From Data to Decisions: Development of Surrogate Models for Process Optimization

by

Bianca Williams

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 6, 2022

Keywords: surrogate models, optimization, random forests, multivariate adaptive regression splines

Approved by

Selen Cremaschi, Chair, B. Redd & Susan W. Redd Professor of Chemical Engineering
Mark Carpenter, Professor of Mathematics and Statistics
Peter He, Associate Professor of Chemical Engineering
Elizabeth Lipke, Mary and John H. Sanders Professor of Chemical Engineering

To my first teacher, Ms. Karen Saxton,
you always believed in me.

To my first research advisor, Dr. Elizabeth Lipke,
you started me on the journey.

To my first best friend, Amber Hicks,
you helped me see it through.

Abstract

Surrogate models are used to map input data to output data when the actual relationship between the two is unknown or computationally expensive to evaluate (Han & Zhang, 2012). Surrogate models can also be constructed for use in surrogate-based optimization when a closed analytical form of the relationship between input data and output data does not exist or is not conducive for use in traditional gradient based optimization methods. The overall goal of this dissertation is to comprehensively investigate and compare the performance of several different surrogate modeling techniques for both approximating functional relationships and surrogate-based optimization, and to link that performance to the characteristics of the data involved in the application. Using the results of the performance comparisons, surrogate modeling techniques are incorporated into a derivative-free optimization framework to use in the application of surrogate-based optimization of chemical processes.

The research activities described here focused on comparison of the performance of eight different surrogate modeling techniques on a collection of generated datasets and construction of a tool to provide recommendations for the appropriate modeling techniques for the datasets based only on the characteristics of the data being modeled. The surrogate modeling techniques include multivariate adaptive regression splines (MARS), random forests (RF), single hidden layer feed forward artificial neural networks (ANN), extreme learning machines (ELM), Gaussian process regression (GP), support vector machines (SVM), Automated Learning of Algebraic Models using Optimization (ALAMO), and radial basis function networks (RBFN). In general, multivariate adaptive regression splines (MARS), artificial neural networks (ANN), and Gaussian process regression (GP) provide the most accurate predictions for approximation, and RF models locate the optimum of a dataset most often. Several of the surrogate modeling techniques were applied

to the prediction of the outcomes of cardiac differentiation experiments. RF and GP models were found to provide the most accurate predictions of those outcomes. With feature selection and data-driven modeling using the surrogate modeling techniques, we were able to build models that could predict insufficient yield for a bioreactor differentiation on day seven (out of 10) of the differentiation protocol with up to a 90% accuracy and a 90% precision, using only 16% of the collected bioreactor features.

Based on the results of the surrogate model comparison study, we identified attributes of datasets appropriate for selecting surrogate models for both surface approximation and surrogate-based optimization. Using these attributes, a recommendation tool, PRESTO, was constructed to recommend surrogate modeling techniques for approximating a dataset with 91% accuracy and 90% precision and for performing surrogate based-optimization with 98% accuracy and 99% precision. A surrogate-based, derivative-free optimization algorithm, pyBOUND, was developed for the solution of expensive black-box optimization problems. pyBOUND combines the capabilities of random forest models to accurately locate the optima of a wide variety of problems with MARS models' high accuracy for making predictions.

Acknowledgements

First, I thank my major professor, Dr. Selen Cremaschi for her belief in me from day one and for her unending support, encouragement, and patience over the last five years. Throughout the many challenges I have experienced over this process, Selen has been unwavering in her belief in me and her support. I am a better researcher and better overall because of her guidance. Additionally, I thank the other members of my committee: Dr. Elizabeth Lipke, who first introduced me to research and the associated possibilities many years ago and has maintained a mentoring presence in my life since then, Dr. Peter He and Dr. Mark Carpenter who have provided input and advice throughout my dissertation process, and Dr. Cheryl Seals for serving as my university reader and giving valuable input on my dissertation.

I would like to give special thanks to my aunt, Tracy Williams, who has been my biggest supporter throughout this process in every way. Graduate school can be a large financial burden, and she has never let me starve or be homeless. I thank my mother and father for their support throughout this process. I thank my brother and sister, Tony and Brea Williams, for being my biggest cheerleaders and never letting me dwell on the stress of research for too long and for always showing up for me.

I also give special thanks to Dr. Logan Dawson, Dr. Benita Bamgbade, and Dr. Stacey Jackson, my doctor friends. Graduate school can be challenging for all, but there are additional, unique challenges and struggles associated with being African American. I greatly appreciate them for being a support system and for showing me that finishing out the degree is possible under any circumstances. Special shout out as well to the Auburn Mocha Doc crew for struggling with me.

One of the people I dedicated this work to is my dearest friend, Amber Hicks, who forced me to start interacting with other people in high school. She has been my greatest source of

motivation over these last five years. Although she is no longer with me, I can still hear her encouragement and feel her smile on me. I know she is proud of me somewhere and will be the loudest one yelling when I walk across the stage. I would also like to acknowledge her parents LaWanda and Jeff Hicks for their continued support and encouragement for me throughout the years and for sharing Amber with me while she was here.

Lastly, I would like to thank the large host of extended family, aunts, uncles, cousins, godparents, church members, friends and everyone else that has played a role in my life throughout the years. I would not have made it to this milestone without their continued prayers and support. It has been quite a ride.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Full Name
ALAMO	Automated Learning of Algebraic Models for Optimization
ANN	Single Layer Feed Forward Artificial Neural Network
CM	Cardiomyocyte
CVD	Cardiovascular diseases
dd	Differentiation day
DFO	Derivative Free Optimization
EI	Expected improvement function
ELM	Extreme Learning Machine
FN	False negative
FP	False positive
FS	Feature set
GP	Gaussian Process Regression
hiPSCs	Human induced pluripotent stem cells
LOO	Leave one out
MARS	Multivariate Adaptive Regression Splines
MAS	Mixed adaptive sampling
MC	Monte Carlo
MCC	Matthews correlation coefficient
MILP	Mixed integer linear problem
MINLP	Mixed integer nonlinear problem
NLP	Nonlinear problem
nRMSE	Normalized root mean squared error
ODIN	Optimization directed incremental sampling
PC	Principal component
PCA	Principal component analysis
PRESTO	Predictive Recommendation of Surrogate Models to Approximate and Optimize
pyBOUND	Python-Based Black Box Optimization using Random Forests
RBFN	Radial Basis Function Network
RF	Random Forest
SSE	Sum of Squared Errors
SVR	Support Vector Regression
TN	True negative
TP	True positive

Chapter 1 - Introduction

Surrogate models, also known as response surfaces, black-box models, metamodels, or emulators, are simplified approximations of more complex, higher order models (C. Wang et al., 2014). These models are used to map input data to output data when the actual relationship between the two is unknown or computationally expensive to evaluate (Han & Zhang, 2012). Surrogate models can also be constructed for use in surrogate-based optimization when a closed analytical form of the relationship between input data and output data does not exist or is not conducive for use in traditional gradient based optimization methods. Surrogate modeling techniques are of particular interest where high-fidelity, thus expensive, simulations are used (Han and Zhang, 2012), for example, in computational fluid dynamics (CFD) or computational structural dynamics (CSD). Surrogates are also of interest when the fundamental relationship between design variables and output variables is not well understood, such as in the design of cell or tissue manufacturing processes (Du et al., 2016; Sokolov et al., 2017; Williams, Lobel, et al., 2020).

Surrogate modeling techniques have been receiving increasing attention in a wide range of applications, for example, in the optimization of process design, scheduling, and control (Burnak et al., 2019). They have successfully been used for both regression and classification tasks. Surrogate models have been used in several recent applications in process systems engineering and the manufacturing industry, including for optimization of a sulfur recovery unit (Rahman et al., 2019), fault detection (Quiroz et al., 2018; Zhang et al., 2018), and surrogate-based optimization of energy consumption in carbon fiber production line (Golkarnarenji et al., 2018).

Although several machine learning and regression techniques have been developed for surrogate model construction, there has been little work on how to best select the appropriate model for a particular application for either surface approximation or optimization. Surface

approximation refers to the application of using a surrogate model to mimic the overall behavior or response of an underlying model. In surrogate-based optimization, a surrogate model can be constructed to represent the objective function or any constraints that may be computationally expensive to evaluate or are unavailable in analytical form. The constructed surrogate can be used as a closed functional form in traditional gradient-based optimization methods.

With all the surrogate modeling techniques currently available, there is a need for a systematic method of selecting the appropriate technique for a given application. The overall goal of this research is to comprehensively investigate and compare the performance of several different surrogate modeling techniques for both approximating functional relationships and surrogate-based optimization, and to link that performance to the characteristics of the data involved in the application. Using the results of the performance comparisons, surrogate modeling techniques are incorporated into a derivative-free optimization framework to use in the application of surrogate-based optimization.

1.1 Objectives

This dissertation will:

(1) Explore and compare the effects of data characteristics on performance of several surrogate modeling techniques.

(2) Develop a tool for systematic recommendation of surrogate modeling techniques for the purposes of surface approximation and surrogate-based approximation.

(3) Develop a derivative-free optimization framework incorporating surrogate modeling techniques.

1.2 Organization

The dissertation is organized as follows. In Chapter 2, Sections 2.1 and 2.2 present background information about selected surrogate modeling techniques and their construction and relevant practical applications. Section 2.3 provides a literature review on derivative-free optimization (DFO) algorithms. Chapter 3 describes work on development of surrogate models for the prediction of the outcomes of cardiac stem cell differentiation experiments. Section 3.1 provides relevant background information on cardiac differentiation and a literature review of previous applications of machine learning in other cell and protein production applications. Sections 3.2, 3.3, and 3.4 present the computational experiments performed for the cardiac study and their results.

In Chapter 4, results are presented for a comparison of the performance of the eight surrogate modeling techniques on a variety of test functions. Section 4.1 gives a detailed description of the test functions generated and used here and throughout the dissertation. Section 4.2 documents the computation methods used in the comparison study, and comparison performance results are presented in section 4.3. Chapter 5 presents the development of PRESTO (Predictive Recommendation of Surrogate models TO approximate and optimize), a framework that provides recommendations for surrogate modeling techniques to use based on dataset characteristics. Section 5.2 defines the dataset characteristics, or attributes, used to describe datasets. Section 5.3 focuses on the construction and training of the PRESTO framework, and Section 5.4 defines the criteria used to evaluate the quality of the model selections made by PRESTO. Section 5.5 discusses the data attributes determined to be relevant in selecting surrogates for the study. A case study is described in Section 5.6, and results for PRESTO's model selection performance for the training and case studies are provided in Section 5.7. Section 5.7 also includes a discussion of a

similarity metric that can be used for determining if a set of data is appropriate for use with PRESTO.

Chapter 6 describes a study applying the random forest surrogate modeling technique for surrogate-based optimization. Section 6.1 describes the unique structure of random forest models and their resulting optimization problem structure. Section 6.2 describes the computational experiments carried out for the optimization study, and results are presented in Section 6.3. Chapter 7 describes the development and performance evaluation of pyBOUND (PYthon-based Black box Optimization Using raNDom forests), a surrogate-based derivative-free optimization algorithm.

Note that parts of the contributions described in this dissertation have been previously published in three journal papers (Williams & Cremaschi, 2021b; Williams, Lobel, et al., 2020; Williams et al., 2021) and three conference papers (Williams & Cremaschi, 2019, 2021a; Williams, Halloin, et al., 2020).

Chapter 2 – Literature Review

Surrogate modeling techniques have been receiving increasing attention in a wide range of applications, for example, in the optimization of process design, scheduling, and control (Burnak et al., 2019). They have successfully been used for both regression and classification tasks. Surrogate models have been used in several recent applications in process systems engineering and the manufacturing industry, including for optimization of a sulfur recovery unit (Rahman et al., 2019), fault detection (Quiroz et al., 2018; Zhang et al., 2018), and surrogate-based optimization of energy consumption in carbon fiber production line (Golkarnarenji et al., 2018).

2.1 Surrogate Model Construction

Construction of a surrogate model is comprised of three steps: (1) selection of the sample points, (2) optimization or "training" of the model parameters, and (3) evaluation of the accuracy of the surrogate model (C. Wang et al., 2014). Although several machine learning and regression techniques have been developed for surrogate model construction, there has been little work on how to select the appropriate model for a particular application for either surface approximation or surrogate-based optimization. Surface approximation refers to the application of using a surrogate model to mimic the overall behavior or response of an underlying model. In surrogate-based optimization, a surrogate model can be constructed to represent the objective function or any constraints that may be computationally expensive to evaluate or are unavailable in analytical form. The constructed surrogate can be used as a closed functional form in traditional gradient-based optimization methods.

Selection of an appropriate number of sample points and sampling method to generate those samples is a critical step in the construction of a surrogate model. In general, a higher number of sample points offers more information about the underlying model being approximated,

although with a higher computational expense. For low-order functions, after reaching a certain sample size, increasing the number of sample points does not contribute much to the approximation accuracy (G. G. Wang & Shan, 2007). Previous studies have investigated the effects of sample size and sampling method on some of the surrogate modeling techniques being studied specifically, including Gaussian process regression (Afzal et al., 2017; Burnaev & Zaytsev, 2015; Iooss et al., 2010) and radial basis function networks (Afzal et al., 2017), as well as on surrogate modeling accuracy in general (Davis et al., 2017). The results of these studies indicate that the accuracy of a surrogate model is dependent upon the number and distribution of samples used in its construction.

2.1.1 Sampling Methods for Surrogate Model Construction

In general, sampling methods can be categorized into two types: one-shot sampling and sequential sampling. In one-shot experimental design, all the experimental points for building a model are generated prior to the execution of the experimental design for the model construction. Examples of one-shot sampling methods include Latin Hypercube sampling (Mckay, 1992) and full factorial design (Das & Dewanjee, 2018). Although these one-shot techniques are very commonly used, they can result in under/oversampling and thus, poor system approximations (Crombecq et al., 2011; Garud et al., 2017b). Sequential sampling attempts to use the minimum number of sample points necessary by starting with a small number of samples and slowly increasing the sample size until the performance of the surrogate model reaches some desired level of a performance metric (Eason & Cremaschi, 2014). Fewer samples translate to decreased computational time required for data collection.

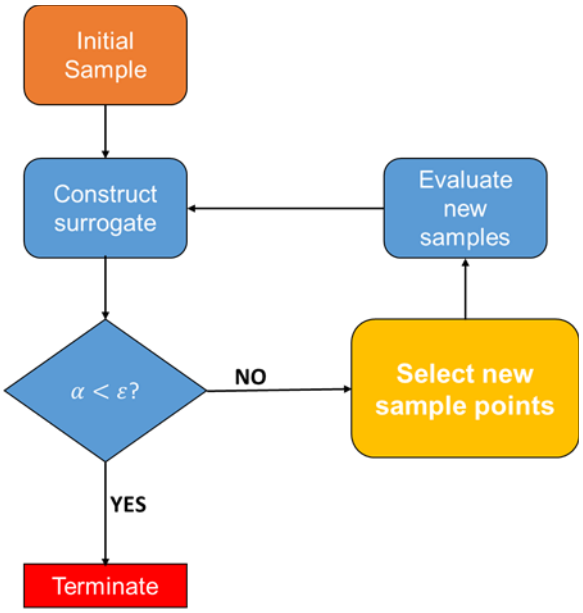


Figure 2.1 - Sequential sampling method for constructing a surrogate model

Figure 2.1 gives an example of a general sequential sampling algorithm. First, an initial set of inputs are generated, usually with a space-filling experimental design, and the experimental output to be modeled is evaluated at those points to generate an initial sample set. Next, additional sample points are selected based on a criterion, and the output is evaluated at these new points. The new points are added to the existing sample set, and a surrogate model is constructed. A performance metric, α, that quantifies the improvement of the surrogate model from the previous iteration is checked against a target value, ε. When α falls below ε, then the sample size is deemed to be sufficient, and the sampling is terminated.

The performance of a sequential sampling algorithm is dependent on selecting appropriate sample locations. Two search strategies may be used for determining the locations of new samples: 1. New samples should be located far enough away from existing ones to avoid redundant samples and adequately fill the design space (exploration) and 2. New samples should be placed in regions of the design space that capture nonlinearities or other deviations from the typical behavior of the

underlying black-box model (exploitation). The contrast between these two approaches is known as the exploration vs. exploitation problem (Crombecq et al., 2011). Several methods have been developed to address this issue of balancing the trade-off between exploration and exploitation, adaptively adding samples to improve the surrogate model performance (Eason & Cremaschi, 2014; Garud et al., 2017b; Hu et al., 2018; Nentwich & Engell, 2019).

2.2 Surrogate Modeling Techniques Considered

Eight commonly used surrogate modeling techniques were chosen for consideration for this work. These techniques include Automated Learning of Algebraic Models using Optimization (ALAMO), single hidden layer feed-forward Artificial Neural Networks (ANN), Extreme Learning Machines (ELM), Gaussian Process Regression (GPR), Multivariate Adaptive Regression Splines (MARS), Radial Basis Function Networks (RBFN), Random Forests (RF), and Support Vector Machine Regression (SVR).

2.2.1 Automated Learning of Algebraic Models for Optimization (ALAMO)

Automated learning of algebraic models (ALAMO) uses a linear summation of nonlinear transformations of the input data to predict output values. Possible nonlinear transformations include polynomial, exponential, logarithmic, ratio, and trigonometric functions (Cozad et al., 2014). The nonlinear transformations allowed for ALAMO models trained for this work were sine, cosine, exponential, logarithmic, polynomial functions. Given a dataset, the approach begins by building a low-complexity, linear model composed of explicit nonlinear transformations of the input variables. Then, the method iteratively refines the model by solving an optimization problem at each iteration to minimize (or maximize) a user-designated error metric. It should be noted that the adaptive sampling scheme of ALAMO is not used in this study. ALAMO is one of the few surrogate modeling techniques developed directly by the chemical engineering community.

2.2.2 Artificial Neural Networks

Artificial neural networks attempt to mimic the behavior of neurons in the brain. The models consist of an input and an output layer that are connected by a number of hidden layers in between. The artificial neurons have weights and biases that create a network between the layers, with the activation function in the hidden layer determining whether or not a neuron will "fire" and produce a signal (Haykin, 2009). Training of a neural network refers to the process that identifies the values of the weights and biases. Three different types of artificial neural networks are considered here, all with a single hidden layer: a feed-forward artificial neural network with a hyperbolic tangent activation function (ANN), an extreme learning machine (ELM), and a radial basis function network (RBFN). In an ELM, the weights between the input layer and hidden layer are randomly assigned, and the weights between the hidden layer and the output layer are fit using linear regression or other regression techniques (Huang et al., 2006). The activation function used in both the ANN and ELM models is a hyperbolic tangent function. An RBFN is a neural network with a radial basis function as the activation function in the hidden layer (Gomm & Yu, 2000). First, the network calculates the Euclidean distance between the input weights and input values. Then it passes those distances through the Gaussian radial basis activation function. The form of the radial basis function is shown in Eqs. (2.1) and (2.2),

$$r = \|x - x'\| \tag{2.1}$$

$$\varphi(r) = e^{-(\varepsilon r)^2} \tag{2.2}$$

where the Euclidean distance, r, between points x and x', is used to calculate the radial basis function, $\varphi(r)$, with the shape tuning parameter ε.

Artificial neural networks have been widely developed and used in a variety of chemical engineering applications. For example, ANNs have been used to estimate the thermodynamic

properties of four binary refrigerant systems (Nikkholgh et al., 2010), and at a larger scale, to heat generated in commercial electric vehicle battery packs (Arora et al., 2017). ELMs have been used to develop soft sensors for chemical processes (He et al., 2016).

2.2.3 Gaussian Process Regression (GPR)

Gaussian process regression (GPR) is a method of interpolation for which the interpolated values are modeled by a Gaussian process governed by prior covariances. Under suitable assumptions on the priors, GPR gives the best linear unbiased prediction of the intermediate values (Rasmussen & Nickisch, 2010). GPR uses a kernel function as measure of similarity between points to predict the value for an unseen point from the training data (Rasmussen & Williams, 2005). GPR is widely used in chemical engineering applications, including modeling and monitoring batch chemical reactors (Masampally et al., 2018; L. Zhou et al., 2015), and uncertainty estimations in erosion rate predictions (Dai et al., 2019). The radial basis function (Eqs. (2.1) and (2.2)) is used as the kernel function for all GP models trained for this work.

2.2.4 Multivariate adaptive regression splines (MARS)

Multivariate adaptive regression spline (MARS) models are made up of a linear summation of basis functions. The three types of possible basis functions are a constant, a hinge function (or "spline"), or a product of two or more hinge functions. The training of a MARS model starts with an initial model that is a basis function equal to the mean of the data outputs. On the first pass, the model overfits to the data, adding basis functions to reduce the sum of the squared errors (SSE) between the given and predicted outputs. Then, a backward, pruning pass is performed to remove terms that have little effect on the SSE until the best model is identified based on cross validation criteria (Friedman, 1991). Recently, MARS models have been used in process systems applications

including optimization of a sulfur recovery unit (Rahman et al., 2019) and surrogate-based sensitivity analysis of a wastewater treatment plant (Al et al., 2019). MARS models have also been successfully implemented in medical applications (Bhat et al., 2013).

2.2.5 Random Forests (RF)

Random forests are machine learning models that make output predictions by combining outcomes from a sequence of regression decision trees, called forests. Each tree is constructed independently and depends on a random vector sampled from the input data, with all the trees in the forest having the same distribution. The predictions from the forests are averaged using bootstrap aggregation and random feature selection (Breiman, 2001). The value that is output for a tree for given inputs is the value of the final leaf node reached, and the output value for the entire RF model is the average value of the outputs for every decision tree in the forest.

Random forests have successfully been used for both regression and classification tasks, performing with high prediction accuracy for both small sample sizes and high dimensional data. They are capable of fitting non-linear data with a minimal number of parameters to tune (Biau & Scornet, 2016). The models have been used in several recent applications in the manufacturing industry, including for fault detection (Puggini et al., 2015; Quiroz et al., 2018; Zhang et al., 2018), prediction of mechanical failures (Wu et al., 2017), and prediction of manufacturing product properties (Maudes et al., 2017). Other areas of research where random forest models have been employed for approximation include development of new pharmaceutical molecules (Svetnik et al., 2004) and thermodynamic property estimation (Palmer et al., 2007).

2.2.6 Support Vector Machine Regression (SVR)

Support vector machine regression transforms input data into m-dimensional space and attempts to construct a set of hyperplanes so that the distance from it to the nearest data point on each side of the plane is maximized using kernel functions (Drucker et al., 2002). The kernel functions transform the data into a higher dimensional feature space to make it possible to perform the linear separation. A recent example of an application of SVM models is using them for surrogate-based optimization of energy consumption in carbon fiber production line (Golkarnarenji et al., 2018).

2.3 Comparison and Selection of Surrogate Modeling Techniques

The current common practice for choosing a model form from the many available techniques relies on process-specific expertise or expensive trial-and-error methods. When selecting a surrogate model with user expertise, only a small subset of the many possible techniques that the user is most familiar with may be considered as candidates. This selection method, as well as trial and error, which is limited by computational resources, may fail to exploit the large pool of surrogate modeling techniques available and lead to a sub-optimal model selection. A systematic, automated procedure for selecting the appropriate surrogate model for a given application would avoid this issue.

Recent advances in automating the surrogate model selection process include the development of the tool Concurrent Surrogate Model Selection (COSMOS), which uses a genetic search algorithm with sequential k-fold cross-validation to identify the best model for an application (Mehmani et al., 2018). While this method allows users to explore a wide range of candidate surrogates to select the best one, it still involves a considerable computational expense for training multiple models. Progress has been made in recent works in generalizing the process

for selecting a surrogate model to approximate a surface by using meta-learning approaches to build selection frameworks, avoiding expensive trial-and-error methods (Cui et al., 2016; Garud et al., 2018). These meta-learning approaches rely on the knowledge pyramid, where the selection framework learns how to best select surrogate models based on past modeling computational experiments results (Vilalta & Drissi, 2002). These frameworks provide "best" recommendations for surrogate modeling techniques based on characteristics, or attributes, calculated from the modeled data. Furthermore, the framework developed by Garud et al. (2018) gives a ranking of all the considered surrogate models based on the predicted accuracy of the model. However, neither framework takes model complexity into account, which can lead to overfitting, or considers that multiple models might perform similarly to the one identified as best in terms of their accuracies. The selection of surrogate models for surrogate-based optimization remains an open challenge.

2.4 Derivative-Free Optimization using Surrogate Models

Optimization is required for several chemical engineering applications, including process design and process synthesis, operations, and supply chain management. These applications usually involve complex, high-fidelity simulations and/or physical experiments, which can both require significant resources in terms of cost and time, as well as a large computational expense to collect data. Optimization using traditional gradient-based methods is impractical for these applications because gradient information is not readily available, and approximating gradients is infeasible due to the required expense for multiple simulation evaluations or experiments. In addition, the direct use of deterministic global optimization methods is restrictive in these cases because the computational cost for obtaining data limits the total number of model runs necessary to optimize the system efficiently (Conn et al., 2009; Forrester et al., 2008). To overcome these challenges, derivative-free optimization methods can be employed.

Derivative-Free Optimization (DFO) algorithms rely on search heuristics based around improving the current best set of decision variables (Conn et al., 2009). The general framework of a DFO algorithm is illustrated in Figure 2.2. Each algorithm begins by using an initial sampling strategy to generate an initial set of decision variables. From there, the variable sets are passed to the problem, model, or simulation, which is treated as a black-box model, and the objective function value corresponding to those decision variables is evaluated. The objective function values are then passed back to the DFO algorithm, where the algorithm-specific search heuristic begins. The search heuristic uses the input-output pairs of the evaluated decision variable sets and corresponding objective function values to then determine the next set(s) of decision variables to evaluate. This process is iterated until a termination criterion has been reached. There are several termination criteria that can be evaluated for a DFO algorithm, including a maximum number of black-box evaluations, a minimum search step size, or exceeding a specified wall time (Rios & Sahinidis, 2013). Comprehensive reviews and comparisons of available DFO algorithms can be found in Kolda et al. (2003), Rios and Sahinidis (2013), and Boukouvala et al. (2016).

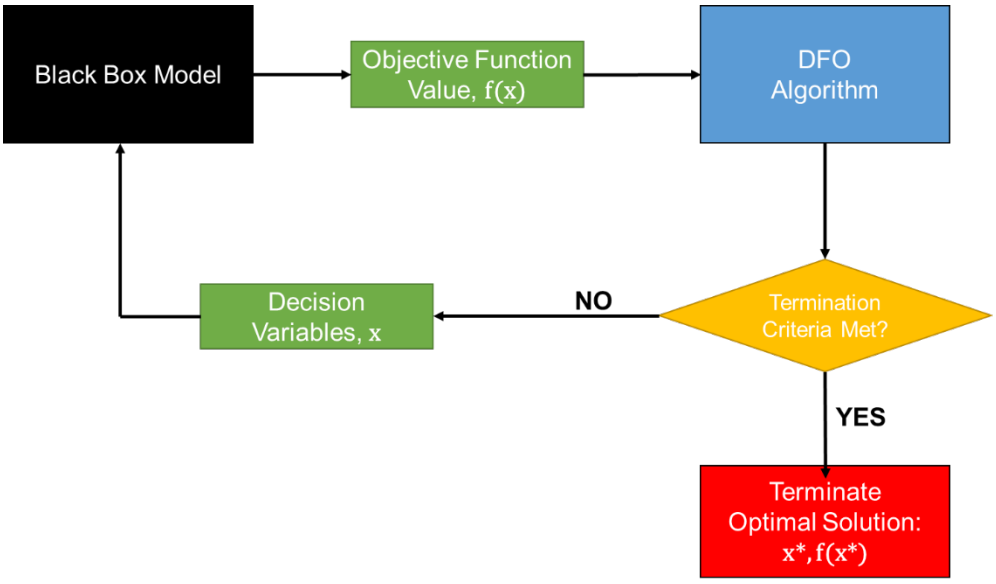


Figure 2.2 - General derivative-free optimization framework

DFO algorithms can be separated into categories based on the method they use to search the design space of the problem (direct-search or model-based) and on whether they search the entire design space (global search) or a local sub-region of it (local search) (Boukouvala et al., 2016). Model-based searches employ the use of surrogate models to assist in the search for optimal decision variables. Examples of model-based search DFO algorithms include the development of a kriging (Boukouvala & Ierapetritou, 2014) and radial basis function (Le Thi et al., 2012) based DFO algorithms for optimization of expensive constrained problems.

There has been considerable progress in developing approaches for derivative-free optimization by using surrogate models to approximate any explicitly unknown relationships in the systems of interest (Bajaj et al., 2018; Boukouvala et al., 2017; Rios & Sahinidis, 2013). The aim of the surrogate approximations is to guide the search toward the optimum of the original model. Existing literature on model-based DFO algorithms predominantly employs local optimization methods for the optimization of the formed surrogate approximations (Conn et al.,

2009). In addition, they can be difficult to scale to solve problems with high dimensions (Bhosekar & Ierapetritou, 2018b; Qian et al., 2016b).

Chapter 3 - Data-Driven Surrogate Model Development for Cardiomyocyte Production Experimental Outcome Prediction

Cardiovascular diseases (CVD) are the leading cause of death worldwide, meaning there are more deaths annually due to CVD than any other cause (Roth et al., 2020). These diseases can lead to heart attacks, which can result in the loss of more than one billion heart cells, leading to congestive heart failure (Kempf, Andree, et al., 2016). Patients who suffer from advanced stages of heart failure have a poor prognosis for survival, and the large disparity between numbers of donors and recipients leaves few viable treatments. Artificial prosthetic hearts and heart assist devices have demonstrated some success in prolonging the lives of patients receiving treatment, but their development is slow and clinical trials have been limited. Due to the nature of heart transplants and the stigma surrounding artificial organs, engineered heart tissue may provide an encompassing treatment for heart failure (Kempf, Andree, et al., 2016).

Mature cardiomyocytes, the contracting cells in the heart, are some of the least regenerative cells in the body. This characteristic carries over into the laboratory environment and thus limits *in vitro* expansion capabilities of cardiomyocytes. Difficulties in direct culture of cardiomyocytes can be overcome by differentiation from human pluripotent stem cells (hiPSCs) (Kempf, Andree, et al., 2016). The indefinite turnover potential of pluripotent cells allows for the expansion of large quantities for differentiation into therapeutic engineered tissues. However, the differentiation of hiPSCs into specific cell types is a highly complex and costly process that is sensitive to the impact of a high number of factors (Gaspari et al., 2018), and significant difficulties exist in reliably and consistently producing the large number of cardiomyocytes needed for therapeutic purposes (Kempf, Andree, et al., 2016).

Data-driven modeling with machine learning techniques has the potential to identify factors and patterns that most significantly affect the outcomes of these differentiation experiments. Previously, machine learning techniques have successfully been used to identify key factors and assist in the optimization of the production of several proteins and cell lines (Sokolov et al., 2017; Y. Zhou et al., 2018). The goal of this work is to use machine learning techniques to identify key process parameters to be used in predictive modeling of bioreactor cardiac differentiation outcomes. The high number of experimental factors influencing the differentiation results in a large set of possible inputs to be considered for modeling. This high data dimensionality, in addition to the low number of data points due to the time-consuming nature of these experiments, represents significant challenges for modeling the differentiation process. The specific aim is to use machine learning models to predict whether or not the cardiomyocyte content at the end of differentiation process will be sufficiently high. We define *insufficient* production as having a cardiomyocyte content on the tenth day of differentiation (dd10) that is less than 90%, meaning less than 90% of the cells produced at the end of the differentiation are cardiomyocytes. Predicting if the cardiomyocyte content will be *insufficient* before the end of the differentiation will provide cost and time savings, as each day the differentiation continues requires significant resources.

Using existing data from bioreactor experiments, we have applied feature selection techniques, including correlations, principal component analysis, and built-in feature selection method in machine learning models, to identify the conditions in the bioreactor, which we define as bioreactor features, that are most influential and predictive of the cardiomyocyte content. Bioreactor features considered include values related to the cell concentration, size of cell aggregates, pH, dissolved oxygen concentration, and concentrations and timings of certain nutrients, such as glucose, and small molecules known to direct the differentiation. We then used

the identified features as inputs to build models to classify the resulting cardiomyocyte content of a particular bioreactor run as being *sufficient* or *insufficient* to justify continuing with the differentiation. The general process for this study is summarized in Fig 3.1.

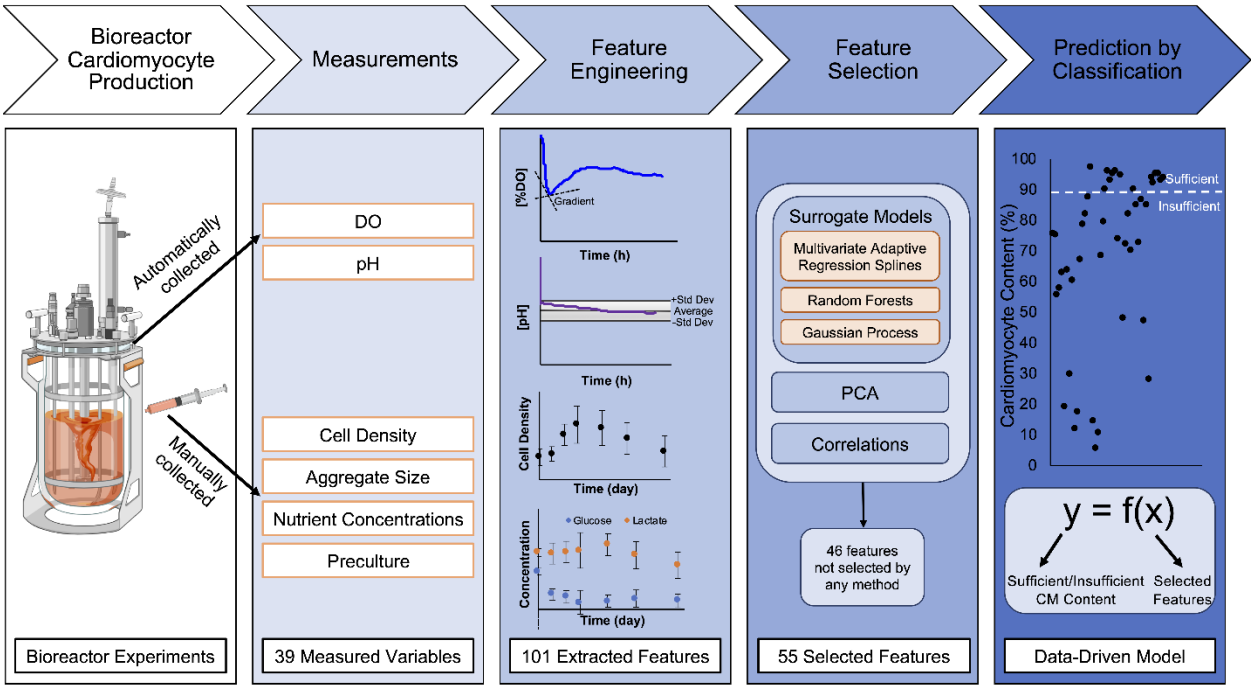


Figure 3.1 – Cardiomyocyte study summary

3.1 Computational Experiments and Theory

3.1.1 Experimental Data Collection

Experimental data were generated and collected from 58 cardiac differentiation experiments (Halloin et al., 2019). The differentiation experiments were carried out in chemically defined conditions in stirred tank bioreactors. Details of the experiments are described in Halloin et al. (2019). The set of independent variables includes experimental conditions such as the rotation speed in the bioreactor and measurements such as differentiation day dependent cell densities and aggregate sizes, and continuous time measurements of dissolved oxygen (DO) concentration and pH. The set of independent variables measured from the experiments was expanded to include

engineered features such as estimated gradients in cell densities and DO concentrations, resulting in a total of 101 variables, which we refer to as bioreactor features. The dependent variable is the percentage of the cells in the bioreactor that have differentiated into cardiomyocytes, or the cardiomyocyte content, on the last day of the differentiation experiment, dd10. Data from 42 of the experiments were designated as training data and used for feature selection and classification model construction. The remaining experiments were reserved as test data for testing the classification models.

3.1.2 Feature Engineering

Experimental data used in computational analysis and model development were collected from 58 cardiac differentiation processes in bioreactors; notably, all processes performed in the relevant experimental setup were included in the study without any type of pre-selection procedure to explicitly exclude any investigator-dependent bias. Each of the differentiation processes represents a single experimental datapoint to be used for model construction. In the first step, data sets from 42 of these processes were randomly chosen and used for constructing predictive models, while data sets from the remaining 16 processes were reserved for testing the models' performance.

From the data, a set of potential input variables, which we refer to as "bioprocess features", for use in predictive models was generated with the goal of this set fully describing the experimental conditions over the entire differentiation process. For model construction using machine learning, a feature is an individual measurable or derived (using measured properties) property of the system that is being modeled. Available experimental conditions included the rotation speed in the bioreactor and measurements such as differentiation day (dd) dependent cell densities, aggregate sizes, and nutrient concentrations, and measurements of DO concentration and pH over the course of the experiment.

The DO concentration and pH measurements were included as features by averaging their values over each day of the differentiation. Additional features were engineered from this data, as well as other time-dependent measurements, to capture how the conditions in the bioreactor were changing over time. These additional features were generated by estimating time gradients and second derivatives for the cell density, aggregate size, DO concentration, and pH measurements, resulting in a final set of 101 potential bioprocess features. The full list of bioprocess features is provided in Appendix A. Time gradients and second derivatives were estimated using Eqs. (3.1) and (3.2),

$$g_{t_i} = \frac{y_{t_i} - y_{t_{i-1}}}{t_i - t_{i-1}} \tag{3.1}$$

$$h_{t_i} = \frac{g_{t_i} - g_{t_{i-1}}}{t_i - t_{i-1}} \tag{3.2}$$

where g_{t_i} and h_{t_i} are the gradient and second derivative, respectively, of bioreactor condition y at timepoint t_i.

3.1.3 Feature Selection Methods

Because of the large number of features (101) compared to the number of experimental data points (58), feature selection was performed on the available data to discover which of the bioprocess features were most influential and predictive of the cardiomyocyte content on dd10. The feature selection methods employed include correlation coefficients, PCA, and the built-in feature selection capabilities of the machine learning techniques investigated for predictive modeling. Two sets of features, Feature Set 1 and Feature Set 2, were considered, each with bioprocess features measured at earlier points in the differentiation process. Feature Set 1 consists

of all the collected bioprocess features measured up until the seventh differentiation day 7 (dd7). Feature Set 2 consists of bioprocess features measured up until the fifth differentiation day 5 (dd5).

The dd7 and dd5 timepoints were chosen in order to use as much data as possible without using any data near the endpoint of the differentiation, such as dd9. Preliminary proof-of-concept studies revealed that classification was possible using data collected up to and including dd7, initiating analysis investigating the possibility of earlier predictions. Based on this analysis, dd5 was chosen as the earliest possible timepoint, as classification using data from earlier points in the differentiation did not yield satisfactory predictive capabilities.

3.1.3.1 Correlations

Pearson correlation coefficient

The Pearson correlation coefficient measures the linear relationship between two variables. Its value ranges from -1 to 1. A value of -1 corresponds to a perfect negative linear relationship between the variables, while a value of 1 indicates a positive linear relationship. A value of 0 demonstrates no linear correlation between the variables (Soper et al., 1915).

Spearman correlation coefficient

The Spearman correlation coefficient measures the strength and direction of a monotonic relationship between two variables. Its value ranges from -1 to 1. A value of 0 indicates no correlation between the variables. Values of -1 or 1 indicate a perfect negative or positive correlation, respectively (Spearman, 1904).

3.1.3.2 Principal component analysis (PCA)

The principal component analysis is a statistical dimension reduction tool. The method transforms a set of possibly correlated variables into uncorrelated principal components (PCs). It

identifies a new set of orthogonal axes in the direction of the highest variance of the data. Each of the axes, which is a linear combination of original axes, represents a PC. Principal components are assigned in ordinal format, with the first PC explaining the highest percentage of the variance and the last PC the least. The PCs with the lower ranks are generally not considered in further analysis reducing the number of dimensions while preserving much of the original variance (Hotelling, 1933).

3.1.3.3 Machine learning techniques

Multivariate adaptive regression splines (MARS) models are nonparametric statistical models that consist of a linear summation of basis functions (Friedman, 1991). In general, basis functions are either a constant, a hinge function, or the product of two or more hinge functions. For the MARS models trained in this study, the Sci-Kit Learn pyEarth software package was used (Pedregosa et al., 2011). Detailed information on MARS models and the other machine learning techniques described in this section are provided in Chapter 2.

