A Machine Learning–Genetic Algorithm (ML-GA) Approach for Rapid Optimization Using High-Performance Computing

Author, co-author (Do NOT enter this information. It will be pulled from participant tab in MyTechZone)

Affiliation (Do NOT enter this information. It will be pulled from participant tab in MyTechZone)

Abstract

A Machine Learning–Genetic Algorithm (ML-GA) approach was developed to virtually discover optimum designs using training data generated from multi-dimensional simulations. Machine learning (ML) presents a pathway to transform complex physical processes that occur in a combustion engine into compact informational processes. In the present work, a total of over 2000 sector-mesh computational fluid dynamics (CFD) simulations of a heavy-duty engine were performed. These were run concurrently on a supercomputer to reduce overall turnaround time. The engine being optimized was run on a low-octane (RON70) gasoline fuel under partially-premixed compression ignition mode. A total of nine input parameters were varied, and the CFD simulation cases were generated by randomly sampling points from this nine-dimensional input space. These input parameters included fuel injection strategy, injector design and various in-cylinder flow and thermodynamic conditions at intake valve closure (IVC). The outputs (targets) of interest from these simulations included five metrics related to engine performance and emissions. Over 2000 samples generated from CFD were then used to train an ML model that could predict these five targets based on the nine input features. A robust super learner approach was employed to build the ML model, where results from a collection of different ML algorithms were pooled together. Thereafter, a stochastic global optimization genetic algorithm (GA) was used, with the ML model as the objective function, to optimize the input parameters based on a merit function so as to minimize fuel consumption while satisfying CO and NO_x emissions constraints. The optimized configuration from ML-GA was found to be very close to that obtained from a sequentially performed CFD-GA approach, where a CFD simulation served as the objective function. In addition, the overall turnaround time was (at least) 75% lower with the ML-GA approach, as the training data was generated from concurrent CFD simulations and employing the ML model as the objective function significantly accelerated the GA optimization. This study demonstrates the potential of ML-GA and high-performance computing (HPC) to reduce the number of CFD simulations to be performed for optimization problems without loss in accuracy, thereby providing significant cost savings compared to traditional approaches.

Introduction

Consumer demand and government regulations are driving automakers to explore new engine designs that simultaneously reduce fuel consumption as well as emissions. To develop these new designs, automakers use a combination of experimental prototyping and numerical modeling. Of late, numerical simulation has assumed a much greater significance in engine design optimization, particularly with the impetus towards advanced and novel combustion concepts coupled with new fuels, where engineers cannot rely solely on past expertise to narrow down the design space for experimental prototyping.

A commonly used approach to engineering design optimization is to employ the design of experiments (DOE) technique [1]. In a simulation-based DOE, the entire design space can be explored by running a large number of simulations to fill the DOE hyper-volume using suitable space-filling techniques. A response surface is then fitted to the simulation data and used for design optimization. This response surface connects the inputs to the outputs and is usually built based on linear regression. However, these linear regression based response surface methods (RSMs) are not well suited to characterize any non-linearities and interactions between various inputs without a tuning effort by the designer, such as adding cross-terms and higher order terms. This can often lead to large errors when dealing with inputs that non-linearly interact, as can often be the case in engine combustion. Another robust approach for design optimization is based on genetic algorithms (GAs) [2-7]. In a GA-based optimization, a CFD simulation is used as an objective function and the merit of this objective function is evaluated on completion of the CFD run. The CFD runs are performed sequentially in waves of several generations, where each generation has some prescribed number of individuals/samples to evaluate (i.e., CFD runs to perform). The more the number of samples in each generation, the fewer the number of generations needed to find the optimum. Using a stochastic approach where the population is varied using a set of genetic operations often leads to much better optimum solutions than DOE-based approaches. However, a GA may take a considerable amount of time (over two or three months) even with a reasonably sophisticated cluster (100-1000 processors) for a problem with a moderate number of input or design features (5-10) to optimize for. This is attributed to the fact that physics-based CFD simulations tend to be very expensive.

In this context, data-driven machine learning (ML) models can play an important role. An ML model can be thought of as a function that, after being trained, can learn from various patterns and structures in the data and capture input-output relationships. Depending on the complexity of the chosen ML model, it is possible to capture non-linear relationships including any interaction effects between the inputs. This results in a very reasonable fit of the input-output data space and provides an accurate and faster running surrogate model for CFD. Over the past few decades, there have been significant developments in using artificial neural networks (ANNs), a type of ML approach, in understanding and solving combustion problems [8-12]. Here ANNs are used for predicting various combustion related outputs after being trained on sample data. New applications for ML have been receiving substantial interest and as a result many new promising models have been developed [13-17]. ML models can be considered as fast-running surrogate models for the more time-consuming methods used in generating their training data viz., experiments or CFD models. In the past, ML models have been used in real-time control of engines [9, 10,

18-21]. Vaughan et al. [19, 20] used a form of support vector machine (SVM) to understand the bifurcation behavior of the combustion stability limit in a homogenous charge compression ignition (HCCI) engine and used it for prediction of cycle-to-cycle variations of combustion phasing and for misfire detection. Validi et al. [21] used an ANN approach to detect and optimize the start of combustion in HCCI engines. He et al. [10] created engine cylinder models by capturing various in-cylinder physical processes using ANNs, which were used in the prediction of engine performance and emissions over a transient federal test procedure (FTP) cycle.

Recent efforts have coupled optimization techniques with ML models to optimize the system outputs within the target design space [1, 22-24]. In these studies, the ML model served as the objective function for GA optimization, as opposed to a CFD model. Brahma et al. [23] used a set of ANNs together with a hybrid GA/hill-climbing type algorithm to optimize operating parameters over the entire speed-torque map of a diesel engine. Alonso et al. [22] followed a similar approach while using experimental datasets for a high-dimensional diesel engine GA optimization problem. Costa et al. [24] looked into the diesel optimization problem from the point of view of soot-NOx trade-off and used an optimization code to obtain pareto fronts of soot-NO_x predictions by an ANN for various piston bowls. However, one of the common features of these studies was that only one type of ML approach was employed. Moreover, the optimization accuracy of these models was not validated.

In the present work, a novel ML-GA model was employed to perform numerical optimization of a gasoline compression ignition (GCI) engine operating with a low-octane gasoline-like fuel. PPCI is an advanced combustion mode which has the potential to achieve diesel-like fuel efficiency with ultra-low nitrogen oxides (NOx) and particulate matter (PM) emissions. However, the primary challenges for practical implementation of PPCI include achieving robust control of ignition timing, preventing excessive pressure rise rates, and mitigating hydrocarbon (HC) and carbon monoxide (CO) emissions. In this study, a total of 2048 samples were randomly generated using Monte-Carlo sampling and 3D CFD simulations for these samples were performed concurrently on Argonne's Mira [25] supercomputer. The samples were distributed over nine input parameters related to the fuel injector design, injection strategy, initial in-cylinder chamber pressure and temperature, and swirl flow. A merit function was formulated consisting of 5 targets related to engine performance and emissions. A stacked generalization approach called "Super Learning" was employed to develop the ML model based on CFD simulations. Employing a stacked generalization approach instead of a stand-alone ML model (as in the above cited works) results in higher error compensation of the ML algorithm for robust performance. The stacked ML model was trained and tested on the input-output CFD data. The overall output of the ML model was the merit value after considering the individual outputs which constituted the merit function. The ML model was characterized by gauging the bias-variance trade-off using learning curves. Subsequently, a stochastic global optimization GA was used with the ML model as an objective function to perform optimization of the merit value. The ML-GA approach was implemented using R programming language [26]; custom scripts were written to combine the approaches efficiently and to validate the concept. The ML-GA optimum was compared to that from a conventional CFD-GA run using CFD as the objective function. In addition, CFD simulation for the ML-GA-predicted optimum point was also performed to further validate the optimization. A parametric study with varying sample sizes for ML model training was also carried out to find out the minimum number of simulations needed to efficiently and accurately carry out the ML-GA optimization. Finally, the runtimes of the conventional CFD-GA and ML-GA approaches were compared keeping various computational resources in mind.

Methodology

Engine Design Space & Optimization Strategy

The present numerical study aims to optimize a heavy-duty engine operating at medium load conditions with a low-octane gasoline-like fuel. The engine considered is a four-stroke, six-cylinder Cummins ISX15 engine with a variable-geometry turbocharger, high-pressure cooled exhaust gas recirculation (EGR) loop and charge air cooler [27, 28]. The details of the engine configuration and baseline operating conditions are listed in Table 1.

Table 1: Engine configuration and baseline operating conditions.

Engine model	Cummins ISX15
Cylinders	6
Displacement	14.9 L
Bore	137 mm
Stroke	169 mm
Connecting rod	262 mm
Compression ratio	17.3:1
Engine speed	1375 rpm
Intake valve closing (IVC)	-137 ^{0}CA after top-dead center (ATDC)
Exhaust valve opening (EVO)	148 ^{0}CA ATDC
Start of injection (SOI) timing	-9 ^{0}CA ATDC
Injection duration	15.58 ^{0}CA
Mass of fuel injected	0.498 g/cycle
Fuel injection temperature	360 K
Injection pressure	1600 bar
Nozzle inclusion angle	152^0
IVC pressure	323 K
IVC temperature	2.15 bar
Exhaust gas recirculation (EGR)	41%
Global equivalence ratio	0.57

The design parameters considered in the present work are listed in Table 2, along with their respective ranges of variation.

