A Novel Active Optimization Approach for Rapid and Efficient Design Space Exploration Using Ensemble Machine Learning

Opeoluwa Owoyele[1]
Energy Systems Division,
Argonne National Laboratory,
9700 S. Cass Avenue,
Lemont, IL 60439
e-mail: oowoyele@anl.gov

Pinaki Pal
Energy Systems Division,
Argonne National Laboratory,
9700 S. Cass Avenue,
Lemont, IL 60439
e-mail: pal@anl.gov

In this work, a novel design optimization technique based on active learning, which involves dynamic exploration and exploitation of the design space of interest using an ensemble of machine learning algorithms, is presented. In this approach, a hybrid methodology incorporating an explorative weak learner (regularized basis function model) that fits high-level information about the response surface and an exploitative strong learner (based on committee machine) that fits finer details around promising regions identified by the weak learner is employed. For each design iteration, an aristocratic approach is used to select a set of nominees, where points that meet a threshold merit value as predicted by the weak learner are selected for evaluation. In addition to these points, the global optimum as predicted by the strong learner is also evaluated to enable rapid convergence to the actual global optimum once the most promising region has been identified by the optimizer. This methodology is first tested by applying it to the optimization of a two-dimensional multi-modal surface and, subsequently, to a complex internal combustion (IC) engine combustion optimization case with nine control parameters related to fuel injection, initial thermodynamic conditions, and in-cylinder flow. It is found that the new approach significantly lowers the number of function evaluations that are needed to reach the optimum design configuration (by up to 80%) when compared to conventional optimization techniques, such as particle swarm and genetic algorithm-based optimization techniques.
[DOI: 10.1115/1.4049178]

Keywords: engine design optimization, surrogate-based optimization, ensemble machine learning, active learning, energy systems analysis, fuel combustion

Introduction

With ever-rising consumer demand for better fuel economy and tightening of emissions regulations, the automotive industry has been pushing toward the exploration of novel design and combustion strategies to improve the efficiency of internal combustion (IC) engines. However, engine performance is highly sensitive to a large number of parameters associated with the geometry, fuel type/system, air system, operating conditions, and their complex interactions [1]. Fuel-engine co-optimization within such a huge design space solely with experiments becomes a very challenging task. In this context, computational fluid dynamics (CFD) modeling and numerical optimization techniques can play a significant role in cutting down both cost and time-to-design.

The two most commonly used numerical approaches for engine design optimization are design of experiments (DOE) [2–5] and genetic algorithm (GA) [6–14]. In a DOE, the entire design space is explored by running a large number of CFD simulations to fill the DOE hypervolume using a chosen space-filling technique. A response surface, usually based on linear regression, is then fit on the simulation data and used for design optimization. These methods are, however, not capable of capturing nonlinear interactions between various design parameters without incorporating tuning techniques, such as the addition of cross-terms and/or higher-order terms. This can often lead to a loss of accuracy. On the other hand, in a GA-based optimization, a CFD simulation is used as an objective function and a combination of outputs from the simulations forms the merit value to be optimized. CFD simulations are performed sequentially in batches known as generations, with each generation having a prescribed number of individuals/samples to evaluate (i.e., CFD runs to perform). Using a stochastic approach where the population is varied using a set of genetic operations often leads to much better optimum solutions than DOE. However, the convergence of GA typically takes a long time (over 2 or 3 months), even for an optimization problem with a moderate number ($\sim$5–10) of input or design features.

As an alternative, data-driven approaches that employ machine learning (ML) have been employed in many studies as a means to improve the performance and runtime of numerical design optimization [15–20]. ML models can be considered as fast-running surrogates for the more time-consuming methods (experiments or CFD simulations) used to generate the data for their training. Moreover, depending on the complexity of the chosen ML model, it is possible to capture nonlinear relationships including interaction among the design parameters. In the past, ML models have been used in real-time control of engines [21–26]. There have also been significant developments in using artificial neural networks (ANNs) in understanding and solving combustion problems [27,28]. More recently, Moiz et al. [15] developed a novel ML-GA workflow in which an ensemble machine learning technique known as Super Learner [29] was employed to generate the surrogate model for CFD, which was then coupled with a Malschains [30] GA to rapidly optimize a heavy-duty engine operating with a gasoline-like fuel. High-performance computing resources at Argonne National Laboratory were leveraged to enable quick

[1]Corresponding author.

Contributed by the Internal Combustion Engine Division of ASME for publication in the Journal of Energy Resources Technology. Manuscript received July 16, 2020; final manuscript received September 1, 2020; published online December 16, 2020. Assoc. Editor: Samer F. Ahmed.

generation of the data for ML training by way of running a large number of CFD simulations concurrently. ML-GA led to design optimization with a turnaround time which was 95% lower than that of the traditional CFD-GA approach. Kavuri and Kokjohn [16] introduced a technique for accelerating micro-genetic algorithm (μGA) optimization of IC engines by replacing the CFD model with a Gaussian process regression model at locations within design space where the estimated error from using the surrogate ML model was low. Using this method, they were able to reduce the total runtime by about 62%, only at a minor degradation in the optimum merit value achieved. However, relaxing the error criterion was found to impact the model predictions negatively.

A common thread among most of the above ML-based numerical optimization studies was the use of more expensive CFD simulations to generate data for training the ML models. However, in such scenarios, the amount of training data required is not known *a priori* and may vary depending on the dimensionality of the design space. Therefore, sampling points uniformly from the whole design space lead to redundant inclusion of points in regions far away from the global optimum and significantly larger number of CFD simulations than actually needed, thereby increasing the overall computational cost. A CFD-GA approach would circumvent this issue, but it suffers from slow convergence as noted earlier, which may again result in a sub-optimally large number of CFD evaluations. Given these shortcomings, adaptive surrogate-assisted optimization [31] is a promising way to reduce the number of function evaluations for problems that involve expensive simulations. In this approach, the surrogate model being optimized in place of the CFD simulations is updated after each design iteration to incorporate the latest information about the design surface. In addition, the surrogate models assist in determining the design parameters for the next iteration of CFD simulations.