Random forests (RFs) are a machine learning method that utilizes a set of decision trees for predicting an output based on input data. Each tree is built independently based on a random subspace of the training data. The final output of a random forest model is determined by averaging the output value of every tree in the forest (Breiman, 2001). The features are selected according to the importance level calculated by the random forest model. The importance level is based on the impact of a feature on improving the separation of the data in each decision node of the tree. For the RF models trained in this study, the Sci-Kit Learn RandomForestRegressor software package was used to train forests with 5 trees (Pedregosa et al., 2011).

Gaussian process regression (GPR) is a nonparametric machine learning method where the prediction of the output corresponding to an unknown input is calculated based on a weighted

average of outputs for known inputs using a similarity metric: the kernel function (Rasmussen and Williams, 2005). The kernel function used for all GPR models in this paper is a radial basis function.

GPR can be used for feature selection with its built-in automatic relevance determination (ARD) method. Further sensitivity analysis (Eq. (4)) on the ARD results (Blix and Eltoft, 2018) provides an even greater separation of the features for selection. For the GPR models trained in this study, the Sci-Kit Learn GaussianProcessRegressor software package was used (Pedregosa et al., 2011).

3.2 Cardiomyocyte Content Classification

A binary process classification based on the CM content (%) at process endpoint (dd10) was applied, and the two classes defined were: “sufficient” for CM content equal to and above 90%, and "insufficient" for CM content below 90%. A binary classification model was chosen after an initial analysis with a multiclass model revealed that the available bioprocess data was not rich enough to train a multiclass model at this time.

To enable CM content prediction based on early process data, two regression models using MARS and GPR were built, and the data points were assigned to their classes (i.e., sufficient or insufficient) based on this predicted CM content value. For RFs, the classification is conducted directly using the classifier models constructed by the RF.

To evaluate and compare the performances of the classification models, four metrics were considered: accuracy, precision, recall, and the Matthews correlation coefficient (MCC). The range for the first three metrics is zero to one, and MCC is between -1 and 1. These metrics are calculated based on the confusion matrix (Sokolova and Lapalme, 2009), which is illustrated in Fig. 3.2. The

confusion matrix describes the performance of a classification model (algorithm). In this section, we assign the *insufficient* CM content class as the positive class and the *sufficient* class as the negative one. The error of the predictions is broken down for each class using the confusion matrix. The four cells of the confusion matrix correspond to true positive, false negative, false positive, and true negative. The values associated with each of the components give information about how many of the positive/negative classification results were correctly predicted by the model.

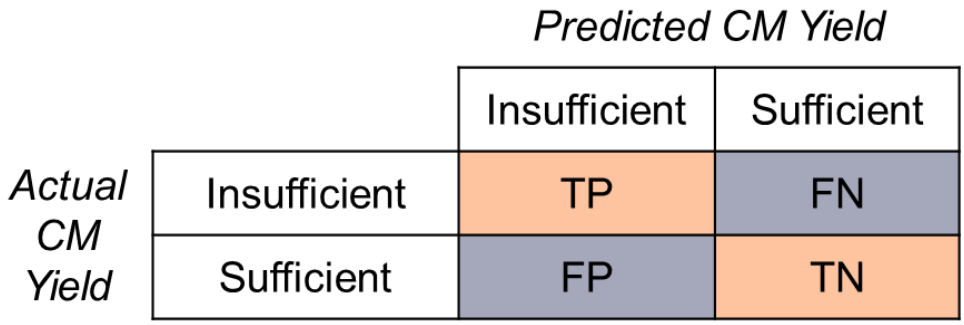


Figure 3.2 - Classification confusion matrix (TP = True Positive, FP = False Positive, TN = True Negative, FN = False Negative)

3.2.1 Classification model performance metrics

Accuracy: Accuracy calculates the proportion of correct classifications (Sokolova & Lapalme, 2009). According to Eq. (3.3), accuracy is the number of all true positives and negatives compared to all prediction results. Accuracy of one indicates that the classification has been conducted accurately and that all the points with *sufficient* or *insufficient* CM content have been included in the right class ($FP + FN = 0$). Zero accuracy defines a totally wrong classification model, which is not able to predict the label of the points correctly.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{3.3}$$

Precision: Precision (Eq. (3.4)) gives information about the proportion of the times the points identified as positive were truly positive (Sokolova & Lapalme, 2009). Precision of one means that all the positive results are actually positive outcomes. When a classifier model with precision of one predicts *insufficient* CM content for a point, it is supposed to have *insufficient* CM content in practice. Value of zero for precision indicates that all the identified positive outcomes are false.

$$Precision = \frac{TP}{(TP + FP)} \tag{3.4}$$

Recall: Recall, Eq. (3.5), is the proportion of actual positive results which were identified as positives (Sokolova & Lapalme, 2009). The value of one for recall demonstrated that the model is able to classify all the actual positive results as positive. In CM content case, all the *insufficient* points would be identified as *insufficient* using a model with recall equal to one. When all the positive classes are falsely identified negative, the value of recall equals zero.

$$Recall = \frac{TP}{(TP + FN)} \tag{3.5}$$

Matthews's correlation coefficient (MCC): Matthews's correlation coefficient (Eq. (3.6)) defines the correlation between the predicted and actual classifications for all data points (Matthews, 1975). Value of one for MCC means there is a strong correlation between the predicted results and the actual values, indicating that the predicted label is correct for all the points. Value of -1 for MCC metric demonstrates a strong inverse correlation. Value of zero for MCC corresponds to no correlation between the predicted and actual results.

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.6}$$

The classification model performance metrics, accuracy, precision, recall, and MCC, were calculated using two different cross-validation techniques: (1) leave one out (LOO) cross-validation and (2) Monte Carlo (MC) cross-validation. Cross-validation is a tool for assessing how well a model can be generalized to new data, which the model has never seen. In MC cross-validation, a set of data is selected randomly to be excluded for validation, and this data set is called the validation set. The model is built using the remaining data, and the model is used to predict the classes for the validation set (Burman, 1989). These predictions are used to calculate performance metrics. In LOO cross-validation, a single data point is set aside (i.e., left out) for validation. The model is built using the remaining data points, and a prediction is obtained for the data point that was left out. This process is repeated for each data point, resulting in a prediction for each.

3.3 Results and Discussion

3.3.1 Feature Selection Results

Two different sets of features were considered for building the classification models. Feature Set 1 contained all potential bioprocess features measured through dd7. Feature Set 2 contained all features measured through dd5. Feature selection was performed on each feature set separately to identify potential features for predicting CM content class on dd10. Classification models were then built using these potential feature sets. We employed PCA, and built-in capabilities of MARS, RFs, and GPR for feature selection. Feature selection resulted in eight potential feature sets for classifying the CM content on dd10. A visual summary of the feature selection results is provided in Fig. 3.3.

PCA yielded five principal components (FS1-PCA) that explained 94% of the variance in the input data for Feature Set 1 (Fig. 3.3A) and yielded four principal components (FS2-PCA) that

explained 94% of the variance for Feature Set 2 (Fig. 3.3B). None of the principal components or bioprocess features strongly correlated with the CM content. The strongest linear correlation between a feature and the CM content was -0.51, and that feature was the time that the differentiation media was supplemented with WNT inhibitor IWP2. This lack of correlation indicates that none of the individual bioprocess features alone suffices to make a prediction on the CM content and that other means, such as machine learning techniques, are necessary to investigate the relationship.

The number of features selected by each machine learning technique is provided in Fig. 3.3. Common features that were selected as significant include the cell densities and their gradients during the first two days of the differentiation protocol (dd0 and dd1). This selection agrees with previous experimental studies concluding that cell density during early differentiation influences differentiation into specific cell lineages (Kempf, Olmer, et al., 2016). A list of features selected by each method is available in Appendix A in Table A1.

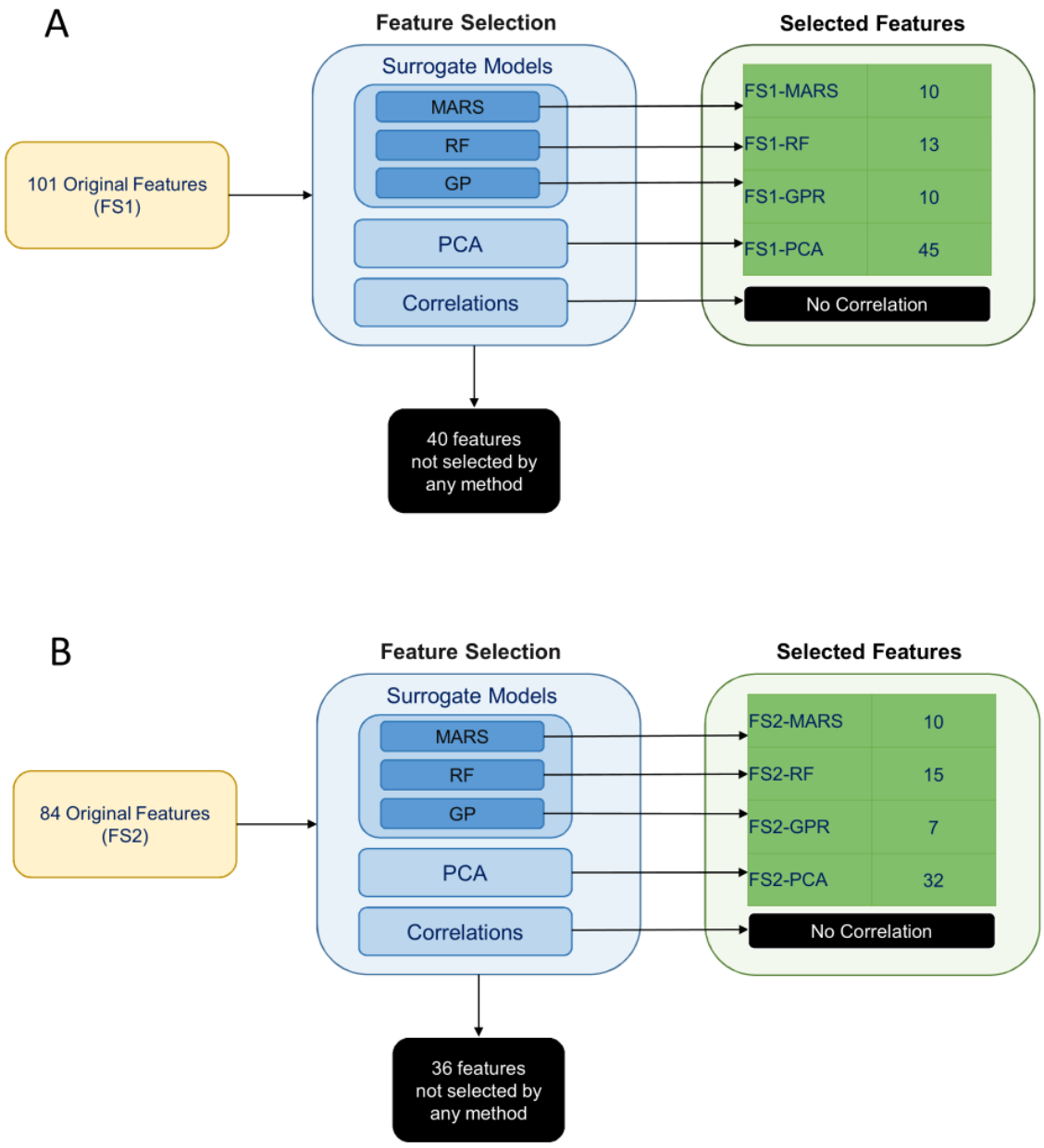
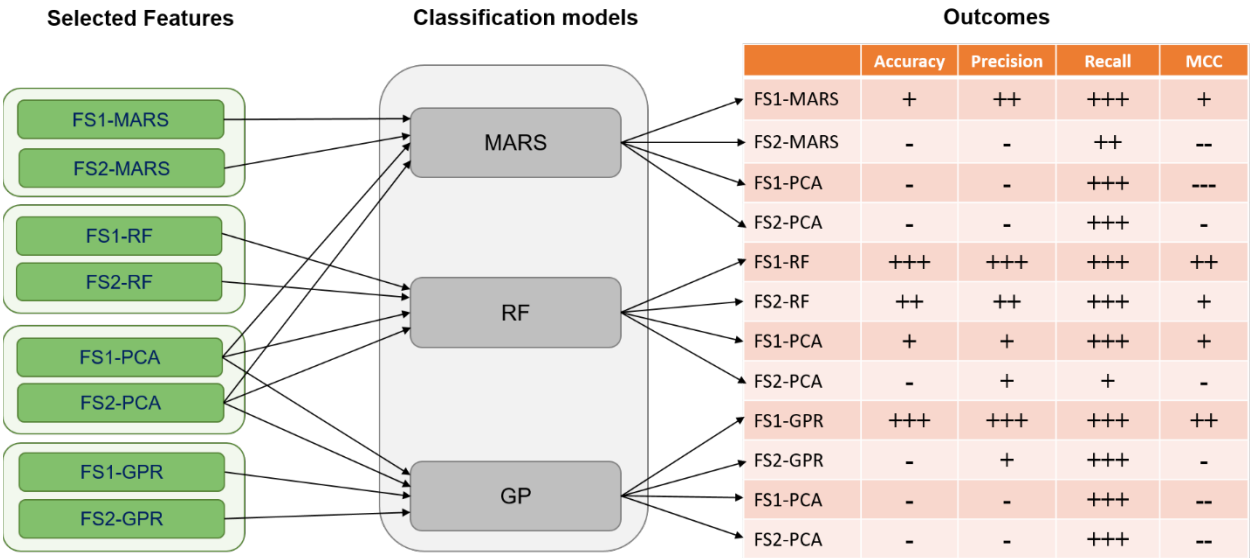


Figure 3.3 - Feature selection results for (A) FS1 and (B) FS2

3.3.2 Classification Model Results

Classification models were constructed for predicting the outcome of the bioreactor experiments on dd10 using features measured up to dd7 and up to dd5, using each of the machine learning techniques described in Section 3.1.3.3. The models were built using the bioprocess

features selected from Feature Set 1 (for predicting using features measured until dd7) and Feature Set 2 (for predicting using features measured until dd5). Results for classification model performance for each of the eight feature sets from the feature selection are summarized in Tables 3.1 and 3.2. Results were obtained using LOO cross-validation and are presented for both the bioprocess features selected by the built-in feature selection for each model, as well as for the PCs obtained from PCA. Both feature sets contained 42 data points chosen from the original set of 58 experiments for training. A visual summary of the results for each classification method with its associated feature sets is depicted in Fig. 3.4.



	Accuracy	Precision	Recall	MCC
FS1-MARS	+	++	+++	+
FS2-MARS	-	-	++	--
FS1-PCA	-	-	+++	---
FS2-PCA	-	-	+++	-
FS1-RF	+++	+++	+++	++
FS2-RF	++	++	+++	+
FS1-PCA	+	+	+++	+
FS2-PCA	-	+	+	-
FS1-GPR	+++	+++	+++	++
FS2-GPR	-	+	+++	-
FS1-PCA	-	-	+++	--
FS2-PCA	-	-	+++	--

Figure 3.4 - Classification results were based on four metrics including accuracy, precision, recall, and MCC for selected features of FS1 and FS2. For accuracy, precision, and recall the values are categorized as follows: (---) for values < 0.3, (--) for 0.3 ≤ values < 0.6, (-) for 0.6 ≤ values < 0.7, (+) for 0.7 ≤ values < 0.8, (++) for 0.8 ≤ values < 0.9, and (+++) for value ≥ 0.9. Moreover, for the MCC metric, the categorization was done as: (---) for values < 0, (--) for 0 ≤ values < 0.1, (-) for 0.1 ≤ values < 0.3, (+) for 0.3 ≤ values < 0.7, (++) for 0.7 ≤ values < 0.9, and (+++) for value ≥ 0.9

For all of the feature sets generated from Feature Set 1, for all of the techniques investigated, classification using the model-selected features always had a better performance than the principal components from PCA. Only two classification model-feature set combinations achieved favorable results for all four of the performance metrics, which is illustrated in Fig. 3.4. RFs trained with feature set FS1-RF and GPR trained with FS1-GPR perform similarly for predicting if CM content will be insufficient for continuing the experiment. Both methods obtained accuracies of 90% and precisions around 90%, meaning that if a model predicts the CM content will be insufficient, there is a 90% probability that it is insufficient.

Similar to those generated from Feature Set 1, the model-selected feature sets for Feature Set 2 resulted in a better performance than the PCs. This indicates that while the PCs successfully explain the variance in the data, they fail to accurately characterize the relationship between the features and the cardiomyocyte content. When only the features up to dd5 are considered, RFs most successfully predict if the CM content will be sufficient. The decrease in the performance of GPR models is possibly due to the removal of the dd7 average value of the DO concentration gradient. This dd7 feature was identified as relevant for predicting dd10 CM content using a GPR and could be an indicator of levels of cell metabolism.

*Table 3.1***.** Classification model performance (calculated with LOO cross-validation) for models trained with features from Feature Set 1.

	MARS		**RFs**		**GPR**	
	FS1-MARS	**FS1-PCA**	**FS1-RF**	**FS1-PCA**	**FS1-GPR**	**FS1-PCA**
Accuracy	0.74	0.64	**0.90**	0.74	**0.90**	0.67
Precision	0.81	0.66	**0.90**	0.74	**0.93**	0.67
Recall	0.93	0.96	**0.96**	0.93	**0.93**	1.0
MCC	0.55	-0.11	**0.78**	0.36	**0.79**	0

Table 3.2 **-** Classification model performance (calculated with LOO cross-validation) for models trained with features from Feature Set 2.

	MARS		**RFs**		**GPR**	
	FS2-MARS	**FS2-PCA**	**FS2-RF**	**FS2-PCA**	**FS2-GPR**	**FS2-PCA**
Accuracy	0.62	0.67	**0.84**	0.67	0.69	0.67
Precision	0.68	0.68	**0.82**	0.73	0.70	0.67
Recall	0.82	0.93	**0.96**	0.79	0.93	1.0
MCC	0.04	0.11	**0.62**	0.22	0.23	0

Table 3.3 contains examples of bioreactor experiments and the predictions for those experiments given by GPR models using FS1-GPR, as well as the values of the bioprocess features indicated by the GPR model to be relevant. For some of the experiments with different prediction types, the feature values are quite similar, for example, the “preculture time” of experiments 36 and 28. However, for some experiments with the same prediction, the features have a wide range

of values, for example, the “dd2 cell density normalized DO gradient” of experiments 16 and 28. These disparities indicate that the individual features alone are not sufficient to determine what will make a good or bad prediction and that all the selected features need to be considered as a whole.

Table 3.3 – Selected features from GPR-FS1

Exp. No	Prediction	Preculture Time [h]	DO gradient/cell count dd2	Average DO concentration gradient dd7	DO gradient/cell count dd5	dd0-dd1 Cell Density Gradient	dd5 Average pH Gradient	d1 Average pH Gradient
1	**TP**	53.000	-1.131	-1.331	-2.275	0.214	0.020	-0.004
36	**TP**	48.000	-0.610	-6.509	-3.477	1.602	-0.007	-0.006
22	**TN**	47.500	-1.199	-0.564	-3.212	0.724	-0.013	-0.004
40	**TN**	48.000	-0.495	-1.706	-3.077	1.345	-0.010	-0.005
16	**FP**	46.000	0.582	0.631	-1.536	0.412	-0.290	-0.207
28	**FP**	48.000	-0.893	0.054	-3.769	-0.176	-0.012	-0.007
34	**FN**	48.000	-0.410	-0.426	-0.664	2.625	-0.014	-0.024
Min		45	-47.24	-6.51	-17.33	-0.76	-0.29	-0.21
Max		56	7.00	9.61	3.73	2.63	0.02	0.01

For model selection purposes, the MCC gives the most important information about how the models perform, as it gives a measure of the correlation between the predicted and actual classes, similar to an R^2 coefficient for a regression model. The other performance metrics should be assessed for their importance based on what the experimental goal of the bioprocess is. For example, if the differentiation process is being studied primarily for data collection and evaluating the outcomes, then maximizing the number of experimental datapoints being retained becomes more important, meaning that the precision of the model needs to be prioritized. However, for another application, such as an in vivo study, it would be more beneficial to stop unsuccessful experiments and start over, meaning that recall and accuracy of the model in identifying which experiments would not produce high CM contents would be prioritized. RF and GPR models were

confirmed to be the most predictive of the dd10 cardiomyocyte content because their MCC values (around 0.80) and their accuracy and precisions of about 90% were higher than the other models investigated.

After testing how the models performed on the original dataset, their performance was evaluated using the test data. The test data consisted of data from the 16 processes that were not used for feature selection and model construction. The values of the selected features from those 16 "control processes" were used as inputs to make predictions of the final CM classification employing the trained models, and those predictions were compared to the actual classifications from the process data. Since MARS had the worst performance for both feature sets, it was excluded from the analysis. The results are presented in Table 3.4. RFs and GPR had similar performance for the test data for feature set FS1-GPR, both with an accuracy of 89%, precision and recall near 90%, and MCC values of 0.72. However, for the sets selected from feature set 2, RFs outperformed GPR. The results obtained for the test data are comparable to those for the data the models were trained on with LOO cross-validation, indicating that the models accurately captured the relationship between the features and the CM content necessary to make the classifications while avoiding overfitting.

Table 3.4 – Model performance on test data

	RFs		**GPR**	
	FS1-RF	FS2-RF	FS1-GPR	FS2-GPR
Accuracy	**0.89**	**0.83**	0.89	0.72
Precision	**0.92**	**0.81**	0.87	0.72
Recall	**0.92**	**1.0**	1.0	1
MCC	**0.72**	**0.57**	0.72	0.11

The "IWP2 treatment time" feature was consistently chosen as having high importance for the prediction of the CM content. This feature describes the amount of time that the IWP2 molecule was allowed to remain in the bioreactor system, i.e., impact the differentiation process. However, this feature was only modulated for a fraction of the process runs and held constant at exactly 48 hours for the rest. To evaluate if our models were able to classify the CM content without using that feature, an additional dataset was constructed. This data set was thus exclusively derived from the original set of 58 processes using only those process runs where the time of IWP2 presence was held constant at 48 hours, and the "IWP2 treatment time" feature was excluded in the analysis. Since RFs performed well for all the previously considered feature sets, the performance was only evaluated using this model. LOO cross-validation and Monte Carlo cross-validation were used to calculate the performance metrics. The Monte Carlo cross-validation used a test set size of 5 and 40 Monte Carlo trials. The results are summarized in Table 3.5. It is thus worth highlighting that

when the IWP2 feature is removed, RFs still successfully predict insufficient CM content with comparable performances for both LOO and Monte Carlo cross-validation for all the feature sets.

Table 3.5 - Model performance with constant IWP2 time.

	FS1-RF		**FS2-RF**	
	LOO	Monte Carlo	LOO	Monte Carlo
Accuracy	0.90	0.90	0.92	0.85
Precision	0.91	0.90	0.91	0.86
Recall	1.0	0.93	0.95	0.93
MCC	0.90	0.82	0.84	0.75

3.4 Conclusions and Future Work

This chapter describes the construction of data-driven models for predicting the CM content on dd10 of hPSC differentiation processes, using existing data sets from bioreactor-based experiments. Using features up to dd7, we were able to identify if an experiment would have an *insufficient* final CM content of less than 90% with 90% accuracy and >90% precision with both RF and GPR models. Furthermore, we were able to identify if an experiment would have an *insufficient* final CM content on dd5 with 84% accuracy with a RF model. Through feature selection methods, these predictions used less than 16% of the collected data, potentially reducing the amount of resource-intense manual collection of data.

Although these models can accurately and precisely predict final CM content, they do not provide any insight into the overall quantity of CMs produced or the resulting functionality and maturity of these cells. In addition, the prediction models were only constructed using a small set of data with limited ranges of all the features. However, the ability to model the outcome of

differentiation experiments at an early stage of differentiation, enables the timely interruption of failing experiments, providing savings in both time and resources.

A trial-and-error method was utilized to select surrogate modeling techniques for building the classification models, which required training multiple models, incurring unnecessary time and computational expense. Although successful models were obtained, this work indicates that a systematic selection of models may provide a more efficient means of outcome prediction modeling. These systematic selection methods will be explored in the following chapters.

Chapter 4 - Comparison of Surrogate Modeling Techniques for Surface Approximation and Surrogate-Based Optimization

The objectives of this work are to comprehensively investigate and compare the performance of several different surrogate modeling techniques for both approximating functional relationships and surrogate-based optimization and to link that performance to the characteristics of the data involved in the application. The results of this analysis are used to develop general "rules of thumb" for selecting an appropriate surrogate modeling technique based on the characteristics of the data being modeled and the desired application. Data sets for training surrogate models are generated from a suite of optimization test functions with different features, such as function shape and number of inputs.

The specific data characteristics being investigated in this study are the shape of the underlying function being modeled, the number of input dimensions, the sampling method used to select sample points to be used in the model training, and the number of sample points. The surrogate modeling techniques considered include Automated Learning of Algebraic Models using Optimization (ALAMO), Artificial Neural Networks (ANN), Extreme Learning Machines (ELM), Gaussian Process Regression (GP), Multivariate Adaptive Regression Splines (MARS), Radial Basis Function Networks (RBFN), Random Forests (RF), and Support Vector Machine Regression (SVR). The following sections contain descriptions of the sampling methods used to select the training data sets and the test function sets. Then, the computational experiments and the results are presented, followed by conclusions and future directions.

4.1 Test Functions

The test functions used to generate data for constructing the surrogate models are from the Virtual Library of Simulation Experiments optimization test suite (Surjanovic & Bingham, 2013).

These test functions are benchmarking optimization problems presented in the form of analytic functions (Hussain et al., 2017). These test functions are used for analysis throughout the rest of the dissertation. Functions with two, four, six, eight, ten, fifteen, and twenty input dimensions were used in evaluations, resulting in a total of 127 test functions. The functions are divided by their shapes, which include the categories: *multi-local minima* with 39 functions, *bowl*-shaped with 41 functions, *plate*-shaped with 11 functions, *valley*-shaped with 16 functions, and *other*-shaped with 20 functions that do not fit into the other four categories. Example functions from each shape category are provided in Fig. 4.1.

The shape categories are defined by multiple characteristics of the test functions, including modality, basins, and valleys, which describe the resulting surface. Modality refers to the number of peaks on the surface. Multimodal functions have many local solutions but one global one, making the global solution difficult to identify as algorithms may become trapped in local solutions. A basin is a relatively steep decline surrounding a large area. These basin regions can severely obstruct optimization algorithms due to a lack of information to direct the search toward the optimum (Jamil & Yang, 2013). A valley occurs when a narrow area of little change is surrounded by regions of steep descent. The progress of an optimization algorithm may be hampered significantly on the floor of the valley (Hussain et al., 2017).

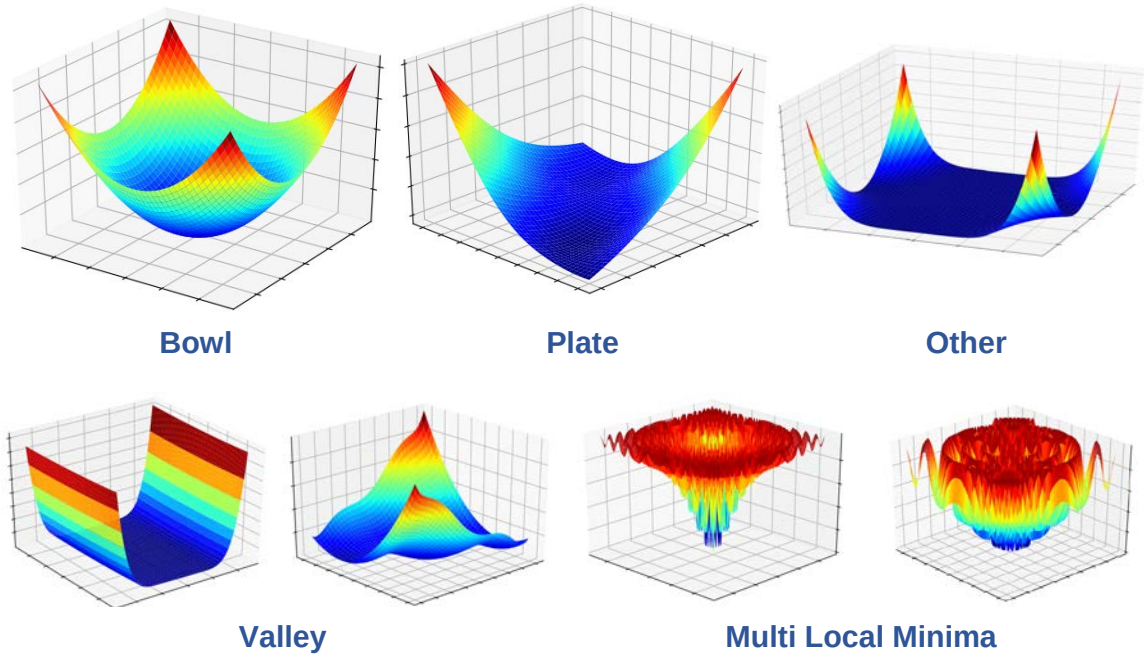


Figure 4.1 - Shape categories for test functions

The *bowl*-shaped functions are unimodal, convex surfaces that can represent applications where changes in inputs produce smooth, regular changes in output values. The *multi-local minima* functions are multimodal and nonconvex and more representative of real data applications with significant noise in the output. The *plate*-shaped functions contain large basin regions. The *plate*-shape function may be representative of processes where several values of the process inputs or a large section of the design space give a constant value for outputs, creating difficulties with optimization searches. *Valley*-shaped functions have valleys, which may be applicable to processes where small changes in input values produce very large variations in output values. Both the *plate*- and *valley*-shaped categories contain unimodal and multimodal functions. The *other*-shaped functions contain combinations of the characteristics of the other categories and non-smooth functional behavior, which could encompass several processes where the shape of the output surface is not well-known.

4.2 Computation Experiments

4.2.1 Surrogate Model Construction

For evaluating the performances of surrogate modeling techniques, input-output pairs were generated from each test function using three different sampling methods at seven different sample sizes (50, 100, 400, 800, 1200, and 1600 samples). The sample sizes were chosen in order to give a range of values for the ratio of sample size to input dimension for each input dimension being studied. In general, a sample size to input dimension ratio of 10 is considered an adequate number of samples for most regression techniques (Harrell et al., 1984). Any ratio smaller than 10 can be considered to be a small sample size, with large sample sizes being any ratio of sample size to input dimension larger than 10. Surrogate models were trained using these pairs with each of the surrogate modeling techniques for each generated dataset. This process resulted in a total of 18,984 trained models. Each of the techniques has unique hyperparameters that were optimized in training the models for each dataset to construct the best possible surrogate without overfitting the model. For the MARS models, the number of hinge functions that could be multiplied together was limited to two to avoid overfitting with higher-order hinge functions. The numbers of ANN, ELM, and RBFN nodes, as well as the number of trees in the RF models, were increased until the root mean squared error of a validation dataset stopped improving. For these models, the validation error was estimated using ten-fold cross-validation on the training set. The number of nodes (or trees) was increased until the average value of the last five validation errors either began to increase or changed by less than 1%.

All of the surrogate modeling techniques except ALAMO and RBFN were implemented in Python with the Sci-Kit Learn library version 0.32.2 (Pedregosa et al., 2011). RBFN models were implemented with MATLAB 2017b, and ALAMO has its own software for model

construction (Cozad et al., 2014). All of the training options except for the ones discussed were set to the default values indicated by the implementation packages. The specific implementation package used for each technique is listed in Table 4.1.

Table 4.1 – Software implementations for surrogate model training

Surrogate Model	Implementation
MARS	py-Earth
RF	Scikit-learn RandomForestRegressor
ANN	Scikit-learn MLPRegressor
ELM	Scikit-learn ELMRegressor
GP	Scikit-learn GaussianProcessRegressor
SVM	Scikit-learn SVR
ALAMO	ALAMO
RBFN	MATLAB newrb

The three sampling methods used were Halton Sequence Sampling (Halton), Latin Hypercube Sampling (LHS), and Sobol Sequence Sampling (Sobol). LHS partitions the domain of each input variable into N subsets to be sampled from, where N is the number of sampling points (Mckay, 1992). Both Halton and Sobol sequence sampling are quasi-random, low discrepancy sequences that attempt to distribute the sampling points uniformly across the sample space (Halton & Smith, 1964; Joe & Kuo, 2008). These sampling methods were chosen because they have been shown to sample input space uniformly for functions up to ten dimensions (Diwekar, 2003; Garud et al., 2017a).

4.2.2 Evaluation of Surface Approximation and Surrogate-Based Optimization Performance

After the surrogate models were trained for each dataset, sample size, and sampling method, a densely sampled set of 100,000 input-output pairs were generated as test dataset for assessing the accuracy of the models. Because there was no significant difference between the

samples or results obtained from any of the sampling methods at this large size, only results for the dense set produced using Sobol sequence sampling are presented here. The root mean squared error, adjusted R^2 value, and the maximum percent error were calculated for each dataset-surrogate model combination based on the difference between the outputs of the given function and the outputs predicted by the surrogate model.

The global minimum of each test function was estimated using the trained surrogate models. The mathematical programs for estimating the minima were constructed in Pyomo (version 5.6) (Hart et al., 2017; Hart et al., 2011), a Python-based optimization language. The estimated minimum location and value are compared to the actual global minimum and value of each function for accuracy to provide some insight into the effectiveness of each surrogate modeling technique for surrogate-based optimization. Computations were carried out on the Auburn University Hopper HPC Cluster (Lenovo System X HPC Cluster) using Intel E5-2650 V3, 2.3 GHz 20 core processors and implemented in Python 3.5 and MATLAB 2017b (for RBFN surrogate models).

4.2.3 Surface Approximation Performance Metrics

Two performance metrics were used for evaluating the surface approximation ability of the surrogate models: normalized root mean square error (nRMSE) and adjusted-R^2. The adjusted-R^2 (Miles, 2014) takes into account both the surrogate model accuracy and the size, or complexity, of the model. Balancing the complexity of the model with the sample size is essential in ensuring that the model is not overfit, as overfit models do not generalize well to new conditions. However, adjusted-R^2 can unfairly penalize some of the surrogate models that are larger by nature of their structure, for example, Random Forests, which need to grow larger because of their decision tree framework. The nRMSE metric was chosen to assess how well the surrogates approximated the

test function without penalizing them for their size. The formula for (nRMSE) is given in Eq. (4.1). The nRMSE value for each dataset-surrogate model combination is normalized by the range of output values for easier comparison across datasets with a variety of ranges for output values.

$$nRMSE = \sqrt{\frac{\sum_{n=1}^{N}(\hat{y}_n - y_n)^2}{N}} \Big/ (y_{max} - y_{min}) \tag{4.1}$$

In Eq. (4.1), y_n is the output for point n for a dataset, $\hat{y}_n$ is the output predicted by a surrogate model for point n, N is the total number of sample points in the dataset, and y_{max} and y_{min} are the maximum and minimum output values in a dataset, respectively.

The formula for calculating adjusted-R^2 ($\hat{R}^2$) is shown in Eq. (4.2).

$$\hat{R}^2 = 1 - (1 - R^2)\left[\frac{N-1}{N-(k+1)}\right] \tag{4.2}$$

In Eq. (4.2), R^2 is the R-squared regression coefficient, N is the number of data points in the training set, and k is the number of model parameters (or hyperparameters). R^2 values typically fall between zero and one, with an R^2 of one indicating a perfect fit. However, with the adjustment for model size, adjusted-R^2 values can become negative. The number of model hyperparameters, k, was estimated as the number of nodes in the trained ANN, RBFN, and ELM models. For MARS models, k was estimated as the total number of hinge functions. The k for the ALAMO models was estimated as the number of nonlinear transformation terms in the final model. The k for SVR models was estimated as the number of support vectors in the trained model. For GP models, k was estimated as the number of input dimensions, which corresponds to the number of hyperparameters that are fit for the length scale used in the radial basis function (the kernel function used in the GP models). For RF models, k was estimated as the average number of decision threshold values per tree in the forest.

The nRMSE and adjusted-R^2 metrics were calculated using the densely sampled 100,000 point test sets generated using Sobol Sequence sampling. One-way analysis of variance (ANOVA) was applied to determine which dataset characteristics had a statistically significant effect on the surrogate model performance metrics at a 95% confidence level.

4.2.4 Surrogate-Based Optimization Performance Metrics

We define D_{opt} as the Mahalanobis distance, D_M, (De Maesschalck et al., 2000) between the location of the global minimum of a test function, x_{opt}, and the location estimated using a trained surrogate model, $\hat{x}_{opt}$. This value is normalized by the maximum Mahalanobis distance between any two points (x_i, x_j) in the dataset (Eq. 4.3),

$$D_{opt} = \frac{D_M(x_{opt}, \hat{x}_{opt})}{\max\limits_{i,j} D_M(x_i, x_j)} \tag{4.3}$$

where x_i and x_j are points in the domain space of the dataset.

We define G_{opt}, Eq. (4.4), as the normalized gap between the global minimum value and the estimated one. This value is normalized by the range of output values in the dataset.

$$G_{opt} = \frac{y_{opt} - \hat{y}_{opt}}{y_{max} - y_{min}} \tag{4.4}$$

In Eq. (4.4), y_{opt} is the actual global minimum value, $\hat{y}_{opt}$ is the one calculated by the surrogate model, and y_{max} and y_{min} are the maximum and minimum output values in a dataset, respectively.

4.3 Results and Discussion

4.3.1 Effect of Sampling Method and Sample Size

The surface approximation performance for MARS models is shown in Fig. 4.2a as a function of both sample size and sampling method. The average value of the performance metrics for all 127 test functions is given as a function of the sample size for models trained with datasets

generated from each of the three sample methods tested. Similar to previous studies, the results here show a general trend of improving performance with the addition of more sample points for training the surrogate model. Surface approximation performance for all the surrogate modeling techniques showed a comparable behavior to that of the MARS models. The 90% confidence interval error bars for the sampling methods have some overlap at all of the sample sizes. From this result, there does not appear to be any significant difference in the surrogate modeling performance among the three sampling methods investigated. ANOVA analysis did not indicate any statistically significant difference in surface approximation performance for any of the surrogate modeling techniques with changing sampling methods ($p > 0.05$). The selection of the space-filling sampling method does not appear to have any effect on the approximation performance.

For surrogate-based optimization, only RF and RBFN models showed any statistically significant differences in the surrogate model performance among the three sampling methods. The other six surrogate modeling techniques' performance was not significantly affected by the choice of the sampling method. The average D_{opt} value for RF and RBFN models as a function of the training set sample size is shown in Fig. 4.2b and Fig. 4.2c for Sobol, Halton, and LHS sampling. For RF models (Fig. 4.2b), the D_{opt} values for models trained with Sobol sequence sampling data tend to be in general lower than those of models trained with data generated from the other two sampling methods, meaning that the optimum locations predicted by RF models trained with Sobol samples are on average closer than those given by other sampling methods. ANOVA analysis further confirmed that the models trained using Sobol sequence samples had statistically significantly lower values for D_{opt} at each of the sample sizes investigated. The models trained using Sobol sequence samples also had statistically lower values of D_{opt} for RBFN models

(Fig. 4.2c) at a sample size of 50 ($p = 0.002$). These results indicate that while the sampling method does not affect surface approximation performance, for some surrogate modeling techniques, the choice of sampling method can have a substantial impact on the performance of the model for surrogate-based optimization, especially at lower sample sizes.