Table 2: Input parameter ranges for the engine design space.

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	°CA ATDC
Nang	Nozzle inclusion angle	145	166	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	-

In total, nine input variables were chosen pertaining to fuel injector design (number of nozzle holes, total nozzle area, nozzle inclusion angle), fuel injection strategy (injection pressure, start of injection timing), and initial thermodynamic and flow conditions (intake valve closing temperature and pressure, exhaust gas recirculation fraction, and in-cylinder swirl). The ranges of variation included the baseline conditions. Note that in Table 2, the total nozzle area is normalized with respect to its baseline value, therefore the baseline value is 1.

Throughout the optimization study, the total mass of fuel injected, i.e., the engine load, was kept constant.

For optimization, an objective merit function, as shown in Eq. (1), was defined using indicated specific fuel consumption (ISFC, g/kW-hr) as the performance variable (to be minimized), and constraint variables based on emissions (soot, NO_x) and engine mechanical limits (peak cylinder pressure (PMAX), maximum pressure rise rate (MPRR)). The merit function incurs a penalty only if soot, NO_x, PMAX and MPRR exceed their constraints of 0.0268 g/kW-hr, 1.34 g/kW-hr, 220 bar, and 15 bar/^{0}CA, respectively. No penalty is incurred if these are within their prescribed limits, in which case the merit function varies only with ISFC. In addition, the constraint variables are assigned different weights to reflect their relative importance in the optimization process. A similar merit function formulation was also employed in a previous engine design optimization study [1].

$$Merit = 100 * \begin{bmatrix} \frac{160}{ISFC} - 100 * f(PMAX) - 10 * f(MPRR) \\ - f(SOOT) - f(NO_x) \end{bmatrix}$$

Where

$$f(PMAX) = \begin{Bmatrix} \frac{PMAX}{220} - 1, if\ PMAX > 220 \\ 0, if\ PMAX \leq 220 \end{Bmatrix}$$

$$f(MPRR) = \begin{Bmatrix} \frac{MPRR}{15} - 1, if\ MPRR > 15 \\ 0, if\ MPRR \leq 15 \end{Bmatrix}$$

$$f(SOOT) = \begin{Bmatrix} \frac{SOOT}{0.0268} - 1, if\ SOOT > 0.0268 \\ 0, if\ SOOT \leq 0.0268 \end{Bmatrix}$$

$$f(NO_x) = \begin{Bmatrix} \frac{NO_x}{1.34} - 1, if\ NO_x > 1.34 \\ 0, if\ NO_x \leq 1.34 \end{Bmatrix}$$

- (1)

Numerical Model Setup

A 3-D CFD code, CONVERGE (version 2.3) [29], was used to perform numerical simulations for the closed part of the cycle, from IVC to EVO. Assuming axisymmetry, the simulation domain was modeled using a sector mesh representing a single cylinder and accounting for only one spray plume, in order to reduce computational cost. Periodic boundary conditions were imposed in the azimuthal direction. Uniform mixture and temperature distributions were specified at IVC. A base mesh size of 1.4 mm was employed. One level of fixed embedding was prescribed near the cylinder head and piston, while two levels of fixed embedding were employed to resolve the flow near the fuel injector. In addition, two levels of adaptive mesh refinement (AMR) were employed based on velocity and temperature gradients of 1 m/s and 2.5 K, respectively. This resulted in the minimum grid size of 0.35 mm and peak cell count per simulation of ~ 500,000. The simulation time step was automatically adjusted based on the maximum convective, diffusive and mach Courant-Friedrichs-Lewy (CFL) numbers of 1, 2 and 50, respectively.

In-cylinder turbulence was modeled using the Reynolds-Averaged Navier Stokes (RANS) based re-normalized group (RNG) *k-ε* model [30] with wall functions. The liquid spray was treated in a Lagrangian fashion and the "blob" injection model developed by Reitz and Diwakar [31] was used, which initializes the diameter of a liquid droplet to the effective nozzle diameter. The Kelvin Helmholtz (KH) – Rayleigh Taylor (RT) breakup model [32] and "no-time counter" collision model of Schmidt and Rutland [33] were employed to describe the subsequent spray atomization and collision processes, respectively. Droplet evaporation was modeled using the Frossling correlation [34], and models for dynamic drop drag and droplet turbulent dispersion [35] were also included. A RON70 primary reference fuel (PRF) blend comprising of 70% iso-octane and 30% n-heptane (by mass) was employed as the surrogate for gasoline-like fuel [36]. A reduced kinetic mechanism for primary reference fuel (PRF) consisting of 48 species and 152 reactions based on Liu et al. [37] was used to account for fuel chemistry. In addition, the NO_x formation was modeled using a reduced mechanism comprising 4 species and 13 reactions [38]. The empirical Hiroyasu soot model [39], coupled with the Nagle and Strickland-constable model [40], was used to determine the soot formation and oxidation rates with acetylene (C_2H_2) as the precursor for soot formation. For combustion modeling, the SAGE detailed chemistry solver [41] was employed along with a multi-zone (MZ) approach, with bins of 5 K in temperature and 0.05 in equivalence ratio. The CFD model predictions were validated against experimental data in previous studies [27, 28, 42] and hence are not shown here for the sake of brevity. In addition, an exhaustive global sensitivity analysis (GSA) was also performed by the authors for the same baseline engine operating condition, with respect to the input parameters (and their corresponding ranges) listed in Table 1 and targets considered in the present work. These details can be found in Ref. [42].

A separate GA optimization (herein referred to as CFD-GA) using CFD simulation as the objective function was performed. The CFD-GA also used the same merit function described in Eq. (1). More details about the CFD-GA approach are given in the following section.

CFD-GA Approach

The Converge CONGO utility [29] was employed to perform the CFD-GA optimization based on an elitist micro-GA approach [43]. The micro-GA has a small population of 9 individuals and is run for a large number of generations. Each individual is a CFD simulation case with a distinct set of input parameters. The initial population of 9 individuals is generated randomly. The merit values for the individuals are evaluated after each generation is completed and the population is monitored for similarity between the individuals using a statistical calculation based on the input parameters. Micro-convergence is achieved when the population reaches a sufficient convergence level (convergence criterion of 0.97). At that point, the population is declared to have converged and is replaced with randomly generated individuals (except for the elite individual with the best merit value, which is always kept in the population). This provides randomness to avoid local optima being erroneously identified as the global optimum. However, until a micro-convergence event is encountered, the individuals of the new population are generated from parents which are selected based on a tournament. A uniform crossover of the DNA (i.e., the input parameters) is used to create children from the parents. The micro-GA used in this work does not allow mutations. The tournament size is set as the size of the population and as parents are selected, the tournament size decreases without re-shuffling. The population is re-shuffled only after depleting the selection pool. The GA is considered to have reached a global optimum when at least five micro-convergence events occur, without any improvement in the maximum merit value.

In this work, the CFD-GA was run for 98 generations (784 CFD simulations) until it converged. The runtime of each CFD simulation was approximately 12 hours on 128 processors. So, the full GA took

around 50 days to run. The GA reached a maximum merit value of 104.0 at 57th generation. Afterwards, 5 micro-convergence events were encountered without any further improvement in the merit value. At that point, it was assumed that the GA had finally found the global optimum. The evolution of maximum merit value during the progress of the CFD-GA optimization is shown in Figure 1.

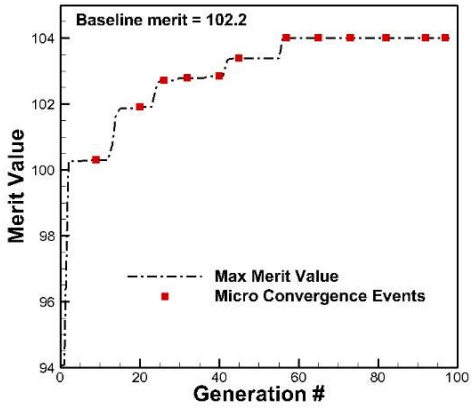


Figure 1: Merit evolution using the micro-GA approach (CFD-GA).

The input parameters and the corresponding outputs for both the baseline and best (optimum) cases are listed in Table 3. Evidently, the best case found by CFD-GA yielded an ISFC benefit of 1.7% over the baseline design. Moreover, none of the emissions and cylinder pressure constraints were exceeded by the GA optimum.

The temporal evolution of in-cylinder pressure for both the baseline and best cases are shown in Figure 2. Clearly, the total work done in the best case is much higher, thereby resulting in lower ISFC, considering that the total fuel mass injected was kept constant. It must be noted that pressure at IVC for the best case is higher than the baseline. In addition, Figure 3 shows the comparison of in-cylinder temperature and equivalence ratio distributions at 12 ^{0}CA ATDC on a vertical cut plane, between the two cases. It is evident that the islands of fuel-rich mixture are much smaller in the best case, owing to a better fuel-air mixing caused by earlier SOI timing, higher swirl and higher number of nozzle holes. Moreover, in-cylinder temperatures are also higher relative to the baseline configuration. All these factors contribute to higher peak cylinder pressure, lower ISFC, lower soot emissions and slightly higher NO_x emissions, in case of the GA optimum.

Table 3. Input parameters and outputs for the baseline and CFD-GA best cases.