In the present work, a novel numerical methodology is presented that improves upon current widely used optimization techniques by markedly reducing the number of function evaluations required to reach the global optimum, thereby minimizing runtime and computational expense of the design process. This is achieved by employing a deterministic approach, where adaptive machine learning models that make use of prior knowledge of the design space are used to determine which regions are more likely to contain the global optimum and consequently guide computational resources toward those regions. This leads to a dense sampling close to the global optimum and sparse sampling in less promising regions. Due to the two-way coupling between the ML model and the CFD solver, this approach is referred to as Active Optimization (ActivO). While there have been previous studies in the broader field of optimization that explore adaptive surrogate-based optimization [32–35], the optimization algorithm presented in the work is a novel approach that has not been introduced before, to the best of the authors' knowledge. The remainder of the paper is organized as follows. The next section provides a description of the ActivO algorithm. Then, details of the two test cases to which ActivO is applied are presented. Subsequently, results from the optimization studies are discussed, and the performance of ActivO relative to conventional GA and particle swarm optimization (PSO) approaches is assessed. Finally, the paper concludes with a summary of the main findings and directions for future work.

Active Optimization Methodology

Basic Algorithm. The proposed algorithm, ActivO, involves the use of machine learning models to fit the design space based on previously sampled points. These models are, in turn, used to predict which points to sample next, i.e., design parameters for the next set of function evaluations. The goal is to spend more time and resources deciding the next point to sample so that the global optimum can be reached in fewer function evaluations. This is done by a hybrid process that includes a constrained random search based on a weak surrogate model and global optimization of the surface obtained from a second, stronger surrogate model. Each design iteration has two phases: a guided exploration phase and a focused exploitation phase, each of them using different machine learning models that complement each other. For guided exploration, the weak learner is used to produce an underfitted representation that does not accurately reflect every minute detail of the actual response surface but provides a general measure of the objective/merit function of the region in the neighborhood of a point. It acts as a guide to regions of space where random sampling is to be performed. During the focused exploitation stage, a stronger learner is trained on the available data and a global optimization scheme is used to identify the optimum of the predicted surface. This is done to exploit the region identified by the weak learner. Thus, the exploitation focuses on providing highly localized information about the surface close to the optimum, while the exploration phase focuses on the broader representation, randomly exploring the region in which the global optimum is likely to lie. The ActivO approach is motivated by the reasoning that it is not required to have an accurate representation of the surface in the entire design space. Rather, the important information in question is *where* the global optimum is located. Therefore, the global optimum can be reached even if the surrogate model is inaccurate in certain regions due to the inadequacy of data samples, as long as there are sufficient points in regions close to the optimum design parameters so that a qualitatively accurate representation can be obtained in these regions.

In this work, a set of third-order polynomial features obtained from the original variables were used as the basis set for the basis function model (BFM). This model was trained using a Ridge regression method. Since the polynomial is only third order, it is limited in the amount of surface detail it can fit and acts as the weak learner. On the other hand, an ANN was used as a strong learner. It should be noted that while these machine learning algorithms were used here, other methods could also be used in their place. For example, a support vector machine could be used as the weak learner, and tree-based algorithms such as random forest could be used in the place of ANN as the strong learner.

As mentioned earlier, a random search is performed in regions where the optimum is likely to be located, based on the latest information about the response surface. This makes use of an aristocratic strategy, where a large pool of random nominee points are generated in the design space, and only those that meet a minimum merit value, λ, are chosen. While ground-truth information about the merit values of nominees is not available, the best performing points are identified based on the current surface predicted by the BFM. The merit values of the nominees are computed using the BFM and then sorted in descending order. Among these, the top k-percent are considered, from which a cut-off criterion, λ_k, that represents the merit value required to be in the (100-k)th percentile of nominees is defined. The input parameters used in the exploration phase of the next design iteration are constrained to come from the region demarcated by $\lambda > \lambda_k$. It should be noted that this step is relatively inexpensive since the evaluation of the merit functions, in this case, is based on the BFM. Furthermore, while k is a parameter that may be arbitrarily chosen, it was found that as long as k is chosen within reasonable bounds of 5–20, ActivO maintained a good performance. In this work, k was chosen as 15 for demonstrative purposes, and so random sampling was constrained to regions where the merit value was projected to be above the 85th percentile. This process of randomly sampling points within the promising regions of the design space helps the optimizer to escape local optima.

The scikit-learn library [36] was used to build the BFM, while TensorFlow [37] was used to develop and train the ANN employed in this study. To summarize, the basic steps involved in the ActivO algorithm are as follows:

(1) Start with a random set of N function evaluations to generate an initial set of points.
(2) Generate a trained regularized basis function model using the data.

(3) Generate an ANN model and fit all the data points obtained thus far.
(4) Randomly sample $N-1$ points that the BFM predicts to be in the 85th percentile within the design space.
(5) Find the optimum of the surface predicted by the trained ANN using a global optimization scheme. Add this optimum parameter set to the $N-1$ points from Step (4) to get N points to be evaluated.
(6) Perform function evaluations for the N points from Step (5).
(7) Add the new solutions to the database and repeat Steps (2)–(6) until the solver converges or reaches the maximum number of design iterations.

These steps are also depicted in the form of a flowchart in Fig. 1.

Refinements. A number of refinements were added to the basic algorithm described in the previous section to make it more robust and efficient. The first refinement was related to the operation of the BFM used to find promising regions in the sample space. The basic algorithm uses a pseudorandom number generator to generate data for the initial sample points and the subsequent constrained exploration. During preliminary tests of ActivO, it was noticed that this was inefficient since very often, the pseudorandom number generator would generate random points near points for which expensive function evaluations had already been performed. To counter this effect, for each iteration, a large number of points, $Np \gg N$, are generated and points that are farthest from the points already sampled are picked. Thus, another stage of nominee generation and selection is introduced, where a large number of nominees that have predicted merit values exceeding λ_k go through the second stage of selection. For each nominee at this stage, the sum of its Euclidean distances from all the points for which CFD data is already available, $d_{\min}$, is calculated. This distance needs to be maximized. Thus, the winning nominee is defined as

$$x_{\text{new}} = \text{argmax}(d_{\min}) \tag{1}$$

This process is repeated until $N-1$ points with predicted merit values greater than λ_k are obtained. This ensures that the optimizer is not needlessly duplicating information regarding the merit function surface and promotes a well-balanced exploration of the constrained design space. This also helps the optimizer in escaping local optima. In situations where the region defined by λ_k suddenly expands due to new information about the design space, this function will tend to explore the extreme portions of the new boundary, thus making the exploration more effective.