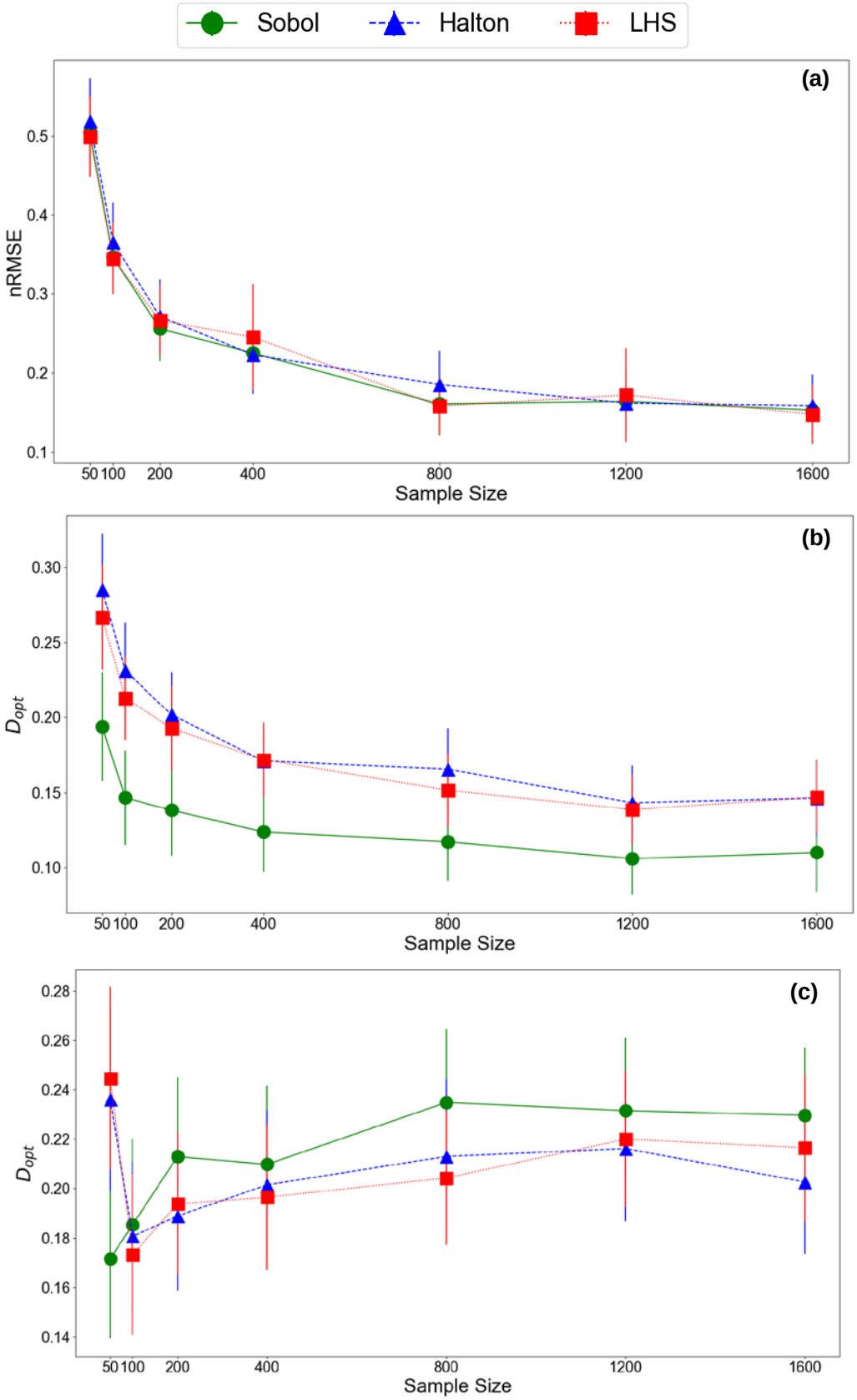


Figure 4.2 - (a) MARS performance for different sampling methods as a function of sample size for average nRMSE on all 127 test functions. (b) Average D_{opt}vs. sample size for RF models. (c) Average D_{opt}vs. sample size for RBFN models. Error bars represent 90% confidence intervals on the averages.

4.3.2 Comparison of surrogate modeling technique performance for surface approximation

There was no significant difference in the surface approximation performance of the surrogate models trained using the sample points generated using Sobol and Halton sequences and LHS. Therefore, results presented in this section only include surrogate models trained with datasets generated via Sobol sequence sampling. Results are presented in this section for three selected sample sizes. The surface approximation performance metric results are presented in violin plot format. The shape of each violin represents the probability density distribution of the data values. The top and bottom of the violin represent minimum and maximum values, with the black bar within each violin representing the interquartile range of the values. Median values are indicated on each violin by a white circle.

4.3.4 Selection of Surrogate Modeling Technique for Surface Approximation by Adjusted-R^2

Results obtained based on the adjusted-R^2 are summarized in Fig. 4.3. The adjusted-R^2 was used to take into account the model size and complexity in addition to its accuracy (Miles, 2014). This metric can be used to select a "best" model to use for approximation while controlling for overfitting by selecting the technique that provides the highest value of adjusted-R^2. Adjusted-R^2 values were calculated for all the trained surrogate models for each dataset. For each dataset category (either input dimension or shape), the number of times each surrogate modeling technique was selected as best (had the highest adjusted-R^2) was tabulated for the datasets in that category. These tabulated values were divided by the total number of datasets in the category to calculate the fraction of datasets for which each surrogate modeling technique was selected as best performing. The number of datasets included in each category is given below the *x*-axis for Fig. 4.3.

Figure 4.3 shows which surrogate modeling techniques are selected most frequently when the datasets are grouped by their input dimension and function shape. Although at the smallest sample size tested, RBFN models give the highest adjusted-R^2 more frequently at the higher input dimensions, ALAMO provides the highest adjusted-R^2, and thus the closest approximation, the highest percentage of the time at low input dimension. The superior approximation performance of RBFN models is not observed at higher sample sizes when considering selecting the "best" model by the adjusted-R^2. In general, ALAMO provides the most robust performance for yielding the highest adjusted-R^2. However, as the dimension and sample size increase, GP and MARS models begin to perform as well or better than ALAMO models.

At the largest sample size (Fig. 4.3b), GP and MARS models demonstrate opposite trends with increasing input dimension until eight inputs. The selection frequency of GP models as having the highest adjusted-R^2 (Fig. 4.5b) deteriorates while that of MARS improves. At input dimensions higher than eight, both GP and MARS models have decreasing selection frequencies. GP models use interpolation between the given training points to estimate outputs at new input conditions. As the number of input dimensions increases, GP models require a higher number of training points to more accurately capture a surface's behavior in a region for interpolating to new conditions. This may explain why MARS models begin to outperform GP models at higher dimensions, as the hinge functions of the MARS models are not dependent on interpolation.

Figs. 4.3c and 4.3d show which surrogate modeling techniques are selected most frequently when the datasets are grouped by function shape. When the datasets are grouped by the function shape, different techniques yield the best adjusted-R^2 values at different sample sizes. For *bowl* and *multi-local minima* shaped functions, MARS and ALAMO models give the highest values for the largest percentage of the datasets at smaller sample sizes. The hinge functions of the MARS

models and the several available nonlinear transformations of ALAMO models may make them particularly suitable for mimicking the convex behavior of the *bowl*-shaped functions and for approximating the somewhat "noisy" surface of the *multi-local minima* functions. When the sample size grows, GP models also begin to perform well for multi-local minima functions as they gain more information for more accurate interpolations. When the sample size grows, GP models also begin to perform well for *multi-local minima* functions. Also, GP models are selected the most frequently at all sample sizes for the *other* functions, which do not fit into any of the other four defined shape categories. ANN models provide the best models for *plate-shaped* functions with smaller samples but are outperformed as sample size increases. The model selection for *valley* functions is spread fairly evenly among a few modeling techniques, which may suggest that additional characteristics should be considered when selecting a surrogate model for a similar dataset.

RF models did not perform the best for any of the datasets considered. SVR performed best for very few, indicating that if adjusted-R^2 is the performance metric of interest, these models may not be suitable choices. These results indicate that there is some dependence of the surrogate model surface approximation performance on the overall shape of the function the dataset was generated from, the input dimension, and the sample size, especially when all these factors are considered together.

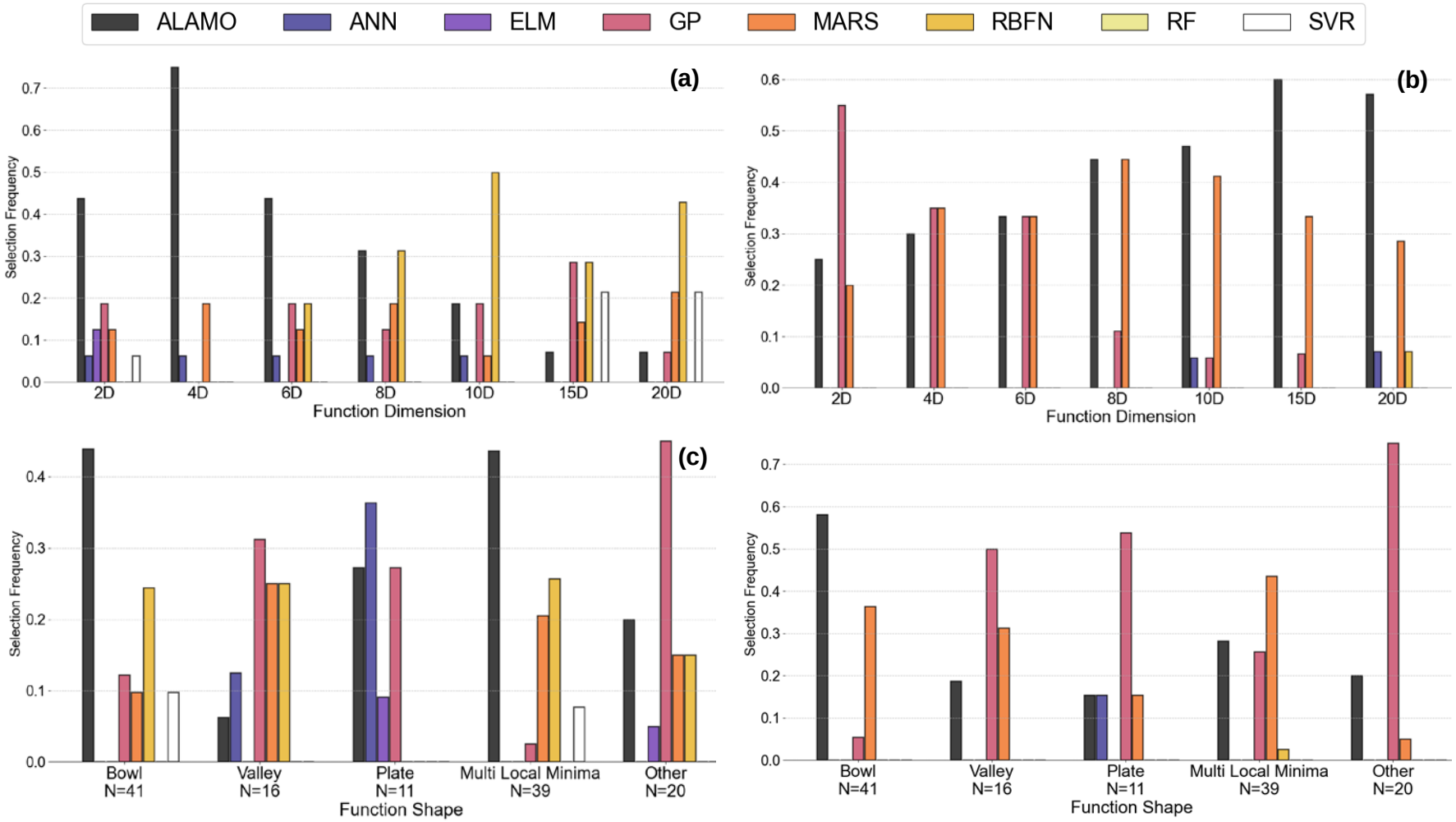


Figure 4.3 - Percentage of datasets grouped by input dimension for which each surrogate modeling technique had the highest adjusted-R^2 for sample sizes: (a) 50 and (b) 1600. Percentage of datasets grouped function shape for which each surrogate modeling technique had the highest adjusted-R^2 for sample sizes: (c) 50 and (d) 1600.

4.3.5 Effect of Underlying Function Input Dimension and Function Shape on Surface Approximation Performance

Results obtained for the effect of input dimension of the test function (and resulting training dataset) on the nRMSE and adjusted-R^2 for each surrogate modeling technique at a sample size of 50 are summarized in Fig. 4.4. RBFN and MARS models have better performance than the other techniques at the smaller sample sizes tested. Although many of the techniques appear to perform comparably for approximation based on their nRMSEs, the performance metric values deteriorate when adjusted for the model size with the adjusted-R^2. This indicates that while many of the techniques can capture the general surface of the test functions at small sample sizes, they do so at the expense of overfitting. This overfitting trend is particularly apparent for ELM and ANN models, for example. With increasing sample sizes, the adjusted-R^2 values and nRMSE follow similar trends, as increased sample sizes allow for larger models that can still avoid overfitting.

The results for adjusted-R^2 are summarized for sample sizes of 400 and 1600 in Fig. 4.5. In general, at these larger sample sizes, MARS models perform the most robustly with respect to the input dimensions. ANOVA analysis confirms this robust behavior with respect to dimensions for MARS models, revealing no significant difference between the nRMSE values of each dimension ($p = 0.43$). MARS and GP models at lower input dimensions yield higher values, close to one, of adjusted-R^2. However, the GP model performance worsens as the dimension increases, which matches the trend from the results for model selection (Fig. 4.3b), illustrating the dependence of model performance on dimension. The robust performance of MARS models may be due to their effective partitioning of the design space with the hinge functions and the accurate modeling of nonlinearities in these partitions by the products of hinge functions. Input dimension has different levels of effects on the surrogate modeling technique performance at larger sample

sizes. RF and RBFN model performance becomes progressively worse with increasing dimensions, while ALAMO model performance does not change much at different input dimensions. ALAMO's robust approximation performance with respect to input dimension may be due to its ability to perform multiple nonlinear transformations for each input dimension separately.

While the selection of a modeling form by adjusted-R^2 can be useful, selecting a single surrogate model as the best for a dataset may be misleading, as multiple models may perform almost identically for the same dataset. For example, although ALAMO models are selected most frequently as best for *bowl*-shaped test functions (Fig. 4.3c and Fig. 4.3d), MARS and GP models are selected most frequently as second-best when ALAMO models are the best performing. However, statistical analysis revealed that, on average, MARS models give higher adjusted-R^2 values than ALAMO for *bowl*-shaped functions, and GP model performance was not significantly different from that of ALAMO (at a significant level of 0.05). Furthermore, ANN models were selected as best (with the highest adjusted-R^2) for *plate*-shaped functions most frequently (Fig. 3b and Fig. 3c), but their adjusted-R^2 values (Fig. 4.5b) were only significantly different from RF, SVR, and ELM models for that shape category. Based on these results, multiple surrogate modeling techniques can be successfully applied to a dataset to produce similarly accurate approximations, and one may not need to rely on a single best choice.

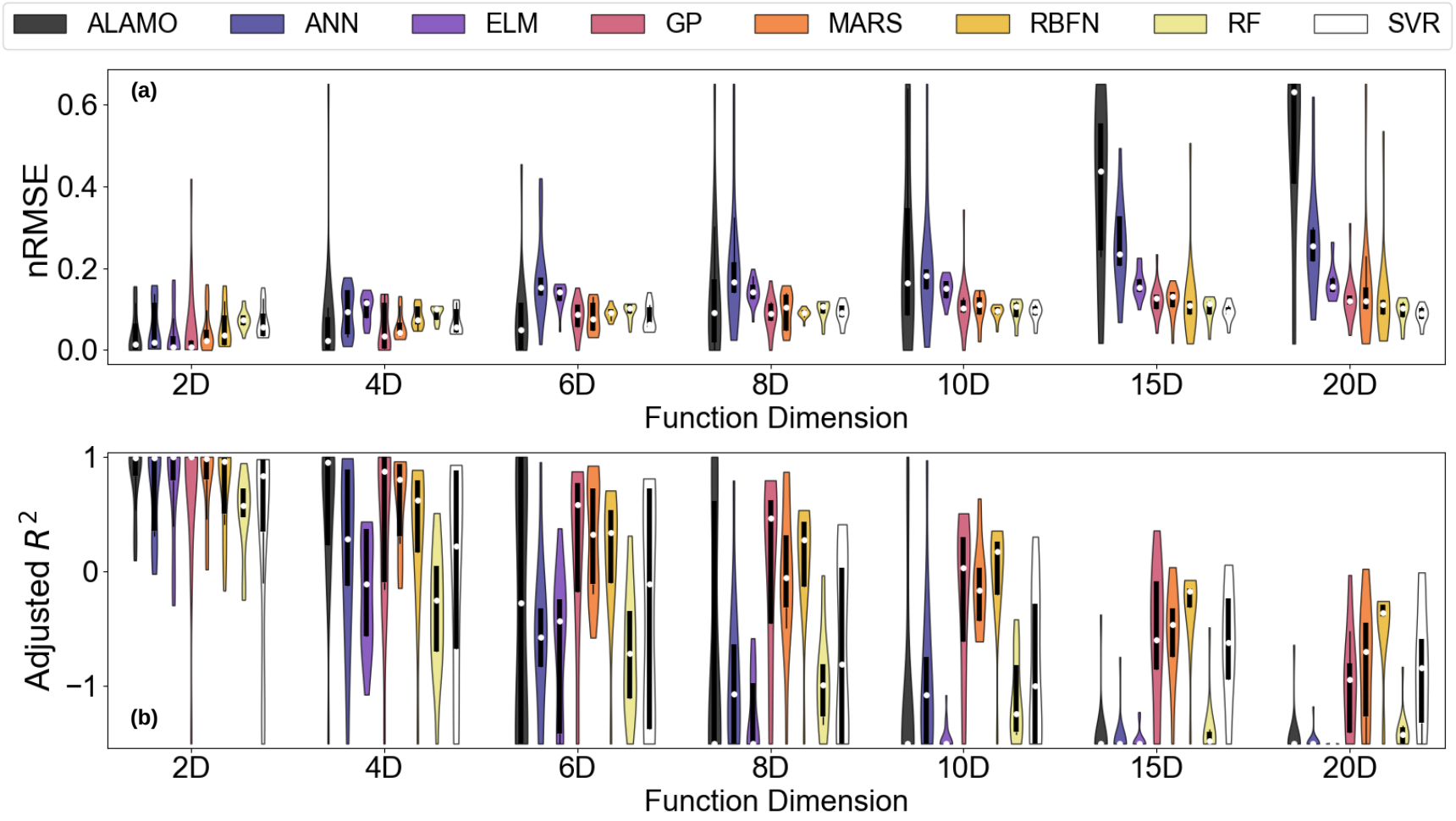


Figure 4.4 - (a) nRMSE and (b) adjusted-R^2 for datasets grouped by underlying function dimension

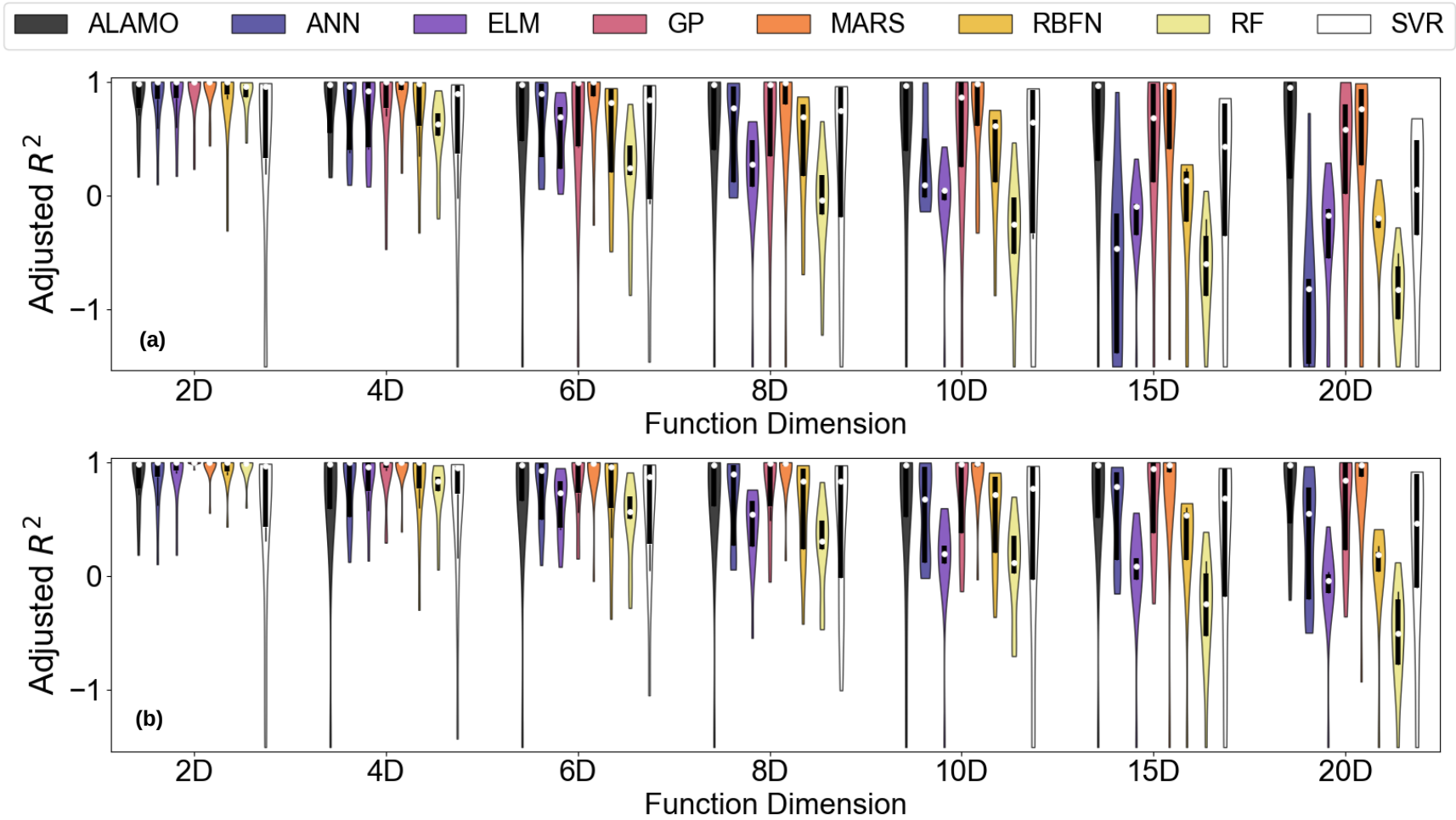


Figure 4.5 - Adjusted-R^2 for models trained with sample sizes of (a) 400 and (b) 1600

Results for the effects of the underlying function shape on the performance of the surrogate models for surface approximation are summarized in Fig. 4.6. All of the surrogate models had poor approximation performance and a high level of overfitting, indicated by negative adjusted-R^2 values, with respect to the function shape at sample sizes less than 200. Therefore, adjusted-R^2 results for sample sizes of 400 and 1600 are presented. GP and MARS models provide the most robust performance when considering the test function shape, though none of the techniques perform well overall for the test functions with *multi-local minima* shape.

The function shape does have an impact on the surrogate models' performance for some of the other techniques. Although overall, GP and MARS models give significantly lower values of nRMSE than the other techniques ($p < 0.05$), when considering only *bowl*-shaped functions, ALAMO models provided the lowest nRMSE values and best performance ($p < 0.05$). In addition, while ELMs have poor performance in general for *bowl* and *valley*-shaped functions, they perform very well in approximating *plate*-shaped functions, with adjusted-R^2 values close to one. Both ANN and ELM models demonstrate improved performance for *plate*-shaped functions in comparison to the other shape categories. The on-or-off nature of the nodes and activation functions in these model types may make them especially suitable to approximate the flat or nearly so portions of the *plate*-shaped surfaces. Results for surface approximation suggest that for datasets where specific characteristics are not available, a MARS or GP model would be appropriate to select as a general guideline. However, if characteristics are available, other models might provide a better approximation.

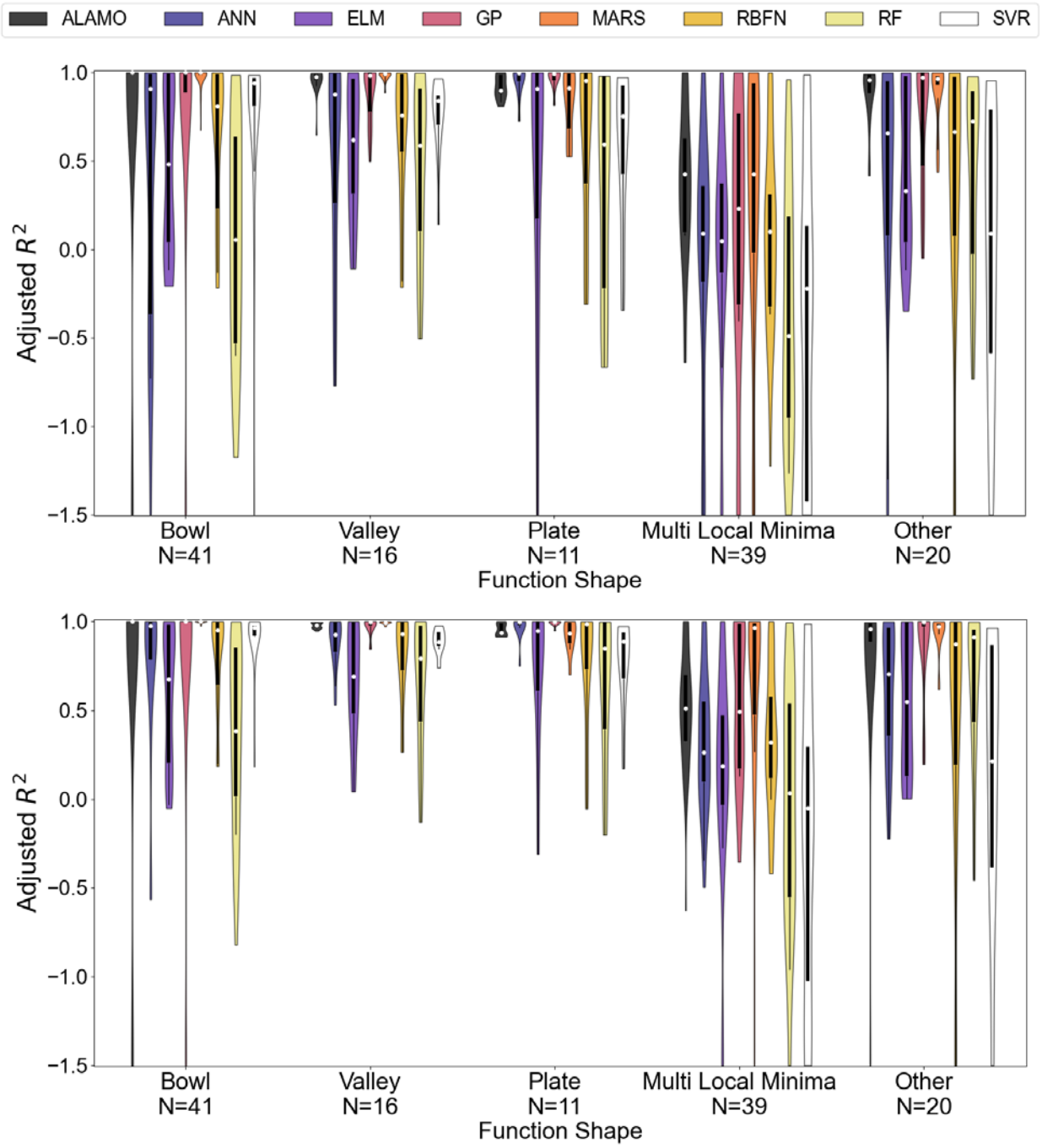


Figure 4.6 - Adjusted R^2 for models trained with sample sizes of (a) 400 and (b) 1600 group by underlying function shape. N values below the function dimensions indicate the number of test functions used for each shape category

4.3.6 Comparison of surrogate modeling technique performance for surrogate-based optimization

The computational experiments for surrogate-based optimization were executed by using each surrogate model to estimate the minimum of each function and the location of the minimum. Then, these results were compared to the global minimum and its true location using two metrics, D_{opt} (Eq. 4.3) and G_{opt} (Eq. 4.4). Results are summarized in Figs. 4.7, 4.8 and 4.9, where we define a model as having located the optimum when it obtains a D_{opt} or G_{opt} value less than a threshold. The thresholds for D_{opt} and G_{opt} are 5% and 0.01%, respectively. Threshold for D_{opt} was selected in terms of how close the estimate needed to be to still get a reasonable estimate of the optimum value with the predicted location (within 1% error of the output range) and to also yield a reasonable separation in the performance of models. The G_{opt} threshold was selected using 1% of the output range as a measure of a good model fit.

Figure 4.7 shows the results for how well the surrogate models locate the global minimum of each test function when they are grouped by the function dimension for sample sizes of 50 (Fig. 4.7a) and 400 (Fig. 4.7b). Surrogate-based optimization performance with respect to underlying function shape did not differ significantly with sample size, so only results for 1600 samples are presented here (Figs. 4.8 and 4.9). RF and SVR models, in general, locate the minima for the highest fraction of the datasets when datasets are grouped by both shape (Fig. 4.9a) and input dimension (Fig. 4.8a). ANOVA analysis on the mean D_{opt} of those two techniques versus that of the others indicates that the locations given by SVR and RF values are significantly lower ($p < 0.05$). Contrastingly, both techniques had some of the worst performances for approximating the design space, with higher values for nRMSE and lower values of adjusted-R^2.

While the RF models perform well in capturing the overall curvature of the underlying function in each dataset, they perform poorly for predicting the actual output values. This may be

due to their utilization of decision trees. The “rules” of the decision trees that determine movement between nodes provide less accurate, more noisy predictions for outputs but may be effective in dividing the domain of the dataset in a way that allows the solver to pinpoint the location of the minimum accurately. The support vectors in SVR models may serve a similar function to the decision tree rules in RF models. GP models perform most robustly in estimating the actual global minima values, in general, with respect to both shape and dimension, which may be related to their ability to approximate the surfaces for the datasets accurately.

Both function input dimension and function shape impacted the surrogate models’ estimation of optimum values. While ANN models only identify the optimum value for about 25% of the *bowl*-shaped test functions (Fig. 4.9b), they can identify close to 80% of the optimum values for the *plate*-shaped functions. On the other hand, ALAMO models can identify optimum values much more accurately for *bowl*-shaped functions than for *plate*-shaped ones. The optimum value estimation seems to be more closely linked to the approximation performance than is the estimation of the optimum location, as ALAMO models were more accurate in approximating *bowl*-shaped functions and ANN models were more accurate for *plate*-shaped ones. At the higher input dimensions of 10 and 15, the optimization problems of the many surrogate models were not solved to 0.001% optimality gap within 48 hours (wall time). Specifically, none of the optimization problems for GP, ELM or RBFN models and very few of the SVR and ANN models could be solved within the allotted computational time. In contrast, the optimization problems constructed using RF, MARS, and ALAMO models were solved to optimality within 72 hours (wall time) for all test functions at high input dimensions. Therefore, our computational test results recommend using only those three techniques for surrogate-based optimization at input dimensions higher than

10. A summary of these findings for surrogate-based optimization, as well as those for surface approximation, is provided in Table 4.2.

Table 4.2 - Summary of findings for surrogate modeling technique performance

Model	Advantages	Disadvantages
ALAMO	-Accurate for approximation and optimization of convex ("bowl"-shaped functions) -Relatively short optimization solution times	
ANN	-Accurate approximation and optimization of *plate*-shaped functions	-Requires a relatively large number of samples for approximating several function types accurately -High computational time for optimization solutions, particularly at high input dimension
ELM	-Accurate approximation of *plate*-shaped functions -Relatively short model training times	-Requires a relatively large number of samples for approximating several function types accurately -High computational time for optimization solutions, particularly at high input dimension
GP	-Accurate approximation of several function types	-High computational time for optimization solutions, particularly at high input dimension -High model training times

MARS	-Accurate approximation of several function types -Optimization problems remain tractable, even at high input dimension	-Not as accurate for optimization of test functions
RBFN	-More accurate than other techniques for optimization at smaller sample sizes	-High computational time for optimization solutions, particularly at high input dimension
RF	-MILP structure of optimization problem provides accurate optimization solutions with relatively low solution times	-Less accurate than other techniques for approximation surfaces in general
SVR	-Relatively short model training times -Accurate optimization of several function types, particularly at small sample sizes	-Less accurate than other techniques for approximation surfaces in general -High computational time for optimization solutions, particularly at high input dimension

4.3.7 Computational Efficiency of Solving the Resulting Optimization Problems

The solvers used for optimization are provided in Table 4.3. For each modeling technique, the selected solver was the most appropriate for the resulting optimization model (Table 4.3). The average computational times required for solving the optimization problems to estimate the global minima of the test functions for each surrogate modeling technique are also included in Table 4.3. The average solution times reported in Table 4.3 are for the optimization problems that were solved to optimality within 48 hours. The solution time is dependent on final model size and structure, with larger, more complex models taking a much longer time to solve than linear models or models with fewer parameters.

GP models have the highest average solution times because the radial basis kernel function used and the interpolation of the model based on the training data points result in a large, highly nonlinear optimization model. The high degree of nonlinearity and number of parameters in the optimization problems of ANN, ELM, RBFN, and SVR models also presented difficulties, with a large proportion of the problems not being solved within 48 hours (wall time). The range of optimality gaps for the models that did not reach a gap of 0.001% within the set time limit were 1% - 10% for ANN, 0.5% - 14% for ELM, 0.009% - $1x10^7$ % for GP, 4% - $1x10^3$ % for RBFN, and 0.14% to 195% for SVR.

Although they are some of the more accurate models for locating optima and the resulting optimization models do not, in general, become computationally intractable, the optimization problems of RF models have the highest average value for the solution time. The solution time for RF-based optimization problems may be reduced by developing specialized algorithms that exploit the special structure of RF model MILPs as RF models were successful in pinpointing the location

of the minimum. While MARS models had relatively low optimization solution times, the solutions given by MARS models were less accurate than those of other methods.

Table 4.3 - Solvers and solution times for surrogate-based optimization (NLP = Non-linear program, MINLP = Mixed integer non-linear program, MILP = Mixed integer linear program).

Surrogate Model	Resulting Optimization Model	Solver	Average Solution Time (min)
ALAMO	NLP	BARON	4.4
ANN	NLP	BARON	664
ELM	NLP	BARON	9.4
GP	NLP	BARON	2169
MARS	MINLP	ANTIGONE	7.9
RBFN	NLP	BARON	33
RF	MILP	CPLEX	27
SVR	NLP	BARON	288

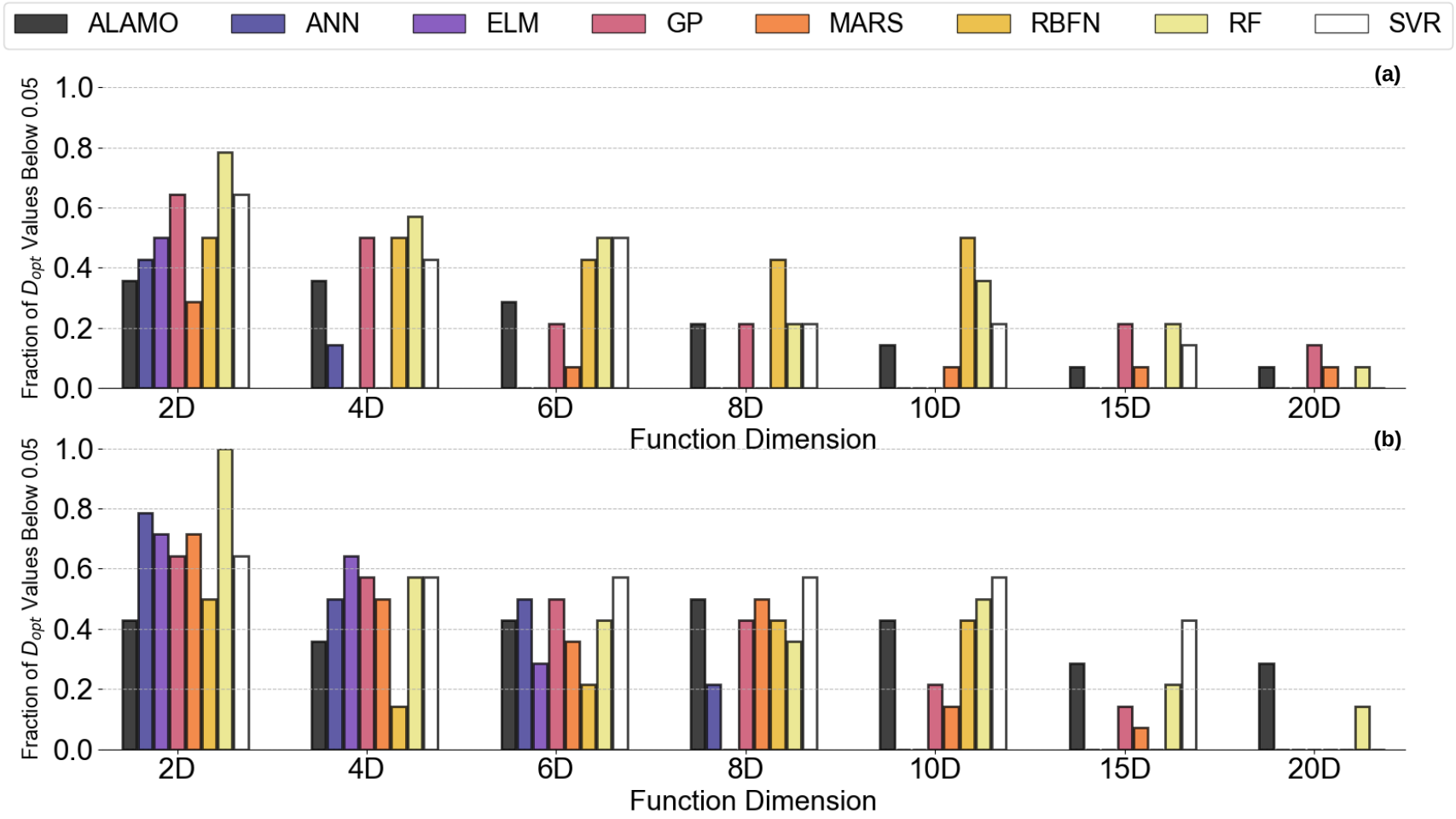


Figure 4.7 - Fraction of datasets with D_{opt} less than 5% grouped by input dimension for sample size (a) 50 and (b) 400

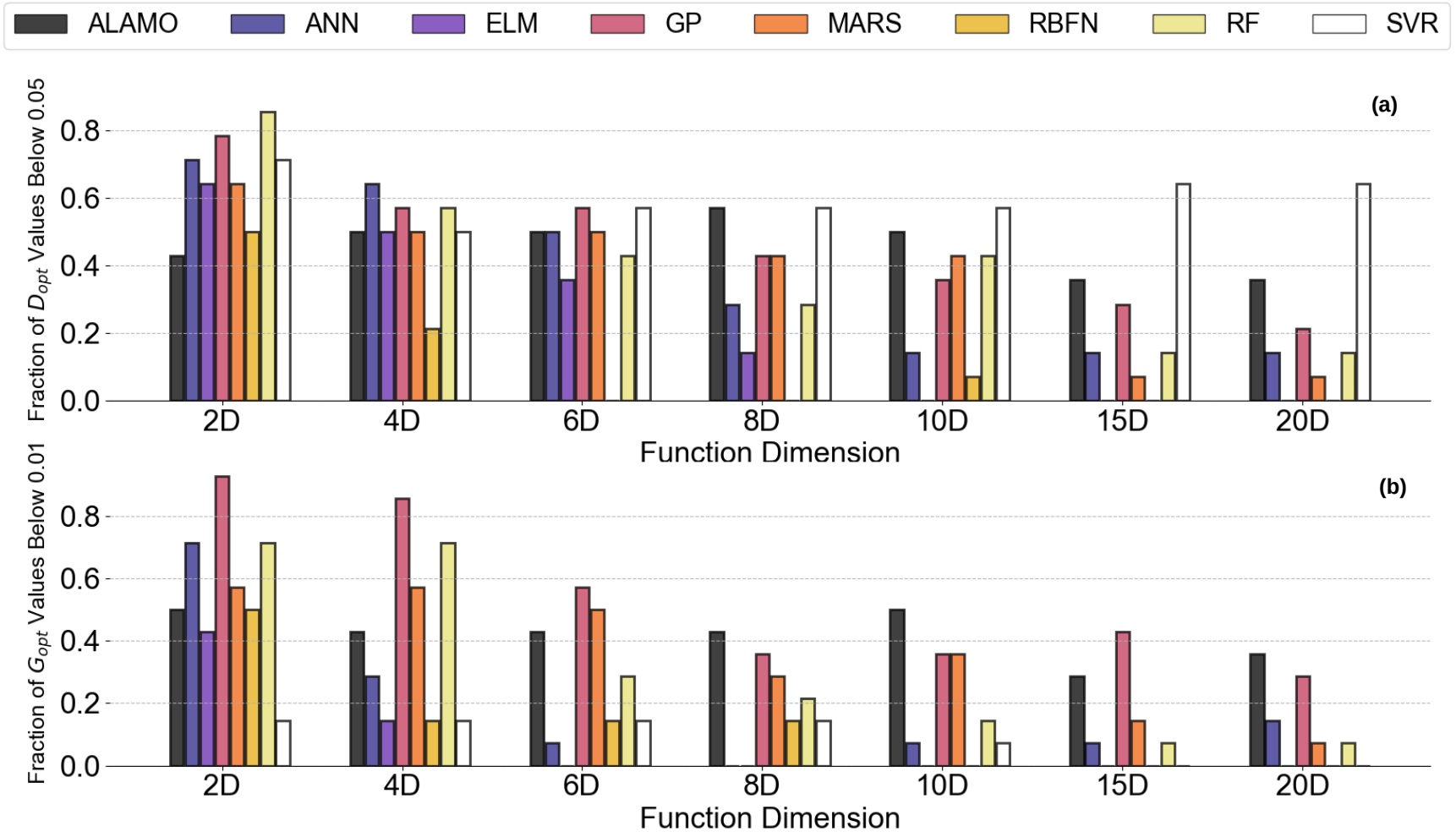


Figure 4.8 - Fraction of datasets with (a) D_{opt} and (b) G_{opt} less than threshold grouped by input dimension for sample size of 1600

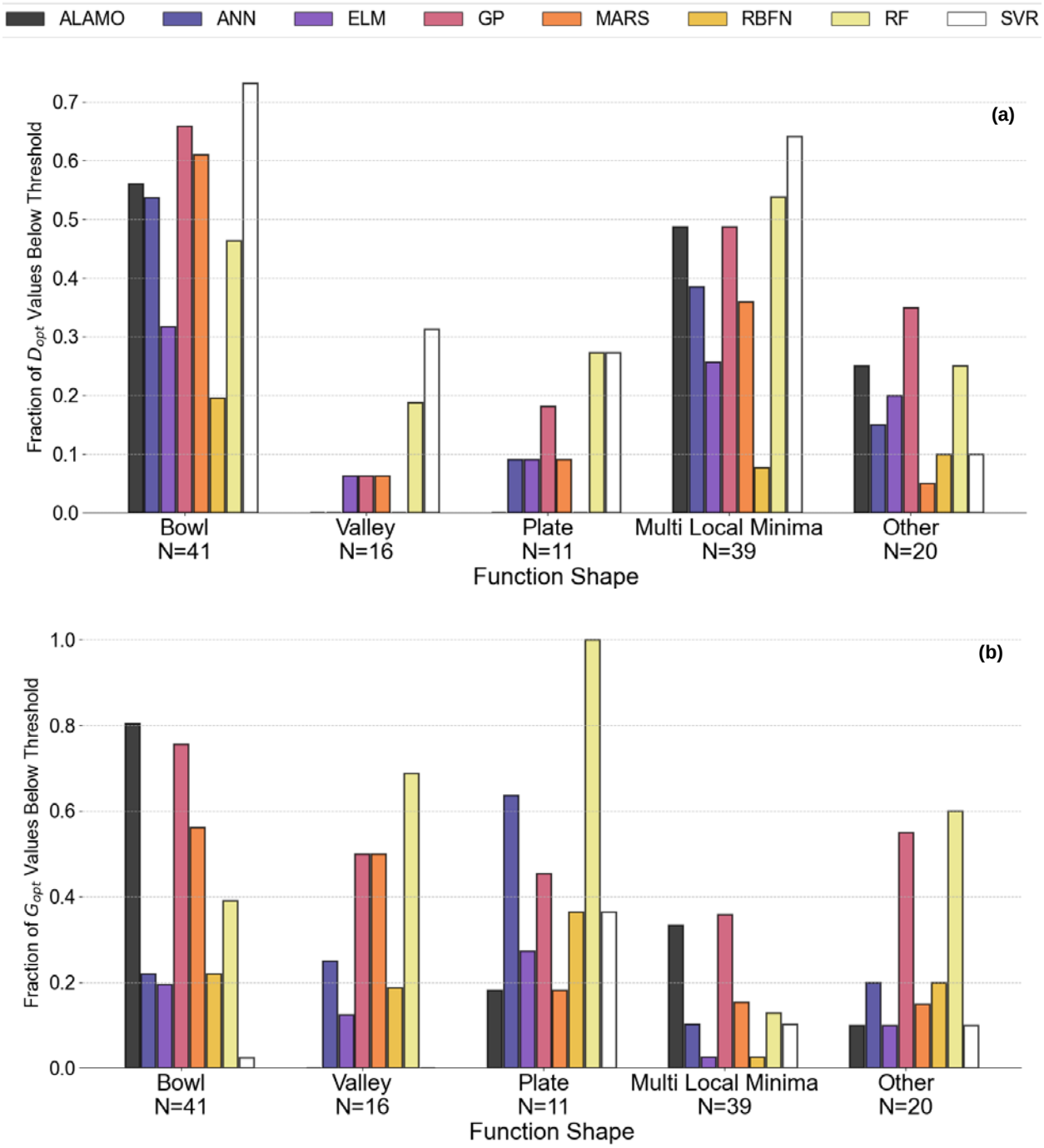


Figure 4.9 - Fraction of datasets with (a) D_{opt} and (b) G_{opt} less than threshold grouped by function shape for sample size of 1600. N values below the function dimensions indicate the number of test functions used for that input dimension

4.3.8 Functions for Which None of the Surrogate Modeling Techniques were Accurate

For both the surface approximation and surrogate-based optimization applications, there were some test functions for which none of the surrogate modeling techniques investigated were able to achieve accurate estimates, even at the largest sample size. The two-dimensional projections of the three functions that none of the surrogate modeling techniques were able to fit with an adjusted-R^2 of at least 0.90 are shown in Fig. 4.10(a) – (c). These functions all come from the *multi-local minima* shape category. The frequency of these functions' peaks may make the surfaces too noisy for approximating with any of the techniques, and other modeling approaches may be necessary to get an accurate approximation.