Parameter	Baseline	Best case
Inputs		
nNoz	9	10
TNA	1.0	1.0
Pinj	1600 bar	1490 bar
SOI	-9^0 CA ATDC	-10.3^0 CA ATDC
Nang	152^0	158^0
SR	-1	-1.66
EGR	0.41	0.44
Tivc	323 K	323.5 K
Pivc	2.15 bar	2.3 bar
Outputs		
ISFC	156.53 g/kWh	153.85 g/kWh
PMAX	152.31 bar	162.03 bar
MPRR	11.22 bar/^{0}CA	11.31 bar/^{0}CA
SOOT	0.0235 g/kWh	0.0220 g/kWh
NOx	1.07 g/kWh	1.28 g/kWh
Merit	102.2	104.0

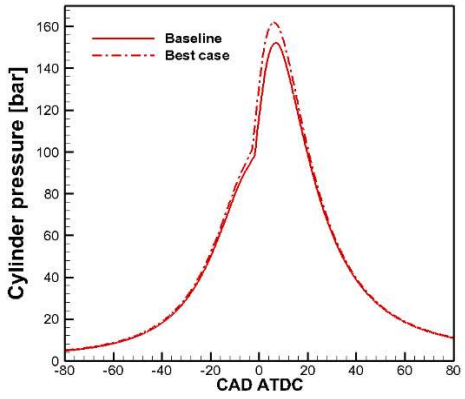


Figure 2: Temporal evolution of in-cylinder pressure for baseline and CFD-GA best cases.

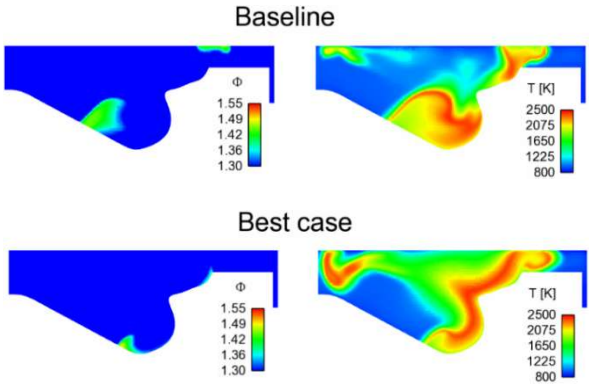


Figure 3: In-cylinder equivalence ratio and temperature distributions along a vertical cut plane at 12 ^{0}CA ATDC for the baseline and CFD-GA best case.

ML-GA Approach

In this approach, an ML model was employed as a surrogate for a CFD simulation and was used to compute merit values of the individuals within a GA optimization routine. The details of the data generation for ML model training are presented next. A description of the ML model and GA technique is provided subsequently.

Data Generation for ML Model Training

The nine input variables were perturbed simultaneously within their respective ranges of variation using Monte Carlo (MC) method to generate 2048 sample (parameter) sets. The input files for the 2048 simulation cases were generated using the Converge CONGO utility [29]. The CFD simulations were run in six batches of 256 cases. Each batch was run on half of a rack of Mira [25], an IBM BG/Q supercomputer at the Argonne Leadership Computing Facility (ALCF) and Office of Science User Facility at Argonne, with each case running on 32 processors. The total runtime for all the simulations was around two weeks including queue time. It must be noted that the 2048 simulations can potentially be simulated all at once if resources allow, which can bring down the simulation time to as low as one day. The minimum and maximum values of the five outputs extracted from the simulation data are given in Table 4.

Table 4: Range of outputs generated from 2048 MC runs.

Output	min	max	units
ISFC	153.3	180.6	g/kWh
SOOT	0	0.2	g/kWh
NOx	0.06	5.25	g/kWh
MPRR	7.9	20.4	bar/deg
PMAX	121	170.3	bar

The surface of the merit function versus two inputs is shown in Figure 4, which sheds light on the highly non-convex nature of the problem. The goal of the ML model is to capture the input-output interactions and reproduce this non-linear surface, while the GA is expected to find the optimum merit based on this surface.

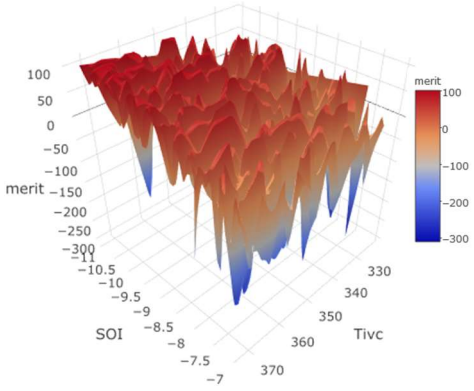


Figure 4: The response surface of merit function w.r.t. SOI and Tivc based on CFD simulations of the uniform MC generated 2048 samples.

ML-GA Model

The ML-GA consists of an ML model acting as a surrogate for CFD model and as an objective function with a GA optimization algorithm. In the following section, the ML model is described first followed by the GA algorithm.

Machine Learning Model

The ML approach of super learner [44] was employed in the present work and was implemented in the R package [45]. The super learning technique falls under the broader approach of stacked generalization of multiple models. Super learning is a loss-based learning method [46] which calculates the optimal combination of a pool of prediction algorithms. The optimal combination minimizes the cross-validated risk (error) during the training phase of the multiple models. Figure 5 shows a simple representation of the stacked generalization approach followed by the super learner algorithm.

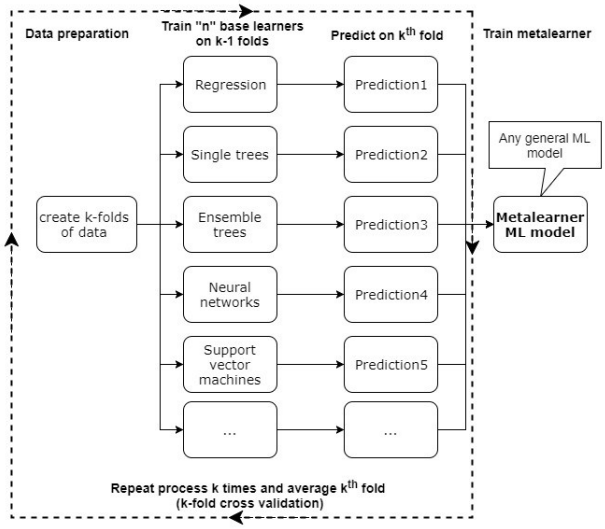


Figure 5: A schematic of the super learner model.

Following Figure 5, in the beginning of the model fitting procedure, the data was split into training and test sets. An 80%/20% split was used during the model accuracy characterization process (accuracy characterization is discussed in a subsequent section). A k-fold cross validation was then performed to assess the prediction of the individual base learners or sub-models. Cross-validation is a technique to evaluate model performance by splitting the original dataset into a training set to train the model, and a cross-validation set to evaluate it. Random splitting of the original (80% training) dataset into 10 (k=10) equal sized subsample folds was performed. Out of the 10 subsample folds, a single subsample fold (called the 10th fold) was retained as the cross-validation dataset and the remaining nine sub-samples were used as training data. In this way, the 80% original training data was split into further training and cross-validation datasets for performing cross-validation. This cross-validation process was then repeated 10 (k=10) times over all the 10 sub-sample folds. In this way, each of the 10 subsamples was used exactly once as the cross-validation dataset. Thus, all observations were used for both training and cross-validation, and each observation was used for cross-validation exactly once. The results from individual folds were then averaged to produce a single estimation. The predictions of the 10th fold for *each* of the algorithms were saved. Following this, weighted combinations were prescribed as coefficients to these predictions of individual base learner algorithms. A non-negative least squares (NNLS) method was employed to compute these coefficients. This was done to minimize the cross-validation error (i.e., to minimize the error in the 10th fold of predictions). The weightage given to each of the base learners would

thus change as the test data is varied and it is possible that some of the base learners get a zero weightage in the linear equation of NNLS, if they have high error. The NNLS model here is called the meta-learner. In practice, any general ML model can be used as a meta-learner [47]. The meta-learner which operates on the predictions (not the raw data) is the final step in the super learning process and is ultimately used to predict new outcomes.

Architecture of the Super Learner

Six different ML algorithms were considered for the individual base learner system. Baseline values of parameters of each of the algorithms were used during their implementation in the super learner model and expensive grid search techniques over the model parameter space to optimize the performance of each model was avoided. A brief description of these base learners is provided below. For more details on the theory and mathematical formulation of these models, the reader is directed to the cited literature in the relevant model descriptions below.

Linear model: A linear model is the simplest form of regression model. The model is obtained by minimizing the sum of squares of the differences between the actual observed values and those predicted by a linear equation with a set of explanatory variables as coefficients.

Ridge regression [13]: Ridge regression is an ordinary least square linear regression problem with built-in regularization term (lambda) to avoid over-fitting or high variance during predictions.

k-nearest neighbor (kNN) [14]: kNN is a non-parametric method in which the input consists of *clusters* of training examples containing "k" samples each, in the feature space. In kNN regression, the prediction is the average of these cluster values which have "k" points each (which are called nearest neighbors).

Random Forest [15]: Random forest is an ensemble of user-specified decision trees, in which each tree uses a random number of samples from the given dataset. The individual decision tree is trained on various parts of the same training set. Random forest models circumvent the over-fitting problem by averaging results from separate decision trees.