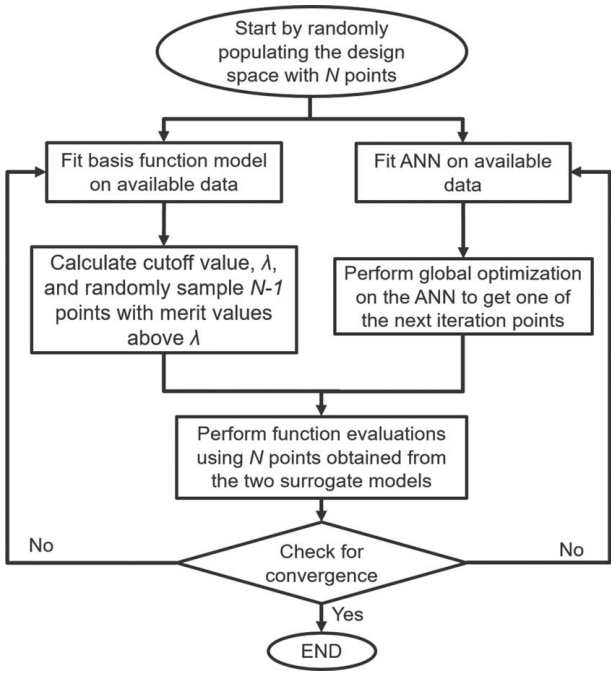

Fig. 1 Flowchart demonstrating the steps involved in the ActivO algorithm

A second refinement was needed to address the problem of sparsity in non-promising regions. The high predictive capability of ANNs carries the caveat of making them prone to overfitting. During the initial tests conducted, it was found that the ANN was prone to overfitting in sparse, non-promising regions, thus producing a number of false optima for the predicted merit surface. Optimizing this overfit surface would sometimes lead to a global optimum in these non-promising regions of the surrogate surface. In such cases, the optimum predicted from the ANN surface was unrelated to the actual surface and occurred as a result of the sparsity of samples in the region of this false optimum. Such a mechanism leads to wasted resources, where some design iterations are wasted running expensive function evaluations in regions that have poor merit values. Related to this is the problem of hyperparameter selection: What neural network architecture should be used (i.e., the number of layers and neurons per layer)? Using a model that is too small will make the learner too weak and impair the model's ability to adequately fit the surface. Using a model that is too strong will lead to overfitting in sparsely sampled regions, as indicated previously. It should be noted that overfitting is not a problem, as long as it is kept below a level so that it does not lead to a false global optimum. In this work, these problems were tackled by using a committee machine. In the committee machine approach used here, the risk of overfitting is mitigated by training multiple networks with different initial weights in parallel and combining the outputs from these individual predictors to get the final prediction. The prediction of a committee machine made up of M networks is given by

$$\varphi(x) = \sum_{i=1}^{M} \varphi_i / M \tag{2}$$

In the above equation, φ_i is the prediction from ANN i, and φ is the overall prediction. The reasoning behind using a committee machine is that overfitting, due to its nature, is not likely to be repeatable since it is non-physical. If the data are fit using a network with different initial conditions or optimizer parameters, overfitting will often occur at a different region or may occur at significantly lower levels. Thus, it is expected that the prediction of the networks will have a higher standard deviation in sparse regions, while they agree at regions of relative certainty where the data are sufficiently dense. Another effect is that the algorithm is not sensitive to the size of the network used. The user can set an arbitrarily large network for each problem since the use of multiple networks can help mitigate the problem of overfitting. In this work, five networks were used. A lower number would increase the chances of false global optima, and based on initial tests of the algorithm, using at least three networks is recommended. From an accuracy point of view, there is no limit to the number of networks that can be used; the only limiting factor is that a large number of networks would require more computational resources and/or take a longer time to train.

Test Cases

In this study, the capability of the ActivO algorithm was assessed for two different optimization problems. These test cases are described below.

Two-Dimensional Multi-Modal Function. ActivO was first applied to find the input parameters that correspond to the maximum objective function for a two-dimensional (2D) surface. The test case chosen here is a challenging multi-modal problem for which the global optimum is known [38]. The function has 25 peaks that can potentially act as local maxima and trap an optimizer.

This function is described by

$$f_x = \exp\left(\frac{-4\log 2(x-0.0667)^2}{0.64}\right)\sin(5.1\pi x + 0.5)^6 \quad (3)$$

$$f_y = \exp\left(\frac{-4\log 2(y-0.0667)^2}{0.64}\right)\sin(5.1\pi y + 0.5)^6 \quad (4)$$

$$z = f_x f_y \quad (5)$$

In the above equations, x and y are parameters of the input space, and z is the merit function to be maximized. In this case, the maximum function can be analytically derived to be $z = 1$, located at $x = y \approx 0.0668$. A surface plot of this function is shown in Fig. 2.

Internal Combustion Engine Combustion Optimization. As a second validation study, ActivO was applied to the engine combustion optimization case from Moiz et al. [15], where the goal was to minimize fuel consumption by a heavy-duty engine operating on a gasoline-like fuel, while satisfying the constraints on emissions (NO_x and soot in g/kWh), peak cylinder pressure (PMAX in bar), and maximum pressure rise rate (MPRR in bar/CA). The nine input parameters included in the design space along with their considered ranges are listed in Table 1. A merit function was defined to quantify the overall performance of a particular engine design, as shown in Eq. (6). An ML surrogate model for the merit function response surface was developed based on the Super Learner [29] approach, which was trained on 2048 engine CFD simulations [15]. This surrogate model was then coupled with a GA to optimize the input parameters within the nine-dimensional design space so that the merit value was maximized. For more details on this optimization study, the readers are referred to Moiz et al. [15].

The merit function is given by

$$\text{Merit} = 100 * \left\{\frac{160}{\text{ISFC}} - 100 * f(\text{PMAX}) - 10 * f(\text{MPRR}) - f(\text{SOOT}) - f(\text{NO}_x)\right\} \quad (6)$$

where,

$$f(\text{PMAX}) = \left\{\begin{array}{c}\dfrac{\text{PMAX}}{220} - 1, \text{ if PMAX} > 220 \\ 0, \text{ if PMAX} \le 220\end{array}\right\} \quad (7)$$

$$f(\text{MPRR}) = \left\{\begin{array}{c}\dfrac{\text{MPRR}}{15} - 1, \text{ if MPRR} > 15 \\ 0, \text{ if MPRR} \le 15\end{array}\right\} \quad (8)$$

$$f(\text{SOOT}) = \left\{\begin{array}{c}\dfrac{\text{SOOT}}{0.0268} - 1, \text{ if SOOT} > 0.0268 \\ 0, \text{ if SOOT} \le 0.0268\end{array}\right\} \quad (9)$$

$$f(\text{NO}_x) = \left\{\begin{array}{c}\dfrac{\text{NO}_x}{1.34} - 1, \text{ if NO}_x > 1.34 \\ 0, \text{ if NO}_x \le 1.34\end{array}\right\} \quad (10)$$

One possible approach to test ActivO could be to directly couple it with the CFD model. However, in the present study, the Super Learner ML surrogate model from Moiz et al. [15] was used instead. This was primarily done to perform the proof-of-concept study of ActivO without incurring any computational cost due to CFD. In addition, using the surrogate model allowed for running multiple trials of ActivO and presenting meaningful projections of its performance. This is very important since, for different trials, optimizers may converge at very different rates when tested on the same problem. Some trials may fortuitously converge much quicker than the optimizer's average performance, while some may uncharacteristically take very long to converge. Therefore, in testing a new optimizer, it is necessary to be able to run multiple trials to have a relevant metric to compare with existing optimizers.