The two-dimensional projections of a selection of the functions that none of the surrogate modeling techniques located the optimum within a D_{opt} value of 5% are shown in Fig. 4.10(d) – (f). There were seven of these functions. When compared to the rest of the test functions, these seven had a range of output values that were several orders of magnitude higher, which may have given the solvers used difficulty in locating the optimum point. Most came from the *plate* and *valley*-shaped function categories. The large flat segments of these surfaces could have caused difficulty in locating the optimums, causing the solvers to get trapped in them. There was no overlap between the functions that were not modeled accurately for approximation and the functions whose optimum locations could not be found, further indicating that selection of a surrogate model for the two different applications may be unrelated. Although there were some common characteristics for the functions that could not be adequately modeled using these approaches, further work is needed on the specific characteristics of a dataset that may make it an inappropriate candidate for these traditional surrogate modeling methods.

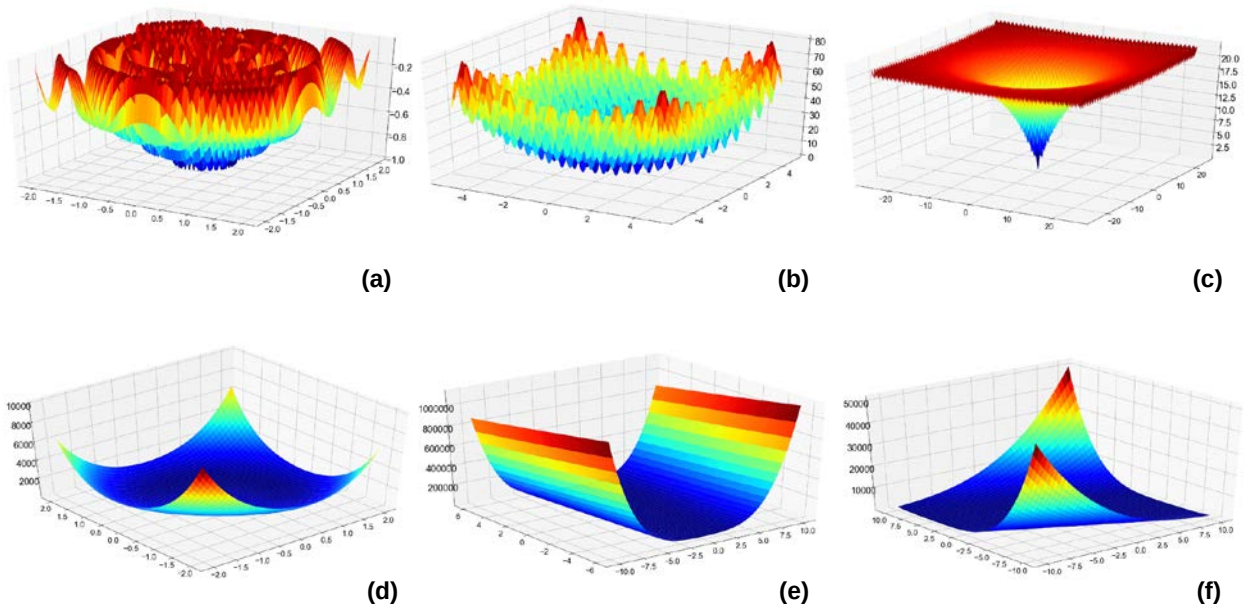


Figure 4.10 - Functions that could not be approximated by any of the surrogate models (a) – (c) or for which the optimum could not be located (d) – (f). (a) Eggholder function (*multi local minima*-shaped) (b) Rastrigin Function (*multi local minima*-shaped) (c) Ackley function (*multi local minima*-shaped) (d) Perm function (*bowl*-shaped) (e) Rosenbrock function (*valley*-shaped) (f) Zakharov function (*plate*-shaped)

4.4 Conclusions and Future Directions

The selection of the appropriate surrogate modeling technique depends on both the desired application of the surrogate model and the characteristics of the dataset being modeled. Although surface approximation using surrogate models is not significantly impacted by the choice of space-filling sampling method, the quality of solutions obtained from surrogate-based optimization can be dependent upon the sampling method, particularly at small sample sizes. For general selection rules, MARS and GP models give the most accurate predictions for design space approximation, and RF, SVR, and GP models give the most accurate estimations for surrogate-based optimization.

The main limitation of this study is that the analysis was carried out only on relatively smooth functions (with the exception of a few) with only continuous outputs. The results may not be applicable to more noisy data or to data that has binary or integer inputs and/or outputs. In addition, the “shape” data characteristic is not one that can readily be applied to other data in determining which surrogate modeling technique might be the most appropriate. Next chapter of this dissertation focuses on developing specific, quantifiable data characteristics related to the shape that can be calculated based only on available inputs and outputs and capturing the overall data behavior to make the recommendations for surrogate modeling selection more generalizable to other data.

Chapter 5 – Development of PRESTO (Predictive REcommendation of Surrogate models to approximate and Optimize)

Various techniques have been developed for constructing surrogate models for both regression and classification tasks (Breiman, 2001; Cozad et al., 2014; Drucker et al., 2002; Rasmussen & Williams, 2005). The current common practice for choosing a model form from the many available techniques relies on process-specific expertise or expensive trial-and-error methods. When selecting a surrogate model with user expertise, only a small subset of the many possible techniques that the user is most familiar with may be considered as candidates. This selection method, as well as trial and error, which is limited by computational resources, may fail to exploit the large pool of surrogate modeling techniques available and lead to a sub-optimal model selection. A systematic, automated procedure for selecting the appropriate surrogate model for a given application would avoid this issue.

This work aims to develop a framework to automatically select the set of surrogate models that will perform the best for a particular set of data based on the characteristics of the data and the application that the surrogate model would be used for. Our work comparing surrogate model performance demonstrated that there is a link between the characteristics of the dataset and how well different surrogate modeling techniques will perform for it for both surface approximation and surrogate-based optimization (Williams & Cremaschi, 2021b). To achieve this aim, we developed PRESTO (Predictive REcommendation of Surrogate models to approximate and Optimize), a random forest-based surrogate model selection tool. Given a set of data, PRESTO classifies each surrogate modeling technique in a set of candidate models as either recommended or not recommended based on the application, surface approximation or surrogate-based

optimization. The set of candidate surrogate modeling techniques considered by PRESTO includes Automated Learning of Algebraic Models using Optimization (ALAMO), single hidden layer feed-forward Artificial Neural Networks (ANN), Extreme Learning Machines (ELM), Gaussian Process Regression (GPR), Multivariate Adaptive Regression Splines (MARS), Radial Basis Function Networks (RBFN), Random Forests (RF), and Support Vector Machine Regression (SVR). The tool provides these recommendations without training any of the models, avoiding much computational expense.

5.1 PRESTO Construction Data and Surrogate Model Training

Datasets were generated for training surrogate models from a suite of optimization test functions (Surjanovic & Bingham, 2013). The functions with two, four, six, eight, ten, fifteen, and twenty input dimensions were utilized, resulting in 127 test functions. The test functions are grouped by their underlying functional shape. In this analysis, we have considered five shape categories: *bowl*-shaped, *plate*-shaped, *valley*-shaped, *multi-local-minima*-shaped, and *other*-shaped. Full descriptions of the characteristics of each shape category are provided in Williams and Cremaschi (2021). Input-output pairs were generated from each test function to create datasets at seven different sample sizes (50, 100, 400, 800, 1200, and 1600 samples) using Sobol sequence sampling (Joe & Kuo, 2008), a quasi-random low discrepancy sequence, and resulting in 791 generated datasets. Detailed information on the choice of sample sizes and sampling methods is described in Section 4.2.

A model was trained using each of the eight surrogate modeling techniques for each of the generated datasets, resulting in 6328 trained models. Each technique has a unique set of hyperparameters that was optimized while training the models for each dataset to construct the best possible surrogate model without overfitting. After the models were trained for each dataset

and sampling method, 100,000 input-output pairs were generated from the test functions using the Sobol sequence sampling method to test the accuracy of the surrogate models' predictions. To evaluate the performance of the surrogate models for surrogate-based optimization, the optimization models to determine the global minimum of each trained surrogate model were constructed in Pyomo (version 5.6), a Python-based optimization language (Hart et al., 2017; Hart et al., 2011). The resulting optimization problems were solved with a global solver most appropriate for the form of the problem (MINLP, MILP, or NLP) (Williams & Cremaschi, 2021b). Computations were carried out on the Auburn University Hopper HPC Cluster (Lenovo System X HPC Cluster) using Intel E5-2650 V3, 2.3 GHz 20 core processors and implemented in Python 3.7 and MATLAB 2017b (for RBFN surrogate models).

5.2 Feature Engineering and Attribute Extraction for Training PRESTO

Attributes calculated based only on the input and output values of each dataset were used as inputs for PRESTO's surrogate model recommendation classification. For surface approximation, the performance metric used to determine if a model would be considered recommended or not is the estimated adjusted R-squared for the model (Eq. 5.1). The adjustments used for each surrogate modeling technique are listed in Section 4.2. The performance metric used to make recommendations for surrogate-based optimization is the normalized Mahalanobis distance between the optimum point(s) estimated by the surrogate models and the actual optimum location of the underlying test function used to generate the model (Eq. 5.2).

$$\hat{R}^2 = 1 - (1 - R^2)\left[\frac{N-1}{N-(k+1)}\right] \tag{5.1}$$

$$D_{opt} = \frac{M\left(x_{opt}, \hat{x}_{opt}\right)}{\max\limits_{i,j} M\left(x_i, x_j\right)} \tag{5.2}$$

In Eq. (5.1), R^2 is the R-squared regression coefficient, N is the number of data points in the training set, and k is the number of model parameters (or hyperparameters). In Eq. (5.2), x_i and x_j are points in the domain space of the dataset, M is the Mahalanobis distance (De Maesschalck et al., 2000) between the location of the global minimum of a test function, x_{opt}, and the location estimated using a trained surrogate model, $\hat{x}_{opt}$. M is normalized by the maximum Mahalanobis distance between any two points (x_i, x_j) in the dataset (Eq. 5.3). Mahalanobis distance is the distance between two points in multivariate space. This distance between two objects, x and y, can be calculated as

$$M(x,y) = \sqrt{(x-y)^T C^{-1}(x-y)} \tag{5.3}$$

where C^{-1} is the sample covariance matrix. It has an advantage over Euclidean distance as it considers correlations in the dataset, and large scaling differences between the dimensions, because the distances are normalized with variance. The Mahalanobis distance is thus unitless and scale-invariant (De Maesschalck et al., 2000).

The attributes are used to capture the overall behavior of the datasets using numeric measures. A total of 38 attributes were defined for the datasets. Twenty of the attributes were previously defined and described in detail in Garud et al. (2018). These include attributes related to estimated gradients and curvatures, attributes related to the distribution of the outputs (such as the first four moments of the output value distribution), and attributes related to the dataset's extreme minimum and maximum values.

An additional 18 new attributes are defined in this work. LEAPS2, the model selection framework described in (Garud et al., 2018), was only trained to select models for surface approximation. In addition, there were no attributes directly related to the distributions of the

inputs in the dataset. These additional attributes were constructed to include information about the arrangement of the inputs for the datasets and provide characteristics of the data that may have a larger effect on the optimization performance.

5.2.1 Input Related Attributes

In Eqs. (5.4) – (5.6), $\mathrm{M}(x_i, x_j)$ is the Mahalanobis distance between any two points x_i and x_j in the domain space. Let D be equal to the number of input dimensions in the dataset and N be equal to the total number of data points.

Minimum Mahalanobis distance: This is the minimum Mahalanobis distance between any two points in the input space (Eq. 5.4).

$$M_{min} = \min_{i,j} \mathrm{M}(x_i, x_j) \tag{5.4}$$

Maximum Mahalanobis distance: This is the maximum Mahalanobis distance between any two points in the input space (Eq. 5.5).

$$M_{max} = \max_{i,j} \mathrm{M}(x_i, x_j) \tag{5.5}$$

Average Mahalanobis distance: This is the average Mahalanobis distance between any two points in the input space (Eq. 5.6).

$$M_{avg} = \frac{1}{N^2} \sum \mathrm{M}(x_i, x_j) \tag{5.6}$$

Euclidean to Mahalanobis distance ratio: This ratio of the average pairwise Euclidean distance to the average Mahalanobis distance estimates the level of correlation, if any, between the dataset inputs and the magnitude of variance. When there is no correlation between the variables, the covariance matrix used to calculate the Mahalanobis distance becomes the identity covariance

matrix, and the Euclidean and Mahalanobis distances become equal to each other, which makes the value of this ratio one. The average Euclidean distance is calculated as

$$E_{avg} = \frac{1}{N^2}\sum\|x_i - x_j\| \tag{5.7}$$

The Euclidean to Mahalanobis distance ratio can then be estimated by

$$R_{E/M} = \frac{E_{avg}}{M_{avg}} \tag{5.8}$$

5.2.2 Gradient-Based Attributes

From Garud et al. (2018), for any point in the data set, $x^{(i)}$, let $x^{(j)}$ be its nearest neighbor based on the Euclidean distance and $y^{(i)}$ be its response. Then, the gradient vector of the response, $g^{(i)}$, at $x^{(i)}$ can be estimated using Eq. (5.9), where $x_d^{(i)}$ is the value of input dimension d for point $x^{(i)}$, and e is a small number related to the precision of the numbers in the dataset. These gradient-based attributes were added to the attribute set because the gradients indicate the overall shape of the surface, and our studies have shown that surrogate-based optimization performance is dependent on the underlying shape of the surface being modeled (Williams & Cremaschi, 2021b).

$$g^{(i)} = \left[g_d^{(i)} = \frac{\left(y^{(i)} - y^{(j)}\right) sign\left(x_d^{(i)} - x_d^{(j)}\right)}{max\left[e, \left|x_d^{(i)} - x_d^{(j)}\right|\right]} \middle| \, d = 1,2,\dots,D \right] \tag{5.9}$$

In Eq. (5.9), the sign function is defined as

$$sign\left(x_d^{(i)} - x_d^{(j)}\right) = \begin{cases} 1, & if\ x_d^{(i)} - x_d^{(j)} \geq 0 \\ -1, & otherwise \end{cases} \tag{5.10}$$

Average magnitude of gradient vector: This attribute (Eq. 5.11), which is the average value of the magnitude of the gradient vector $g^{(i)}$, across all sample points, aims to provide a measure of the average steepness of the surface being modeled.

$$g_{avg} = \frac{1}{N}\sum_{i=1}^{N}\left|g^{(i)}\right| \tag{5.11}$$

Standard deviation of gradient vector magnitudes: The standard deviation of the magnitude of the gradient vector $g^{(i)}$ across all sample points gives an estimate of the non-linearity of the surface (Eq. 5.12).

$$g_{std} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(\left|g^{(i)}\right| - g_{avg}\right)^2} \tag{5.12}$$

Minimum and maximum gradient vector magnitudes: These attributes describe the minimum (Eq. 5.13) and maximum (Eq. 5.14) values for the magnitudes of the gradient vectors for sample points in the data set.

$$g_{min} = \min_{N} g^{(i)} \tag{5.13}$$

$$g_{max} = \max_{N} g^{(i)} \tag{5.14}$$

Ratios of gradient vector magnitudes: These attributes aim to capture the average "bumpiness" or noisiness of the surface by measuring how sharply the gradients change on average throughout the surface (Eqs. 5.15-5.17). Higher values of these ratios indicate a noisier surface.

$$g_{avg,min} = \frac{g_{avg}}{g_{min}} \tag{5.15}$$

$$g_{avg,max} = \frac{g_{avg}}{g_{max}} \tag{5.16}$$

$$g_{max,min} = \frac{g_{max}}{g_{min}} \tag{5.17}$$

Skewness of gradient vector magnitudes: The skewness of the gradient magnitudes estimates a measure of the asymmetry of the distribution of the gradient vector magnitudes (Eq. 18).

$$g_{skew} = \frac{\sum_{i=1}^{N}\left(\left|g^{(i)}\right| - g_{avg}\right)^3}{N(g_{std})^3} \tag{5.18}$$

5.2.3 Response (Output)-Based Attributes

These response-based attributes were developed and added to the attribute set to provide insights into how concentrated (or sparse) the output values are distributed at the extreme high and low output values. Data that is concentrated in certain areas and not well-distributed over the entire possible output space may produce models whose predictions do not generalize well over the space. However, if data is concentrated at extreme values, a trained model may be better able to closely locate the optimum for the dataset.

Upper and lower tail average: These attributes calculate the average value of the response values in the top 5% and bottom 5% of responses.

Upper and lower tail relative size: This is the ratio of the number of output responses in the top (Eq. 5.19) and bottom (Eq. 5.20) 5% of values to the total number of data points. These attributes aim to estimate how well distributed the response values are over the range of responses. In Eqs. (5.19) and (5.20), N_u and N_l represent the number of output responses in the top 5% and bottom 5% of values, respectively.

$$u_{size} = \frac{N_u}{N} \tag{5.19}$$

$$l_{size} = \frac{N_l}{N} \tag{5.20}$$

Ratio of lower to upper tail size: This ratio (Eq. 5.21) describes how evenly the output responses are distributed between the upper and lower extremes of the output values.

$$r_{l/u} = \frac{l_{size}}{u_{size}} \tag{5.21}$$

5.2.4 Other Attributes

Average local convex deviation: This deviation (Eq. 5.27) aims to estimate the convexity of the function from which the input-output data was generated. We hypothesize that this metric may be important for determining the appropriate surrogate modeling technique for surrogate-based optimization.

Let $c^{(m)}, m = 1,2, \dots, M$ be some sample points generated on the input domain of the dataset using Latin hypercube sampling.

$$d_{min} = \min_{m \neq n} \left\| c^{(m)} - c^{(n)} \right\| \tag{5.22}$$

We can construct a hypersphere of diameter d_{min} (Eq. 5.22) around each point $c^{(m)}$ to create a local "neighborhood" of dataset points $x^{(i)}$ around each center, where d_{min} is the minimum distance between $c^{(m)}$ and any other generated sample point, $c^{(n)}$. Then, we define the convex difference for each point in the neighborhood as in Eq. (5.23),

$$C_{diff}^{(i),(m)} = \left| y^{(i)} - y_{convex}^{(i)} \right| \tag{5.23}$$

and the average convex difference in the neighborhood is given in Eq. (5.24),

$$\bar{C}_{diff}^{(m)} = \frac{1}{K} \sum_{i=1}^{K} C_{diff}^{(i),(m)} \tag{5.24}$$

where $y_{convex}^{(i)}$ is the response of a known convex function (Eq. (25))

$$y_{convex}^{(i)} = 0.1\left(x^{(i)}\right)^4 \tag{5.25}$$

for the input $x^{(i)}$. Figure 5.1 illustrates the process for calculating $C_{diff}^{(i),(m)}$.

The local convex deviation in each neighborhood can then be calculated as in Eq. (5.26)

$$C_{dev}^{(m)} = \sqrt{\frac{1}{K-1}\sum_{i=1}^{K}\left(C_{diff}^{(i),(m)} - \bar{C}_{diff}^{(m)}\right)^2} \tag{5.26}$$

where K is the number of points from the dataset in the neighborhood m. The average local convex deviation is given in Eq. (5.27).

$$\bar{C}_{dev} = \frac{1}{M}\sum_{i=1}^{M} C_{dev}^{(m)} \tag{5.27}$$

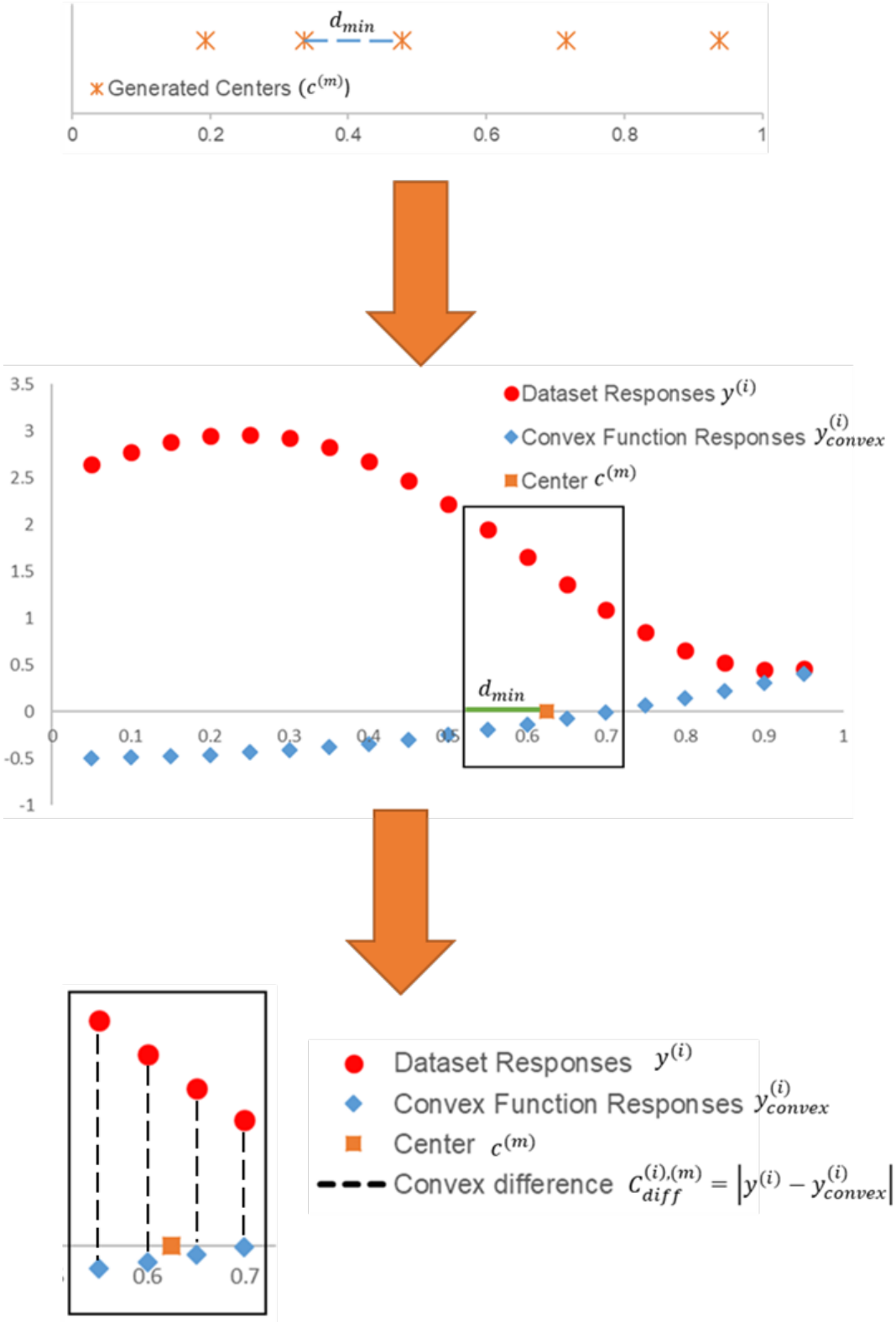


Figure 5.1 - Steps for generating neighborhoods for convex difference calculations

5.3 PRESTO Framework Construction

Random forest classification models were trained for each surrogate modeling technique to predict whether the surrogate should be recommended or not recommended for a dataset. Random forests are surrogate models that make output predictions based on inputs by combining predictions from a collection of decision trees. Each tree in a random forest model is constructed independently and depends on a random vector sampled from the input data, with all the trees in the forest having the same distribution (Brieman, 2001). Random forests have successfully been used for both regression and classification tasks, performing with high prediction accuracy for both small sample sizes and high dimensional data.

Separate classification models were trained for surface approximation and surrogate-based optimization. The calculated attributes were used as inputs, and the assigned recommendation classes ("recommended" or "not recommended") were used as outputs. To assign recommendation classes for a dataset, the highest or lowest value out of all the eight surrogate techniques for $\hat{R}^2$ and D_{opt}, respectively, were assigned as "recommended", as they had the "best" performance for that dataset. Then, surrogate models with performance metric values within 1% of those best values were also assigned as "recommended." Any surrogate models with metric values outside of the 1% range were assigned a recommendation class of "not recommended."

The built-in feature selection method of random forest models was performed to determine which attributes had the most influence on the predicted recommendation class for each surrogate modeling technique. Input features are assigned an importance value in random forest models based on how much they reduce the Gini impurity (Menze et al., 2009) at each decision node in the forest. The Gini impurity measures how well the decision threshold separates the training samples into the two classes at a particular node (Menze et al., 2009). The feature importances of

all the input features (in this case, the attributes) sum to 100%. Attributes were ranked from highest to lowest feature importance. The attributes were added to the input feature set starting with the highest importance up to a sum of 90% of the total importance to reduce the attribute set for the classification model for each surrogate modeling technique. Here, the goal is to consider the attributes with the highest impact on the classification model outcome in the random forest classifier model. The remaining features in the lower 10% of the importance sum were discarded. Figure 5.2 summarizes the PRESTO construction steps. PRESTO is available for testing in the Cremaschi research group GitHub repository (https://github.com/CremaschiLab/PRESTO).

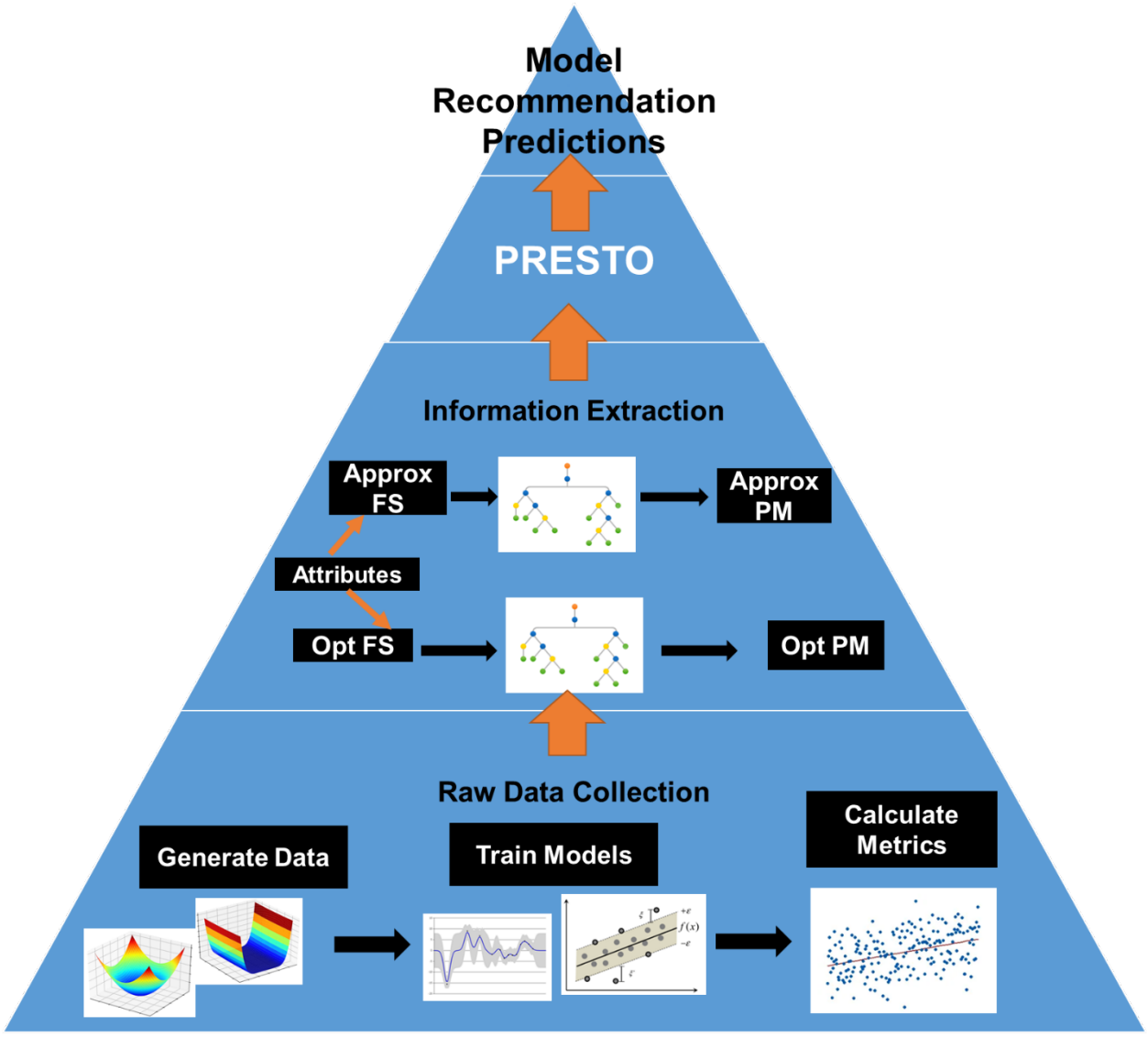


Figure 5.2 - Summary of PRESTO construction (FS = Feature Selection, Approx = Surface Approximation, Opt = Surrogate-based optimization, PM = Classification Performance Metrics)

5.4 PRESTO Performance Evaluation Criteria

The performance of PRESTO was evaluated using three performance metrics: accuracy, precision, and hit ratio. These metrics are calculated based on the classification confusion matrix (Sokolova & Lapalme, 2009) (Figure 5.3), which describes the quality of classifications made by a classification model.

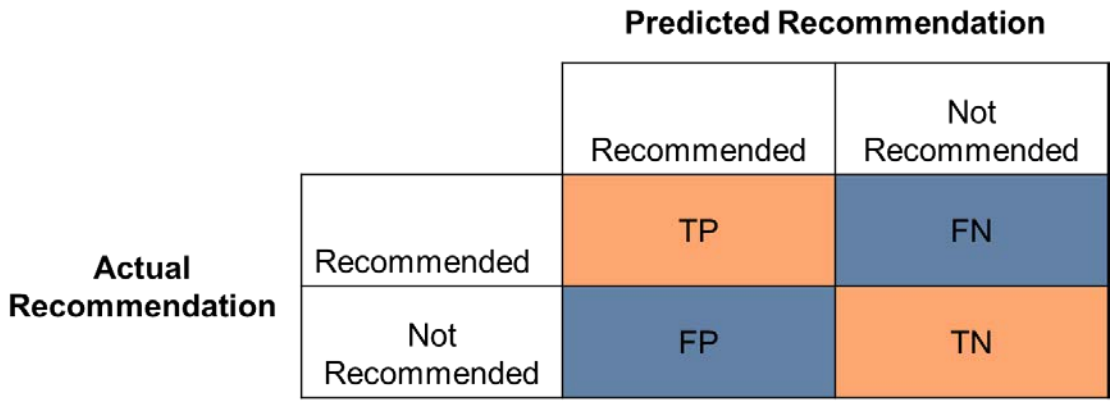


Figure 5.3 - Classification confusion matrix (TP = true positive, TN = true negative, FP = false positive, FN = false negative)

The accuracy (Eq. 5.28) measures the percentage of recommendation classifications made by PRESTO that is correct. The precision (Eq. 5.29) is the probability that a model classified as recommended should actually be recommended and will perform well for a dataset. The hit ratio (Eq. 30) is the percentage of models that will perform well for a dataset that PRESTO is recommending.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.28}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.29}$$

$$Hit\ ratio = \frac{TP}{TP + FN} \tag{5.30}$$

Figure 5.4 provides a flowchart depicting how the performance metrics were calculated for evaluating PRESTO's performance. The first step is similar to how a user would use PRESTO, where input-output data is generated and passed to PRESTO. PRESTO then provides recommendations for which surrogate modeling techniques to employ. For the second step in the

analysis, all eight of the candidate surrogate modeling techniques were used to train models, and their performance metrics were calculated. These metrics of actual performance were compared to the recommendations for models PRESTO predicted would perform well for the data to evaluate the quality of PRESTO's recommendations. Training all of the models as was done in this analysis is not necessary for a PRESTO user. In practice, a user could train as few or as many of the recommended models as desired.

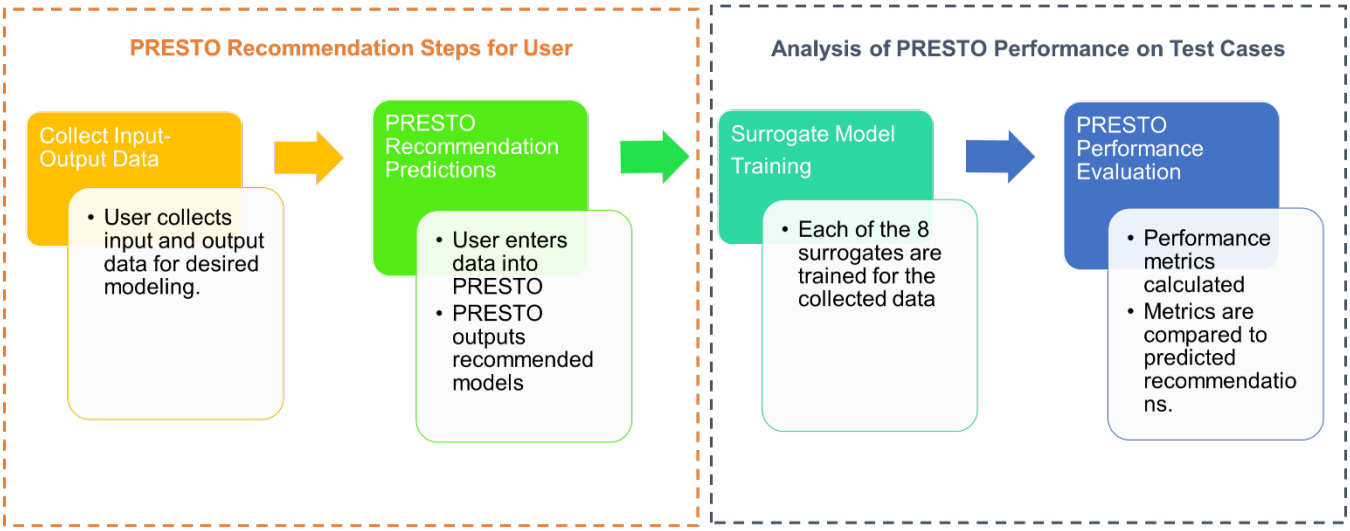


Figure 5.4 - Summary of PRESTO performance evaluation

5.5 Application Dependent PRESTO Attributes for Surrogate Modeling Techniques

The numbers of attributes selected for classifying the eight candidate surrogate modeling techniques as being recommended or not recommended for surface approximation and surrogate-based optimization are given in Table 5.1. For example, for surface approximation, 21 attributes out of the initial set of 38 were selected as inputs for the classifier trained to make predictions regarding ALAMO. Based on these results, the random forest classifier required approximately 39

- 55% of the attributes for making recommendations. There were no significant differences between the number of attributes selected for each surrogate modeling technique or application.