Extreme gradient boosting (XGboost) [12]: XGboost is an advanced implementation of decision tree algorithms (similar to random forest), but is developed keeping in mind speed and performance. The general concept of gradient boosted trees is that initially the model predictions are made with simple tree structures and later on new tree structures are created that predict the errors of prior models. These are subsequently added together to make the final predictions. The term gradient boosting comes from the fact that the loss minimization during addition/ensembling of the new models is achieved using a gradient descent algorithm.

Support vector machines (SVM) [16]: The idea behind an SVM is to find a separation line that splits the data, for example between two different clusters in a training dataset. This line separates the clusters of data in such a way that the individual clusters are farthest away from the line. A new test data point is predicted depending on where it will be placed on either side of the line. The determination of this line is done through quadratic programming and resembles an optimization problem.

Neural Net [11]: Neural net (also called artificial neural network) is a network of simple units called neurons which are subjected to raw inputs. Based on the inputs, the neurons then vary their internal state and produce an output. A network, in the form of a directed weighted graph, is formed by connecting the outputs to the inputs. The internal aspects of the learning process of neural nets are usually optimized through a process called learning where researchers have applied methods such as fuzzy logic, Bayesian method, and genetic algorithms.

The ML algorithms used and their model parameters are presented in Table 5.

Table 5: Sub-model parameters used in the super learner.

Model	Details
Random Forest	ntree=1000, mtry=3
Suport vector machines (nu-regression)	nu=0.5, degree=3,cost=16, coef0=0, kernel=radial
Ridge regression	lambda=1 to 20 (steps of 0.1)
xgboost	ntrees=1000,max_depth=4,shrinkage=0.1, minobspernode=10
Neural net	size=9,hiddenlayers=1
Linear model	ordinary least square
K-nearest neighbor	k=10

The GA used with the ML model was not an elitist micro-GA unlike in the CFD-GA approach. Instead, a GA technique available from the R package was chosen which was compatible with the ML model. A previous comparative study [48] was taken as an index to first down-select a few good GA model candidates exhibiting good run-time/accuracy trade-offs. An in-house testing exercise was subsequently carried out with this subset of better performing GA models from the study [48] to finalize the GA to be used for the present work. Details of this GA model are presented next.

Genetic Algorithm Model

The genetic algorithm used here is called malschains [49], which stands for memetic algorithms with local search chains. The implementation in R (Rmalschains) was used in this work [50]. Malschains uses a combination of local and global optimization techniques. The idea behind the algorithm is to apply a local search method on the most promising regions which are found to have highest fitness value using a (global) genetic algorithm. Malschains uses a steady state genetic algorithm as an evolution algorithm which executes the global optimization. The GA in malschains is different from a standard genetic algorithm, where the individuals of the population are subjected to genetic operations simultaneously. In the present GA method (a steady-state GA), only single individuals are used at a time to generate offspring, which replace other single individuals of the population.
The malschains algorithm randomly generates an initial population of individuals. The genetic algorithm then evaluates the merit values (fitness) of these individuals and builds a set of individuals which can be further refined by the local search method. For local optimization, solis-wet algorithm is used. The local search method is iteratively applied on these best-fit cluster of individuals obtained from the GA and a best individual is picked and recorded. This solution replaces the worst solution in the next application of the GA and local search loop. This final solution is also used for the initialization of a subsequent local search application which creates a chain of local search solution. This allows for improvement of the same solution several times.

The GA model acts as a wrapper around the ML model to constitute the ML-GA. A complete overview of the ML-GA pipeline is explained in the flow chart shown in Figure 6.

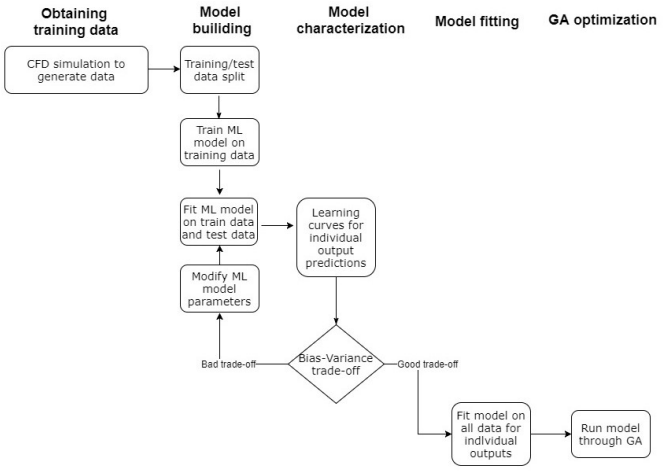
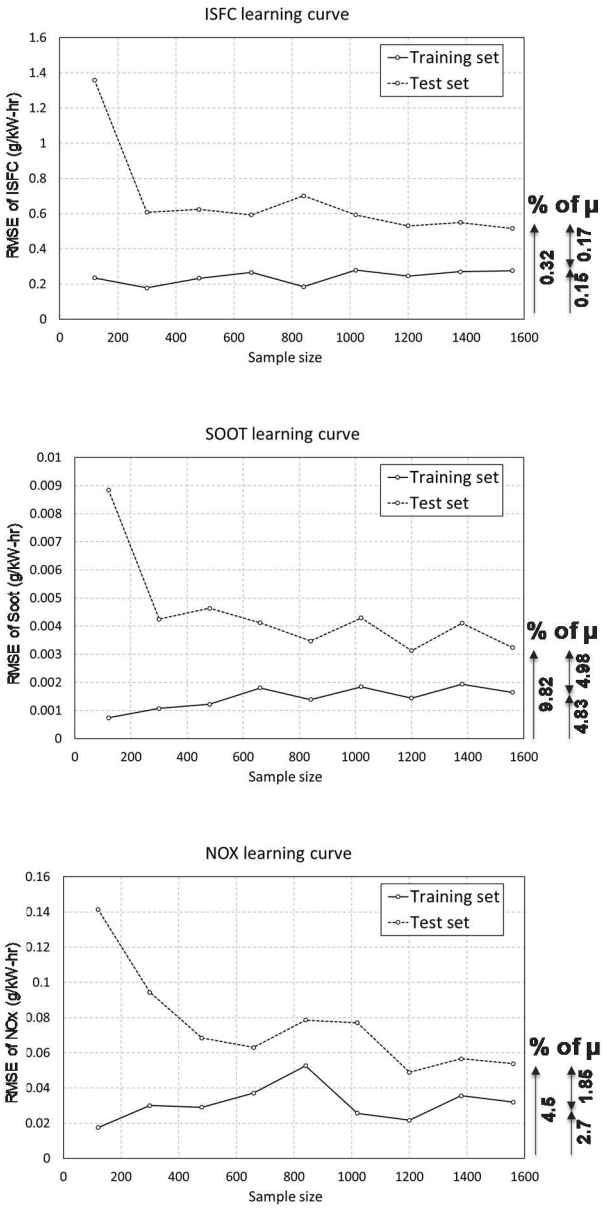


Figure 6: A schematic of the ML-GA technique.

To apply the ML method, CFD or experimental data is needed. After running 2048 CFD simulations, the input-output data was extracted from the simulation results and split into training and test datasets. In this study, 80% of the data was randomly sampled to generate a training set and the remaining 20% of the data was used as the test set. A super learner model was then trained on each of the outputs/targets (ISFC, PMAX, MPRR, soot, NO_x), i.e., one super learner model for predicting each output parameter. After the models were trained, they were tested on both the training (in-sample) and test (out-of-sample) datasets to generate learning curves for individual output predictions. For a learning curve, training data size was increased from a minimum sample size to maximum of 80% of the total CFD samples. The model was trained each time and a training root mean squared error (RMSE) was calculated. For each of these training stages, 20% of test data was evaluated and the test RMSEs were noted. Learning curves were then obtained by plotting train and test RMSEs versus training sample size. A bias-variance trade-off analysis was carried out using the learning curves to understand the model behavior. If both the test and train RMSEs were within 10% of the mean value of the CFD outputs (for reduced bias) and their test RMSEs were within 5% of the train RMSEs (for reduced variance), then the model was considered to have a good bias-variance trade-off. A high bias would cause under-fitting, leading to its inability to capture the trends in the data points effectively. To remedy under-fitting, it is desirable to make the model complex so that it captures all the important interactions in the input-output space. In this work, adding more models to the super learner (to increase complexity) was an option. The strength of the super learner approach is to replace the costly model parameter tuning exercise and work with the baseline model setting, combine the predictions and provide high error compensation. However, if the model is too "coarse" and predicts with high error, then slight tuning will be necessary. If a model predicts with high error, then it will not be selected for predictions by the meta-learner in the first place and so tuning will allow the individual model to be picked up by the meta-learner, increasing the model complexity and thus avoid under-fitting. Tuning of model parameters of the individual algorithms was however, not performed here. On the other hand, a high variance leads to over-fitting, which is a result of the model not generalizing the data very well and exhibiting large changes in outputs as the inputs change. To remedy over-fitting, it is desirable to add more data during model training. Thus, if the bias-variance trade-off is not acceptable, the model parameters can be tuned or more data can be sought and the process is repeated until a good trade-off is obtained. The steps until now characterize the model behaviors (further discussed in the next section). After the model characterization was done, all the data (in our case 2048 simulations) was considered for training so that the ML model was trained on as much data as possible to cover majority of input combinations which the GA would (randomly) generate. This trained ML model was then employed as a surrogate for the CFD model to compute and optimize the objective merit function in the GA optimization.