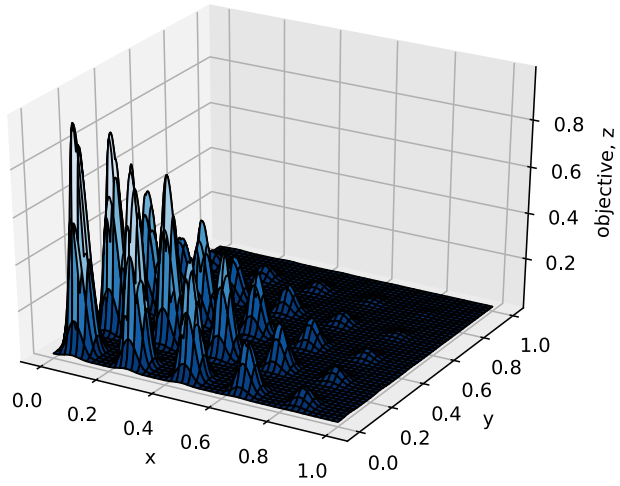

Fig. 2 Surface plot of the 2D multi-modal function

Table 1 Input parameters for IC engine combustion optimization study [15]

Parameter	Description	Min	Max	Units
nNoz	Number of nozzle holes	8	10	–
TNA	Total nozzle area	1	1.3	–
Pinj	Injection pressure	1400	1800	bar
SOI	Start of injection timing	−11	−7	deg CA ATDC
NozzleAngle	Nozzle half-inclusion angle	72.5	83.0	deg
EGR	EGR fraction	0.35	0.5	–
Tivc	IVC temperature	323	373	K
Pivc	IVC pressure	2.0	2.3	bar
SR	Swirl ratio	−2.4	−1	–

Results and Discussion

First Validation Study: Two-Dimensional Multi-Modal Merit Surface. In this section, the results of the optimization for the first test case are presented. In Fig. 3, the maximum objective function evolution versus the number of objective function evaluations is shown for ActivO and compared with μGA [39] and PSO

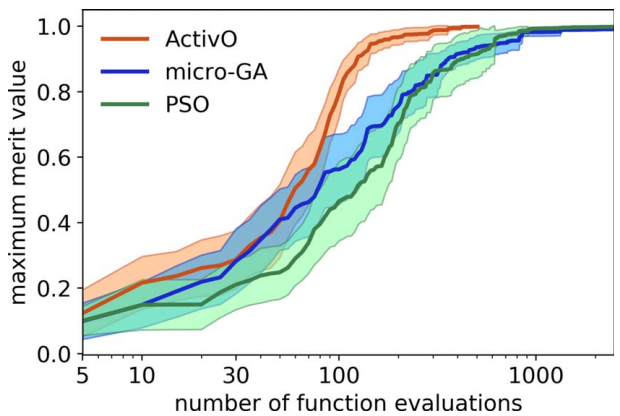

Fig. 3 The evolution of maximum objective function versus the number of function evaluations for ActivO, μGA, and PSO

Table 2 Performance of ActivO compared to μGA and PSO

	Number of generations		
Maximum merit value	ActivO	GA	PSO
0.99	60	396	183
0.95	31	144	124
0.9	25	72	81
0.85	22	59	60

[40,41]. Each design iteration consisted of five function evaluations in parallel, and for this problem, 25 trials were run for each optimization algorithm. The solid lines in Fig. 3 represent the average value of the maximum objective function over the 25 trials, while the shaded regions around the solid lines depict the objective function values represented by a 95% confidence interval. For μGA, micro-convergence was assumed to be reached whenever the variation in the chromosomes of a given generation became less than 5%. For PSO, the inertia weight, which determines the balance between exploration and exploitation, was set to 1.0 for this problem to allow for more exploration. In particular, it was found that the performance of PSO was highly sensitive to the inertia weight: for lower values, the algorithm got trapped in local maxima more often than not.

The results show that on average, ActivO performs much better than μGA and PSO in terms of reaching the global optimum faster. The maximum objective function value at the local maxima surrounding the global optimum was $\sim$0.84, so it was assumed that the optimizer had reached very close to the actual global optimum when it crossed this threshold. Considering this cut-off, it can be seen from Table 2 that it takes around 60 iterations for the μGA and PSO to correctly find the region where the global optimum is located, while ActivO is able to identify this region much faster, in about 22 iterations. More pronounced, however, is the difference in the number of iterations it takes to reach the actual global optimum once the optimizer is in the vicinity of the global optimum. For this problem, it was assumed that the global optimum had been found when the merit value reached 0.99. Evidently, ActivO takes a total of 60 design iterations on average (300 function evaluations) to reach a maximum merit value of 0.99, while the PSO takes 183 iterations and μGA takes 396 iterations (915 and 1980 function evaluations, respectively). While it appears that PSO performs much better than μGA, in practice this may not be the case for PSO algorithms using a constant inertia weight, since the algorithm is sensitive to the inertia weight and the best value may not be known a priori.

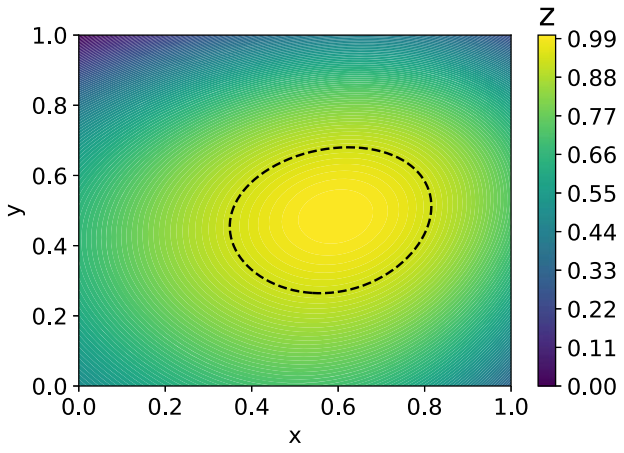

Fig. 4 Contour plots obtained from BFM prediction at initialization. The dotted black line denotes the region demarcated by λ_k.

