” *Pacific Journal of Optimization*, vol. 7, no. 3, pp. 467–478, 2011.

[14] J. Müller and M. Day, “Surrogate optimization of computationally expensive black-box problems with hidden constraints,” *INFORMS Journal on Computing*, vol. 31, no. 4, pp. 689–702, 2019.

[15] J. Liu, Q. Zhang, J. Pei, H. Tong, X. Feng, and F. Wu, “fSDE: efficient evolutionary optimisation for many-objective aero-engine calibration,” *Complex & Intelligent Systems*, pp. 1–17, 2021.

[16] M.-H. Tayarani-N, X. Yao, and H. Xu, “Meta-heuristic algorithms in car engine design: A literature survey,” *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 609–629, 2014.

[17] R. J. Lygoe, M. Cary, and P. J. Fleming, “A many-objective optimisation decision-making process applied to automotive diesel engine calibration,” in *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, 2010, pp. 638–646.

[18] H. Langouët, L. Métivier, D. Sinoquet, and Q.-H. Tran, “Engine calibration: multi-objective constrained optimization of engine maps,” *Optimization and Engineering*, vol. 12, no. 3, pp. 407–424, 2011.

[19] K. Rezapour, “Exergy based SI engine model optimisation: exergy based simulation and modelling of bi-fuel si engine for optimisation of equivalence ratio and ignition time using artificial neural network (ANN) emulation and particle swarm optimisation (PSO),” Ph.D. dissertation, University of Bradford, 2012.

[20] H. Ma, Z. Li, M. Tayarani, G. Lu, H. Xu, and X. Yao, “Computational intelligence nonmodel-based calibration approach for internal combustion engines,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 140, no. 4, 2018.

[21] K. I. Wong, P. K. Wong, C. S. Cheung, and C. M. Vong, “Modeling and optimization of biodiesel engine performance using advanced machine learning methods,” *Energy*, vol. 55, pp. 519–528, 2013.

[22] E. Samadani, A. H. Shamekhi, M. H. Behroozi, and R. Chini, “A method for pre-calibration of DI diesel engine emissions and performance using neural network and multi-objective genetic algorithm,” *Iranian Journal of Chemistry and Chemical Engineering*, vol. 28, no. 4, pp. 61–70, 2009.

[23] C. Audet and J. E. Dennis, “Mesh adaptive direct search algorithms for constrained optimization,” *SIAM Journal on Optimization*, vol. 17, no. 1, pp. 188–217, Jan. 2006.

[24] H. Tong, J. Liu, and X. Yao, “Algorithm portfolio for individual-based surrogate-assisted evolutionary algorithms,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 943–950.