ML Prediction Accuracy Characterization

To assess the behavior of the ML model, learning curves were generated to characterize bias-variance trade-offs. A brief description of learning curves and the importance of bias-variance trade-off was given in the previous section. Specifically, a high bias would cause under-fitting, rendering the ML model incapable of capturing the trends in the data effectively. On the other hand, a high variance would lead to over-fitting, which is a result of the model not generalizing to data very well and exhibiting large changes in output predictions as the inputs change. The learning curves for different outputs are plotted in Figure 7. Since there are multiple models within the super learner which are involved in making a prediction, the learning curves are not smooth. The authors have found that using a single ML model for predictions usually gives smoother learning curves.



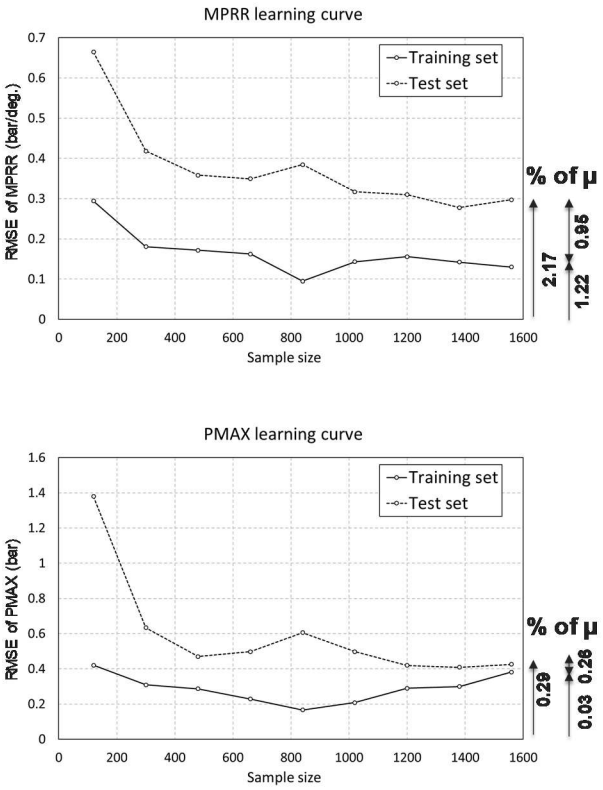


Figure 7: Learning curves of the individual outputs of ISFC, SOOT, NOx, MPRR and PMAX. Single arrows on the right Y-axis signify the training set and test set RMSEs as percentages of the mean of the corresponding output (to account for bias). Double arrow shows the difference between the test set and training set RMSE percentages (to account for variance).

Learning curves play an important role in model characterization. As can be seen in Figure 7, the learning curves for the five outputs exhibit good bias-variance trade-off as per the targets set. On the Y-axis of each of the sub-plots in Figure 7, single arrows denote the training set and test set RMSEs as percentages of the mean of the corresponding target, used as quantitative metrics to assess bias in the model. Double arrow, on the other hand, shows the difference between test and train RMSE percentages, thereby representing the model variance. It can be observed that both test and train RMSEs are within 10% of the mean value and test RMSEs are within 5% of the train RMSEs (shown on secondary Y-axis). The R-squared values of the predictions shown in Figure 7, over all the sample sizes and 5 output parameters, were above 98%. An R-squared value is not a good measure of goodness of the fit though, be it variance or bias; a learning curve sheds more light on those aspects. Additionally, it can be observed that all the five outputs reach a quasi-steady RMSE state after ~300 samples; this indicates that ~300 samples might be good enough for the ML model to capture various input-output non-linearities in the dataset satisfactorily, for this particular engine simulation case. A later section introduces a test done by reducing sample sizes to explore this possibility.

RESULTS AND DISCUSSSION

Based on the process specified in Figure 6, since the learning curves exhibited good behavior, all the data was considered for training of the ML model. After training the models on each of the five outputs, the 5 models were employed in the GA solver to compute the merit values of the individual CFD input sets. The goal of the optimization was to maximize the merit value.

Before discussing the results of the GA optimization, it is worthwhile to check the performance of the individual models within the super learner and their contributions to the model predictions as a whole. This will emphasize the need for using a super learner type approach. Figure 8 shows a plot of model importance. Model importance is gauged as the value of the coefficient in the linear non-negative least squares (NNLS) equation which optimizes the selection of the individual sub-models.

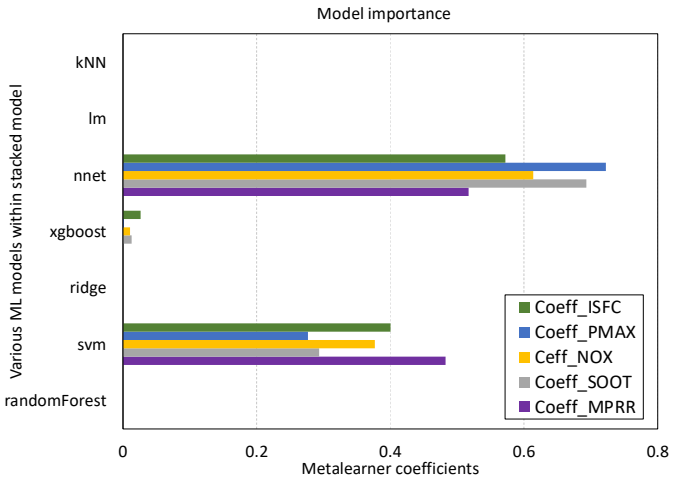


Figure 8: Weights of each ML sub-model comprising the super learner model for each output.

It is evident that out of the 6 different ML models employed within the super learner, only neural network, SVM and xgboost (in that order) contribute to the overall prediction. The rest of the models have no contribution and so have zero weightage. The selection of the models based on their individual contributions, i.e., the decision of how important these models are in the super learner model, was made by considering the cross-validation error. Weights were decided according to the mathematical optimization technique of NNLS which was based on the Lawson-Hanson algorithm. NNLS is a constrained version of the least squares problem, in which the coefficients are not allowed to become negative. The cross validated errors are mean square error values and are shown as an error risk estimate in Figure 9. The error bars in the figure signify the variation in each validation loop of the k-folds. The risk is meant to be a measure of model accuracy. By minimizing this risk, the model makes fewer erroneous predictions. It can be clearly seen that the model with the lowest CV errors are the ones chosen by the NNLS meta-learner model to maximize the prediction accuracy. In short, the super learner uses 10-fold cross-validation to estimate the risk (or error) on future data. The super learner then employs a meta-learner and reduces the error of the stacked models by assigning proper weights to the best performing models. Figure 9 also shows that the error using super learner is less than or equal to the errors from the best performing models viz., neural net, SVM and xgboost, considered individually and so indicates its benefits over using individual ML models.

This super learner model was then applied along with the GA (malschains). Although, ~3500 evaluations would have been sufficient to get to true convergence, a much higher number of 35000 evaluations was chosen, as repeated runs of the GA would converge at slightly different number of evaluations. Note that the ML-GA optimization is not a time-consuming process: the runtime of the ML-GA was between 1 and 10 minutes. The major portion of time for this approach was spent generating the initial CFD data for training. The training of the ML model took between 30 seconds and 2 minutes.

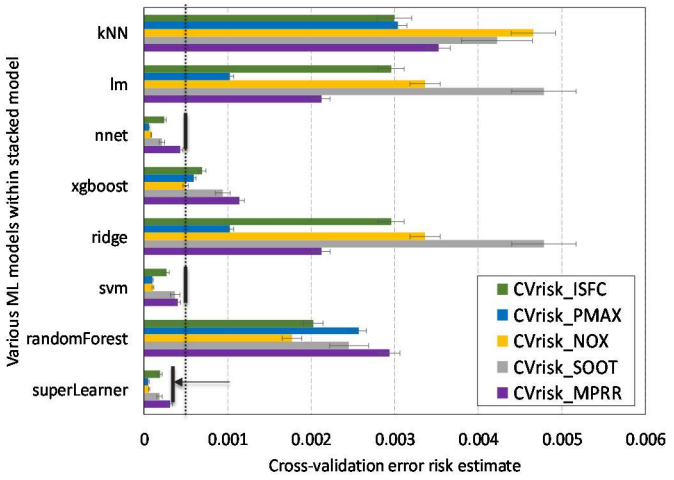


Figure 9: Mean square error representing a risk estimate in cross-validation procedure of individual sub-models and the super learner model. The arrow points towards a reduced error using super learner model.

The resulting optimized values of the inputs are presented in Figure 10 in the form of a normalized plot, i.e., the values are scaled between corresponding minimum and maximum values with centering done at the minimum value. Ten repeats of the GA model (with different random initializations) were performed to make sure that the GA did not output different values for each run, in which case, the GA might be getting stuck at various local optima. The standard deviations of the repeats are also depicted in Figure 10, but are not so obvious due to their low values. In addition, the optimized values from the CFD-GA approach are also shown, which can be considered as a form of validation of ML-GA. It can be seen from Figure 10 that the optimum values predicted by the ML-GA approach are very close to the corresponding CFD-GA values.