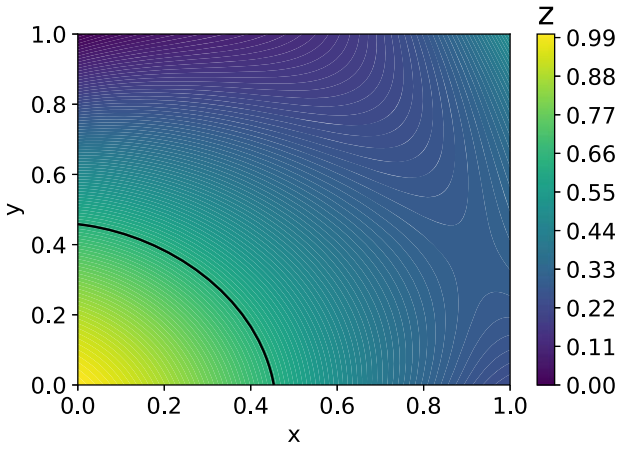

Fig. 5 Contour plots obtained from BFM prediction after the global optimum has been found. The solid black line denotes the region demarcated by λ_k.

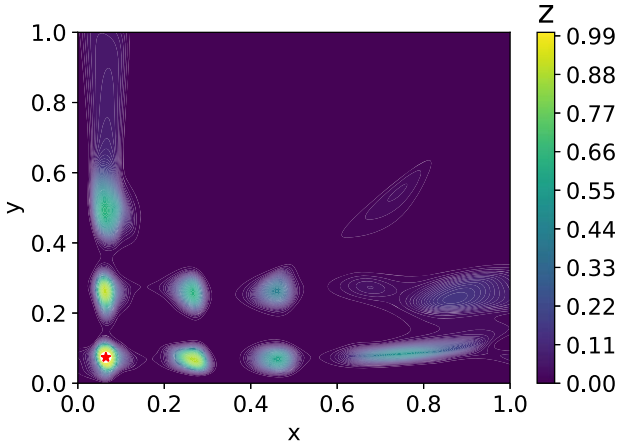

Fig. 6 Contour plots obtained from committee machine prediction after the optimum has been determined. The star marker indicates the final optimum as found for this surface.

It should be noted that the numbers shown in Table 2 represent maximum merit values averaged over multiple trials, and individual trials may sometimes reach the optimum in a fewer number of function evaluations. However, in general, ActivO converges significantly faster than the PSO and μGA, and this difference is compounded by the fact that there were some cases where PSO and μGA took an unusually large number of function evaluations to reach the

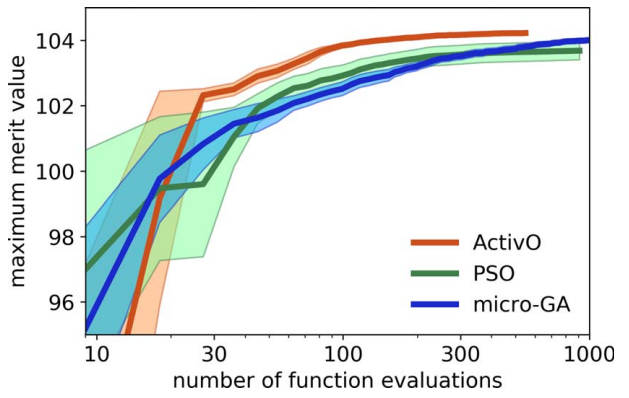

Fig. 7 The evolution of maximum objective function versus the number of function evaluations for ActivO, μGA, and PSO