Table 5.1 - Number of attributes selected for recommendation predictions

	Attributes Selected	
	Surface Approximation	Surrogate-Based Optimization
ALAMO	21	21
ANN	20	19
ELM	19	20
GPR	21	20
MARS	21	20
RBFN	20	20
RF	15	21
SVR	21	21

Tables 5.2 and 5.3 list the five highest important attributes selected for surface approximation and surrogate-based optimization, respectively. Attributes related to the dataset inputs, including the average, minimum, and maximum Mahalanobis distances between input data points, were selected frequently for the majority of the surrogate modeling techniques for both surface approximation and surrogate-based optimization performance predictions. Other commonly selected features include those related to the distributions of output values, specifically the relative size of the output distribution tails and the output distribution skewness and kurtosis. These results suggest that the distribution and location of the sample points and the relative steepness and smoothness of the surface significantly influence how well each of the surrogate models can approximate that surface and locate the optimum of the underlying function.

Attributes related to the distributions of the input and output locations were commonly selected among all of the candidate techniques. These attributes may affect surface approximation performance as having data unevenly concentrated (or sparse) at the extreme values may skew models to predict more accurately in areas of data concentration and less so for other areas of the design space. For example, in the case of RF models, uneven tails could cause decision nodes in the model trees to split more frequently at the extremes of the output values while more finely split partitions were really needed elsewhere, such as where the gradients were steep. For the neural network-based models, the on-off nature of the hidden layer nodes may make them more suitable for making accurate predictions for surfaces where large areas of the design space have similar output values, creating flat or nearly flat areas, similar to the *plate*-shaped functions.

For surface approximation, the attributes of the empirical mean of fractional local fluctuations and the empirical standard deviation in fractional local fluctuations were also frequently selected in the top five attributes. These attributes measure the average bumpiness and non-linearity variations, respectively, of the surface being modeled (Garud et al., 2018) and can be considered to give a measure of the noisiness of the surface. The noise level has a significant effect on some models' ability to fit a surface. For example, for SVR model performance, the support vectors fitted in the model construction can easily become sensitive to noise as they are only dependent on a small set of the data used to train the model (Sabzekar et al., 2011).

Table 5.2 - Five highest important attributes selected for surface approximation

ALAMO	
Attribute	Importance
Euclidean to Mahalanobis ratio	11.1%
Skewness of outputs	8.7%
Kurtosis of outputs	7.2%
Upper tail average	5.8%
Coefficient of variation of outputs	5.4%

ANN	
Attribute	Importance
Coefficient of variation of outputs	10.4%
Upper tail average	9.3%
Average Mahalanobis distance	6.6%
Average gradient cosine direction	6.0%
Kurtosis of outputs	5.5%

ELM	
Attribute	Importance
Average Mahalanobis distance	12.2%
Minimum Mahalanobis distance	12.0%
Input dimensions	9.7%
Empirical standard deviation in fractional local fluctuations	8.9%
Kurtosis of outputs	6.6%

GPR	
Attribute	Importance
Upper tail average	7.6%
Coefficient of variation of outputs	7.6%
Empirical mean of fractional local fluctuations	6.4%
Skewness of outputs	6.3%
Empirical standard deviation in fractional local fluctuations	5.7%

Table 5.2 cont'd. - Five highest important attributes selected for surface approximation

MARS	
Attribute	Importance
Kurtosis of outputs	11.5%
Empirical standard deviation in fractional local fluctuations	7.6%
Skewness of outputs	7.1%
Relative size of upper tail	6.7%
Upper tail average	5.0%

RBFN	
Attribute	Importance
Average Mahalanobis distance	10.4%
Minimum Mahalanobis distance	9.2%
Empirical mean of fractional local fluctuations	7.0%
Empirical standard deviation in fractional local fluctuations	7.0%
Skewness of outputs	5.5%

RF	
Attribute	Importance
Empirical standard deviation in fractional local fluctuations	21.3%
Empirical mean of fractional local fluctuations	15.6%
Coefficient of variation of gradient magnitudes	7.3%
Average Mahalanobis distance	7.1%
Minimum Mahalanobis distance	7.0%

SVR	
Attribute	Importance
Skewness of outputs	8.0%
Empirical mean of fractional local fluctuations	7.7%
Kurtosis of outputs	6.3%
Empirical standard deviation in fractional local fluctuations	5.7%
Skewness of gradient magnitudes	5.1%

Table 5.3 - Five highest important attributes selected for surrogate-based optimization

ALAMO	
Attribute	Importance
Upper tail average	9.3%
Coefficient of variation of outputs	9.2%
Skewness of outputs	8.0%
Lower tail relative size	6.2%
Average Mahalanobis distance	5.8%

ANN	
Attribute	Importance
Input dimensions	12.3%
Average Mahalanobis distance	12.1%
Maximum Mahalanobis distance	9.3%
Minimum Mahalanobis distance	7.7%
Kurtosis of outputs	7.7%

ELM	
Attribute	Importance
Average Mahalanobis distance	18.6%
Maximum Mahalanobis distance	12.9%
Minimum Mahalanobis distance	10.7%
Input dimensions	10.0%
Coefficient of variation of outputs	4.4%

GPR	
Attribute	Importance
Input dimensions	9.5%
Average Mahalanobis distance	9.1%
Coefficient of variation of outputs	7.0%
Maximum Mahalanobis distance	6.6%
Skewness of outputs	6.5%

MARS	
Attribute	Importance
Input dimensions	13.8%
Average Mahalanobis distance	13.0%
Minimum Mahalanobis distance	8.0%
Maximum Mahalanobis distance	6.2%
Kurtosis of outputs	5.2%

RBFN	
Attribute	Importance
Minimum Mahalanobis distance	13.9%
Average Mahalanobis distance	10.4%
Input dimensions	10.3%
Maximum Mahalanobis distance	6.6%
Skewness of outputs	4.8%

RF		SVR	
Attribute	Importance	Attribute	Importance
Average Mahalanobis distance	8.8%	Skewness of outputs	11.0%
Minimum Mahalanobis distance	6.4%	Upper tail average	8.8%
Input dimensions	6.3%	Coefficient of variation of outputs	8.0%
Maximum Mahalanobis distance	5.7%	Average Mahalanobis distance	5.3%
Euclidean to Mahalanobis ratio	5.6%	Kurtosis of outputs	5.2%

For surrogate-based optimization, although not selected as one of the top five important attributes, the average local convex deviation, and attributes related to the estimated gradients were selected frequently in the set of important attributes. The convexity of a model has a significant effect on the relative "ease" of finding its global minimum. Gradient information is also crucial in the application of gradient-based methods for optimization. The complete listing of all attributes selected is available in Appendix B.

5.6 Performance Evaluation of PRESTO for a Chemical Engineering Application - Cumene Production Case Study

A simulation model of the cumene production process was employed to test the performance of PRESTO's recommendations for a chemical engineering application and on datasets that were not used for its training. The entire process for cumene production was simulated in gPROMS Process. Input-output datasets were generated for a subset of the unit operations in the flowsheet, using the gPROMS Global System Analysis capabilities. PRESTO was used to provide surrogate modeling technique recommendations to predict each output for surface approximation. Then, surrogate models were trained for each output using the eight candidate

surrogate modeling techniques, and the corresponding adjusted R-squared values were calculated using Monte Carlo cross-validation (Xu et al., 2004) with a test set size of 20% of the dataset and 50 Monte Carlo trials. The adjusted R-squared results were compared to the recommendation classifications made by PRESTO for surface approximation.

5.6.1 Process and Simulation Description for Cumene Production

The production of cumene, a petrochemical used in the production of several chemicals, involves the reaction of benzene with propylene to form cumene and the undesirable reaction of cumene with propylene to form p-diisopropyl benzene (PDIB) (Luyben, 2011). The flowsheet for the process is given in Figure 5.5. The case study focuses on the cooled tubular reactor (Reactor) and two distillation columns (C1 and C2). These are the three most complex unit-level models in the overall flowsheet, meaning that replacing them with surrogate models will have the greatest impact on improving computational speed. In the process, the liquid fresh feed streams are combined with a benzene recycle stream, vaporized, preheated to 360 °C, and fed to the cooled tubular reactor. The first distillation column, C1, produces a mostly benzene distillate, which is recycled back to the reactor. This recycle stream is necessary to keep benzene from exiting in the bottoms product and affecting the purity of the cumene product in the distillate of the second column, C2. Column C2 is designed to attain high-purity cumene in the distillate and minimize the loss of cumene in the bottoms (Luyben, 2011).

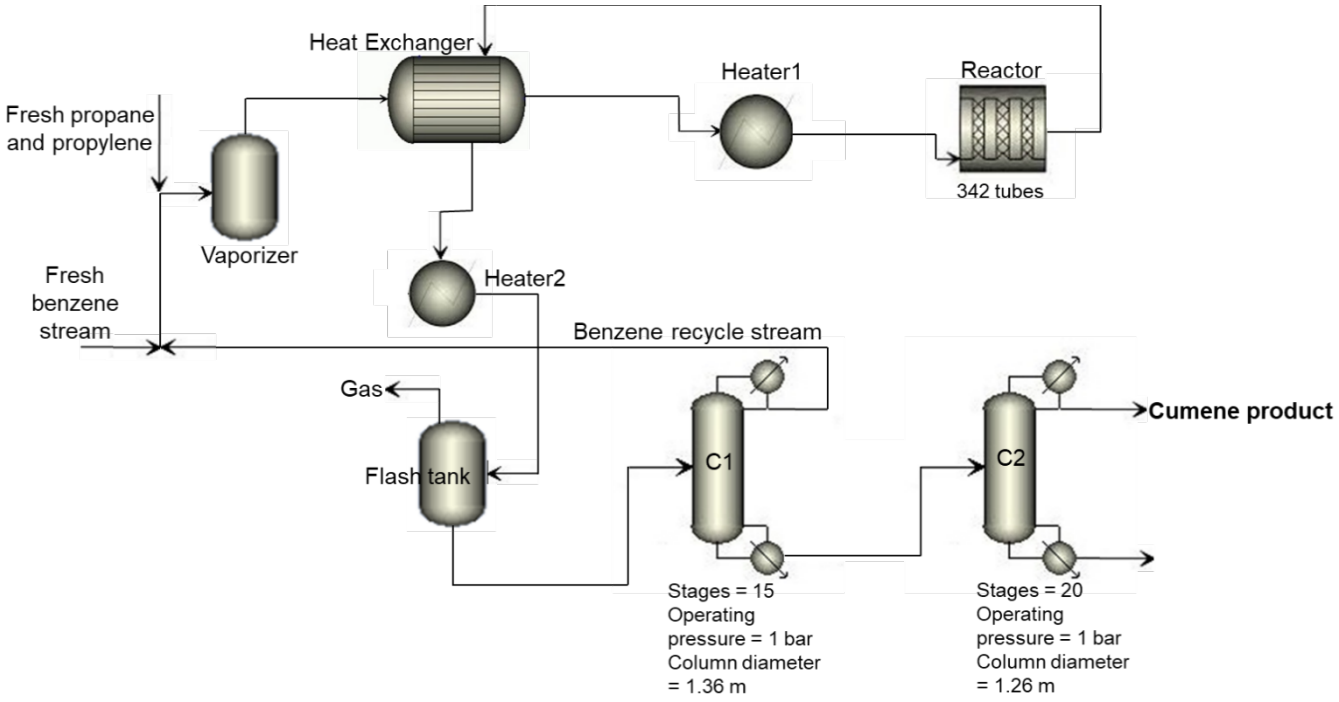


Figure 5.5 - Flowsheet for cumene production case study

The entire process was simulated in gPROMS Process, where the data for the three complex unit operations was generated. Data were generated for a total of 27 outputs, with seven outputs for the reactor, 11 outputs for the first distillation column (C1), and nine outputs for the second distillation column (C2). Outputs for the distillation columns include heat duties and top and bottoms product compositions. Outputs for the reactor include outlet temperatures and reaction product compositions. For each output, data was randomly selected at four different sample sizes (100, 300, 1000, and 3000 samples) for a total of 108 case study datasets for evaluating PRESTO's performance. The input values for each dataset are operating specifications for a fixed design of the respective unit operations, such as inlet temperatures and inlet compositions.

5.7 Results and Discussion

5.7.1 PRESTO Recommendation Classification Results

The selected attributes were used as inputs to train random forest classification models for the eight techniques to classify each technique as being “recommended” or “not recommended” for a given dataset. Separate classifiers were trained for surface approximation and surrogate-based optimization. This recommendation scheme allows for multiple similarly performing surrogate modeling techniques to be suggested for use. The performance metrics were calculated using Monte Carlo cross-validation with 75 Monte Carlo trials. Each trial had a test set size of 20% of the total dataset.

PRESTO identified which techniques should be recommended for the simulated datasets for surface approximation with an accuracy of 91%. The precision, or the probability that a recommended technique should actually be recommended, was 90%. The hit ratio, the percentage of the surrogate models that should have been recommended for surface approximation that PRESTO successfully captured, was 87%.

For surrogate-based optimization, PRESTO recommended surrogate modeling techniques with an accuracy of 98% and a precision of 99%. This result indicates that if PRESTO identifies a model as being recommended for a dataset, there is a 99% probability that the model will accurately locate the global optimum of the underlying model for the dataset. The hit ratio for surrogate-based optimization was 98%.

5.7.2 Cumene Case Study Performance Results

Table 5.4 lists the performance metrics for PRESTO on the case study data compared to the metrics for the data that PRESTO was trained on, or the PRESTO data. Performance metrics

were calculated for both to compare how well PRESTO performed on the simulated data from the test functions to how it performed on data from a real-world application. PRESTO's performance on the cumene case study data was similar to that of the tool training data for accuracy and precision. However, the hit ratio was slightly lower. The lower hit ratio for the case study data indicates that PRESTO was not identifying all the possible models that could be recommended, only a subset of them, which also resulted in a higher precision. These results suggest that PRESTO may successfully identify a set of surrogate models that will perform well for approximating the behavior of a data set for some relevant cases without the need for expensive trial-and-error methods.

The lower hit ratio may be due to the fact that the values for the Mahalanobis distances between the data points for the case study data were outside the ranges of the distances for the data that the tool was trained on. For example, the maximum value for the PRESTO data for the maximum Mahalanobis distance between data points is approximately 6.7, while both the average and maximum value of that same attribute for the case study data are higher at 6.8 and 7.1, respectively. The simulated data was generated using the same space-filling method, while the inputs for the case study data were generated randomly. We can conclude that the position of the sample points, or the distances between them, are critical in providing accurate recommendations as features related to the Mahalanobis distances were selected as important for almost all of the classification models that were trained. The performance of PRESTO could be improved by adding datasets that use a variety of sampling methods in order to provide better ranges for the attributes related to data point distribution.

In addition, all of the datasets in the PRESTO data were created using relatively smooth, continuous functions. Data from real applications may not share the same characteristics. An

example of this difference in behavior can be seen in the average local convex deviation attribute. The average value for this attribute is $2.66x10^6$ for the PRESTO data and several orders of magnitude higher for the case study data at $9.1x10^{12}$. The recommendations of PRESTO for real data could be enhanced by the addition of real datasets to the PRESTO training data. However, with a 94% precision for the case study data, PRESTO's predictions for which surrogate models to use are still accurate. All of the compiled results for adjusted R-squared and recommended models for the case study data are available in the supplementary materials for Williams et al. (2021).

Table 5.4 - PRESTO case study performance comparison

	Case Study Data	PRESTO Data
Accuracy	89%	91%
Precision	94%	90%
Hit Ratio	76%	87%

PRESTO did not recommend any candidate surrogate models for two process outputs: the bottom product temperature and top liquid recovery of cumene. These classifications of "not recommended" for all the surrogates were correct, as when the models were trained, none of the techniques could successfully approximate these outputs with an adjusted R-squared above 0.7. Our work demonstrated that there are some test functions that were used to train PRESTO, for which none of the surrogate modeling techniques were able to approximate the surface (Williams & Cremaschi, 2021b). In these instances, alternative modeling strategies, such as ensemble modeling, deep learning algorithms, or another surrogate modeling technique not included in the

candidate set, may be considered. It should be noted that when PRESTO does not recommend any surrogate modeling techniques, it could also indicate that the current data set size is too small for the techniques to model the input-output relationship accurately. We observed that for some datasets, increasing its size also increased the number of recommended models. Hence, we also recommend increasing the dataset size and re-running PRESTO. The case study results reveal that PRESTO can capture the qualities of datasets that would make them unsuitable for modeling with the eight candidate techniques studied.

5.7.2.1 PRESTO Cosine Similarity Analysis

In order to further analyze the discrepancy in PRESTO's performance on the cumene case study data, the calculated dataset attributes of the cumene data were compared to the attributes of the PRESTO training data using the cosine similarity (Eq. 5.31). The cosine similarity between two vectors of dataset attributes X and Y falls between values of -1 and 1, with similarity values closer to 1 indicating a higher measure of similarity

$$\cos(X, Y) = \frac{X \cdot Y}{\|X\|\|Y\|} \tag{5.31}$$

and for vector $X = \langle x_1, x_2, \ldots\ldots, x_n \rangle$

$$\|X\| = \sqrt{\sum_{;=1}^{n} x_i^2} \tag{5.32}.$$

When calculating the pairwise cosine similarity between each of the 791 datasets used to construct PRESTO, their average similarity was 0.805, indicating that there is a high degree of similarity between the PRESTO training datasets. Figure 5.6 shows the frequency distribution of the cosine similarities for the PRESTO training data. For comparison of the cumene case study

data to the PRESTO training data, the cosine similarity for cumene data set to each of the PRESTO datasets was calculated, and the average was taken. This calculation was repeated for each of the cumene datasets. Table 5.5 lists the performance metric values for the cumene datasets when separated by those that had a negative or positive average cosine similarity to the PRESTO data.

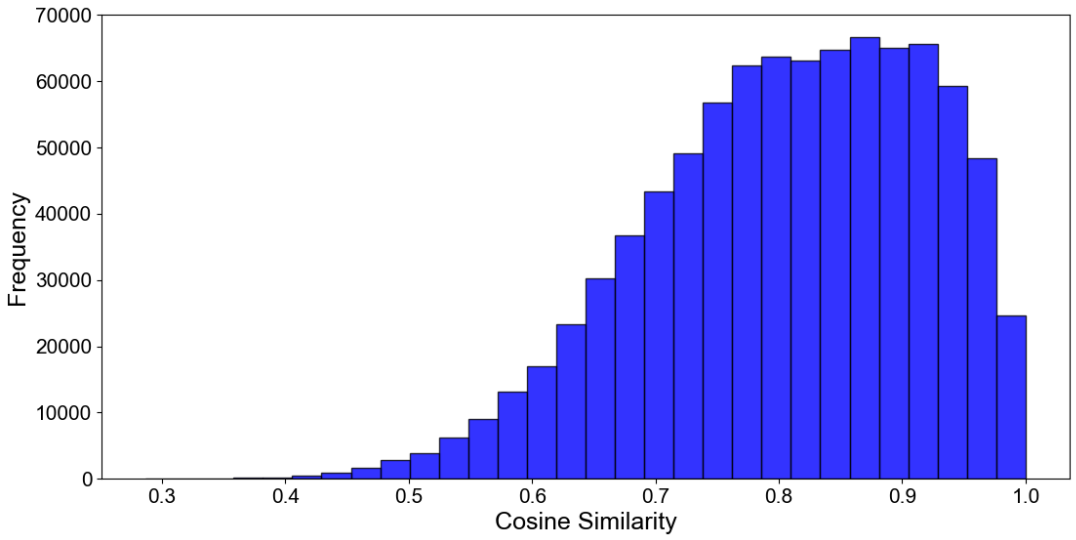


Figure 5.6 – Histogram of cosine similarity scores for PRESTO training data

Table 5.5 - Cumene data performance by cosine similarity score

	Cumene Data		PRESTO Data
	Negative Average Cosine Similarity	Positive Average Cosine Similarity	
Accuracy	86%	92%	91%
Precision	86%	94%	90%
Hit Ratio	67%	85%	87%

The performance metrics of the cumene datasets with positive average cosine similarity to the PRESTO training data were higher than those of the datasets with negative average similarity and

more comparable to the cross-validation performance of the PRESTO training data (Table 5.5). This analysis provides some indication that datasets with negative similarity to the PRESTO training data may not be suitable for use with PRESTO to make surrogate modeling recommendations. Prescreening data with this similarity metric may provide some insight into PRESTO's potential accuracy in selecting surrogates for a data set.

5.8 Conclusions and Future Work

Selecting an appropriate surrogate modeling technique depends on the characteristics of the dataset being modeled and the application domain of the surrogate model, surface approximation vs. optimization. We identified attributes of datasets appropriate for selecting surrogate models for both surface approximation and surrogate-based optimization. Using these attributes, a recommendation tool, PRESTO, was constructed to recommend surrogate modeling techniques for approximating a dataset with 91% accuracy and 90% precision and for performing surrogate based-optimization with 98% accuracy and 99% precision. Although PRESTO could not capture the full set of models that could be recommended on a set of test data generated from a cumene production process simulation, the recommended models did provide higher values of adjusted R-squared and better predictions for the outputs. Future work on PRESTO will include adding more real datasets to the training data for the tool, focusing on using a wider variety of sampling methods, not just space-filling ones, and incorporating the impact of noisy data.

Chapter 6 – Surrogate-Based Optimization Using Random Forests

Random forests (RFs) are surrogate models that make output predictions based on inputs by combining predictions from a collection of decision trees. Surrogate models are used to approximate the relationship between input and output data when the actual one between the two is unknown or computationally expensive to evaluate (C. Wang et al., 2014). These models can also be used in surrogate-based optimization approaches to approximate the objective function and/or constraints when they are not available in closed, analytical form or are not conducive for use in traditional gradient-based optimization methods, for example, if gradient information is not available. Each tree in a RF model is constructed independently and depends on a random vector sampled from the input data. (Breiman, 2001).

Random forests have successfully been used for regression and classification tasks, performing with high prediction accuracy for both small sample sizes and high dimensional data (Biau & Scornet, 2016). They can fit nonlinear data with a minimal number of parameters to tune (Biau & Scornet, 2016). The models have been used in several recent applications in the manufacturing industry, including for fault detection (Puggini et al., 2015; Quiroz et al., 2018; Zhang et al., 2018), prediction of mechanical failures (Wu et al., 2017), and prediction of manufacturing product properties (Maudes et al., 2017). Other areas of research where RF models have been employed for approximation include the development of new pharmaceutical molecules (Svetnik et al., 2004) and thermodynamic property estimation (Palmer et al., 2007).

Because of their prediction accuracy for a wide array of applications, RFs could represent an exceptional candidate for a surrogate model to approximate the objective function for surrogate-based optimization approaches. The RF decision tree structure allows it to be formulated as a mixed-integer linear program (MILP), which can be readily solved using existing commercial

solvers. However, fully trained models can result in large-scale MILPs that may become computationally intractable. Biggs and Hariss (2018) propose a method for optimizing RF objective functions by using Benders' decomposition (Benders, 1962) on only a subset of the decision trees in the RF. While they can successfully optimize RF objective functions with their approach, employing their solution method requires a significant amount of input on expertise on the part of the user, as do other developed techniques for optimizing decision-tree based models (Bertsimas & Dunn, 2017; Robertson et al., 2017). This drawback could represent an obstacle to using these models for optimization in practical applications.

To address this obstacle, this chapter introduces a MILP formulation corresponding to a RF model of an objective function and describes a Python library that can be used to construct this MILP formulation and solve it automatically. This code is available in the Cremaschi Research Group Github repository (https://github.com/CremaschiLab/Random-Forest-Optimization). The chapter is organized as follows. Section 6.1 presents the general structure of a RF model and the MILP formulation. Section 6.2 describes computational experiments carried out to test the performance of the RF objective function approximation. Section 6.3 presents how the RF objective function approximation performs for locating the global minimum of several test functions and the results for several solution approaches designed to reduce the solution time for the RF model MILPs. Conclusions are drawn in Section 6.4.

6.1 Random Forest Structure and MILP Formulation

Each RF model comprises several trees with a series of decision nodes. The decision node consists of a parent, or test, node, with a left child node and a right child node. If the indicated input value is less than or equal to a threshold value for the given test node, then the left child node is selected; otherwise, the right child node is selected. Decisions are made at each branch in a tree

until a final node, or leaf node, is reached. An example of a simple decision tree structure is illustrated in Figure 6.1. The output value for a tree for given inputs is the value of the final leaf node reached, and the output value for the entire RF model is the average value of the outputs for every decision tree in the forest.

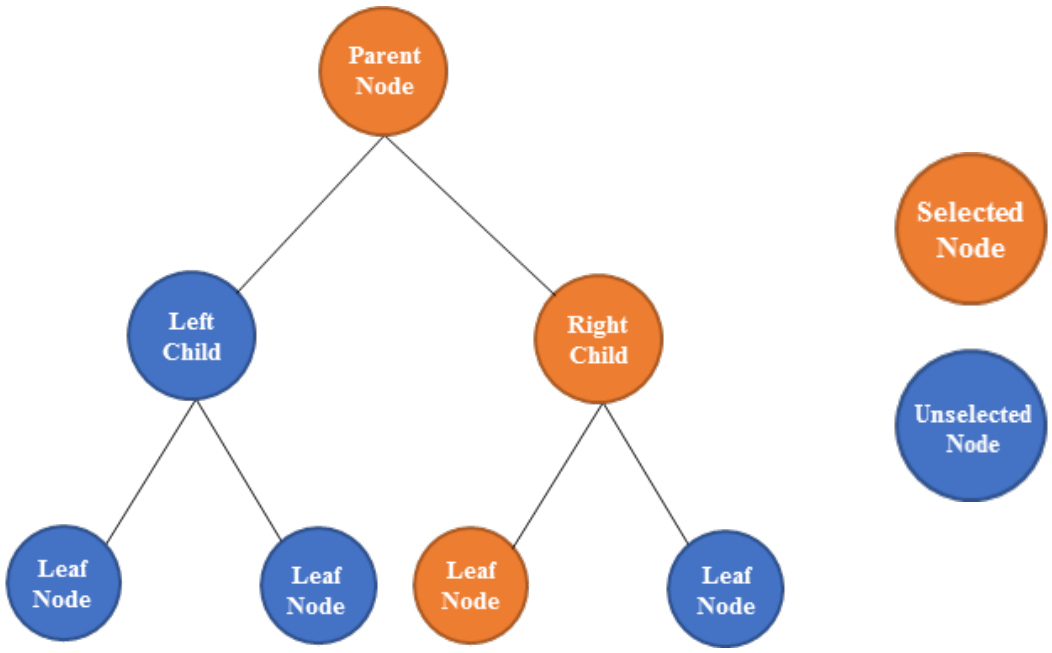


Figure 6.1 – Random forest decision tree structure

Figure 6.2 illustrates a simplified example of a RF model constructed to approximate the function

$$z = x_1 + x_2 \tag{6.1}$$

where x_1 and x_2 are input values restricted to the range [0, 5]. For the example shown, x_1 = 2 and $x_2 = 4.5$. The output for the RF model will be equal to the average value of the selected leaf nodes: $c_{1,7}$, $c_{2,5}$, and $c_{3,5}$. The RF estimates the value of z at x_1 = 2 and $x_2 = 4.5$ to be equal to 6.11, which is a close approximation of its actual value of 6.5.

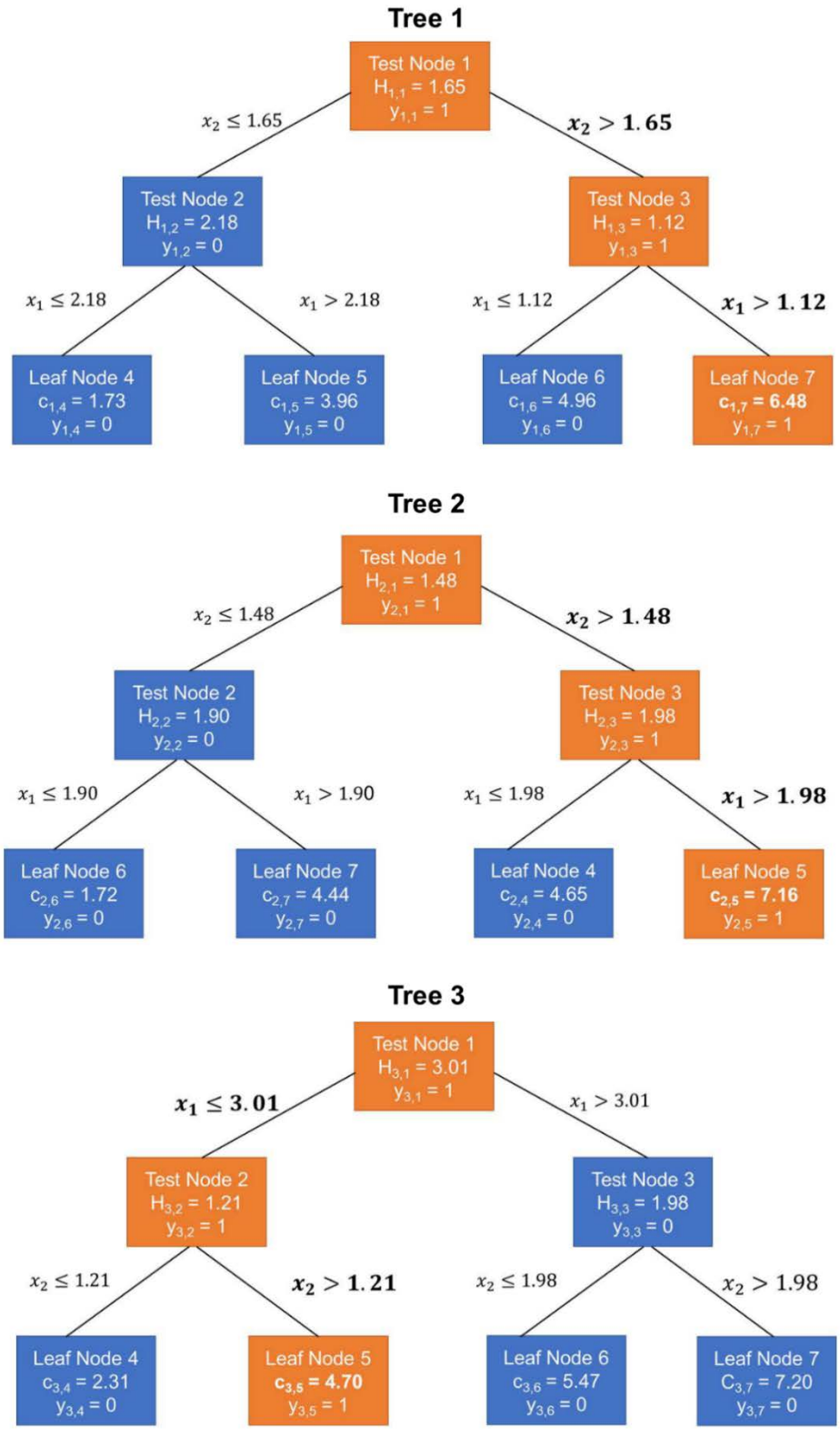


Figure 6.2 - Random forest model approximation of the function, $z = x_1 + x_2$. Orange boxes indicate test or leaf nodes that are selected (x_1 = 2 and $x_2 = 4.5$).

The optimization problem for the RF objective function is formulated as follows:

$$\min_{x} \frac{1}{N_{trees}} \sum_{t}^{T} \sum_{l}^{L} c_{t,l} y_{t,l} \tag{6.2}$$

s.t. $y_{t,1} = 1 \qquad \forall\, t \in T \tag{6.3}$

$$y_{t,lc} + y_{t,rc} = y_{t,p} \qquad \forall\, (t,p,lc,rc,i) \in P \tag{6.4}$$

$$x_i \geq H_p - M\left(1 - y_{t,rc}\right) - M\left(1 - y_{t,p}\right) + N y_{t,rc} \qquad \forall\, (t,p,lc,rc,i) \in P \tag{6.5}$$

$$x_i \leq H_P + M y_{t,rc} + M\left(1 - y_{t,p}\right) \qquad \forall\, (t,p,lc,rc,i) \in P \tag{6.6}$$

$$x_i \in [x_{min}, x_{max}] \tag{6.7}$$

The objective function, Eq. (6.2), of the formulated optimization problem is the model's output, which is the average value of the final leaf nodes selected by the model. In the formulation, $c_{t,l}$ is the value of a leaf node *l* in tree *t*, $y_{t,l}$ is a binary decision variable with a value of 1 if leaf node l is selected and 0 otherwise. H_p is the threshold value for parent node p, M and N are constants specific to the dataset, and each x_i is a continuous decision variable within a range specific to the dataset representing an input to the RF model. The index i for x_i indicates the input dimension. For example, x_1 is the input value for the first input to the model. T is the set of trees in the RF, and *L* is the set of every leaf node in the RF. *P* is the set of all groupings for *tree*, *parent node*, *left child*, *right child*, *input value* combination for every tree. Sets *T*, *L*, and *P* are constructed using the decision tree structure of each tree in the RF model.

Equation (6.3) forces the first node in every tree to be selected, ensuring that each tree in the model contributes to the RF model output and objective function value. Equation (6.4) enforces a constraint on the number of child nodes that can be selected for a parent node, with only one

child being chosen if the parent node is selected and neither being selected if the parent node is not. Equations (6.5) and (6.6) represent constraints that determine whether a left child node or right child node is selected based on the threshold value for their respective parent node. The value of M in each of these constraints ensures that the threshold constraint will not be enforced if the parent node for that threshold is not selected. A recommended M value would be to set it equal to the range of x values ($x_{max} - x_{min}$) the RF model is defined over.

6.2 Computational Experiments

Using RF models in surrogate-based optimization requires training an RF model to approximate a function. Based on the final structure of the model, an optimization problem is formulated with the output of the RF model as the objective function. This optimization problem is outlined in Section 6.1.

6.2.1 Test Functions

To evaluate the performance of the RF surrogate models for locating an optimum, datasets were generated from a set of test functions from an optimization test suite described in Section 4.2 (Surjanovic & Bingham, 2013). Functions with input dimensions of two, four, six, eight, and ten were used in evaluations.

One thousand input-output pairs were generated for each test function using Sobol sequence sampling (Joe & Kuo, 2008) to sample input values and obtain the function outputs for the given inputs, resulting in 99 total datasets. A RF model was trained for each dataset employing the generated pairs using the Sci-kit learn RandomForestRegressor implementation (Pedregosa et al., 2011). A densely sampled test set of 100,000 data points for each test function was generated using Sobol sequence sampling to analyze how well the RF models approximated the test

functions. The output predictions for the 100,000 test points by the trained models were compared to the actual outputs, and the normalized root mean square error (nRMSE), Eq. (6.8), for the model was calculated to quantify the performance. The nRMSE value for each dataset-surrogate model combination is normalized by the range of output values for easier comparison across datasets with various ranges for output values.

$$nRMSE = \sqrt{\frac{\sum_{n=1}^{N}(\hat{z}_n - z_n)^2}{N}} \Big/ (z_{max} - z_{min}) \tag{6.8}$$

In Eq. (6.8), z_n is the output for point n for a dataset, $\hat{z}_n$ is the output predicted by the RF model for point n, N is the total number of sample points in the dataset, and z_{max} and z_{min} are the maximum and minimum output values in a dataset, respectively.

6.2.2 Surrogate-Based Optimization with Random Forest Models

The global minimum of the underlying function used to produce each dataset was estimated using the trained RF models by solving the MILP model given in Section 6.1. The mathematical programs were constructed in Pyomo (version 5.6) (Hart et al., 2017; Hart et al., 2011), a Python-based optimization modeling language. The estimated minima were compared to the actual global minima for accuracy using a performance metric, D_{opt} (Eq. 6.9), we define as:

$$D_{opt} = \frac{D_M(x_{opt}, x_{opt'})}{\max\limits_{i,j} D_M(x_i, x_j)} \tag{6.9}$$

where $D_M(x_{opt}, x_{opt'})$ is the Mahalanobis distance (De Maesschalck et al., 2000) between the location of the actual global minimum of the function, x_{opt}, and the location estimated using the surrogate-based optimization, $x_{opt'}$. This value is normalized by the maximum Mahalanobis

distance between any two points (x_i, x_j) in the dataset. Computations for solving the MILPs were carried out with CPLEX on the Auburn University HPC Cluster (Lenovo System X HPC Cluster) using Intel E5-2650 V3, 2.3 GHz 20 core processors and implemented in Python 3.6.

6.3 Results and Discussion

6.3.1 Effect of Random Forest Model Size on Surface Approximation Performance

Each RF model has two tunable hyperparameters, the depth of each tree and the number of trees in the forest. To investigate how the RF prediction performance changed with the tree depth, we constructed RF models for each generated dataset with a maximum of 50, 100, 200, 300, 400, 600, 700, and 800 leaf nodes allowed in the model. Each of these models contained 100 trees. For investigating the effect of the number of trees, models with trees ranging from 1 to 100 trees were trained for each test dataset. The leaf nodes in these models were allowed to expand until all the test nodes in each tree were determined to be "pure." Purity is determined during the RF regression model training according to the mean squared error (Pedregosa et al., 2011). The RF training algorithm attempts to minimize the impurity at each decision node in building the model. For the models with "pure" test nodes, the mean squared error at each decision node is minimized.

Example results for how well the RF models approximated the test functions are summarized in Figs. 6.3 and 6.4. The nRMSE was calculated for each trained model to predict the densely sampled 100,000 point test set and the training data set. In addition, 50-fold cross-validation was used to calculate an average validation nRMSE, as well as a 95% confidence interval on that average. Figures 6.3 and 6.4 illustrate trends observed among the test functions for how the nRMSE changes with an increase in the number of leaf nodes and trees. Behavior for both increasing the number of leaf nodes and the number of trees in the RF models exhibit similar trends. The nRMSE for the training data set is always the lowest value for both increasing numbers

of leaf nodes and trees. When increasing the number of leaf nodes in the model, the nRMSE error for the dense sample set falls either within (Fig 6.3a) or below (Fig. 6.3b) the 95% confidence interval for the validation error. However, the nRMSE for the dense sample set with increases in the number of trees at a fixed number of leaf nodes has three different trends: it is either above the 95% confidence interval for the validation error (Fig. 6.4a), within the interval (Fig. 6.4b), or below it (Fig. 6.4c).

The test functions fall into one of these three categories of behavior for surface approximation performance. All of the cases follow a general trend of a sharp decrease in the error with increasing model size at low numbers of trees (or leaf nodes), followed by an eventual leveling off in the error value after a certain threshold number of trees (or leaf nodes) is reached. This threshold value is different for each test function. These results for the effect of the model size on the approximation performance indicate that allowing the RF models to expand to "pure" leaf nodes will not significantly increase overfitting. This assessment is valid, especially for models with higher numbers of trees as the nRMSE for the training data, densely sampled data, and validation data remain relatively evenly separated from each other for the same test function as the numbers of leaf nodes increase.