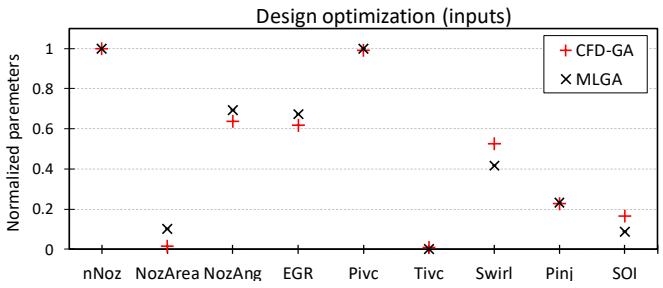


Figure 10: Comparison of the normalized values of the optimized input variables between CFD-GA and ML-GA.

Corresponding to Figure 10, in Table 6, absolute values of the optimized input variables are shown along with the corresponding values of the outputs from both ML-GA and CFD-GA. For reference, the highest merit obtained in the 2048 simulation dataset for the ML training was 103.2. ML-GA optimized inputs and corresponding predicted outputs are shown in column labeled "a". The CFD predicted outputs for the ML-GA optimized inputs are compared and shown in the column labeled "b". The CFD-GA optimized set of inputs and outputs are shown in the column labeled "c". The percentage differences among these columns are shown in the next three columns. The comparison between the ML-GA predicted outputs of the "a" column and CFD predicted outputs for the ML-GA optimized inputs from the "b" column gives an idea of the level of confidence one might have in using ML-GA as a surrogate for CFD-GA.

Table 6: Comparison of the inputs and outputs from the ML-GA optimum, CFD prediction of ML-GA optimized inputs and CFD-GA optimum. Their percentage differences are also shown.

Parameter	a ML-GA optimum	b CFD_{MLGA}	c CFD-GA optimum	% (a-b) diff	% (a-c) diff	% (b-c) diff
Inputs						
nNoz (-)	10	10	10	-	0	0.00
TNA (-)	1.05	1.05	1.0	-	4.58	4.58
Pinj (bar)	1492.5	1492.5	1490	-	0.17	0.17
SOI (deg.)	-10.65	-10.65	-10.3	-	2.99	3.00
Nang(deg.)	159.26	159.26	158.0	-	0.73	0.73
SR (-)	-1.81	-1.81	-1.66	-	9.03	9.04
EGR (-)	0.45	0.45	0.44	-	2.27	2.27
Tivc (K)	323	323	323.5	-	-0.15	-0.15
Pivc (bar)	2.3	2.3	2.3	-	0.44	0.44
Outputs						
ISFC (g/kWh)	153.375	153.97	153.85	-0.38	-0.31	0.08
Pmax (bar)	166.73	165.23	162.03	0.9	2.90	1.97
MPRR (bar/deg.)	13.28	12.22	11.31	8.67	17.42	8.04
Soot (g/kWh)	0.011	0.020	0.022	-50.62	-50.00	-9.09
NOx (g/kWh)	1.32	1.23	1.28	8.24	3.125	-3.91
Merit	104.32	103.91	104.0	0.39	0.32	-0.08

Comparing columns "a" and "b", the differences in most of the outputs are within 10%, but soot has a high error of 50%. It must be noted that inaccurate predictions of outputs might deviate the GA in a wrong direction. However, if the ML predictions are below the set constraints for the outputs, the corresponding terms in the merit function would not be penalized, thereby not affecting the overall GA process. In the present case, for the high error in soot predictions, the overall effect on the merit value due to this under-prediction of soot would be negligible as long as the ML soot under-prediction lies below the threshold soot constraint of 0.0268 g/kW-hr. On the other hand, the current GA has a strong influence from ISFC since it is not constrained in the merit function. Thus, a good ISFC prediction is very important to ensure that the merit calculation is done properly and the GA trajectory represents reality, as is evident from the ISFC and Merit rows of Table 6. Considering the comparison of ML-GA inputs and outputs of column "a" versus CFD-GA inputs and outputs of column "c" from Table 6, it can be seen that most of the predicted values of the inputs are within 10% of the CFD-GA predicted values. Regarding the outputs, soot predictions show the maximum error of 50%. It is however important to note that the CFD run of ML-GA optimized point from column "b" falls very close to the CFD-GA optimized input-output set from column "c" and soot predictions are actually within 10% error based on the CFD run. The high error in predictions of soot form the ML model could be due to an inherent non-linearity of a higher degree in the soot output with respect to the 9 inputs that the ML model could not capture. In order to reduce the error in soot prediction, the values of the inputs were converted to logarithmic scale, after which the ML code normalized them for training. But this approach also did not yield a noticeable impact on the predicted outcomes of the optimized inputs and outputs from ML-GA. The improvement in soot predictions will be investigated and addressed in future studies.

Since optimization using ML-GA showed promising results with merit values and the optimized inputs being consistent with CFD-GA, a parametric study was carried to find out the minimum number of samples needed to obtain a merit value close to what the CFD-GA approach predicted. To further investigate the ML-GA approach for its performance and accuracy, only the worst merit data points were chosen when reducing the data samples. In other words, the training data was sorted in decreasing order of merit values and only the samples less than a particular value were considered to train the ML model and GA was performed using that ML model. For example, if samples with merit value less than -80 are chosen, the original 2048 sample dataset reduces to 345 samples. The ML model is trained on these 345 samples and the GA approach uses this trained ML model to optimize the inputs. Considering an output space constrained in a low merit region and trying to optimize the inputs to a higher merit region is a good test of the ML-GA technique. In a conventional ML-GA optimization approach, one would randomly generate these 345 samples and so the output space will be well distributed among high and low merit values resulting in better ML model training and a better optimization result. From the learning curves of Figure 7, a quasi-steady state was observed after 300 samples. So, around 300 was the lowest number of samples which was expected to perform well with the ML-GA optimization approach for this problem. Sample size reduction was however performed until 66 samples to confirm if there indeed was a knee-point prior to 300 samples beyond which ML-GA became inefficient in performing optimization.

For sample sizes above 275, which correspond to samples above a maximum merit value of -100, except for the nNoz (number of nozzles) parameter, some variation in the optimized input space was observed as the sample size was reduced. In other words, the sample size variation study above 275 did not result in the same optimized inputs as CFD-GA except for number of nozzle holes which was consistently same as the CFD-GA value (nNoz=10). Since there was a wide variation of inputs observed in the optimization process as the sample size was reduced, it was checked if some or all of these optimized inputs would result in a merit closer to that predicted by the CFD-GA approach. The merit predictions for mean values of the optimized input solutions (upon 10 repeated applications of the GA) for various sample sizes is plotted in Figure 11. The percentage error between ML-GA merit predictions and CFD-GA merit predictions is shown as black dashed line. It can be observed that the ML-GA merit predictions are within ~0.5% of CFD-GA merit predictions for sample sizes above 275. For less than 275 samples, ML-GA does not optimize the merit to the level of CFD-GA (black solid line) and hence the error increases sharply. This may be due to a bad prediction of number of nozzle holes among other parameter mis-predictions. To confirm that ML-GA optimized merit values were indeed true, CFD simulations were performed for the optimized input sets and the calculated merit values from CFD were compared to the ML-GA merits. The CFD merit predictions of the ML-GA optimized inputs are shown in Figure 11 as red circles. The ML-GA merits (black cross marks) and their corresponding CFD validations (red circles) show similar values above 275 samples with errors (red dashed line) being within 0.5%. For sample sizes smaller than 275, these errors increase sharply. Additionally, for sample sizes under 275 samples, since the number of nozzles are also predicted incorrectly, it creates a big uncertainty for injector optimization. The error seen as a red dashed line is the merit prediction error related to the ML technique not performing well with less number of samples and thus showing bad validations on comparing with CFD simulations. The error between the merits of ML-GA optimum and CFD-GA optimum is higher than the error seen between ML-GA optimum and its corresponding CFD prediction, since merit mis-prediction error (of that between ML-GA and CFD predictions) propagates through the GA causing a higher error in optimizing the merit value. However, according to this study, as low as 275 (worst) samples would still be enough to optimize the feature set using the ML-GA technique for this CFD engine case. With random sampling of input data, the merits would be fairly well distributed in the output space, which will result in even better learning of the ML model, and this may result in the lowest acceptable sample size limit (here 275) getting even lower.

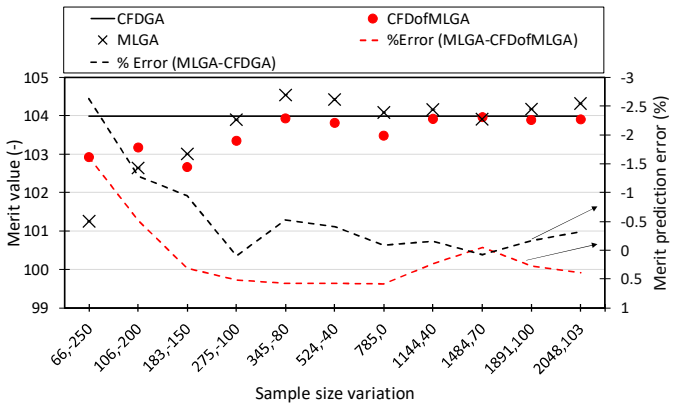


Figure 11: Merit value and its prediction errors between CFD-GA, ML-GA and CFD run of ML-GA optimized points. On X-axis, data is represented in x.y format where x is the ML sample training size and y is the merit value below which samples were considered for training.