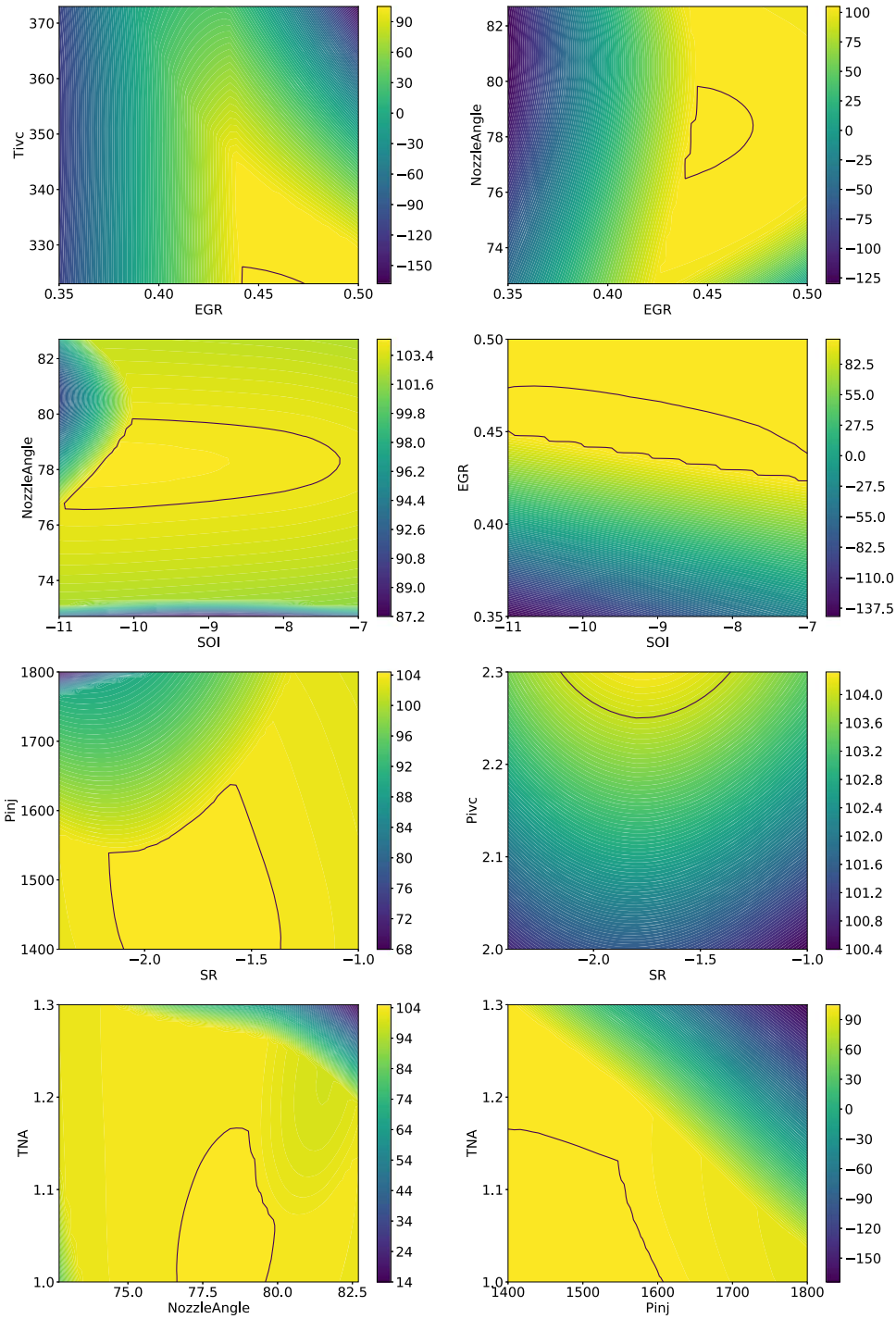

Fig. 8 2D contour plots at optimum obtained using ActivO. Isolines depict the regions where merit value is greater than 104.

global optimum. Furthermore, it is observed that for a given problem, once ActivO is in the vicinity of the global optimum, the solution gets refined fairly quickly and reaches the 0.99 mark, as compared to μGA and PSO. The number of iterations it takes to reach 0.99 is 2.7 times the number it takes to reach 0.85, while this number is about 6.7 for μGA and 3 for PSO. This is because the ActivO technique involves fitting a committee machine on data which has a lot of information about the topology of the surface close to the optimum, due to the constrained random sampling.

Contour plots of the objective function predicted by the BFM and the committee machine are shown in Figs. 4–6. Figure 4 shows the surface predicted by the BFM at the initial design iteration, while Fig. 5 shows the surface predicted by the BFM at the end of the optimization process after the global optimum has been successfully found. The black lines in Figs. 4 and 5 are isolines showing the boundary that separates the promising region from the other regions. The BFM predicts that the points within this region lie in the 85th percentile of all points in the input design space. The boundary moves as more information is gained about the actual surface, from the dotted line as shown in Fig. 4 to the solid line in Fig. 5. It can be seen that as the optimization progresses, the BFM is able to identify that points sampled in the bottom left corner of the contour plot are better than other areas. The random

sampling is thus concentrated in this region, and the committee machine (Fig. 6) is able to predict the topology in this region fairly accurately. However, a comparison of Fig. 6 and Fig. 2 shows that the surface predicted by the committee machine is not accurate everywhere. This is because, in order to find the global optimum, a surrogate model that accurately fits the entire surface is not necessary. At a minimum, what is needed is a surface that is accurate enough to not lead to a false global optimum outside the region defined by λ_k, and a qualitatively accurate surface within the region defined by λ_k.

Second Validation Study: Internal Combustion Engine Combustion Optimization. In this section, the results obtained from applying ActivO to the optimization of the engine surrogate model are discussed and compared with those of PSO and μGA. In Fig. 7, the evolution of the maximum merit value versus the number of evaluations for the three optimization algorithms is shown. The solid lines in the plots represent the average results from 25 trials, whereas the shaded regions around the plots represent a 95% confidence interval. For this problem, the inertia weight for PSO was set to 0.8, a different value from what was used for the 2D multi-modal test problem (as shown in the preceding subsection) because the algorithm performed badly using the previous value of 1.0. Thus, in addition to the sensitivity of the performance of PSO to the inertia weight for a given problem, the performance is also sensitive across different problems. In contrast, for ActivO, the same parameters were used for both problems and the results obtained were superior to those obtained using PSO and μGA for these constant optimizer parameters. For this case, nine evaluations were used for each design iteration in line with the μGA used in the previous study by Moiz et al. [15].

As shown in Fig. 7, ActivO requires fewer iterations to reach a given maximum merit value and is also able to find a higher merit value than the other two algorithms for a given number of function evaluations. For testing purposes, a merit value of 104.0 (close to a global optimum of $\sim$104.32, obtained from extensive searching of the design space) was chosen to compare the three methods. It takes only 16 iterations for the mean maximum merit value to cross the 104.0 threshold in the case of ActivO, while it takes μGA about 80 iterations to cross this threshold and PSO never reaches 104.0 within the 100 iterations it was run for. The maximum average merit value it reaches is 103.7. Thus, the ActivO approach leads to five times speedup relative to μGA in reaching a merit value of 104.0 and a much better optimum than is found by PSO. It must be noted that for certain individual runs, the PSO does reach 104, but this is averaged out by other poorly performing trials.

Close to the optimum design obtained using ActivO, two selected parameters were varied within their design limits at a time and contour plots of the slices thus obtained are shown in Fig. 8. The merit values represented in these slices are based on the Super Learner surface [15] being optimized. In these plots, the variables not shown are kept at their optimum design values. The black lines in the plots correspond to an isoline with a merit value of 104. Therefore, the merit value is greater than 104 in the regions within the isoline. With nine input parameters, there are 35 possible combinations of two variables, but only eight are shown here for the sake of brevity. It can be seen from the plots that the merit function is more sensitive to some variables than others. Some variables (e.g., exhaust gas recirculation (EVR), and NozzleAngle) have a narrow range over which the merit value is higher than 104, while others (SR, SOI) exhibit that for a wider range. In particular, it is observed that the swirl ratio (SR) can range from about -2.2 to -1.4 (over 50% of its specified design range) and still have merit greater than 104.0 at the slice being considered, while a variable like temperature at intake valve closing (IVC) (Tivc) generally needs to be as close to 323 K as possible. While this is a slice at a given plane of the design space, these ranges are indicative of the general topology of the surface close to the global optimum.

Table 3 Optimum design parameters obtained from 25 trials using ActivO

Design parameter	Mean optimum	Normalized standard deviation
nNoz	10	3.61×10^{-2}
TNA	1.03	6.40×10^{-2}
Pinj	1450	9.06×10^{-2}
SOI	−9.88	1.83×10^{-1}
NozzleAngle	78.12	2.51×10^{-2}
EGR	0.445	3.30×10^{-2}
Tivc	323	2.18×10^{-3}
Pivc	2.3	8.73×10^{-3}
SR	−1.77	7.81×10^{-2}

Note: Standard deviations are normalized by the range of the corresponding design parameter.