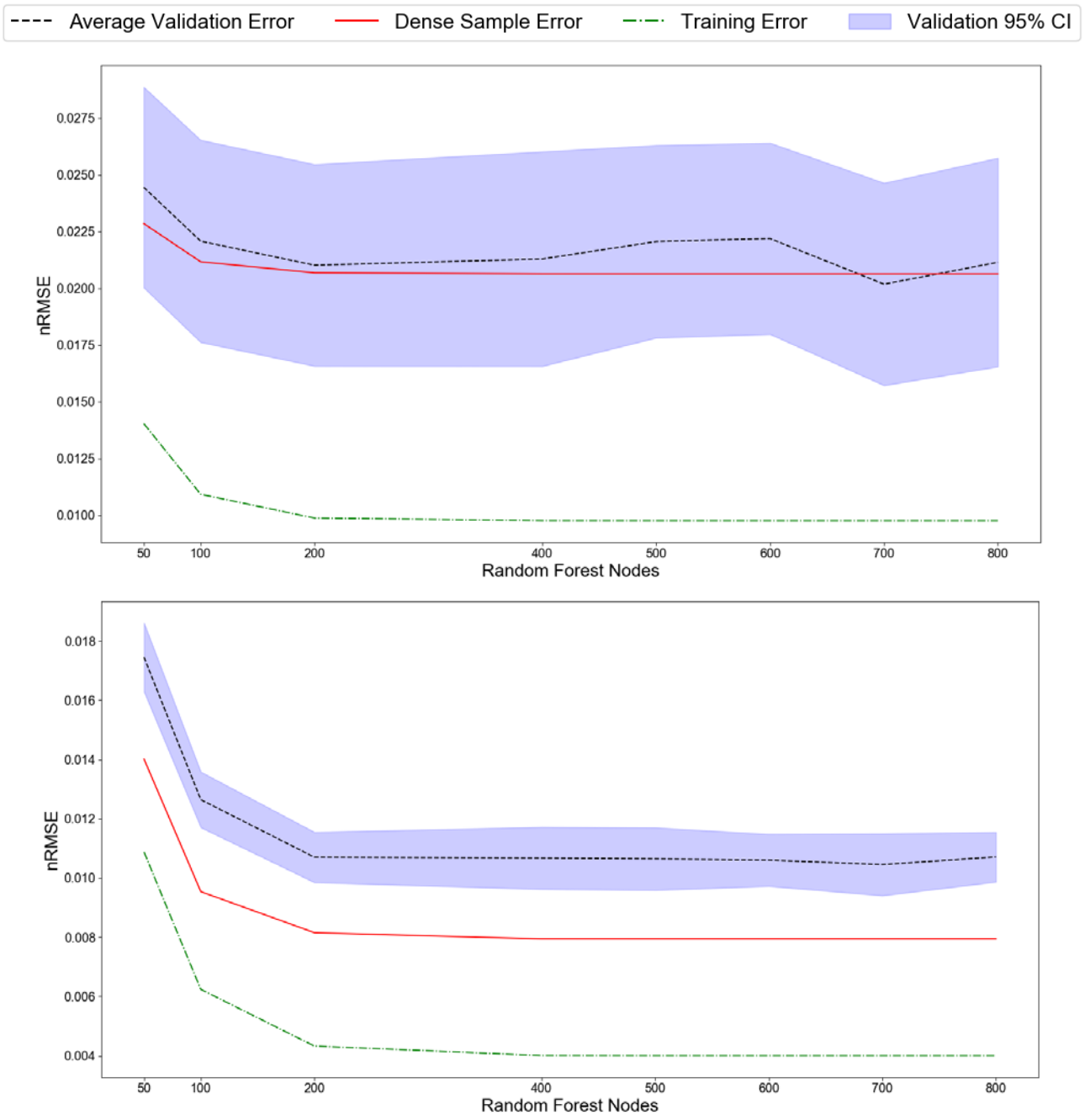


Figure 6.3 - Effect of increasing leaf nodes on random forest surface approximation performance

(CI = confidence interval)

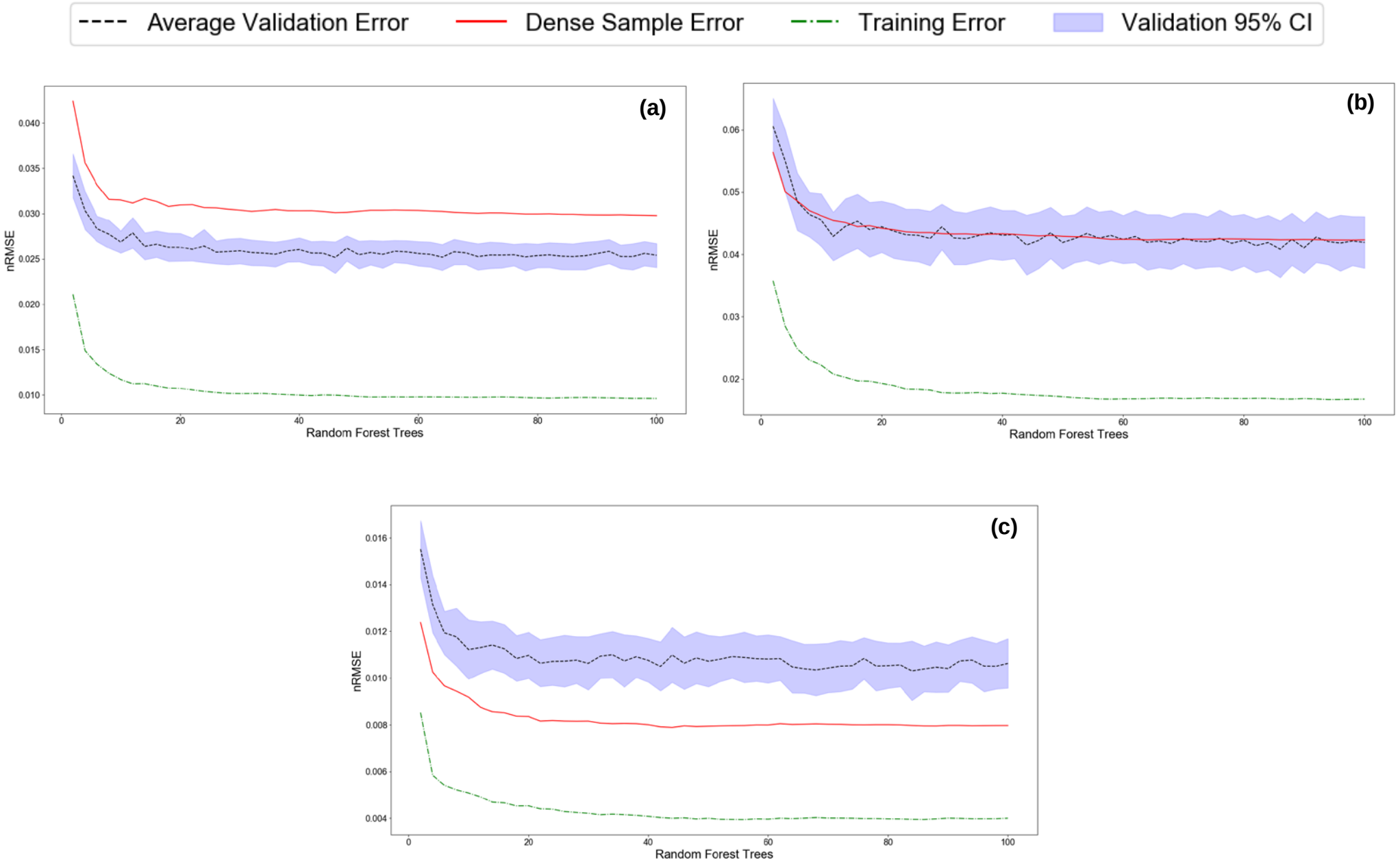


Figure 6.4 - Effect of increasing tree size on random forest surface approximation performance (CI = confidence interval)

6.3.2 Effect of Random Forest Model Size on Surrogate-Based Optimization Performance

Results for the effects of increasing the maximum numbers of leaf nodes and trees on locating the optimum points on average for all test functions are summarized in Fig. 6.5. The surrogate-based optimization performance was assessed by calculating the normalized Mahalanobis distance from the actual minimum point of each test function to the one predicted by the trained RF models, D_{opt}. The average D_{opt} values in Fig. 6.5 are separated by the number of input dimensions of the test functions.

In general, the average D_{opt} decreases with increasing number of leaf nodes for all input dimensions investigated (Fig. 6.5a), suggesting that the predicted optimum locations move closer to the actual ones with increases in the number of leaf nodes allowed in the models. Similarly, when leaf nodes are allowed to expand until they are “pure” and the number of trees is increased, the average D_{opt} decreases, particularly at higher input dimensions (Fig. 6.5b). While the average D_{opt} value reaches a minimum value after 600 maximum leaf nodes for each of the input dimensions tested, the average D_{opt} does not level off to a minimum value after the addition of 100 trees for the higher input dimensions (Fig. 6.5b.) This result suggests that lower values of D_{opt}, and thus, locations closer to the actual minimum locations of the test functions can be achieved by allowing the trees to expand to a maximum number of leaf nodes that produces "pure" trees with an increased number of trees in the forest (above 100). However, the resulting optimization problems may become too computationally expensive.

Figure 6.6 shows three trends for how the estimated optimum locations were affected by an increasing number of trees when the forests were allowed to expand until all leaf nodes were “pure”. The majority of the test functions had an overall decrease in the D_{opt} as the number of

trees in the RF model increased, eventually settling to a minimum D_{opt} after a certain number of trees is reached (Fig. 6.6a).

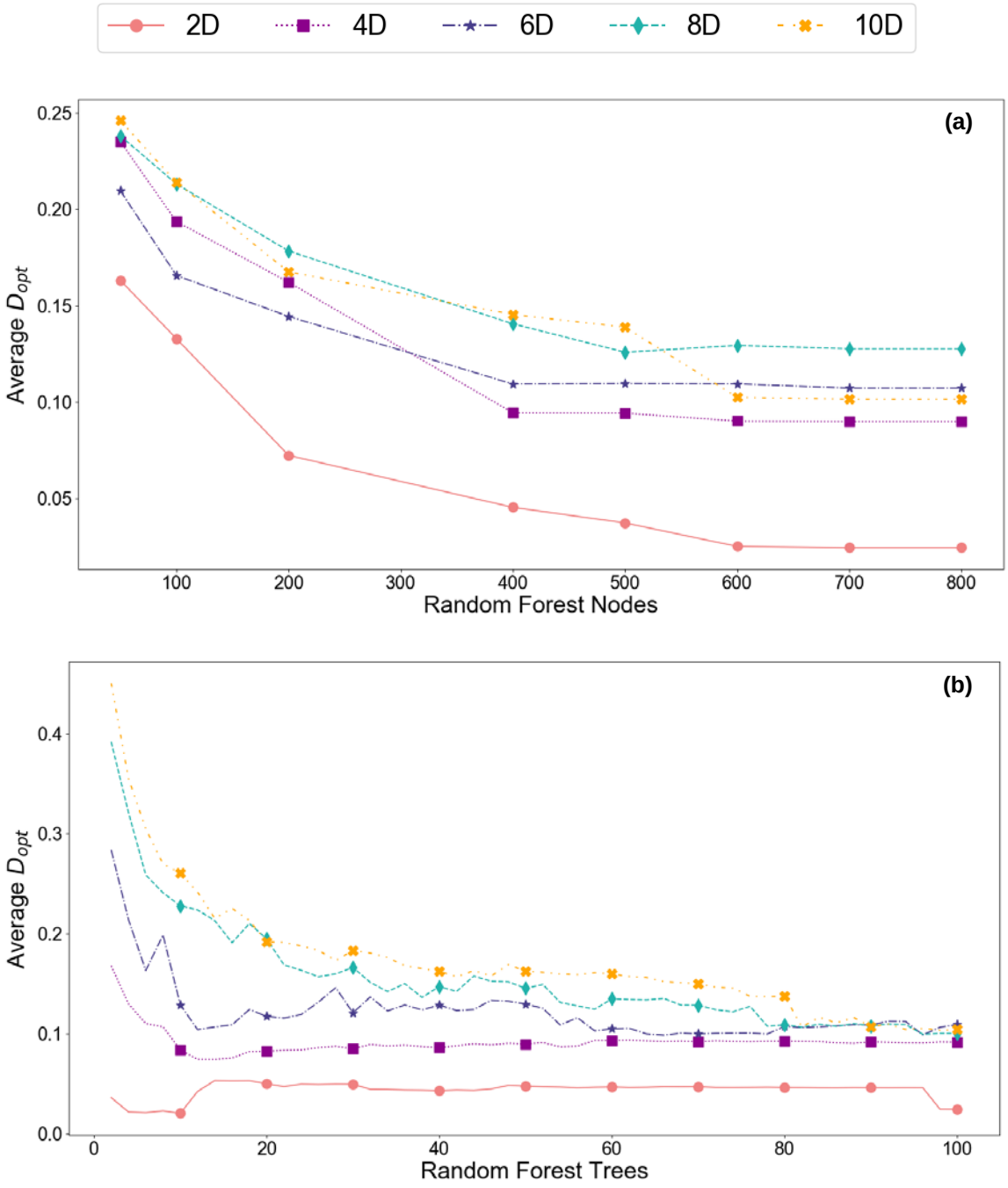


Figure 6.5 - Average value of D_{opt} for all 99 test functions vs (a) maximum number of leaf nodes and (b) number of trees in random forest model

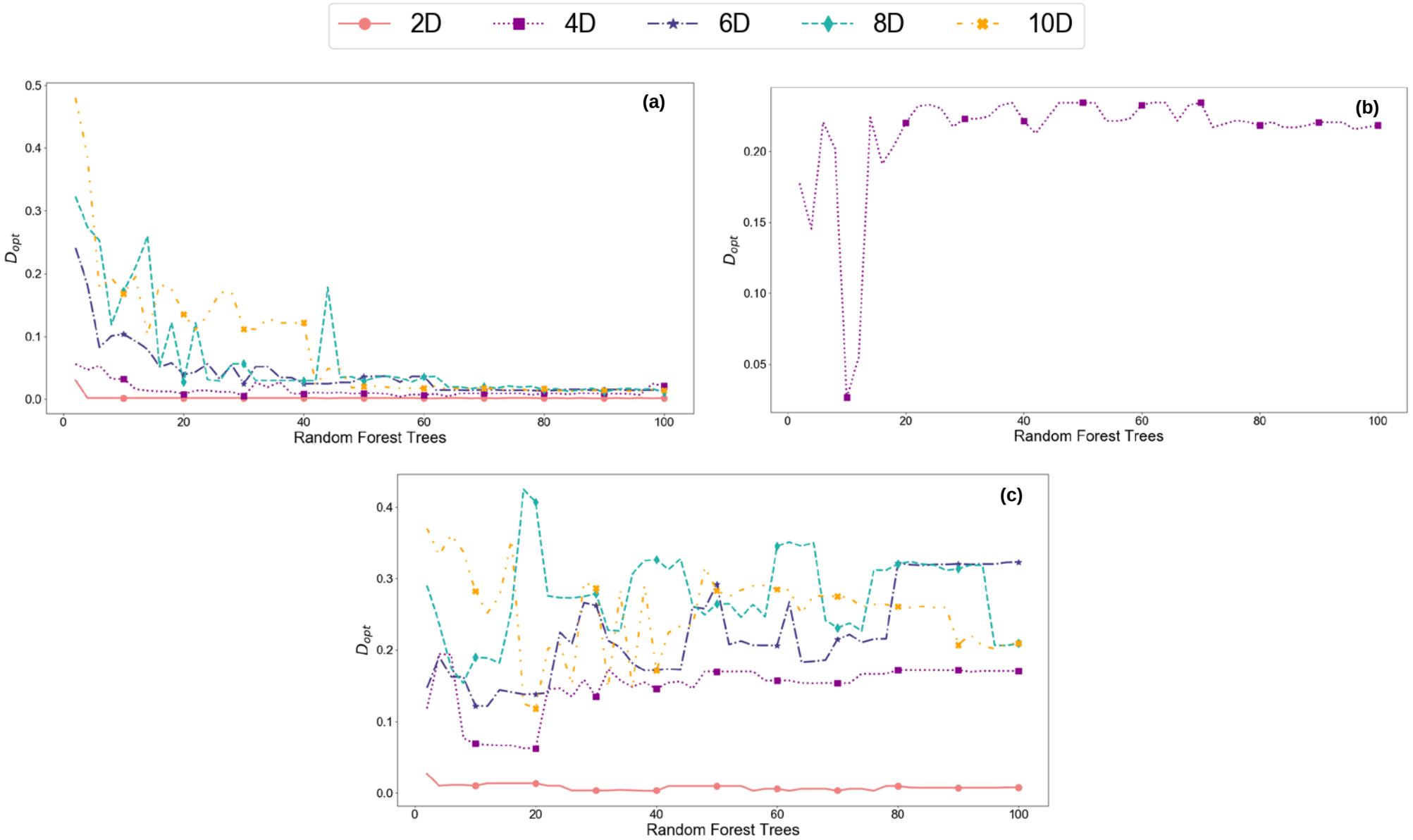


Figure 6.6 - D_{opt} vs. number of random forest trees for (a) Ellipsoid function, (b) Power Sum function, and (c) Zakharov function

Three of the test functions reached a minimum D_{opt} value at a certain number of trees and then D_{opt} began increasing once that number of trees was surpassed (one such function was Power Sum function, whose plot is in Fig. 6.6b). For some of the test functions, the D_{opt} did not decrease even when the RF model included 100 trees, and the RF failed to accurately capture the optimum of the functions (Fig. 6.6c). However, allowing more trees in the model could lead to more accurate solutions at the risk of the resulting optimization problem becoming intractable. The behavior of the majority of the test functions with increasing the number of trees in the RF model suggests that, in general, a higher number of trees results in lower D_{opt}, and thus, a closer, more accurate estimate of the optimum location, albeit at a higher computational expense.

6.3.3 Computational Efficiency of Solving the Random Forest MILP

The average computational time required for solving the RF MILP to estimate the global minima of the test functions for the number of input dimensions investigated is plotted in Fig. 6.7 as a function of the number of trees in the forest. The average sizes of the MILPs for the test functions are given in Table 6.1. Although the size of the problem (the number of constraints and variables) increases linearly with the number of trees (Table 6.1), the solution time begins to increase exponentially beyond approximately 40 trees (Fig. 6.7). This increase in solution time is distinctly apparent at higher input dimensions.

Table 6.1 - Average Size of Random Forest MILP

	Average			
Trees	Solution Time (sec)	Constraints	Binary Variables	Total Variables
2	1.0	3778	2519	2525
4	1.6	7635	5091	5097
8	5.5	15286	10192	10199
16	9.0	30404	20274	20280
32	35	60650	40443	40450
64	447	121166	80798	80804
100	10967	198226	126183	126189

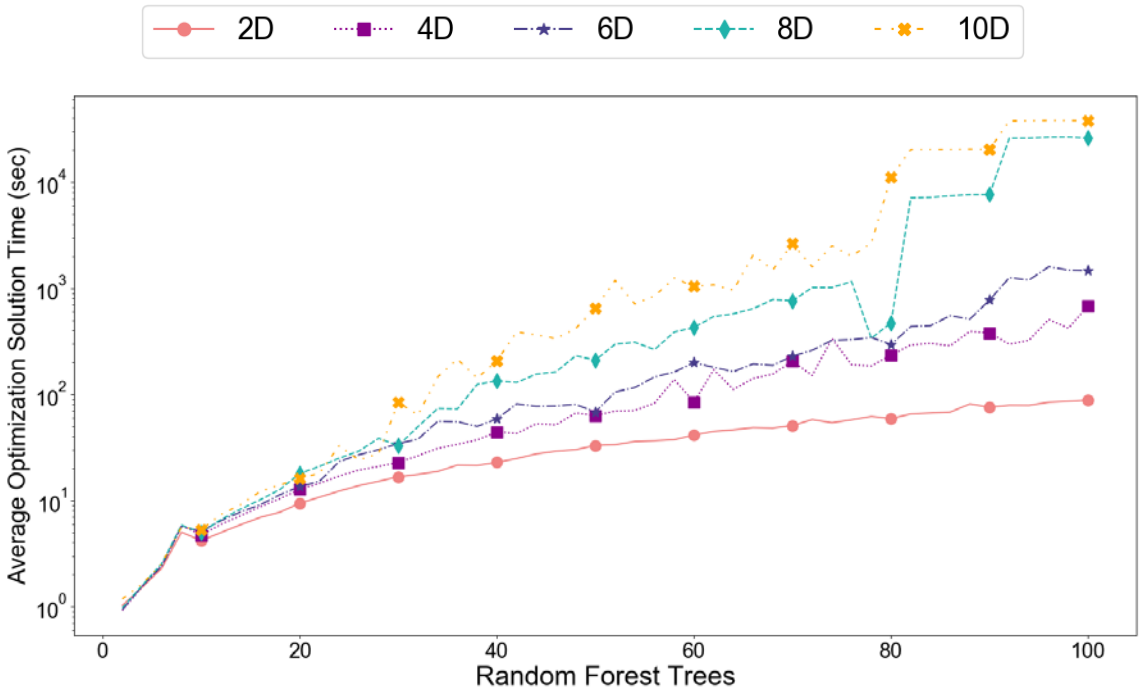


Figure 6.7 - Average time required for the solution of RF MILPs as a function of the number of trees in the random forest model

6.3.4 Performance Profiles for Surrogate-Based Optimization with Random Forests

To determine if a higher number of trees in the model results in overall better estimates of the optimum location, the fraction of the test functions for which the optimum was located is compared for RF models trained with 50 trees, 100 trees, and models where the number of trees was determined using cross-validation. The number of trees was increased from 40 to 100 until the RMSE of a validation dataset stopped improving during cross-validation. For these models, the validation error was estimated using ten-fold cross-validation on the training set. The number of trees was increased until the average value of the last five validation errors either began to increase or changed by less than 1%. This cross-validation prevented the RF models, and hence, the resulting MILPs, from becoming unnecessarily large, with a maximum solution time for the models of about two hours of computational time. We define a model as having located the optimum when it obtains a D_{opt} value less than a threshold. Results are presented in Fig. 6.8.

Both the models trained with 50 and 100 trees located the majority of the optima for the test functions at low input dimension, i.e., number of decision variables, within D_{opt} values of 0.05 (Fig. 6.8a) and 0.01 (Fig. 6.8b). At higher input dimensions, the cross validation-trained RF models and the RF models trained with 100 trees perform comparably for the 5% threshold (Fig. 6.8a), locating about the same fraction of the optima for the test functions. However, when the threshold for locating the optimum is lowered to 1%, the RF models trained with 100 trees exhibit superior performance to the other models at higher input dimensions, as they locate the optimum for a much higher fraction of the test functions (Fig 6.8b). These results indicate that more trees may be required for RF models if a closer, or more accurate, estimate of the location of the optimum point is required, especially at higher input dimensions. The exponential increase in solution time at higher numbers of RF model trees presents a barrier to using larger RF models for

optimization purposes. An investigation of decomposition approaches to exploit the unique MILP structure of RF models to reduce solution times is explored and described in detail in Zeng (2020).

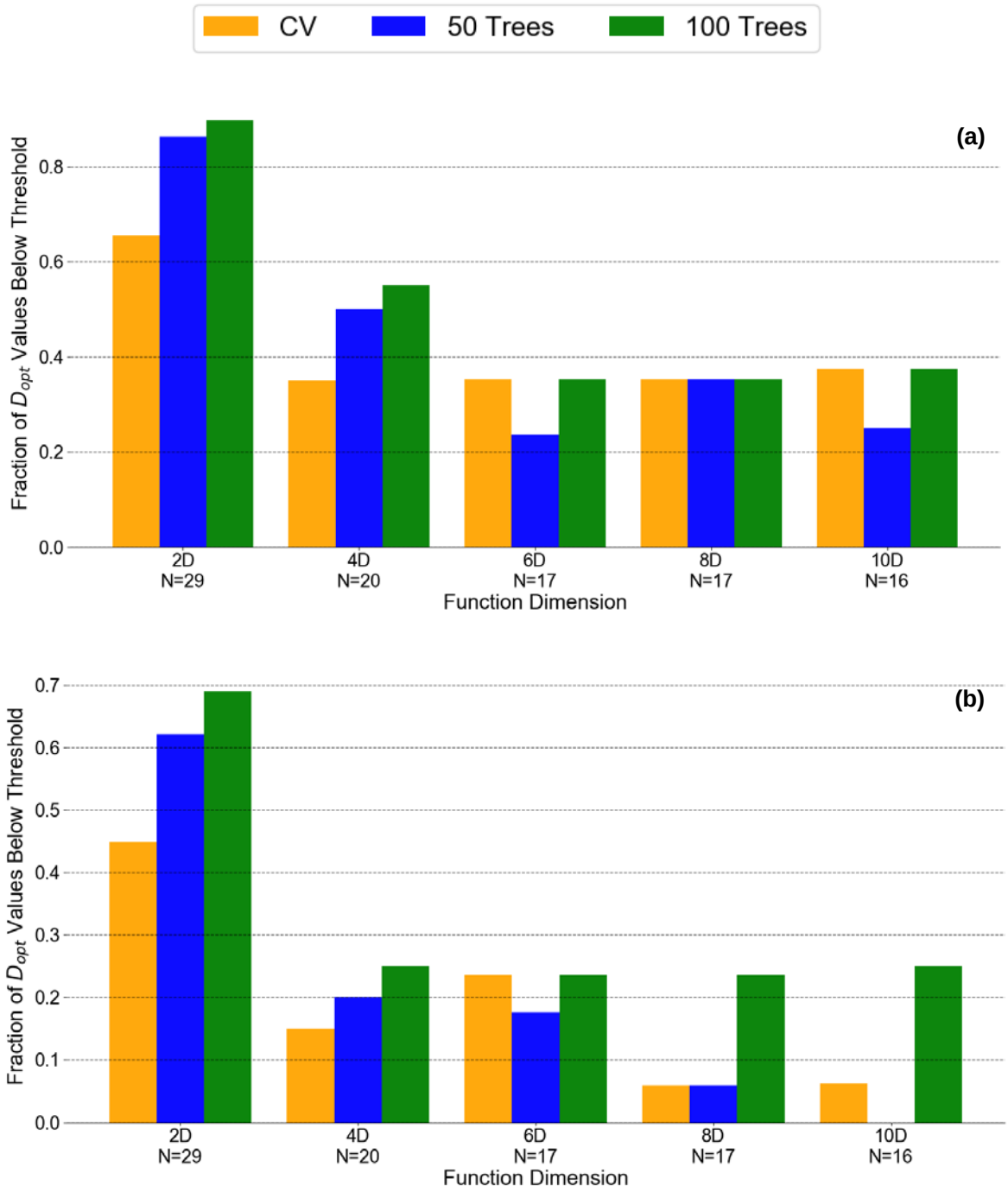


Figure 6.8 - Fraction of datasets with D_{opt} less than **(a) 5%** and **(b) 1%** grouped by input dimension for RF models trained with cross validation, 50 trees, and 100 trees. N values below

the function dimensions indicate the number of test functions used for that input dimension (CV = cross validation)

6.4. Conclusions and Future Work

We have developed a method for automatically generating and solving an optimization problem using a RF model to approximate the objective function. This method can be used for surrogate-based optimization of complex models using only input-output data from those models. While the resulting MILPs provide accurate estimates for the location of the optimum points for a large proportion of the test functions investigated, the large solution times required provide a significant obstacle for solving the necessary models to achieve in an accurate location in some cases. Decomposition solution approaches are being investigated in order to decrease the solution time required for larger RF models.

Chapter 7 – Derivative Free Optimization with pyBOUND (PYthon-based Black box Optimization Using raNDom forests)

Optimization is required for several chemical engineering applications, including process design and synthesis, operations, and supply chain management. These applications usually involve complex, high-fidelity simulations and/or physical experiments, which can both require significant resources in terms of cost and time, as well as a large computational expense to collect data. Optimization using traditional gradient-based methods is impractical for these applications because gradient information is not readily available, and approximating gradients may be infeasible due to the required expense for multiple simulation evaluations or experiments. In addition, the direct use of deterministic global optimization methods is restrictive in these cases because the computational cost for obtaining data limits the total number of model runs necessary to optimize the system efficiently (Conn et al., 2009; Forrester et al., 2008).

To overcome these challenges, derivative-free optimization methods can be employed. Here, a derivative-free optimization (DFO) algorithm is developed using surrogate-based optimization, where a surrogate model can be constructed to represent an objective function that may be computationally expensive to evaluate or are unavailable in analytical form. The constructed surrogate can be used as a closed functional form in traditional gradient-based optimization methods.

7.1 Optimization Problem Formulation

Motivated by the recent progress on surrogate-based optimization of black-box problems, the proposed algorithm uses a surrogate-based approach to solving a constrained black-box

optimization problem. We consider black-box optimization problems of the form given in Eqs. (7.1) and (7.2),

$$\min_{x} \quad f(x) \tag{7.1}$$

$$\text{s.t.} \quad x_i \in [x_i^L, x_i^U] \qquad \forall\, i \in \{1,2, \dots \dots D\} \tag{7.2}$$

where the objective function, $f(x)$, represents a black-box simulation model. It is assumed that the black-box model is expensive to evaluate and/or its derivative information is not available, so traditional gradient-based optimization methods would not be practical for use. D represents the number of continuous input variables in the model with known finite bounds $[x_i^L, x_i^U]$.

7.2 General pyBOUND Framework

There has been considerable progress in developing approaches for derivative-free optimization by using surrogate models to approximate any explicitly unknown relationships in the systems of interest (Boukouvala et al., 2016; Rios & Sahinidis, 2013). The surrogate approximations aim to guide the search towards the optimum of the original model. However, these algorithms can be difficult to scale to problems with high dimensionality (Bhosekar & Ierapetritou, 2018b; Qian et al., 2016a). To address this dimensionality issue, we have developed pyBOUND (PYthon-based Black box Optimization Using raNDom forests), a two-tiered approach for surrogate-based optimization. The first stage aims to shrink the search space through a global search with random forests, and the second stage aims to locate the optimum via a local search within the reduced space using MARS models. The following sections provide information on the algorithmic steps of pyBOUND and results for comparing pyBOUND's performance on a set of test problems to the performance of several common DFO algorithms. A general framework for the pyBOUND algorithm is provided in Fig 7.1.

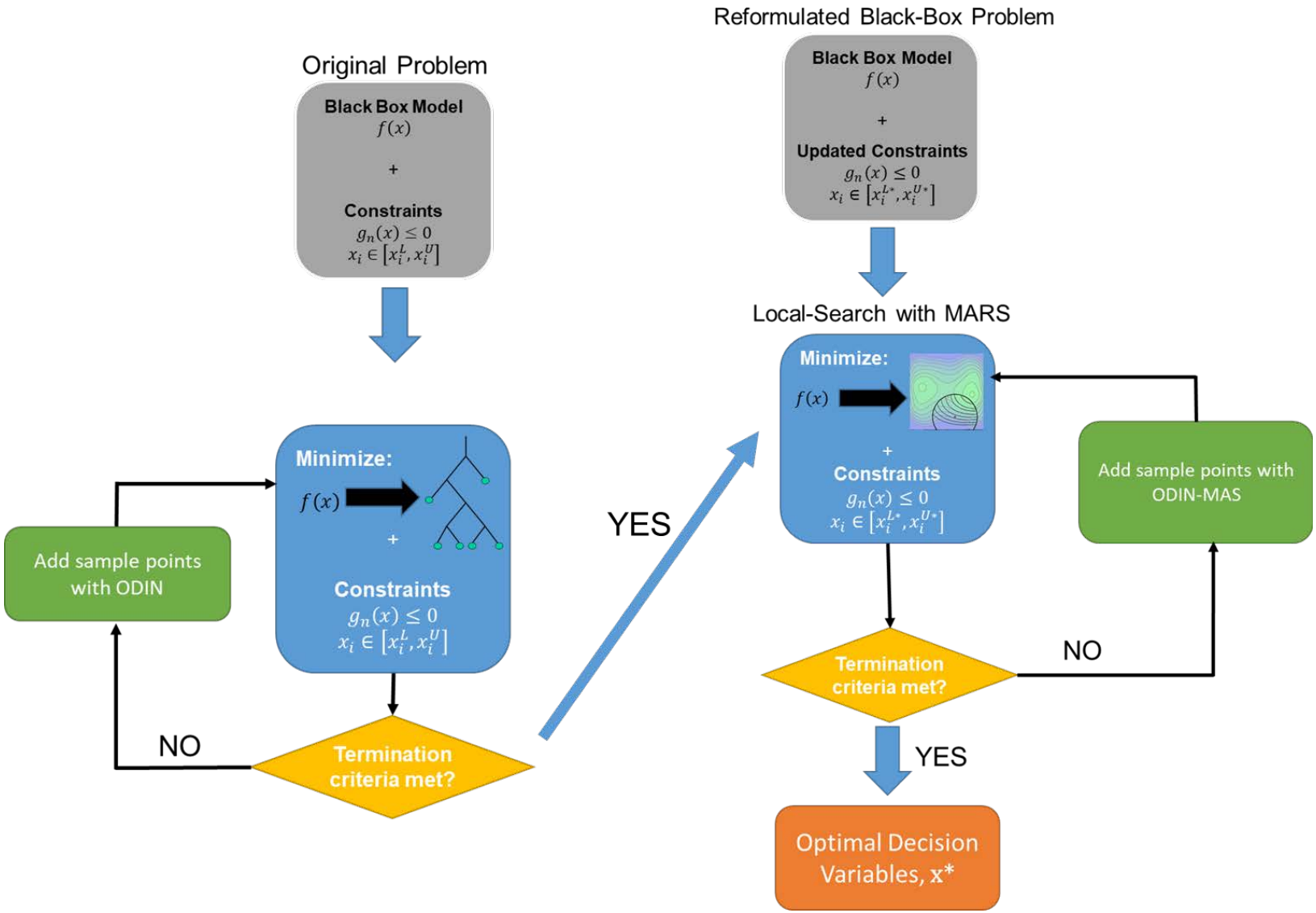


Figure 7.1 - General pyBOUND Framework

7.3 pyBOUND Stage 1: Generation of Decision Variable Bounds with Random Forest Models

In the first stage of pyBOUND, the algorithm reduces the size of the search space by solving a series of global deterministic subproblems using random forest (RF) models (Breiman, 2001). A single iteration of this stage consists of sampling, construction of a RF model surrogate approximation, global optimization of the constrained approximation problem that employs the RF model, and collection of new sampling points. The iterations repeat the entire procedure until certain termination criteria are met. Termination criteria for stage 1 include a maximum number of black box evaluations and a maximum amount of reduction of the original search space.

The initial step for the algorithm for both the global exploration and the trust region framework generates N sample points to construct the surrogate model. Initially, a set of samples is generated using a Sobol sequence design. The overall constrained approximation model, which consists of the RF model objective equation, the RF model constraints, and any original constraints and variable bounds, is solved to optimality using deterministic global optimization methods (Misener & Floudas, 2014; Sahinidis, 1996). We previously developed a method of automatically constructing a RF model and formulating its resulting optimization problem given a set of input-output values. A full discussion of the method and its results can be found in Chapter 6. If the model does not include any original nonlinear constraints, the constrained approximation models with the RF models are MILPs, which are solved with CPLEX (version 12.10.0), in this study.

7.3.1 Approaches for Reducing Decision Variable Bounds

RF models consist of a collection of decision trees, with the final output for the model being the average of the predicted output for each tree in the forest (Breiman, 2001). Bounds can be generated for the decision variables from the optimal variables found by solving the constrained approximation optimization problem using the thresholds given by the decision tree rules for every

tree in the forest. However, each tree in the RF model has its own decision rules and yields its own bounds. The algorithm must determine a single set of bounds based on the threshold bounds given by every RF tree. Three methods for reducing the decision variable bounds, or cutting methods, were investigated for the stage 1 bounds reduction: using the widest bounds set of bounds (Wide), including bounds where at least two trees intersect each other (Intersection), and averaging the bounds given by each tree for a single set of bounds (Average). Figure 7.2 provides an illustrative example of the three methods. The three colored rectangles in Fig. 7.2 represent bounds given by three separate trees in a RF model for two input dimensions (or decision variables). The new, reduced bounds determined by the cutting methods are highlighted in yellow.

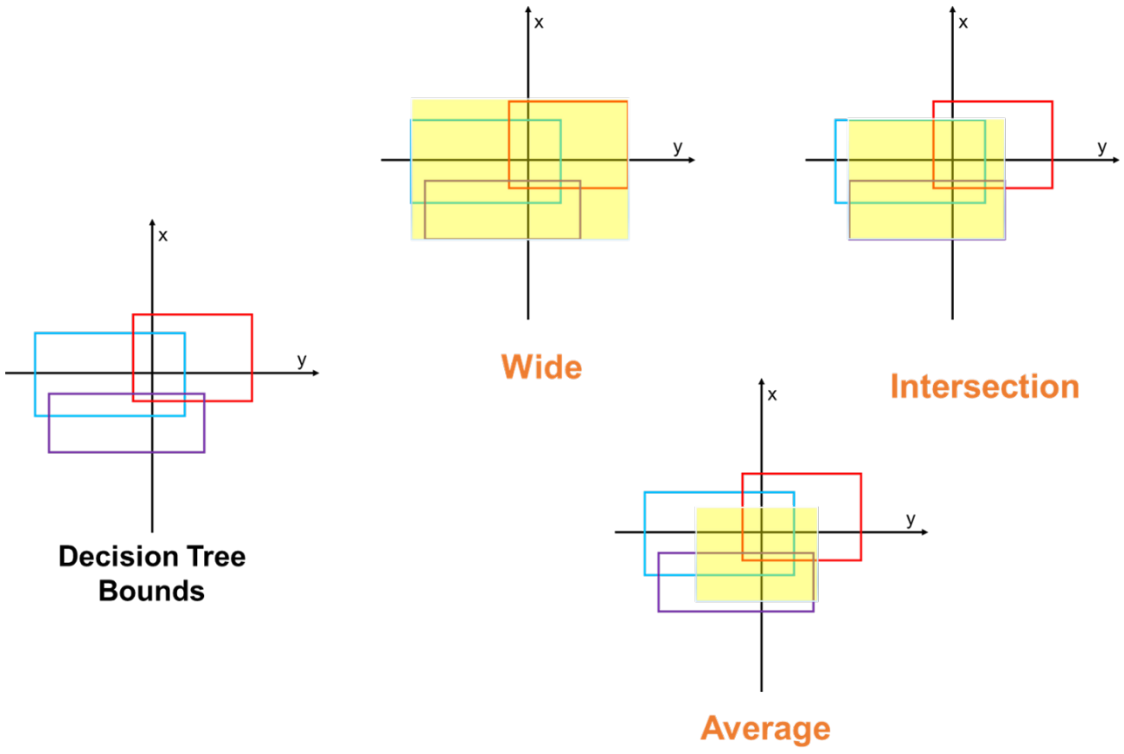


Figure 7.2– Decision variable bounds cutting methods

7.3.2 Adaptive Sampling Methods for Updating RF Model

Based on the solution of the constrained optimization problem that employs the RF surrogate model, the algorithm determines whether to continue by increasing the size of the sample set used to construct the surrogate model or proceed to second stage of pyBOUND with the bounds determined by the current solution. Four adaptive sampling methods for adding new samples points were considered at each iteration of stage 1. These methods include Mixed Adaptive Sampling (MAS), Optimization Directed INcremental (ODIN) sampling, ODIN-MAS, and the maximization of the expected improvement (EI) function. For each of these methods, a set of candidate sampling points is randomly generated. Then, a score designating the quality of each candidate point is calculated based on the objectives of the adaptive sampling method. The RF model training set is updated with the candidate points with the highest scores.

The Mixed Adaptive Sampling (MAS) algorithm was developed using a combination of space-filling and adaptive sampling methods (Eason & Cremaschi, 2014). The sample point score for MAS is based on maximizing the Euclidean distance between the candidate point and the nearest neighbor point in the existing sample set, as well as maximizing the estimated variance of the surrogate model at the candidate point. Optimization Directed INcremental (ODIN) sampling adds new sample points based on a space-filling criterion and by exploiting the regions around the previously identified best solution (Smith, 2015). The ODIN sample point score is based on maximizing the nearest neighbor distance to the existing sample set and minimizing the distance between the candidate sample point and the current best solution for the optimum location of the optimization problem. ODIN-MAS is a hybrid approach combining the MAS and ODIN sampling methods. The sample point score for ODIN-MAS is based on maximizing the nearest neighbor

distance and the estimated variance of the surrogate model at the candidate point while minimizing the distance to the current best solution (Smith, 2015).

The expected improvement (EI) function (Eq. 7.3) is considered to be a robust global optimizer (Kleijnen et al., 2012). The sample point score for the EI sampling method is based on maximizing the value of the EI function (Bhosekar & Ierapetritou, 2018a).

$$EI_n(x) = \Delta_n(x) + \sigma_n(x)\varphi\left(-\frac{|\Delta_n(x)|}{\sigma_n(x)}\right) - |\Delta_n(x)|\phi\left(-\frac{|\Delta_n(x)|}{\sigma_n(x)}\right) \tag{7.3}$$

In Eq. (7.3), φ and ϕ, are the normal cumulative distribution function and probability distribution function, respectively, $\sigma_n(x)$ is the model variance at the set of decision variables, x, and $\Delta_n(x)$ is the difference between the model estimate of $f(x)$ and the best value of $f(.)$ observed thus far (Jones et al., 1998).

7.3.3 Results for Bounds Cutting and Sampling Methods

Decisions were made for the structure of pyBOUND using the suite of 127 test functions described in Section 4.1. Results used for selecting the bounds cutting method, the adaptive sampling method, and the termination criteria for stage 1 of the algorithm are presented in Figs. 7.3, 7.4, and 7.5. Figures 7.3 and 7.4 show the fraction of the test functions for which each of the cutting methods (Wide, Intersection, Average) resulted in an infeasible constrained approximation model and for which each cutting method resulted in no reduction in the decision variable bounds at the end of the RF bounds reduction stage. Each of the four potential sampling methods was applied with all three cutting methods.