Although the optimized inputs are different as sample sizes change, the fact that the optimized merit is still as high as the CFD-GA technique, is the essential outcome supporting the validity of ML-GA optimization technique. An ML-GA optimization with lower sample size may not provide a global optimum but would point to a very close local optima. It can be said that the ML-GA technique works best with higher number of samples, which in this case is 2048. The sample size variation study adds value to the ML-GA technique by showing that a high value of merit is attainable even when the sample sizes are lowered. A mean value of the optimized input solutions (upon repeated application of the GA) was observed to give a merit value with a maximum error of ~0.5% compared to the CFD-GA optimized merit value as shown in Figure 11.

From Figure 1 concerning CFD-GA, a steady state convergence was defined when the GA encountered 5 micro-convergence events without any improvement in the maximum merit value. It can be seen that a minimum of 98 generations, corresponding to 784 CFD simulations run sequentially in batches of 8, were needed for CFD-GA to converge. In this study, 2048 runs were used to perform optimization using ML-GA. The training of the ML model and GA optimization was performed on a single core of Intel Core i7-5600U CPU (2.6 GHz). The runtime for ML model training was between 30 seconds and 2 minutes depending on the size of the data, and the runtime of a single run of the GA routine was between 1 minute and 10 minutes. Since the ML training and GA run times were very low and could in fact, be reduced further if run in a parallel fashion (especially the ML model), they were not included in the runtime calculation of ML-GA and only the time taken for CFD simulations was considered. ML-GA provides the flexibility to run all the simulations at once, if resources allow. For example, simulations can be run all at once (within a day) on a supercomputer. On the other hand, in case of a CFD-GA, the simulations need to be run in batches sequentially over many days. A simulation run on a supercomputer usually needs more cores since memory per core of a supercomputer is less than that of a typical computing cluster. The “time to completion” of job is also higher for a supercomputer compared to a cluster. Nevertheless, the core hours of a supercomputer are cheaper (in terms of dollar value) than the core hours of a typical cluster. So, a direct comparison of core hours used by a job on a supercomputer

versus a cluster is not complete without the economic considerations (which is not done here). A comparison of CFD-GA and ML-GA runtimes considering the supercomputing and cluster resources are presented in Table 8. Shown in the table are the number of simulations required for the CFD-GA (784), the minimum number of simulations needed for training the ML model (~300) to satisfactorily carry out the GA, as well as the full dataset of simulations used for training the ML model in this study (2048). The clusters used in this study can be classified as small clusters, since 128 cores were used at any point of time during the computations unlike bigger clusters which allow for cores in the order of 1000 to be used at once. Comparisons on considering the resources on big and small clusters are also presented in Table 8 along with considering super computing resources.

Table 7: Runtimes for different scenarios of performing CFD simulations.

	Setting	#procs per sim.	hrs per sim.	No. of sim.	sim. per batch	#cores active at a time	RunTime in Day(s)	Core hours
CFD-GA	Small Cluster	16	12	784	8	128	49	150528
ML-GA	Small Cluster	16	12	300	8	128	19	57600
ML-GA	Big Cluster	16	12	300	64	1024	2	57600
ML-GA	Super Computer	32	24	2048	2048	65536	1	1572864

Considering a smaller cluster and keeping the resources the same between CFD-GA and ML-GA, it can be observed that ML-GA can reduce runtimes by about 75% without much sacrifice in the optimization accuracy. It also allows to efficiently increase the optimization accuracy if bigger clusters and supercomputers are used, since it gives the freedom to choose the number of CFD simulations that can be run at once without effecting the quality of the optimization process.

It must be noted that for a CFD-GA, higher number of individuals can be chosen to reduce the generations and hence complete the optimization faster. A micro-GA was used in the CFD-GA approach of the present work and is traditionally employed for engine optimization problems. Micro-GAs are designed to work with a very small number of individuals in the initial population (hence the word "micro"). For the micro-GA of the present work, it has been observed that as the number of individuals in the initial population increases, the generations needed to achieve the optimum decrease in a linear trend (keeping the number of CFD simulations constant). This trend was seen to be valid in the range of 5-13 individuals in the population [51]. For populations above 13 individuals, the micro-GA does not perform well, since the algorithm relies on the population converging to highly similar individuals and larger populations take prohibitively longer to converge. So, the micro-GA in the present form does not provide much leverage in decreasing the number of generations to expedite the optimization time. Nevertheless, there is scope to develop the micro-GA technique to use larger populations in order to reduce the number of generations. An ML-GA approach provides a time-saving equally efficient alternative with added benefits of post-processing (for sensitivity analysis, uncertainty quantification, reliability analysis of the optimized points, etc.), since now one has a faster running mathematical model at hand (although black-box). In addition to that, multiple optima can be readily found by the ML-GA model. This allows for testing on a wider optima pool so that a design can be chosen which is easy to implement and operate in real-world situations.

SUMMARY AND CONCLUSIONS

Machine learning (ML) and genetic algorithm (GA) were used in conjunction to formulate an ML-GA technique. ML-GA was shown to significantly decrease the runtimes of a GCI engine design optimization process by at least 75%, keeping computational resources the same between ML-GA and a traditional CFD-GA. A super learner ML model was employed which pooled the predictions of various individual ML models and thus provided for high error compensation. This super learner was shown to have a better accuracy than traditional ML models when used separately. An exhaustive and necessary accuracy characterization of the ML models using learning curves was also carried out. The ML-GA technique was validated using standalone CFD simulations along with a separate run of a traditional and costlier CFD-GA optimization. The results showed that the accuracy of merit optimization using ML-GA was on par with CFD-GA. The CFD-GA approach requires the CFD simulations to be run in sequential batches to perform the optimization. In contrast, ML-GA allows the CFD simulations to be run in a parallel fashion to train the ML model, provides a surrogate ML model to replace the CFD simulations in the GA and perform optimization using the ML model instead. A parametric study with different sample sizes of ML training data was performed to show that the ML-GA optimum plateaued to a high merit value above a certain (low) number of training samples but the ML-GA merit prediction accuracy decreased below that threshold sample size. Thus, ML-GA allowed for a lower number of CFD simulations while still achieving merit optimization accuracy close to the CFD-GA approach. This has the potential to reduce design optimization times significantly. In addition, ML-GA allows for the optimization process to be scalable to higher computational platforms such as supercomputers, with the potential to complete the optimization in a day, since (a large number of) CFD simulations for training the ML model can be run all at once. The benefit of having a faster running mathematical model at hand also provides the flexibility to carry out other post-processing studies like sensitivity analysis, uncertainty quantification and reliability analysis of the optimized points. ML-GA was also shown to yield different optimized input sets having similar high merit values (when training the ML model with different sample sizes). This can be highly beneficial to an experimentalist, in terms of readily assessing multiple design configurations and choosing the one which is practically most feasible.

References

[1] D. M. Probst, P. K. Senecal, P. Z. Qian, M. X. Xu, and B. P. Leyde, "Optimization and Uncertainty Analysis of a Diesel Engine Operating Point Using CFD," in *ASME 2016 Internal Combustion Engine Division Fall Technical Conference*, 2016, pp. V001T06A009-V001T06A009: American Society of Mechanical Engineers, doi: 10.1115/ICEF2016-9345

[2] Q. Zhang, R. M. Ogren, and S.-C. Kong, "A comparative study of biodiesel engine performance optimization using enhanced hybrid PSO–GA and basic GA," *Applied Energy,* vol. 165, pp. 676-684, 2016, doi: 10.1016/j.apenergy.2015.12.044

[3] D. D. Wickman, P. K. Senecal, and R. D. Reitz, "Diesel engine combustion chamber geometry optimization using genetic algorithms and multi-dimensional spray and combustion modeling," SAE Technical Paper 2001-01-0547, 2001, doi: 10.4271/2001-01-0547

[4] R. Hanson, S. Curran, R. Wagner, S. Kokjohn, D. Splitter, and R. D. Reitz, "Piston bowl optimization for RCCI combustion in a light-duty multi-cylinder engine," *SAE International Journal of Engines,* vol. 5, no. 2012-01-0380, pp. 286-299, 2012, doi: 10.4271/2012-01-0380

[5] A. M. Bertram, Q. Zhang, and S.-C. Kong, "A novel particle swarm and genetic algorithm hybrid method for diesel engine performance optimization," *International Journal of Engine Research,* vol. 17, no. 7, pp. 732-747, 2016, doi: 10.1177/1468087415611031

[6] Y. Shi and R. D. Reitz, "Optimization of a heavy-duty compression–ignition engine fueled with diesel and gasoline-like fuels," *Fuel,* vol. 89, no. 11, pp. 3416-3430, 2010, doi: 10.1016/j.fuel.2010.02.023

[7] Z. Wu, C. J. Rutland, and Z. Han, "Numerical optimization of natural gas and diesel dual-fuel combustion for a heavy-duty engine operated at a medium load," *International Journal of Engine Research,* p. 1468087417729255, 2017, doi: 10.1177/1468087417729255

[8] I. Brahma, C. J. Rutland, D. E. Foster, and Y. He, "A new approach to system level soot modeling," SAE Technical Paper 2005-01-1122, 2005, doi: 10.4271/2005-01-1122

[9] Y. He and C. J. Rutland, "Modeling of a turbocharged DI diesel engine using artificial neural networks," SAE Technical Paper 2002-01-2772, 2002, doi: 10.4271/2002-01-2772

[10] Y. He and C. J. Rutland, "Neural cylinder model and its transient results," SAE Technical Paper 2003-01-3232, 2003, doi: 10.4271/2003-01-3232

[11] J. Rezaei, M. Shahbakhti, B. Bahri, and A. A. Aziz, "Performance prediction of HCCI engines with oxygenated fuels using artificial neural networks," *Applied Energy,* vol. 138, pp. 460-473, 2015, doi: 10.1016/j.apenergy.2014.10.088

[12] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[13] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785-794: ACM, doi: 10.1145/2939672.2939785

[14] A. N. Tikhonov, V. I. A. k. Arsenin, and F. John, *Solutions of ill-posed problems*. Winston Washington, DC, 1977.