Therefore, we expect that an optimizer would converge to highly repeatable values for input variables exhibiting the topology of a sharp valley close to the optimum, while there will be some variation in the optimum values for variables that have a flatter topology around the optimum. In general, this was consistent with the results obtained from ActivO, as shown in Table 3, in that the SR, SOI, and injection pressure have the highest uncertainty in their optimum values as found using ActivO, while the temperature and pressure at IVC, nozzle half-inclusion angle, and EGR fraction have the lowest uncertainties.

Summary

In this work, a novel active learning approach, ActivO, incorporating ensemble machine learning was developed to enable efficient design space exploration and rapid design optimization. The methodology was validated for two test cases, one of which corresponded to an IC engine optimization problem. The new optimization framework lets the machine learning algorithm assist in determining the next points to sample within the design space, based on previously collected data. By concentrating computational resources on regions in which the global optimum is more likely to lie, it was able to achieve higher computational efficiency. ActivO's potential to achieve significant speedup (up to five times) compared to μGA, and to achieve significantly better merit values compared to PSO for engine optimization was demonstrated. It was also shown that ActivO had the potential to provide better refinements, once it was in the vicinity of the global optimum. The speedups recorded can be potentially significant in terms of total CPU-hours, since for engine applications, a CFD simulation can take an order of days to run.

Future work will involve coupling the proposed ActivO algorithm with a CFD solver to develop a compact workflow for engine design optimization. In addition, the current ActivO framework incorporates a cut-off (λ_k) with a constant parameter, $k=15$. This parameter could potentially be determined by some learning rate, where the optimization starts with $k=100$ (all points are sampled) and increases or decreases the value of k as the optimization process progresses, based on the topology of the surface predicted by the BFM and committee machine. This will widen the scope of the applicability of the current approach and also enable the algorithm to accelerate even faster from local optima to more promising regions. Lastly, the scope of using a Super Learner approach instead of a committee machine will also be explored.

Acknowledgment

The authors acknowledge the computing resources provided on "Blues," a high-performance computing cluster operated by the Laboratory Computing Resource Center (LCRC) at Argonne National Laboratory.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The data sets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

Funding Data

The submitted paper has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory (Argonne). This work was supported by the U.S. Department of Energy, Office of Science under contract DE-AC02-06CH11357. The research work was funded by the Department of Energy Technology Commercialization Fund (TCF) project.

References

[1] Pal, P., Probst, D., Pei, Y., Zhang, Y., Traver, M., Cleary, D., and Som, S., 2017, "Numerical Investigation of a Gasoline-Like Fuel in a Heavy-Duty Compression Ignition Engine Using Global Sensitivity Analysis," SAE Int. J. Fuels Lubr., **10**(1), pp. 56–68. .

[2] Probst, D. M., Senecal, P. K., Qian, P. Z., Xu, M. X., and Leyde, B. P., 2016, "Optimization and Uncertainty Analysis of a Diesel Engine Operating Point Using CFD," Proceedings of ASME 2016 Internal Combustion Engine Division Fall Technical Conference, Oct. 9–12, American Society of Mechanical Engineers, New York, pp. V001T006A009–V001T006A009.

[3] Pei, Y., Pal, P., Zhang, Y., Traver, M., Cleary, D., Futterer, C., Brenner, M., Probst, D., and Som, S., 2019, "CFD-Guided Combustion System Optimization of a Gasoline Range Fuel in a Heavy-Duty Compression Ignition Engine Using Automatic Piston Geometry Generation and a Supercomputer," No. 0148-7191, SAE Technical Paper.

[4] Ashok, B., Jeevanantham, A. K., Prabu, K., Shirude, P. M., Shinde, D. D., Nadgauda, N. S., and Karthick, C., 2021, "Multi-objective Optimization on Vibration and Noise Characteristics of Light Duty Biofuel Powered Engine at Idling Condition Using Response Surface Methodology," ASME J. Energy Resour. Technol., **143**(4), p. 042301.

[5] Marri, V. B., Madhu Murthy, K., and Amba Prasad Rao, G., 2020, "Optimization of Operating Parameters of an Off-Road Automotive Diesel Engine Running at Highway Drive Conditions Using Response Surface Methodology," ASME J. Energy Resour. Technol., accepted manuscript.

[6] Zhang, Q., Ogren, R. M., and Kong, S.-C., 2016, "A Comparative Study of Biodiesel Engine Performance Optimization Using Enhanced Hybrid PSO–GA and Basic GA," Appl. Energy, **165**, pp. 676–684.

[7] Broatch, A., Novella, R., Gomez-Soriano, J., Pal, P., and Som, S., 2018, "Numerical Methodology for Optimization of Compression-Ignited Engines Considering Combustion Noise Control," SAE Int. J. Engines, **11**(6), pp. 625–642.

[8] Hanson, R., Curran, S., Wagner, R., Kokjohn, S., Splitter, D., and Reitz, R., 2012, "Piston Bowl Optimization for RCCI Combustion in a Light-Duty Multi-cylinder Engine," SAE Int. J. Engines, **5**(2), pp. 286–299.

[9] Wickman, D. D., Senecal, P. K., and Reitz, R. D., 2001, "Diesel Engine Combustion Chamber Geometry Optimization Using Genetic Algorithms and Multi-dimensional Spray and Combustion Modeling," SAE Transactions, pp. 487–507

[10] Bertram, A. M., Zhang, Q., and Kong, S.-C., 2016, "A Novel Particle Swarm and Genetic Algorithm Hybrid Method for Diesel Engine Performance Optimization," Int. J. Engine Res., **17**(7), pp. 732–747.

[11] Shi, Y., and Reitz, R. D., 2010, "Optimization of a Heavy-Duty Compression–Ignition Engine Fueled With Diesel and Gasoline-Like Fuels," Fuel, **89**(11), pp. 3416–3430.

[12] Wu, Z., Rutland, C. J., and Han, Z., 2018, "Numerical Optimization of Natural Gas and Diesel Dual-Fuel Combustion for a Heavy-Duty Engine Operated at a Medium Load," Int. J. Engine Res., **19**(6), pp. 682–696.

[13] Lu, Y., Li, J., Xiong, L., and Li, B., 2020, "Simulation and Experimental Study of a Diesel Engine Based on an Electro-hydraulic FVVA System Optimization," ASME J. Energy Resour. Technol., **142**(3), p. 032204.

[14] Hamel, J. M., Allphin, D., and Elroy, J., 2018, "Multi-objective Optimization Model Development to Support Sizing Decisions for a Novel Reciprocating Steam Engine Technology," ASME J. Energy Resour. Technol., **140**(7), p. 072204.