The “Average” cutting method resulted in the highest fraction of infeasible models (Fig. 7.3). The majority of these infeasibilities occurred for test functions with lower input dimensions (less than 10). Although the “Wide” cutting method resulted in infeasible runs for a small fraction

of the test functions with only one of the potential sampling methods, it had the highest fraction of test functions for which stage 1 was not able to provide any reduction to the search space (Fig. 7.4). For both the “Wide” and “Intersection” cutting methods, the majority of the functions that did not achieve any reduction in the search space were also at lower input dimensions. Based on these results, the “Intersection” cutting method was selected for optimization problems with less than 10 input dimensions, and the “Average” cutting method was selected and implemented for problems with 10 or more input dimensions.

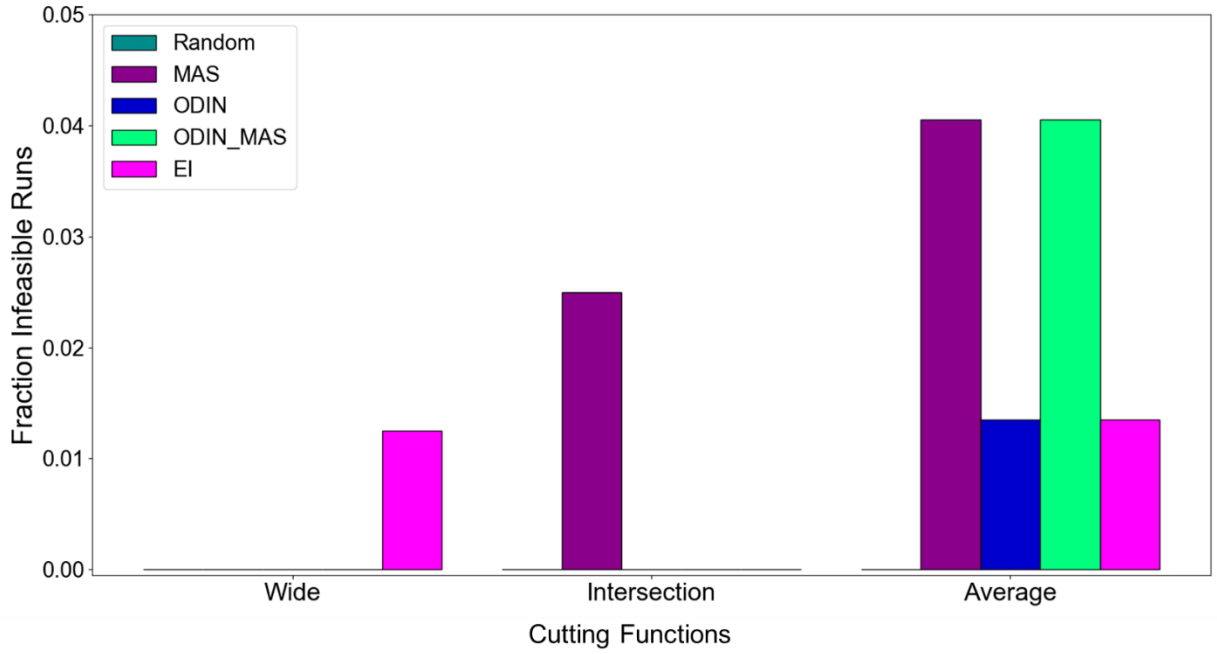


Figure 7.3 - Fraction of test functions with infeasible models for RF stage of pyBOUND

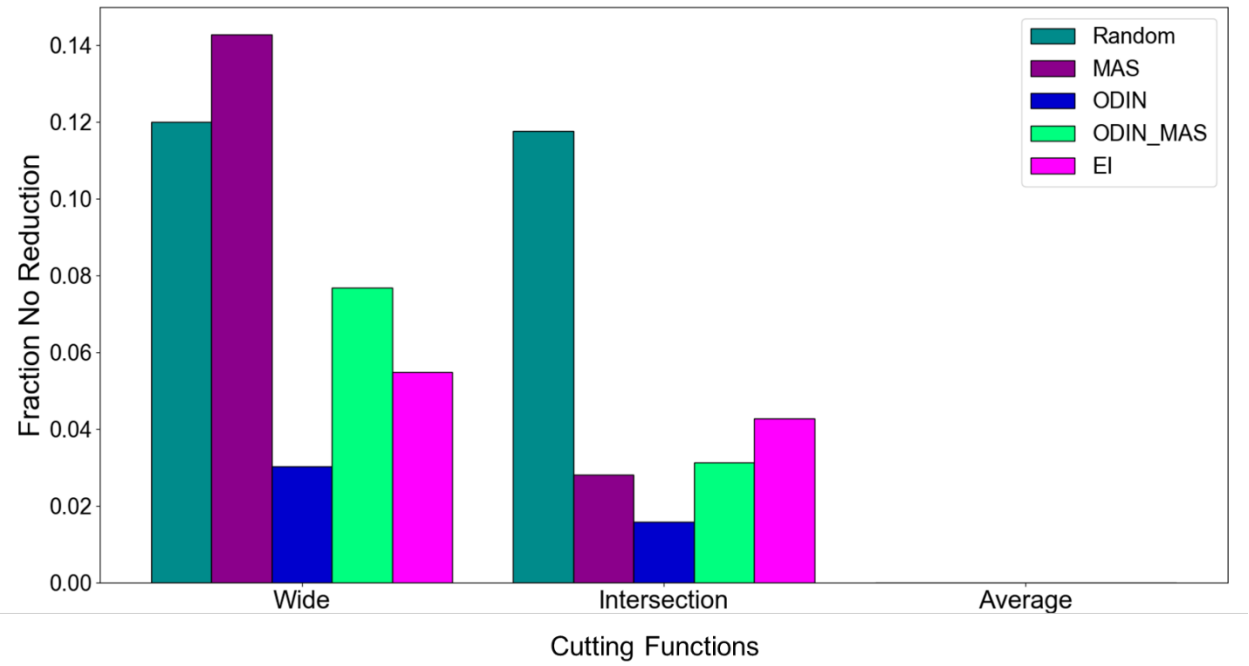


Figure 7.4 - Fraction of test functions with no reduction in decision variables bounds for RF stage of pyBOUND

Figure 7.5 depicts the fraction of the test functions for which the true optimum of the function was cut out of the reduced search space during the RF bounds reduction stage as a function of the fraction of the original search space that is reduced. These results were obtained using the "Wide" cutting method. In general, the ODIN sampling method resulted in the lowest fraction of the true optima being cut out of the search space with increasing search space reduction, with no cutting for any of the functions occurring until 80% of the original search space had been removed by the algorithm. Based on these results, the ODIN sampling method was selected for updating the RF model training set with a termination criterion of an input dimension dependent maximum amount of reduction in the search space. This termination criterion was applied to avoid removing the true solution of the optimization problem from the search space. The final configuration for the RF stage of the algorithm is summarized in Fig. 7.6.

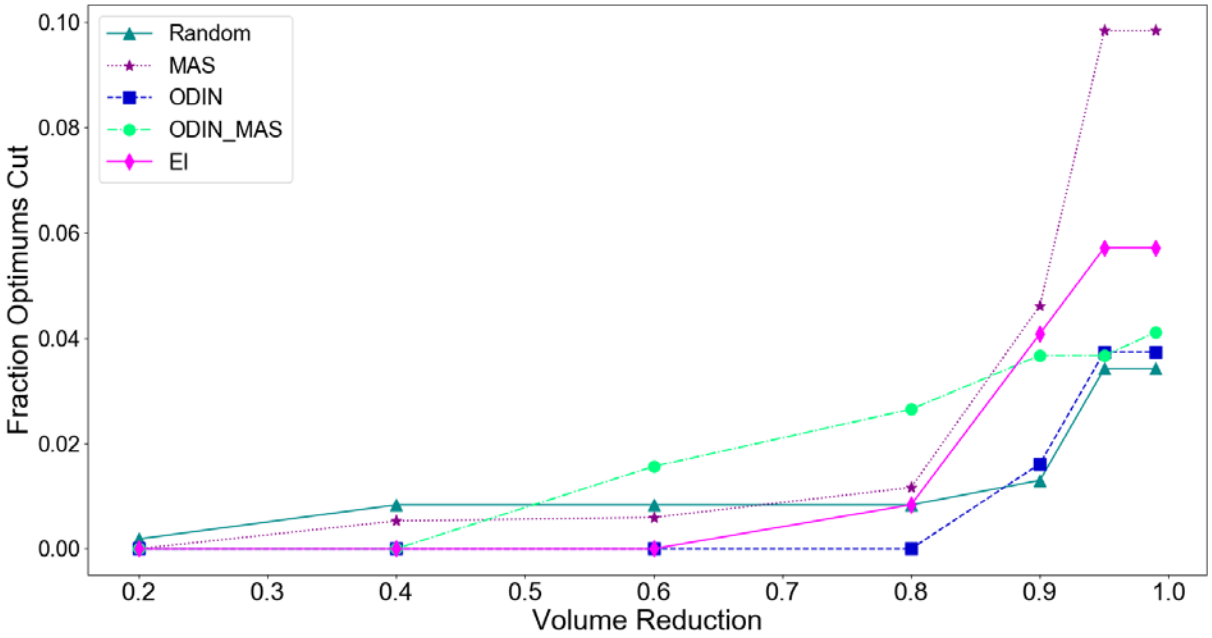


Figure 7.5 - Fraction of test functions with the actual location cut out of the reduced search space vs the fraction of the original search space volume removed ("Wide")

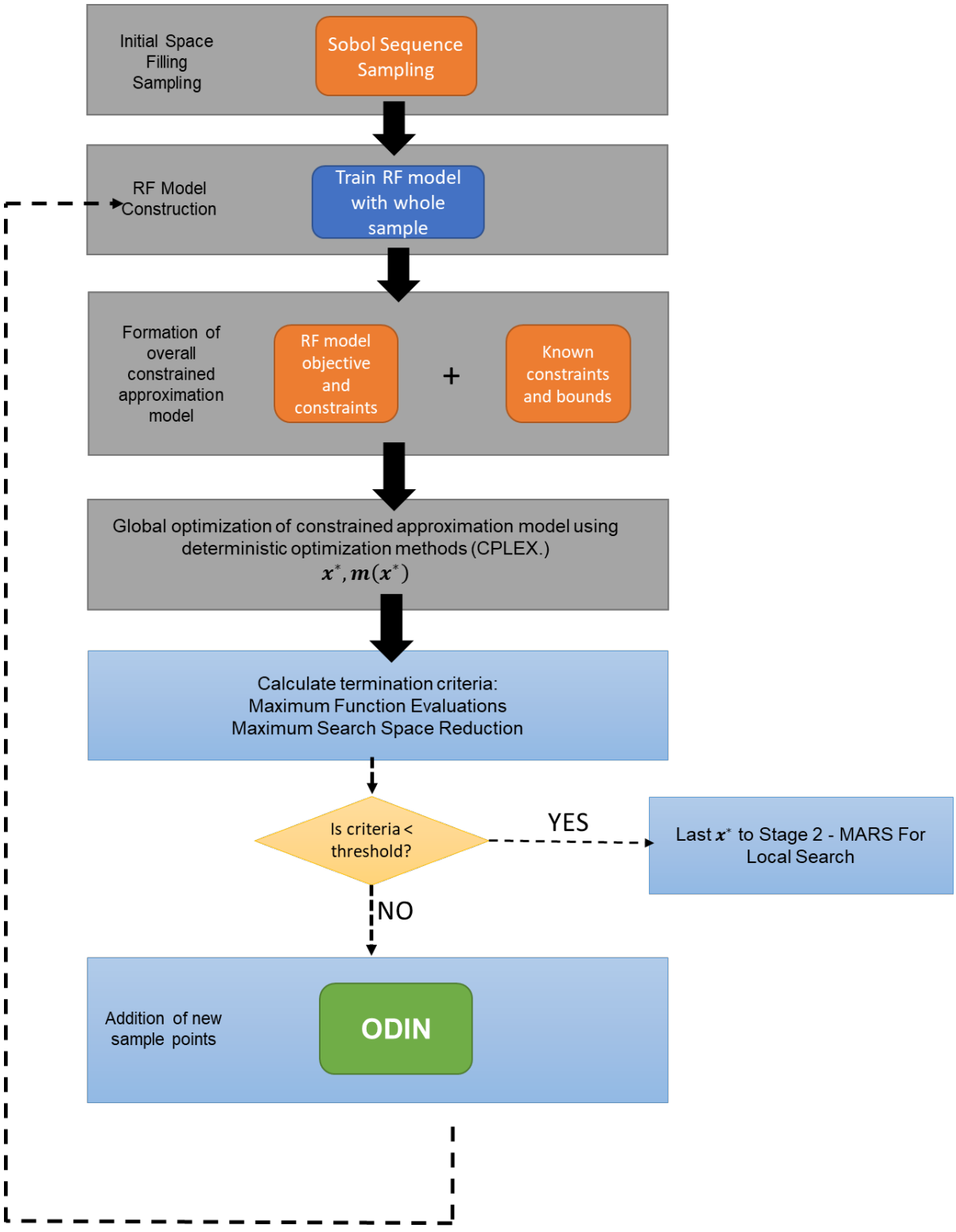


Figure 7.6 - Random forest (RF) model bounds generation step. (ODIN = Optimization Directed INcremental sampling)

7.4 pyBOUND Stage 2: Refinement of Solution with Multivariate Adaptive Regression Splines (MARS) Models

Once a suitable reduction of the search space is achieved from Stage 1, the next step in the pyBOUND algorithm is to refine that solution with a local search in the now reduced search space. MARS models were chosen for this stage of the algorithm as previous results have indictaed that MARS models are able to make accurate predictions for a variety of functional forms (Williams & Cremaschi, 2021b). Similar to the first stage, a surrogate model is constructed and iteratively improved with adaptive sampling. Based on the final form of the trained MARS surrogate model, a deterministic optimization problem is formulated and solved to refine the solution. MARS models result in MINLP optimization models, which are solved with ANTIGONE (Misener & Floudas, 2014). ANTIGONE was selected because specific relaxations are incorporated into its solution algorithm to effectively handle the bilinear terms of the MARS MINLP (Misener & Floudas, 2014).

New sample points are added in the second stage using adaptive sampling based on a hybrid of the ODIN sampling method and Mixed Adaptive Sampling (MAS) Eason and Cremaschi (2014), referred to here as ODIN-MAS. The hybrid algorithm was developed by Smith (2015) on the assumption that combining the exploitation advantages of the ODIN sampling algorithm with those of the MAS algorithm would perform better than the individual algorithms. This hybrid algorithm was chosen as the sampling method for the second stage because MARS models have been observed to exhibit poor approximation performance along the edges of the decision variable boundaries (Williams & Cremaschi, 2021b). The variance-reducing capability of the MAS algorithm is hypothesized to mitigate the edge effects of the MARS models.

ODIN-MAS requires an estimate of the MARS model variance. Currently, pyBOUND utilizes jackknife resampling (Berger, 2007) to estimate MARS model variance. In jackknife resampling, the variance is estimated by using a leave-one-out (LOO) strategy (Wong, 2015). With this method, a single data point is set aside (i.e., left out) for validation. The surrogate model is trained using the remaining data points, and a prediction is obtained for the data point for which the variance is being estimated. This process is repeated for each data point in the training set, resulting in a prediction for each. The model variance is then estimated as the variance of the aggregated LOO predictions.

Termination criteria for the second stage include a maximum number of function evaluations and a maximum accuracy of the MARS model. The accuracy of the MARS models is assessed in the algorithm by calculating the R-squared regression coefficient.

7.5 Computational Experiments

Fifty-four test problems were taken from a suite of optimization test functions in order to assess and compare the performance of pyBOUND for the minimization of black box models. The number of input dimensions of these functions varied from two up to 20 inputs. None of the test problems were used to make any of the decisions in designing the pyBOUND algorithm. Each test problem was used as a black box model for function evaluations in pyBOUND to estimate the global minimum of the test function. All of the test problems were implemented in Python 3.8. A table of the test problems and their related minimum values and optimum locations is provided in Appendix C.

7.5.1 DFO Algorithms for Comparison

The optimization capabilities of pyBOUND were compared to three commonly used DFO algorithms: Stable Noisy Optimization by Branch and Fit (SNOBFIT; (Huyer & Neumaier, 2008)),

Nonlinear optimization with the Mesh Adaptive Search (MADS) Algorithm (NOMAD; (Le Digabel, 2011)), and Implicit Filtering (ImFil; (Kelley, 2011)). SNOBFIT combines a branching strategy to enhance the chance of finding a global minimum with a sequential quadratic programming method based on fitted quadratic models to have good local properties (Huyer & Neumaier, 2008). ImFil is a steepest descent search algorithm that builds a local surrogate of the objective function using a quasi-Newton method to explore the search space for the optimization problem solution. Gradients are approximated for the objective function using interpolation (Gilmore & Kelley, 1995). SNOBFIT and ImFil were implemented using the Scikit-quant package. NOMAD is a direct-search algorithm that generates a series of meshes with varying sizes. At every iteration, the goal of the algorithm is to generate a trial point on each mesh that improves the current best solution. If this goal is not achieved, a finer mesh is generated on the next iteration. NOMAD was implemented using the NOMAD software's built-in Python interface. Computations were carried out on the Auburn University Easley HPC Cluster (Lenovo System X HPC Cluster) in Python 3.8.

7.5.2 Performance Metrics

Two performance metrics were used for evaluating the quality of the optimization solutions obtained by each of the DFO methods: the normalized optimum gap, g_{norm} (Eq. 7.4), and the optimum distance, d_{norm} (Eq. 7.5), between the scaled optimum location found by the DFO algorithm ($\bar{x}'_{opt}$) and the scaled actual optimum location of the test function ($\bar{x}_{opt}$). In Eq. 7.5, the actual and predicted optimum decision variables are scaled to a value between 0 and 1 using the original decision variable bounds designated for the test problem.

$$g_{norm} = \frac{min_{algo} - min_{actual}}{z_{max} - z_{min}} \tag{7.4}$$

$$d_{norm} = \left\| \bar{x}_{opt} - \bar{x}'_{opt} \right\| \tag{7.5}$$

In Eq. 7.5, $\left\| x_i - x_j \right\|$ is the Euclidean distance between any two points x_i and x_j.

7.6 Results and Discussion

7.6.1 Results for Original Test Functions

Figures 7.7 and 7.8 show results for the fraction of the original set of 127 test problems described in Section 4.1 for which pyBOUND was able to obtain g_{norm} (Eq. 7.4) and d_{norm} (Eq. 7.5) values, respectively, below certain thresholds. The thresholds for g_{norm} and d_{norm} are 0.00001 and 0.01, respectively. pyBOUND solved about 80% of the test functions for the objective function value (Fig 7.7) and solved about 60% of the test functions for the true optimum location (Fig 7.8).

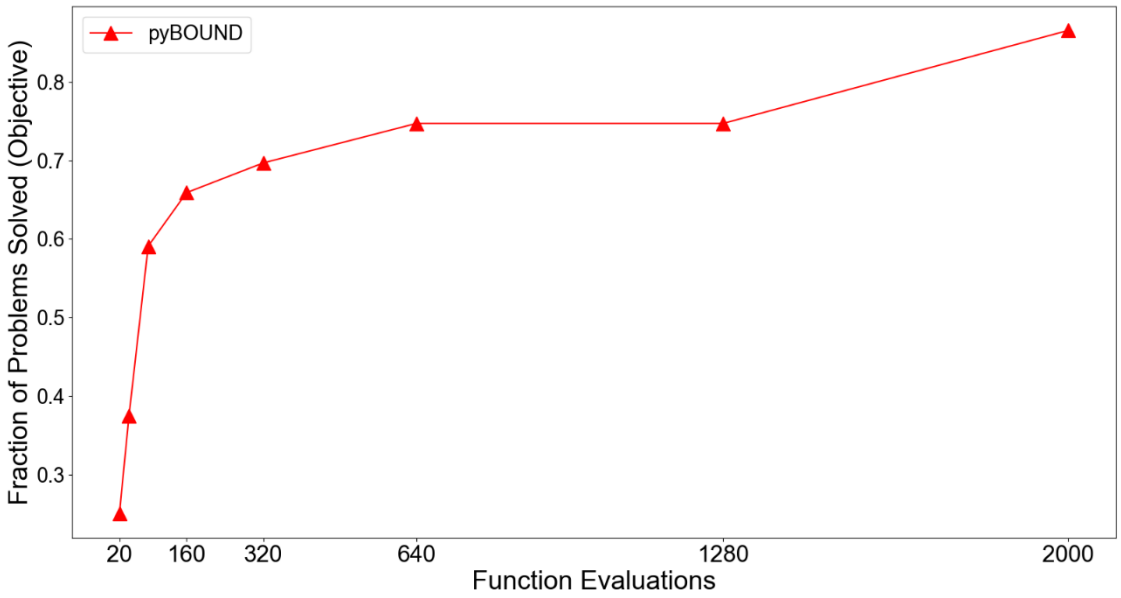


Figure 7.7 - Fraction of original test problems solved with g_{norm} less than 0.00001

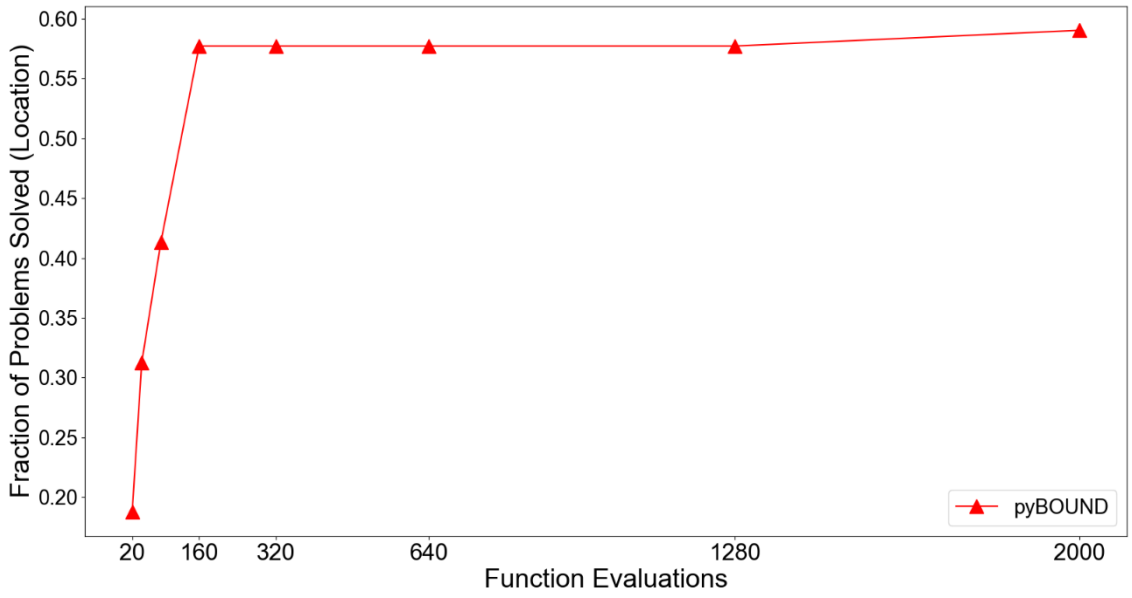


Figure 7.8 - Fraction of original test problems solves with d_{norm} less than 0.01

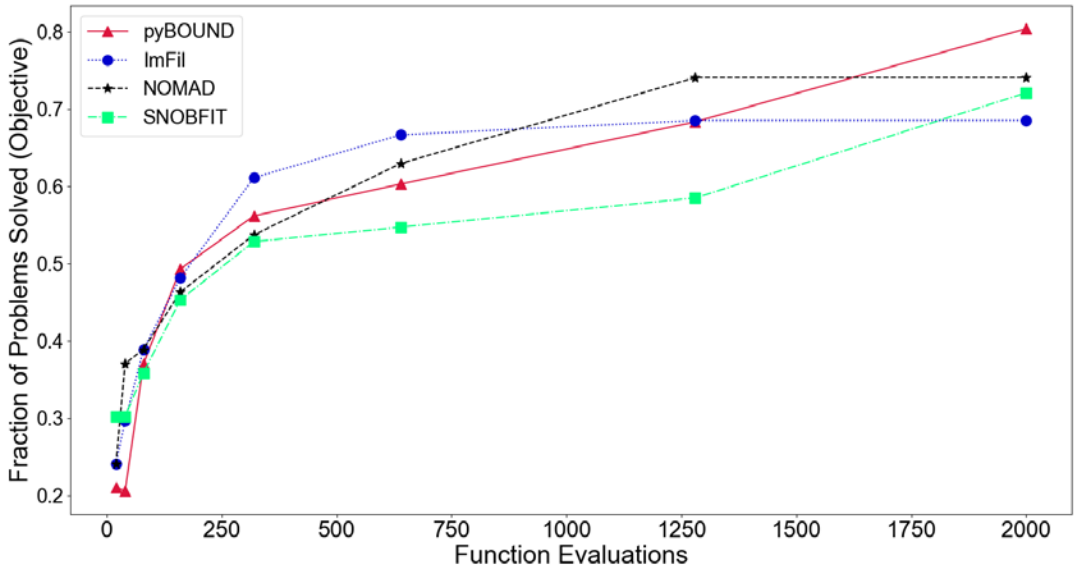
7.6.2 Results for New Test Problems

The computational experiments for the DFO algorithm performance comparisons were executed using each algorithm (pyBOUND, ImFil, NOMAD, SNOBFIT) to estimate the minimum value of the new set of problems and the location of the global minimum with an increasing number of function evaluations. These new optimization problems were not used in the construction of pyBOUND. Then, these results were compared to the global minimum and its true location using two metrics g_{norm} (Eq. 7.4) and d_{norm} (Eq. 7.5). Results are summarized in Figs. 7.8 and 7.9, where we define a model as having located the optimum when it obtains a g_{norm} or d_{norm} value less than a threshold. The thresholds for g_{norm} and d_{norm} are 0.00001 and 0.01, respectively.

Figure 7.9 shows results for how well each of the DFO algorithms estimates the minimum test function values for all the test problems (Fig 7.9a) and for only the test problems with higher input dimensions, greater than five (Fig 7.9b). When considering all test problems, pyBOUND

solves a lower fraction of the problems than both NOMAD and ImFil at less than 1000 function evaluations. However, when function evaluations are allowed to increase to the maximum allowed number of 2000, pyBOUND solves a higher fraction of the problems than any other method, yielding the minimum value of about 80% of the test problems (Fig 7.9a). This value is similar to the fraction of the test problems pyBOUND estimated the minimum function value. When considering test problems with high input dimensions, pyBOUND outperforms the other three algorithms even at lower numbers of function evaluations, estimating the highest fraction of the test problem optimum values. These results indicate that, in general, pyBOUND can successfully find the optimum values for black-box problems with a similar level of performance to existing algorithms. Results for test problems with higher input dimensions may indicate that pyBOUND performs better than existing algorithms for black-box optimization problems with a high dimensionality.

(a)



(b)

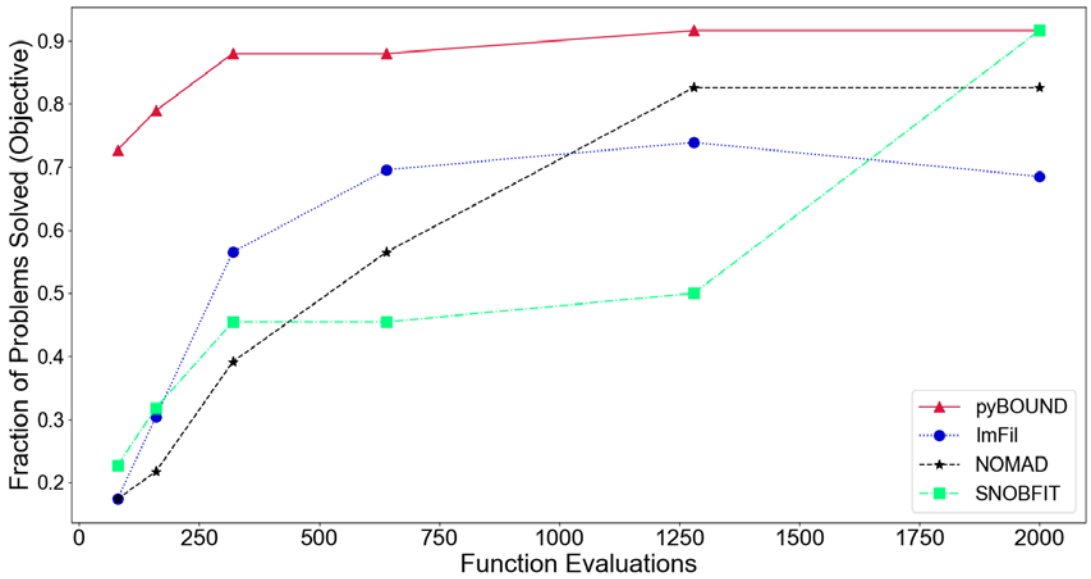


Figure 7.9 - Fraction of new test problems solved with g_{norm} less than 0.00001 for (a) all test problems and (b) test problems with input dimensions greater than 5

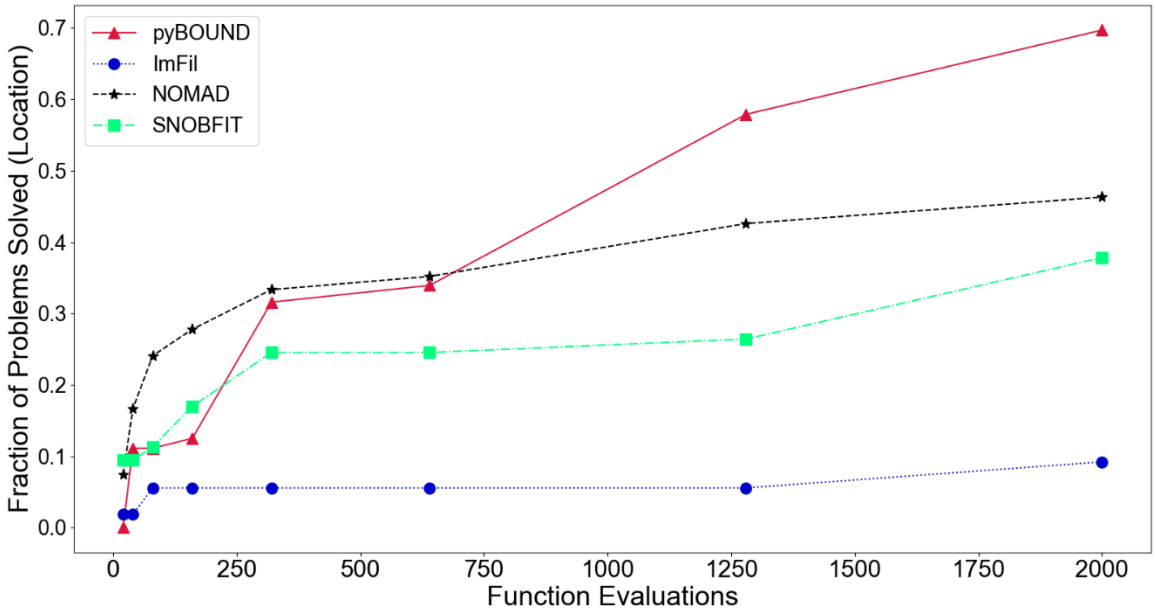


Figure 7.10 - Fraction of new test problems solves with d_{norm} less than 0.01

Figure 7.10 shows results for how well each of the DFO algorithms locates the true optimum decision variable values for the 54 test problems. Overall, pyBOUND is able to estimate the true optimum for the highest fraction of the test problems out of any of the methods tested. While the three comparison algorithms performed comparably to pyBOUND for estimating the minimum objective function value, they did not perform as well for locating the optimum decision variable values. These results may indicate that the comparison algorithms became trapped in local optima a larger fraction of the time than pyBOUND, and, thus, were still able to estimate objective function values in close approximation to the true minimum value but not the true optimum decision variable values. In general, pyBOUND outperformed the existing algorithms for locating the true optimum decision variable values for these 54 test problems.

7.7 Conclusions and Future Directions

Derivative-free optimization of black-box problems using surrogate-based optimization has been successfully applied in many algorithms. The proposed algorithm, pyBOUND, was

developed to address the difficulty in scaling DFO methods to high dimensionality problems. The results for the performance of pyBOUND demonstrate that the algorithm is able to estimate optimum values with similar performance to three commonly used DFO algorithms and provide evidence that pyBOUND may exhibit improved performance to existing algorithms for both high dimensionality problems and for locating the true optimal decision variable values. Future work on pyBOUND will focus on further refinement of the MARS stage of the algorithm, in order to reduce the amount of time for both solution of the MARS models and estimation of the model variance.

Chapter 8 – Conclusions and Recommendations for Future Work

In this dissertation, we developed surrogate models for optimization of processes as well as a surrogate-based optimization algorithm for derivative-free optimization of expensive black-box simulations.

8.1 Systematic Selection of Surrogate Modeling Techniques for Surface Approximation and Surrogate-Based Optimization

In Chapter 4, we comprehensively investigate and compare the performance of several different surrogate modeling techniques for both approximating functional relationships and surrogate-based optimization, and to link that performance to the characteristics of the data involved in the application. The results of the study provided general 'rules of thumb' for selecting modeling techniques. We used the results of Chapter 4's study to construct PRESTO (described in Chapter 5). PRESTO recommends surrogate modeling techniques for approximating a dataset with 91% accuracy and 90% precision and for performing surrogate based-optimization with 98% accuracy and 99% precision.

Recommendations for extending the surrogate modeling selection work include adding more real datasets to the training data for the PRESTO and focusing on using a wider variety of sampling methods, not just space-filling ones. Future work can also focus on including noisy data in the analysis, as all the test data used in this study were taken from smooth, continuous functions.

8.2 Surrogate-Based Optimization Using Random-Forests

In Chapter 6, we developed a method for automatically generating and solving an optimization problem using a RF model to approximate the objective function. This method can be used for surrogate-based optimization of complex models using only input-output data from

those models. While the resulting MILPs provide accurate estimates for the location of the optimum points for a large proportion of the test functions investigated, the large solution times required provide a significant obstacle for solving the necessary models to achieve in an accurate location in some cases. Future work can focus on decomposition solution approaches in order to decrease the solution time required for larger RF models.

8.3 pyBound (PYthon-based Black box Optimization Using raNDom forests)

Chapter 7 describes the development of pyBOUND (PYthon-based Black box Optimization Using raNDom forests). The performance analysis of pyBOUND demonstrated that the algorithm could estimate optimum values with similar performance to three commonly used DFO algorithms. The analysis also provides evidence that pyBOUND may exhibit improved performance to existing algorithms for both high dimensionality problems and for locating the true optimal decision variable values. Recommendations for future work on pyBOUND include further refinement of the MARS stage of the algorithm to reduce the amount of time required to solve the MARS optimization problems and estimate the model variance. The current jackknife resampling method for estimating the MARS variance involves a leave-one-out strategy for estimating the model variance. This strategy requires training a high number of MARS models, which can become computationally expensive. Improvements to the algorithm could focus on the development of less expensive estimates of MARS variance. Developing decomposition approaches for the MARS model MINLP could reduce the algorithm solution time. Additional recommendations include testing the algorithm against a wider range of DFO algorithms with more test problems at higher input dimensions. Further investigation into pyBOUNDS's performance for high dimensionality problems would provide value to its performance.

References

Afzal, A., Kim, K., & Seo, J. (2017). Effects of Latin hypercube sampling on surrogate modeling and optimization. International Journal of Fluid Machinery and Systems, 10, 240-253.

Al, R., Behera, C. R., Zubov, A., Gernaey, K. V., & Sin, G. (2019). Meta-modeling based efficient global sensitivity analysis for wastewater treatment plants - An application to the BSM2 model. Computers & Chemical Engineering, 127, 233-246.

Arora, S., Shen, W. X., & Kapoor, A. (2017). Neural network based computational model for estimation of heat generation in LiFePO4 pouch cells of different nominal capacities. Computers & Chemical Engineering, 101, 81-94.

Bajaj, I., Iyer, S. S., & Hasan, M. M. F. (2018). A trust region-based two phase algorithm for constrained black-box and grey-box optimization with infeasible initial point. Computers & Chemical Engineering, 116, 306-321.

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik, 4, 238-252.

Berger, Y. G. (2007). A jackknife variance estimator for unistage stratified samples with unequal probabilities. Biometrika, 94, 953-964.

Bertsimas, D., & Dunn, J. (2017). Optimal classification trees. Machine Learning, 106, 1039-1082.

Bhat, G., Spratt, H., Tamayo, E., Saade, G., & Menon, R. (2013). Multivariate adaptive regression splines analysis to predict biomarkers of spontaneous preterm birth. American Journal of Obstetrics and Gynecology, 208, S210-S210.

Bhosekar, A., & Ierapetritou, M. (2018a). Advances in surrogate based modeling, feasibility analysis, and optimization: A review. Computers & Chemical Engineering, 108, 250-267.

Bhosekar, A., & Ierapetritou, M. (2018b). Space mapping based derivative-free optimization framework for supply chain optimization. Computer Aided Chemical Engineering, 44, 985-990.

Biau, G., & Scornet, E. (2016). Rejoinder on: A random forest guided tour. Test, 25, 264-268.

Biggs, M., & Hariss, R. (2018). Optimizing Objective Functions Determined from Random Forests. Social Science Research Network, 1-49.

Boukouvala, F., Hasan, M. M. F., & Floudas, C. A. (2017). Global optimization of general constrained grey-box models: new method and its application to constrained PDEs for pressure swing adsorption. Journal of Global Optimization, 67, 3-42.

Boukouvala, F., & Ierapetritou, M. G. (2014). Derivative-free optimization for expensive constrained problems using a novel expected improvement objective function. Aiche Journal, 60, 2462-2474.

Boukouvala, F., Misener, R., & Floudas, C. A. (2016). Global optimization advances in Mixed-Integer Nonlinear Programming, MINLP, and Constrained Derivative-Free Optimization, CDFO. European Journal of Operational Research, 252, 701-727.

Breiman, L. (2001). Random forests. Machine Learning, 45, 5-32.

Burman, P. (1989). A Comparative-Study of Ordinary Cross-Validation, Nu-Fold Cross-Validation and the Repeated Learning-Testing Methods. Biometrika, 76, 503-514.

Burnaev, E. V., & Zaytsev, A. A. (2015). Surrogate modeling of multifidelity data for large samples. Journal of Communications Technology and Electronics, 60, 1348-1355.

Burnak, B., Diangelakis, N. A., Katz, J., & Pistikopoulos, E. N. (2019). Integrated process design, scheduling, and control using multiparametric programming. Computers & Chemical Engineering, 125, 164-184.

Conn, A., Scheinberg, K., & Vicente, L. (2009). Introduction to Derivative-Free Optimization (Vol. 8). Philadelphia, PA, USA: SIAM.

Cozad, A., Sahinidis, N. V., & Miller, D. C. (2014). Learning surrogate models for simulation-based optimization. Aiche Journal, 60, 2211-2227.

Crombecq, K., Laermans, E., & Dhaene, T. (2011). Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling. European Journal of Operational Research, 214, 683-696.

Cui, C., Hu, M. Q., Weir, J. D., & Wu, T. (2016). A recommendation system for meta-modeling: A meta-learning based approach. Expert Systems with Applications, 46, 33-44.

Dai, W., Cremaschi, S., Subramani, H. J., & Gao, H. J. (2019). Estimation of data uncertainty in the absence of replicate experiments. Chemical Engineering Research & Design, 147, 187-199.

Das, A. K., & Dewanjee, S. (2018). Optimization of Extraction Using Mathematical Models and Computation. Computational Phytochemistry, 75-106.

Davis, S., Cremaschi, S., & Eden, M. (2017). Efficient Surrogate Model Development: Optimum Model Form Based on Input Function Characteristics. In A. Espuna, M. Graells & L. Puigjaner (Eds.), 27th European Symposium on Computer Aided Process Engineering (ESCAPE 27) (Vol. 40, pp. 457-462). Barcelona, Spain: Elsevier.

De Maesschalck, R., Jouan-Rimbaud, D., & Massart, D. L. (2000). The Mahalanobis distance. Chemometrics and Intelligent Laboratory Systems, 50, 1-18.

Diwekar, U. M. (2003). A novel sampling approach to combinatorial optimization under uncertainty. Computational Optimization and Applications, 24, 335-371.

Drucker, H., Shahrary, B., & Gibbon, D. C. (2002). Support vector machines: relevance feedback and information retrieval. Information Processing & Management, 38, 305-323.