[15] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician,* vol. 46, no. 3, pp. 175-185, 1992, doi: 10.1080/00031305.1992.10475879

[16] A. Liaw and M. Wiener, "Classification and regression by randomForest," *R news,* vol. 2, no. 3, pp. 18-22, 2002.

[17] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications,* vol. 13, no. 4, pp. 18-28, 1998, doi: 10.1109/5254.708428

[18] 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 (IJCCE),* vol. 28, no. 4, pp. 61-70, 2009.

[19] A. Vaughan and S. V. Bohac, "A cycle-to-cycle method to predict HCCI combustion phasing," in *Proceedings of the ASME Internal Combustion Engine Division 2013 Fall Technical Conference, ICEF2013-19203*, 2013, doi: 10.1115/ICEF2013-19203

[20] A. Vaughan and S. V. Bohac, "An extreme learning machine approach to predicting near chaotic HCCI combustion phasing in real-time," *arXiv preprint arXiv:1310.3567,* 2013.

[21] A. Validi, J.-Y. Chen, and A. Ghafourian, "HCCI Intelligent Rapid Modeling by Artificial Neural Network and Genetic Algorithm," *Journal of Combustion,* vol. 2012, pp. 1-11, 2012, doi: 10.1155/2012/854393

[22] J. M. Alonso, F. Alvarruiz, J. M. Desantes, L. Hernndez, V. Hernndez, and G. Molt, "Combining Neural Networks and Genetic Algorithms to Predict and Reduce Diesel Engine Emissions," *IEEE Transactions on Evolutionary Computation,* vol. 11, no. 1, pp. 46-55, 2007, doi: 10.1109/TEVC.2006.876364

[23] I. Brahma and C. J. Rutland, "Optimization of diesel engine operating parameters using neural networks," SAE Technical Paper 2003-01-3228, 2003, doi: 10.4271/2003-01-3228

[24] M. Costa, G. M. Bianchi, C. Forte, and G. Cazzoli, "A Numerical Methodology for the Multi-objective Optimization of the DI Diesel Engine Combustion," *Energy Procedia,* vol. 45, pp. 711-720, 2014, doi: 10.1016/j.egypro.2014.01.076

[25] S. Coghlan *et al.*, "Argonne applications for the IBM Blue Gene/Q, Mira," *IBM Journal of Research and Development,* vol. 57, no. 1/2, pp. 12: 1-12: 11, 2013, doi: 10.1147/JRD.2013.2238371

[26] R. Ihaka and R. Gentleman, "R: a language for data analysis and graphics," *Journal of computational and graphical statistics,* vol. 5, no. 3, pp. 299-314, 1996, doi: 10.2307/1390807

[27] Y. Zhang, P. Kumar, M. Traver, and D. Cleary, "Conventional and Low Temperature Combustion Using Naphtha Fuels in a Multi-Cylinder Heavy-Duty Diesel Engine," *SAE International Journal of Engines,* vol. 9, no. 2016-01-0764, pp. 1021-1035, 2016, doi: 10.4271/2016-01-0764

[28] Y. Pei *et al.*, "CFD-Guided Heavy Duty Mixing-Controlled Combustion System Optimization with a Gasoline-Like Fuel," *SAE International Journal of Commercial Vehicles,* vol. 10, no. 2017-01-0550, 2017, doi: 10.4271/2017-01-0550

[29] K. Richards, P. Senecal, and E. Pomraning, "Converge theory manual," *Convergent Sciences Inc., Madison, WI, http://www. convergecfd. com,* 2014.

[30] Z. Han and R. D. Reitz, "Turbulence modeling of internal combustion engines using RNG κ-ε models," *Combustion science and technology,* vol. 106, no. 4-6, pp. 267-295, 1995.

[31] R. D. Reitz and R. Diwakar, "Structure of high-pressure fuel sprays," SAE Technical Paper 870598, 1987, doi: 10.4271/870598

[32] R. Reitz, "Modeling atomization processes in high-pressure vaporizing sprays," *Atomisation and Spray Technology,* vol. 3, no. 4, pp. 309-337, 1987.

[33] D. P. Schmidt and C. Rutland, "A new droplet collision algorithm," *Journal of Computational Physics,* vol. 164, no. 1, pp. 62-80, 2000, doi: 10.1006/jcph.2000.6568

[34] N. Frossling, "Evaporation, heat transfer, and velocity distribution in two-dimensional and rotationally symmetrical laminar boundary-layer flow," National Aeronautics And Space Admin Langley Research Center Hampton VA, 1956.

[35] A. A. Amsden, P. O'rourke, and T. Butler, "KIVA-II: A computer program for chemically reactive flows with sprays," Los Alamos National Lab., NM (USA)1989.

[36] C. Bae and J. Kim, "Alternative fuels for internal combustion engines," *Proceedings of the Combustion Institute,* vol. 36, no. 3, pp. 3389-3413, 2017, doi: 10.1016/j.proci.2016.09.009.

[37] Y.-D. Liu, M. Jia, M.-Z. Xie, and B. Pang, "Enhancement on a skeletal kinetic model for primary reference fuel oxidation by using a semidecoupling methodology," *Energy & Fuels,* vol. 26, no. 12, pp. 7069-7083, 2012, doi: 10.1021/ef301242b.

[38] J. B. Heywood, *Internal combustion engine fundamentals*. Mcgraw-hill New York, 1988.

[39] H. Hiroyasu and T. Kadota, "Models for combustion and formation of nitric oxide and soot in direct injection diesel engines," SAE Technical Paper 760129, 1976, doi: 10.4271/760129

[40] J. Nagle, "Oxidation of carbon between 1000-2000°C," in *Proceeding of the 5th Conference on Carbon, 1982*, 1982: Pergamon Press.

[41] A. Babajimopoulos, D. Assanis, D. Flowers, S. Aceves, and R. Hessel, "A fully coupled computational fluid dynamics and multi-zone model with detailed chemical kinetics for the simulation of premixed charge compression ignition engines," *International Journal of Engine Research,* vol. 6, no. 5, pp. 497-512, 2005, doi: 10.1243/146808705X30503

[42] P. Pal, Probst, P., Pei, P., Zhang, P. *et al.*, "Numerical Investigation of a Gasoline-Like Fuel in a Heavy-Duty Compression Ignition Engine Using Global Sensitivity Analysis," *SAE International Journal of Fuels and Lubricants,* vol. 10, no. 2017-01-0578, pp. 56-68, 2017, doi: 10.4271/2017-01-0578

[43] P. K. Senecal and R. D. Reitz, "Simultaneous reduction of engine emissions and fuel consumption using genetic algorithms and multi-dimensional spray and combustion modeling," SAE Technical Paper 2000-01-1890, 2000, doi: 10.4271/2000-01-1890

[44] E. C. Polley and M. J. Van der Laan, "Super learner in prediction," *U.C. Berkeley Division of Biostatistics Working Paper Series.* Working Paper 266, 2010.

[45] E. Polley, E. LeDell, C. Kennedy, S. Lendle, and M. van der Laan, "Package 'SuperLearner'," ed: CRAN, 2017.

[46] M. J. Van der Laan, E. C. Polley, and A. E. Hubbard, "Super learner," *Statistical applications in genetics and molecular biology,* vol. 6, no. 1, 2007, doi: 10.2202/1544-6115.1309

[47] S. Sapp, M. J. van der Laan, and J. Canny, "Subsemble: an ensemble method for combining subset-specific algorithm fits," *Journal of applied statistics,* vol. 41, no. 6, pp. 1247-1259, 2014 ,doi: 10.1080/02664763.2013.864263

[48] K. M. Mullen, "Continuous global optimization in R," *Journal of Statistical Software,* vol. 60, no. 6, pp. 1-45, 2014.

[49] C. N. Bergmeir, D. Molina Cabrera, and J. M. Benítez Sánchez, "Memetic Algorithms with Local Search Chains in R: The Rmalschains Package," American Statistical Association, vol.75, no. 4, 2016, doi: 10.18637/jss.v075.i04

[50] C. Bergmeir, D. Molina, and J. Benıtez, "Rmalschains: Continuous Optimization using Memetic Algorithms with Local Search Chains (MA-LS-Chains) in R," *Journal of statistical software*, 2012.

[51] D. M. Probst, "Optimization and Model Interrogation," Convergent Science Advanced Training Slides, 2017.

Acknowledgments

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory (Argonne). The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

This work was supported by the U.S. Department of Energy, Office of Science under contract DE-AC02-06CH11357. The research used resources of the Argonne Leadership Computing Facility (ALCF), which is a DOE Office of Science User Facility supported under contract DE-AC02-06CH11357. Fusion and Blues High Performance LCRC cluster facilities at Argonne National Laboratory were also used for some of the simulations.