[15] Moiz, A. A., Pal, P., Probst, D., Pei, Y., Zhang, Y., Som, S., and Kodavasal, J., 2018, "A Machine Learning-Genetic Algorithm (ML-GA) Approach for Rapid Optimization Using High-Performance Computing," SAE Int. J. Commer. Veh., **11**(5), pp. 291–306.

[16] Kavuri, C., and Kokjohn, S. L., 2018, "Exploring the Potential of Machine Learning in Reducing the Computational Time/Expense and Improving the Reliability of Engine Optimization Studies," Int. J. Engine Res., **21**(7), p. 1468087418808949.

[17] Probst, D. M., Raju, M., Senecal, P. K., Kodavasal, J., Pal, P., Som, S., Moiz, A. A., and Pei, Y., 2019, "Evaluating Optimization Strategies for Engine Simulations Using Machine Learning Emulators," ASME J. Eng. Gas Turbines Power, **141**(9), p. 091011.

[18] Badra, J., Khaled, F., Tang, M., Pei, Y., Kodavasal, J., Pal, P., Owoyele, O., Fuetterer, C., Brenner, M., and Farooq, A., 2019, "Engine Combustion System Optimization Using CFD and Machine Learning: A Methodological Approach," Proceedings of ASME 2019 Internal Combustion Engine Division Fall Technical Conference, Chicago, IL, Oct. 20–23, American Society of Mechanical Engineers Digital Collection.

[19] Badra, J., Sim, J., Pei, Y., Viollet, Y., Pal, P., Futterer, C., Brenner, M., Som, S., Farooq, A., and Chang, J., 2020, "Combustion System Optimization of a Light-Duty GCI Engine Using CFD and Machine Learning," SAE Technical Paper No. 0148-7191.

[20] Badra, J. A., Khaled, F., Tang, M., Pei, Y., Kodavasal, J., Pal, P., Owoyele, O., Fuetterer, C., Brenner, M., and Farooq, A., 2021, "Engine Combustion System Optimization Using CFD and Machine Learning: A Methodological Approach," ASME J. Energy Resour. Technol., **143**(2), p. 022306.

[21] Vaughan, A., and Bohac, S. V., 2013, "A Cycle-to-Cycle Method to Predict HCCI Combustion Phasing," Proceedings of ASME 2013 Internal Combustion Engine Division Fall Technical Conference, Oct. 13–16, American Society of Mechanical Engineers, p. V001T003A026.

[22] Validi, A., Chen, J.-Y., and Ghafourian, A., 2012, "HCCI Intelligent Rapid Modeling by Artificial Neural Network and Genetic Algorithm," J. Comb., **2012**, pp. 1–11.

[23] Vaughan, A., and Bohac, S. V., 2013, "An Extreme Learning Machine Approach to Predicting Near Chaotic HCCI Combustion Phasing in Real-Time," arXiv preprint arXiv:1310.3567.

[24] Samadani, E., Shamekhi, A. H., Behroozi, M. H., and Chini, R., 2009, "A Method for Pre-calibration of DI Diesel Engine Emissions and Performance Using Neural Network and Multi-objective Genetic Algorithm," Iran. J. Chem. Chem. Eng., **28**(4), pp. 61–70.

[25] He, Y., and Rutland, C. J., 2003, "Neural Cylinder Model and Its Transient Results," SAE Technical Paper No. 0148-7191.

[26] He, Y., and Rutland, C. J., 2002, "Modeling of a Turbocharged Di Diesel Engine Using Artificial Neural Networks," SAE Transactions, pp. 1532–1543, Paper No. 2002-01-2772.

[27] Rezaei, J., Shahbakhti, M., Bahri, B., and Aziz, A. A., 2015, "Performance Prediction of HCCI Engines With Oxygenated Fuels Using Artificial Neural Networks," Appl. Energy, **138**, pp. 460–473.

[28] Brahma, I., Rutland, C. J., Foster, D. E., and He, Y., 2005, "A New Approach to System Level Soot Modeling," SAE Technical Paper No. 0148-7191.

[29] Van der Laan, M. J., Polley, E. C., and Hubbard, A. E., 2007, "Super Learner," Stat. Appl. Genet. Mol. Biol., **6**(1).

[30] Bergmeir, C. N., Molina Cabrera, D., and Benítez Sánchez, J. M. "Memetic Algorithms With Local Search Chains in R: The Rmalschains Package," American Statistical Association.

[31] Joly, M., Sarkar, S., and Mehta, D., 2019, "Machine Learning Enabled Adaptive Optimization of a Transonic Compressor Rotor With Precompression," ASME J. Turbomach., **141**(5), p. 051011.

[32] Jones, D. R., 2001, "A Taxonomy of Global Optimization Methods Based on Response Surfaces," J. Glob. Optim., **21**(4), pp. 345–383.

[33] Holmström, K., Quttineh, N.-H., and Edvall, M. M., 2008, "An Adaptive Radial Basis Algorithm (ARBF) for Expensive Black-Box Mixed-Integer Constrained Global Optimization," Optim. Eng., **9**(4), pp. 311–339.

[34] Wang, C., Duan, Q., Gong, W., Ye, A., Di, Z., and Miao, C., 2014, "An Evaluation of Adaptive Surrogate Modeling Based Optimization With Two Benchmark Problems," Environ. Model. Softw., **60**, pp. 167–179.

[35] Müller, J., 2016, "MISO: Mixed-Integer Surrogate Optimization Framework," Optim. Eng., **17**(1), pp. 177–203.

[36] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., and Dubourg, V., 2011, "Scikit-Learn: Machine Learning in Python," J. Mach. Learn. Res., **12**, pp. 2825–2830.

[37] Girija, S. S., 2016, "Tensorflow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," arXiv preprint arXiv:1603.04467

[38] Senecal, P. K., 2000, *Numerical Optimization Using the GEN4 Micro-Genetic Algorithm Code*, University of Wisconsin-Madison, Madison, WI.

[39] Krishnakumar, K., 1989, "Micro-genetic Algorithms for Stationary and Non-stationary Function Optimization," Proceedings of Intelligent Control and Adaptive Systems, Philadelphia, PA, Nov. 1–3, International Society for Optics and Photonics, pp. 289–297.

[40] Eberhart, R., and Kennedy, J., 1995, "Particle Swarm Optimization," Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, Nov. 27–Dec. 1, pp. 1942–1948.

[41] Clerc, M., 2012, "Beyond Standard Particle Swarm Optimisation," *Innovations and Developments of Swarm Intelligence Applications*, IGI Global, pp. 1–19.