Du, D., Yang, H., Ednie, A. R., & Bennett, E. S. (2016). Statistical Metamodeling and Sequential Design of Computer Experiments to Model Glyco-Altered Gating of Sodium Channels in Cardiac Myocytes. IEEE J Biomed Health Inform, 20, 1439-1452.

Eason, J., & Cremaschi, S. (2014). Adaptive sequential sampling for surrogate model generation with artificial neural networks. Computers & Chemical Engineering, 68, 220-232.

Forrester, A., Sobester, A., & Keane, A. (2008). Engineering Design via Surrogate Modeling - A Practical Guide. Chichester: Wiley.

Garud, S. S., Karimi, I. A., & Kraft, M. (2017a). Design of computer experiments: A review. Computers & Chemical Engineering, 106, 71-95.

Garud, S. S., Karimi, I. A., & Kraft, M. (2017b). Smart Sampling Algorithm for Surrogate Model Development. Computers & Chemical Engineering, 96, 103-114.

Garud, S. S., Karimi, I. A., & Kraft, M. (2018). LEAPS2: Learning based Evolutionary Assistive Paradigm for Surrogate Selection. Computers & Chemical Engineering, 119, 352-370.

Gaspari, E., Franke, A., Robles-Diaz, D., Zweigerdt, R., Roeder, I., Zerjatke, T., & Kempf, H. (2018). Paracrine mechanisms in early differentiation of human pluripotent stem cells: Insights from a mathematical model. Stem Cell Res, 32, 1-7.

Gilmore, P., & Kelley, C. T. (1995). An Implicit Filtering Algorithm for Optimization of Functions with Many Local Minima. Siam Journal on Optimization, 5, 269-285.

Golkarnarenji, G., Naebe, M., Badii, K., Milani, A. S., Jazar, R. N., & Khayyam, H. (2018). Support vector regression modelling and optimization of energy consumption in carbon fiber production line. Computers & Chemical Engineering, 109, 276-288.

Gomm, J. B., & Yu, D. L. (2000). Selecting radial basis function network centers with recursive orthogonal least squares training. Ieee Transactions on Neural Networks, 11, 306-314.

Halloin, C., Schwanke, K., Lobel, W., Franke, A., Szepes, M., Biswanath, S., Wunderlich, S., Merkert, S., Weber, N., Osten, F., de la Roche, J., Polten, F., Wollert, K., Kraft, T., Fischer, M., Martin, U., Gruh, I., Kempf, H., & Zweigerdt, R. (2019). Continuous WNT Control Enables Advanced hPSC Cardiac Processing and Prognostic Surface Marker Identification in Chemically Defined Suspension Culture. Stem Cell Reports.

Halton, J. H., & Smith, G. B. (1964). Algorithm-247 - Radical-Inverse Quasi-Random Point Sequence [G5]. Communications of the Acm, 7, 701-702.

Han, Z., & Zhang, K. (2012). Surrogate-Based Optimization. In O. Roeva (Ed.), Real-World Applications of Genetic Algorithms (pp. 343-362). Rijeka, Croatia: InTech Open.

Harrell, F. E., Lee, K. L., Califf, R. M., Pryor, D. B., & Rosati, R. A. (1984). Regression Modeling Strategies for Improved Prognostic Prediction. Statistics in Medicine, 3, 143-152.

Hart, W. E., Laird, C. D., Watson, J.-P., Woodruff, D., L., Hackebail, G. A., Nicholson, B. L., & Siirola, J. D. (2017). Pyomo - Optimization Modeling in Python (2 ed. Vol. 67). Boston, MA: Springer.

Hart, W. E., Watson, J.-P., & Woodruff, D. L. (2011). Pyomo: modeling and solving mathematical programs in Python. Mathematical Programming Computation, 3, 219 - 260.

Haykin, S. (2009). Neural Networks and Learning Machines (3rd ed.). Upper Saddle River, New Jersey: Pearson Education, Inc.

He, Y. L., Geng, Z. Q., & Zhu, Q. X. (2016). Soft sensor development for the key variables of complex chemical processes using a novel robust bagging nonlinear model integrating improved extreme learning machine with partial least square. Chemometrics and Intelligent Laboratory Systems, 151, 78-88.

Hu, J. X., Zhou, Q., Jiang, P., Shao, X. Y., & Xie, T. L. (2018). An adaptive sampling method for variable-fidelity surrogate models using improved hierarchical kriging. Engineering Optimization, 50, 145-163.

Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006). Extreme learning machine: Theory and applications. Neurocomputing, 70, 489-501.

Hussain, K., Salleh, M. N. M., Cheng, S., & Naseem, R. (2017). Common Benchmark Functions for Metaheuristic Evaluation: A Review. International Journal of Informatics Visualization, 1, 218-223.

Huyer, W., & Neumaier, A. (2008). SNOBFIT - Stable noisy optimization by branch and fit. Acm Transactions on Mathematical Software, 35.

Iooss, B., Boussouf, L., Feuillard, V., & Marrel, A. (2010). Numerical studies of the metamodel fitting and validation process. International Journal of Advances in Systems and Measurements, 3, 11-21.

Jamil, M., & Yang, X.-S. (2013). A Literature Survey of Benchmark Functions for Global Optimization Prolems. International Journal of Mathematical Modelling and Numerical Optimization, 4, 150-194.

Joe, S., & Kuo, F. Y. (2008). Constructing Sobol' Sequences with Better Two-Dimensional Projections. Siam Journal on Scientific Computing, 30, 2635-2654.

Kelley, C. T. (2011). Implicit Filtering. Implicit Filtering, 23, 3-+.

Kempf, H., Andree, B., & Zweigerdt, R. (2016). Large-scale production of human pluripotent stem cell derived cardiomyocytes. Advanced Drug Delivery Reviews, 96, 18-30.

Kempf, H., Olmer, R., Haase, A., Franke, A., Bolesani, E., Schwanke, K., Robles-Diaz, D., Coffee, M., Gohring, G., Drager, G., Potz, O., Joos, T., Martinez-Hackert, E., Haverich, A., Buettner, F. F. R., Martin, U., & Zweigerdt, R. (2016). Bulk cell density and Wnt/TGFbeta signalling regulate mesendodermal patterning of human pluripotent stem cells. Nature Communications, 7.

Kleijnen, J. P. C., van Beers, W., & van Nieuwenhuyse, I. (2012). Expected improvement in efficient global optimization through bootstrapped kriging. Journal of Global Optimization, 54, 59-73.

Kolda, T. G., Lewis, R. M., & Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. Siam Review, 45, 385-482.

Le Digabel, S. (2011). Algorithm 909: NOMAD: Nonlinear Optimization with the MADS Algorithm. Acm Transactions on Mathematical Software, 37.

Le Thi, H. A., Vaz, A. I. F., & Vicente, L. N. (2012). Optimizing radial basis functions by d.c. programming and its use in direct search for global derivative-free optimization. Top, 20, 190-214.

Luyben, W. (2011). Design and Control of the Cumene Process. In Principles and Case Studies of Simultaneous Design (pp. 135-158). Hoboken, NJ: John Wiley & Sons, Inc.

Masampally, V., Pareek, A., & Runkana, V. (2018). Cascade Gaussian Process Regression Framework for Biomass Prediction in a Fed-batch Reactor. In IEEE Symposium Series on Computation Intelligence (SSCI '18) (pp. 129-135).

Matthews, B. W. (1975). Comparison of Predicted and Observed Secondary Structure of T4 Phage Lysozyme. Biochimica Et Biophysica Acta, 405, 442-451.

Maudes, J., Bustillo, A., Guerra, A. J., & Ciurana, J. (2017). Random Forest ensemble prediction of stent dimensions in microfabrication processes. International Journal of Advanced Manufacturing Technology, 91, 879-893.

Mckay, M. D. (1992). Latin Hypercube Sampling as a Tool in Uncertainty Analysis of Computer-Models. 1992 Winter Simulation Conference Proceedings, 557-564.

Mehmani, A., Chowdhury, S., Meinrenken, C., & Messac, A. (2018). Concurrent surrogate model selection (COSMOS): optimizing model type, kernel function, and hyper-parameters. Structural and Multidisciplinary Optimization, 57, 1093-1114.

Menze, B. H., Kelm, B. M., Masuch, R., Himmelreich, U., Bachert, P., Petrich, W., & Hamprecht, F. A. (2009). A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data. Bmc Bioinformatics, 10.

Misener, R., & Floudas, C. A. (2014). ANTIGONE: Algorithms for coNTinuous/Integer Global Optimization of Nonlinear Equations. Journal of Global Optimization, 59, 503-526.

Nentwich, C., & Engell, S. (2019). Surrogate modeling of phase equilibrium calculations using adaptive sampling. Computers & Chemical Engineering, 126, 204-217.

Nikkholgh, M. R., Moghadassi, A. R., Parvizian, F., & Hosseini, S. M. (2010). ESTIMATION OF VAPOUR-LIQUID EQUILIBRIUM DATA FOR BINARY REFRIGERANT SYSTEMS CONTAINING 1,1,1,2,3,3,3-HEPTAFLUOROPROPANE (R227ea) BY USING ARTIFICIAL NEURAL NETWORKS. Canadian Journal of Chemical Engineering, 88, 200-207.

Palmer, D. S., O'Boyle, N. M., Glen, R. C., & Mitchell, J. B. O. (2007). Random forest models to predict aqueous solubility. Journal of Chemical Information and Modeling, 47, 150-158.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

Puggini, L., Doyle, J., & McLoone, S. (2015). Fault Detection using Random Forest Similarity Distance. IFAC-PapersOnLine, 48, 583-588.

Qian, H., Hu, Y., & Yu, Y. (2016a). Derivative-free optimization of high-dimensional non-convex functions by sequential random embeddings. In G. Brewka (Ed.), Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI '16) (pp. 1946-1952). New York, NY: AAAI Press.

Qian, H., Hu, Y., & Yu, Y. (2016b). Derivative-Freee Optimization of High-Dimensional Non-Convex Functions by Sequential Random Embeddings. In S. Kambhampati (Ed.), Twenty-Fifth International Joint Conference on Artificial Intelligence. New York City, NY, USA: AAAI Press.

Quiroz, J. C., Mariun, N., Mehrjou, M. R., Izadi, M., Misron, N., & Radzi, M. A. M. (2018). Fault detection of broken rotor bar in LS-PMSM using random forests. Measurement, 116, 273-280.

Rahman, R. K., Ibrahim, S., & Raj, A. (2019). Multi-objective optimization of sulfur recovery units using a detailed reaction mechanism to reduce energy consumption and destruct feed contaminants. Computers & Chemical Engineering, 128, 21-34.

Rasmussen, C. E., & Nickisch, H. (2010). Gaussian Processes for Machine Learning (GPML) Toolbox. Journal of Machine Learning Research, 11, 3011-3015.

Rasmussen, C. E., & Williams, C. K. I. (2005). Gaussian Processes for Machine Learning. Adaptive Computation and Machine Learning, 1-247.

Rios, L. M., & Sahinidis, N. V. (2013). Derivative-free optimization: a review of algorithms and comparison of software implementations. Journal of Global Optimization, 56, 1247-1293.

Robertson, B. L., Price, C. J., & Reale, M. (2017). Stochastic global optimization using random forests. In G. Syme, D. Hatton MacDonald, B. Fulton & J. Piantadosi (Eds.), MODSIM2017, 22nd International Confress on Modelling and Simulation (pp. 784-789). Hobart, Tasmania, Australia: Modelling and Simulation Society of Australia and New Zealand.

Roth, G. A., Mensah, G. A., & Fuster, V. (2020). The Global Burden of Cardiovascular Diseases and Risks A Compass for Global Action. Journal of the American College of Cardiology, 76, 2980-2981.

Sabzekar, M., Yazdi, H. S., & Naghibzadeh, M. (2011). Relaxed constraints support vector machines for noisy data. Neural Computing & Applications, 20, 671-685.

Sahinidis, N. V. (1996). BARON: A general purpose global optimization software package. Journal of Global Optimization, 8, 201-205.

Smith, J. (2015). Computationally Assisted Biofuel Production: Hydrodynamics, Optimization, and Heuristics. University of Tulsa, Tulsa, OK.

Sokolov, M., Ritscher, J., MacKinnon, N., Souquet, J., Broly, H., Morbidelli, M., & Butte, A. (2017). Enhanced process understanding and multivariate prediction of the relationship between cell culture process and monoclonal antibody quality. Biotechnol Prog, 33, 1368-1380.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45, 427-437.

Surjanovic, S., & Bingham, D. (2013). Virtual Library of Simulation Experiments. In (Vol. 2018). Simon Fraser University.

Svetnik, V., Liaw, A., Tong, C., & Wang, T. (2004). Application of Breiman's random forest to modeling structure-activity relationships of pharmaceutical molecules. Multiple Classifier Systems, Proceedings, 3077, 334-343.

Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. Artificial Intelligence Review, 18, 77-95.

Wang, C., Duan, Q. Y., Gong, W., Ye, A. Z., Di, Z. H., & Miao, C. Y. (2014). An evaluation of adaptive surrogate modeling based optimization with two benchmark problems. Environmental Modelling & Software, 60, 167-179.

Wang, G. G., & Shan, S. (2007). Review of metamodeling techniques in support of engineering design optimization. Journal of Mechanical Design, 129, 370-380.

Williams, B., & Cremaschi, S. (2019). Surrogate Model Selection for Design-Space Approximation and Surrogate-Based Optimization. In S. Munoz, C. Laird & M. Realff (Eds.), Ninth International Conference on Foundations of Computer-Aided Process Design (FOCAPD-19) (pp. 353-358). Copper Mountain, CO, USA: Elsevier B.V.

Williams, B., & Cremaschi, S. (2021a). Novel Tool for Selecting Surrogate Modeling Technqiues for Surface Approximation. 31st European Symposium on Computer Aided Process Engineering, Pts a-C, 49, 451-456.

Williams, B., & Cremaschi, S. (2021b). Selection of surrogate modeling techniques for surface approximation and surrogate-based optimization. Chemical Engineering Research & Design, 170, 76-89.

Williams, B., Halloin, C., Lobel, W., Finklea, F., Lipke, E., Zweigerdt, R., & Cremaschi, S. (2020). Data-Driven Model Development for Cardiomyocyte Production Experimental Failure Prediction. 30th European Symposium on Computer Aided Process Engineering, Pts a-C, 48, 1639-1644.

Williams, B., Lobel, W., Finklea, F., Halloin, C., Ritzenhoff, K., Manstein, F., Mohammadi, S., Hashemi, M., Zweigerdt, R., Lipke, E., & Cremaschi, S. (2020). Prediction of Human Induced Pluripotent Stem Cell Cardiac Differentiation Outcome by Multifactorial Process Modeling. Front Bioeng Biotechnol, 8, 851.

Williams, B., Otashu, J., Leyland, S., Eden, M. R., & Cremaschi, S. (2021). PRESTO: Predictive REcommendation of Surrogate models To approximate and Optimize. Chemical Engineering Science, 249, 117360.

Wong, T. T. (2015). Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. Pattern Recognition, 48, 2839-2846.

Wu, D. Z., Jennings, C., Terpenny, J., Gao, R. X., & Kumara, S. (2017). A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests. Journal of Manufacturing Science and Engineering-Transactions of the Asme, 139.

Xu, Q. S., Liang, Y. Z., & Du, Y. P. (2004). Monte Carlo cross-validation for selecting a model and estimating the prediction error in multivariate calibration. Journal of Chemometrics, 18, 112-120.

Zeng, Z. (2020). Models and Solution Approaches for Large-Scale Multistage Stochastic Programs with Endogenous and Exogenous Uncertainty. Auburn University, Auburn, AL.

Zhang, D. H., Qian, L. Y., Mao, B. J., Huang, C., Huang, B., & Si, Y. L. (2018). A Data-Driven Design for Fault Detection of Wind Turbines Using Random Forests and XGboost. Ieee Access, 6, 21020-21031.

Zhou, L., Chen, J. H., & Song, Z. H. (2015). Recursive Gaussian Process Regression Model for Adaptive Quality Monitoring in Batch Processes. Mathematical Problems in Engineering.

Zhou, Y., Li, G., Dong, J., Xing, X. H., Dai, J., & Zhang, C. (2018). MiYA, an efficient machine-learning workflow in conjunction with the YeastFab assembly strategy for combinatorial optimization of heterologous metabolic pathways in Saccharomyces cerevisiae. Metab Eng, 47, 294-302.

Appendix A – Supplementary Data for Cardiomyocyte Feature Selection

Table A1. Summary of Feature Selection Results for Feature Set 1 (**X** = selected). The minimum and maximum value as well as the Pearson and Spearman correlation with the dd10 cardiomyocyte content is given.

Feature	**Minimum**	**Maximum**	**Pearson**	**Spearman**	**FS1-PCA**	**FS1-MARS**	**FS1-RF**	**FS1-GPR**
dd0 Cell Density	0.30	1.28	0.05	0.04		X		
dd0-dd1 Cell Density Gradient	-0.76	2.63	0.13	0.28				X
dd1 Cell Density	0.29	1.36	0.50	0.54			X	
dd1-dd2 Cell Density Gradient	0.03	3.57	-0.37	-0.01				
dd2 Cell Density	0.35	2.40	0.06	0.24				
dd2-dd3 Cell Density Gradient	-0.46	4.13	-0.30	-0.18				
dd3 Cell Density	0.69	3.00	-0.18	-0.07				
dd3-dd5 Cell Density Gradient	-0.49	0.81	0.38	0.31		X		
dd5 Cell Density	0.68	3.06	0.31	0.25				
dd5-dd7 Cell Density Gradient	-0.79	0.40	0.17	0.22				
dd7 Cell Density	0.16	3.05	0.44	0.54			X	

dd0 Aggregate Size	71.49	148.91	0.37	0.05	X	X		
dd0-dd1 Aggregate Size Gradient	-0.03	0.65	-0.51	-0.38				
dd1 Aggregate Size	79.61	199.12	0.05	-0.06	X			
dd1-dd2 Aggregate Size Gradient	-0.19	0.52	-0.05	0.06				
dd2 Aggregate Size	121.07	177.55	0.12	0.15	X			
dd2-dd3 Aggregate Size Gradient	-0.25	0.46	-0.13	-0.18				
dd3 Aggregate Size	114.85	219.12	-0.08	-0.16	X			
dd3-dd5 Aggregate Size Gradient	-0.22	3.26	-0.20	0.08				
dd5 Aggregate Size	115.77	754.75	-0.23	-0.02	X			
dd7 Aggregate Size	106.29	439.52	0.03	0.21	X			
dd5-dd7 Aggregate Size Gradient	-0.55	0.50	0.35	0.29			X	
Preculture Time [h]	45.00	56.00	-0.18	-0.17				X

Table A1.1. Summary of Feature Selection Results for Feature Set 1 (**X** = selected). The minimum and maximum value as well as the Pearson and Spearman correlation with the dd10 cardiomyocyte content is given.

Feature	**Minimum**	**Maximum**	**Pearson**	**Spearman**	**FS1-PCA**	**FS1-MARS**	**FS1-RF**	**FS1-GPR**

Start Preculture Perfusion [h after inoc] d1-d2	25.00	28.50	-0.51	-0.65				X
IWP2 Treatment Time [h]	43.00	54.00	0.09	0.34		X		X
Average DO concentration d0	19.40	155.99	0.07	0.18	X			
Average DO concentration d1	20.01	146.91	0.07	0.27	X			
Average DO concentration dd0	19.97	144.99	0.06	0.18	X		X	
Average DO concentration dd1	19.89	143.00	0.12	0.23	X			
Average DO concentration dd2	27.27	117.99	-0.09	-0.12	X	X		
Average DO concentration dd3	19.68	126.34	0.01	0.00	X			
Average DO concentration dd4	19.94	197.68	0.14	0.14	X			
Average DO concentration dd5	19.03	237.49	0.03	-0.05	X			
Average DO concentration dd6	22.74	237.25	-0.17	-0.38	X			
Average DO concentration dd7	14.14	214.26	-0.23	-0.45	X			

Average DO concentration time gradient d0	-355.00	238.56	0.31	0.40	X		X	
Average DO concentration time gradient d1	-12.36	69.65	0.07	-0.04	X			
Average DO concentration time gradient dd0	-5.80	2.76	0.00	0.03				
Average DO concentration time gradient dd1	-7.14	3.15	0.01	-0.01				
Average DO concentration time gradient dd2	-80.07	14.37	-0.14	-0.32				
Average DO concentration time gradient dd3	-4.23	8.50	0.14	0.11	X			

Table A1.1. Summary of Feature Selection Results for Feature Set 1 (**X** = selected). The minimum and maximum value as well as the Pearson and Spearman correlation with the dd10 cardiomyocyte content is given.

Average DO concentration time gradient dd4	-52.10	3.47	-0.28	-0.34		X		
Average DO concentration time gradient dd5	-28.23	3.96	-0.03	-0.20	X			
Average DO concentration time gradient dd6	-38.29	-2.38	0.02	-0.04	X		X	
Average DO concentration time gradient dd7	-6.51	9.61	-0.17	0.11	X			X
Cell density normalized DO concentration dd0	26.10	325.81	-0.25	-0.27	X			
Cell density normalized DO concentration dd1	21.51	242.12	-0.27	-0.26	X			
Cell density normalized DO concentration dd2	21.77	110.53	-0.14	-0.11	X			
Cell density normalized DO concentration dd3	15.29	95.74	-0.05	0.02	X			
Cell density normalized DO concentration dd5	11.07	236.31	-0.16	-0.33	X			
Cell density normalized DO concentration dd7	6.69	325.13	-0.16	-0.58	X		X	

Average cell density normalized DO gradient dd0	-5.66	3.10	0.09	0.05				
Average cell density normalized DO gradient dd1	-5.00	3.59	0.10	0.08				
Average cell density normalized DO gradient dd2	-47.24	7.00	-0.10	-0.21			X	X
Average cell density normalized DO gradient dd3	-3.82	7.20	0.07	0.10	X			
Average cell density normalized DO gradient dd5	-17.33	3.73	0.13	-0.15	X			X

Table A1.1. Summary of Feature Selection Results for Feature Set 1 (**X** = selected). The minimum and maximum value as well as the Pearson and Spearman correlation with the dd10 cardiomyocyte content is given.

Average cell density normalized DO gradient dd7	-7.23	18.30	-0.12	0.13	X		X	
dd0 Average acceleration of DO gradient	-2114.51	1270.22	-0.19	-0.30	X			

dd1 Average acceleration of DO gradient	-548.72	159.65	-0.26	-0.23	X			
dd2 Average acceleration of DO gradient	-717.88	569.36	-0.15	-0.13	X			
dd3 Average acceleration of DO gradient	-1403.97	6.14	-0.16	-0.26	X			
dd5 Average acceleration of DO gradient	-408.35	1810.40	0.18	0.12	X			
dd7 Average acceleration of DO gradient	-727.09	281.59	-0.19	-0.10	X			
dd0 Average acceleration of cell density normalized DO gradient	-3410.50	2190.03	-0.14	-0.37	X		X	
dd1 Average acceleration of cell density normalized DO gradient	-518.56	190.06	-0.25	-0.21	X			
dd2 Average acceleration of cell	-487.08	353.64	-0.14	-0.11	X			

density normalized DO gradient								
dd3 Average acceleration of cell density normalized DO gradient	-684.86	8.96	-0.14	-0.26	X			
dd5 Average acceleration of cell density normalized DO gradient	-296.53	1149.46	0.19	0.11	X			
dd7 Average acceleration of cell density normalized DO gradient	-635.01	268.18	-0.19	-0.10	X			
Overall Average pH	6.57	7.23	-0.14	-0.05				
Overall density gradient	-0.80	4.65	0.24	0.37		X		
Overall aggregate size gradient	-0.03	2.99	-0.19	0.07				
d0 Average pH	6.86	7.26	0.04	-0.14				

Table A1.1. Summary of Feature Selection Results for Feature Set 1 (**X** = selected). The minimum and maximum value as well as the Pearson and Spearman correlation with the dd10 cardiomyocyte content is given.

d0 Average pH Gradient	-0.18	0.01	0.21	-0.06			X	
d1 Average pH	6.46	7.86	-0.12	-0.21				
d1 Average pH Gradient	-0.21	0.01	-0.09	-0.01		X		X
dd0 Average pH	6.37	7.73	0.16	0.26				
dd0 Average pH Gradient	-0.23	0.08	-0.17	-0.25				
dd1 Average pH	5.15	7.87	-0.09	-0.11		X		
dd1 Average pH Gradient	-0.26	0.26	0.20	-0.02				
dd2 Average pH	5.99	7.31	-0.42	-0.46		X	X	
dd2 Average pH Gradient	-0.18	0.01	0.51	0.49				
dd3 Average pH	6.50	7.36	-0.27	-0.16				
dd3 Average pH Gradient	-0.17	0.02	0.25	0.12			X	
dd4 Average pH	6.09	7.24	-0.12	0.06				
dd4 Average pH Gradient	-0.18	0.01	0.26	0.24				
dd5 Average pH	6.56	8.25	0.09	0.10				
dd5 Average pH Gradient	-0.29	0.02	0.15	0.10				X
dd6 Average pH	6.39	7.25	-0.23	-0.12				

dd6 Average pH Gradient	-0.11	0.02	-0.02	0.21				
dd7 Average pH	6.74	7.31	0.23	0.20				
dd7 Average pH Gradient	-0.16	0.02	0.34	0.08				
dd0 Lactate Concentration	11.59	15.83	0.31	0.33				
dd1 Lactate Concentration	9.05	19.59	0.10	0.19	X			
dd3 Lactate Concentration	1.00	18.73	0.07	0.19				
dd5 Lactate Concentration	10.46	20.38	0.10	0.04	X			
dd7 Lactate Concentration	5.94	20.59	-0.02	-0.03				

Table A1.1. Summary of Feature Selection Results for Feature Set 1 (**X** = selected). The minimum and maximum value as well as the Pearson and Spearman correlation with the dd10 cardiomyocyte content is given.

dd0 Glucose Concentration	7.14	11.95	-0.24	-0.16				
dd1 Glucose Concentration	0.60	6.20	0.20	0.10				
dd3 Glucose Concentration	0.00	10.27	-0.08	-0.21				

dd5 Glucose Concentration	0.00	6.91	0.03	0.05				
dd7 Glucose Concentration	0.00	8.33	-0.14	-0.23				

Appendix B – PRESTO Training Data

Test Function	ALAMO	ANN	ELM	GP	MARS	RBFN	RF	SVM
Sphere_2D_50	1	0.991883	0.998527	0.999891	0.984819	0.995412	0.567829	0.972371
Sphere_4D_50	1	0.679355	0.374345	0.999743	0.938708	0.885513	-0.20604	0.896715
Sphere_6D_50	1	-0.32103	-1.38109	0.584517	0.6797	0.704757	-0.5477	0.773264
Sphere_8D_50	1	-1.5	-0.99898	0.461843	-0.01391	0.533775	-0.82552	0.411296
Sphere_10D_50	-1.5	-1.5	-1.5	0.212904	0.029458	0.340699	-1.2602	0.302759
Ellipsoid_2D_50	1	0.999386	0.998998	0.999886	0.991959	0.982831	0.647666	0.975139
Ellipsoid_4D_50	1	0.972754	0.095487	0.999729	0.934743	0.796317	0.063112	0.923058
Ellipsoid_6D_50	1	0.012355	-0.24285	0.757335	0.906734	0.572044	-0.25715	0.755811
Ellipsoid_8D_50	1	-1.5	-0.89361	0.610191	0.83531	0.40496	-0.62792	0.106989
Ellipsoid_10D_50	1	-1.5	-1.07773	0.476375	0.636184	0.259156	-0.84942	-0.14107
Sum of Different Powers_2D_50	0.998417	0.96248	0.99683	0.999323	0.983367	0.962979	0.531638	0.975553
Sum of Different Powers_4D_50	-1.5	0.158583	-0.57195	0.749224	0.846154	0.713696	-0.25444	0.811213

Sum of Different Powers_6D_50	-1.5	-1.5	-0.23581	0.412089	0.055331	0.523253	-0.48272	-0.12269
Sum of Different Powers_8D_50	-1.5	-0.88385	-1.5	0.258848	-0.26474	0.444601	-0.75403	-1.05526
Sum of Different Powers_10D_50	-1.5	-0.96565	-1.5	0.077975	-0.18174	0.262125	-1.08251	-0.88772
Sum of Squares_2D_50	1	0.99464	0.998302	0.99989	0.993516	0.963265	0.644871	0.97896
Sum of Squares_4D_50	1	0.891122	0.03013	0.999644	0.928025	0.805632	0.043647	0.928945
Sum of Squares_6D_50	1	-0.86844	-0.40176	0.826707	0.805668	0.611637	-0.37288	0.810521
Sum of Squares_8D_50	1	-1.24093	-1.00597	0.666572	0.840446	0.430393	-0.92552	-0.12561
Sum of Squares_10D_50	-1.5	-1.37587	-1.5	0.346557	0.477318	0.355969	-0.70161	-0.64066
Trid_2D_50	1	0.999703	0.998419	0.999911	0.977877	0.997463	0.713511	0.911079
Trid_4D_50	1	0.88689	0.423573	0.998636	0.756335	0.799795	-0.64207	0.873351
Trid_6D_50	1	-0.63838	-0.41016	0.848318	-0.06127	0.453277	-1.11443	0.716932
Trid_8D_50	-1.5	-1.02606	-0.86796	0.687593	-0.10428	0.293276	-1.41487	0.290363
Trid_10D_50	-1.5	-0.98551	-1.5	0.28193	-0.04517	0.170072	-1.43021	-0.32451
Perm_2D_50	0.999975	0.366541	0.999529	0.999869	1	0.996878	0.752834	0.932505
Perm_4D_50	0.998425	-0.63391	0.435027	-0.10269	0.629348	0.620584	-0.69051	-1.5

Perm_6D_50	-1.5	-0.81575	-0.09012	-1.5	0.066267	0.088246	-1.21029	-1.5
Perm_8D_50	-1.5	-0.64246	-1.5	-1.5	-0.46226	-0.1336	-1.47073	-1.5
Perm_10D_50	-1.5	-0.63198	-1.5	-1.5	-0.48879	-0.20345	-1.63527	-1.5
Bohachevsky_2D_50	1	0.998606	0.9983	0.999891	0.993516	0.963269	0.636029	0.978971
Dixon_Price_2D_50	0.932253	0.998433	0.991308	0.998778	0.98754	0.899464	0.836493	0.776174
Dixon_Price_4D_50	0.567243	0.412705	-0.04346	0.821688	0.948822	0.624614	0.208551	0.770469
Dixon_Price_6D_50	-1.5	-0.31957	-0.45331	0.597333	0.583042	0.4606	-0.22687	0.529303
Dixon_Price_8D_50	-1.5	-1.10185	-0.58201	0.47543	0.502895	0.362318	-0.83221	0.011865
Dixon_Price_10D_50	-1.5	-1.08656	-1.5	-0.0153	0.037613	0.23744	-0.61435	-1.5
Rosenbrock_2D_50	0.992232	0.999572	0.994771	0.999501	0.997448	0.965528	0.944532	0.833608
Rosenbrock_4D_50	0.973538	0.699565	-0.76041	0.949872	0.959903	0.650476	0.509064	0.176287
Rosenbrock_6D_50	-1.5	0.071869	0.002001	0.718071	0.921816	0.51107	0.311985	-0.1023
Rosenbrock_8D_50	-1.5	-0.61521	-0.59402	0.60082	0.868788	0.497417	-0.03383	-0.05657
Rosenbrock_10D_50	-1.5	-0.77959	-1.30608	0.380086	-0.15048	0.257544	-0.41694	-0.12439
3Hump_2D_50	0.990801	0.998725	0.634506	0.997668	0.992859	0.89648	0.810089	0.163489
6Hump_2D_50	0.894824	0.928145	0.715906	0.931298	0.909716	0.744162	0.528215	0.248031
Booth_2D_50	1	0.997037	0.998949	0.999275	0.996986	0.992131	0.635283	0.860427
Matyas_2D_50	1	0.998969	0.997691	0.999909	0.993636	0.994159	0.218241	0.971831

McCormick_2D_50	0.989584	0.992434	0.998515	0.998813	0.983295	0.98368	0.363914	0.884559
Zakharov_2D_50	0.974013	0.998206	0.999339	0.995958	0.980524	0.988421	0.467876	0.777433
Zakharov_4D_50	0.766549	0.987558	0.37432	0.929132	0.697904	0.401263	-0.80022	0.264688
Zakharov_6D_50	-1.5	0.953243	0.376145	0.872863	0.693764	-0.08837	-1.08565	-0.02681
Zakharov_8D_50	-1.5	0.792505	-1.5	0.730693	0.251604	-0.13102	-1.01908	-0.56637
Zakharov_10D_50	-1.5	0.969553	-1.5	0.507132	0.169522	-0.20514	-1.39108	-1.1121
PowerSum_4D_50	0.753261	0.523696	0.268972	0.736593	0.651823	0.353438	-0.26695	0.353632
Beale_2D_50	0.883135	0.379235	0.757562	0.838555	0.533899	0.598431	-0.38028	0.310821
Branin_2D_50	0.902476	0.987024	0.989747	0.995485	0.956918	0.896405	0.436637	-1.5
GoldsteinPrice_2D_50	0.753127	0.935189	0.875279	0.93931	0.901397	0.772101	0.260905	0.676832
StyblinskiTang_2D_50	0.990015	0.477489	0.932555	0.969359	0.921375	0.539851	0.486645	-1.29549
StyblinskiTang_4D_50	0.965141	-0.8427	-0.59956	-0.00381	0.333815	-0.13993	-0.11531	-1.5
StyblinskiTang_6D_50	-1.5	-1.5	-1.5	-0.12084	-0.5768	-0.21063	-0.88605	-1.5
StyblinskiTang_8D_50	-1.5	-0.87544	-1.5	-0.37897	-0.70479	-0.05296	-0.95974	-1.5
StyblinskiTang_10D_50	-1.5	-1.5	-1.5	-0.54389	-0.55778	-0.13132	-1.13684	-1.5
Perm_db_2D_50	0.9826	0.985744	0.998268	0.989014	0.958297	0.929078	0.775503	0.838194
Perm_db_4D_50	0.937618	0.931542	-0.54661	0.932948	0.915928	0.243678	0.45696	0.114804
Perm_db_6D_50	0.770769	-0.64473	-1.5	0.865451	0.805461	-0.1296	0.194415	-1.32697

Perm_db_8D_50	0.485347	-0.40476	-1.5	0.795058	-1.5	-0.20775	-0.23886	-1.41371
Perm_db_10D_50	-1.5	-0.19721	-1.5	-0.01216	-0.61029	-0.23808	-0.52584	-0.34579
Shekel_4D_50	-0.08537	-1.02119	-1.37698	-0.10286	-0.02331	0.323784	-0.99064	-1.5
Colville_4D_50	0.676627	0.111371	-0.41437	0.781178	0.946919	0.529135	0.346001	0.661308
Powell_4D_50	0.7188	0.510888	0.39547	0.954988	0.827737	0.54731	-0.51499	0.793735
Powell_8D_50	-1.5	-0.26491	-1.14723	0.705512	-0.04561	0.064764	-1.04633	0.129459
Hartmann_6D_50	0.035418	-0.11686	-1.20856	0.265293	-0.14614	-0.06027	-1.34079	-0.10586
Ackley_2D_50	0.361025	-0.021	0.081351	-1.5	0.067319	0.146373	-0.04714	0.500436
Ackley_4D_50	-0.03556	-0.03498	-1.07313	-1.5	-0.00648	-1.5	-2.26317	-0.05641
Ackley_6D_50	-0.02439	-1.5	-1.5	-1.5	-0.07574	-1.5	-4.88074	-1.5
Ackley_8D_50	-0.06612	-0.18937	-1.5	-1.5	-0.09281	-1.5	-6.71234	-1.5
Ackley_10D_50	-0.13747	-0.36433	-1.5	-1.5	-0.18032	-1.5	-7.76696	-1.5
Griewank_2D_50	0.999813	0.999208	0.999	0.999718	0.984617	0.995271	0.581613	0.974332
Griewank_4D_50	0.99997	0.038515	0.372949	0.999764	0.950531	0.885808	-0.24061	0.899717
Griewank_6D_50	0.999877	-0.46188	-0.37019	0.585474	0.679556	0.706134	-0.53773	0.77306
Griewank_8D_50	0.999954	-1.5	-1.5	0.461365	-0.01397	0.535432	-1.05924	0.411178
Griewank_10D_50	0.782419	-1.5	-1.5	0.211298	-0.00474	0.342059	-1.2862	0.303024
Schwefel_2D_50	0.096824	0.018579	-0.17047	0.060478	0.017776	-0.16663	-0.2473	-1.5

Schwefel_4D_50	-0.02777	-1.5	-0.46668	-0.30795	-0.14429	-0.42285	-0.85269	-1.5
Schwefel_6D_50	-0.52967	-0.76992	-0.67431	-0.35197	-0.30092	-0.34188	-1.19393	-1.03563
Schwefel_8D_50	-1.06399	-1.5	-1.5	-0.6703	-0.25881	-0.30694	-1.2439	-1.5
Schwefel_10D_50	-1.5	-1.5	-1.5	-0.82853	-0.59806	-0.32616	-1.21411	-1.41275
Rastrigin_2D_50	0.208071	0.187672	-0.29457	0.125208	0.384824	0.143072	-0.19374	-1.08402
Rastrigin_4D_50	0.104287	-0.40768	-0.56606	-0.09094	-0.01708	0.019158	-0.73407	-0.71862
Rastrigin_6D_50	-1.5	-0.51239	-0.47901	-0.41388	-0.24162	0.049814	-1.10641	-1.5
Rastrigin_8D_50	-1.5	-1.5	-1.5	-0.91548	-0.00235	0.111212	-1.14611	-1.5
Rastrigin_10D_50	-1.5	-1.5	-1.31349	-0.98443	-0.21492	0.116979	-1.33907	-1.34839
Levy_2D_50	0.536222	0.306399	0.397654	0.979847	0.460678	0.408935	0.471569	-0.10023
Levy_4D_50	0.267984	-0.00247	-0.1784	-0.15818	0.244106	0.213374	-0.64872	-0.66246
Levy_6D_50	0.237066	-0.37611	-1.5	0.116743	-0.19761	0.229232	-0.96031	-1.04134
Levy_8D_50	-1.5	-1.5	-1.5	-0.1159	-0.49646	0.265537	-1.33308	-1.2973
Levy_10D_50	-1.5	-1.05914	-1.5	-0.31045	-0.41203	0.181455	-1.42015	-1.5
Cross_IT_2D_50	-0.022	-1.5	-0.17428	-0.72824	-0.02877	-0.09398	-0.73056	-1.5
Drop_Wave_2D_50	0.109282	-0.66164	-0.1551	-1.5	0.107558	0.008478	-1.24592	-0.85612
Eggholder_2D_50	-0.07513	-1.5	-0.3907	-0.12616	-0.00036	-0.42873	-2.15016	-1.5
Holder_2D_50	0.152205	0.260596	0.105227	0.18093	0.276957	0.176985	-0.42813	-0.20458

Sphere_15D_50	-179.711	-9.8657	-2.04612	-0.62386	-0.57402	-0.14315	-1.39023	0.05799
Sphere_20D_50	-1023.52	-6.94551	-2.10197	-0.99494	-0.51106	-0.3626	-1.38987	-0.00863
Sum of Squares_15D_50	-273.808	-4.48563	-1.66883	0.073638	0.037495	-0.07413	-1.55792	-0.17834
Sum of Squares_20D_50	-12423.4	-5.27736	-2.99793	-0.39559	0.023466	-0.25744	-1.35164	-0.66812
Sum of Different Powers_15D_50	-491.756	-2.30014	-4.31176	-0.76535	-0.847	-0.15915	-1.47174	-0.98963
Sum of Different Powers_20D_50	-1921.03	-3.785	-2.65157	-0.79926	-1.39439	-0.28044	-1.61379	-0.93145
Trid_15D_50	-1154.78	-4.24324	-1.85249	-0.04945	-0.06158	-0.20677	-1.66054	-0.5681
Trid_20D_50	-19720.6	-4.35131	-2.30716	-0.81235	-1.34902	-0.3887	-1.77885	-0.90441
Zakharov_15D_50	-81.0491	-0.74462	-12.4229	0.357708	-0.79621	-0.34399	-1.42689	-0.66751
Zakharov_20D_50	-100.768	-2.43007	-51.4726	-0.03045	-0.42692	-0.36265	-1.45078	-0.55664
StyblinskiTang_15D_50	-1874.33	-2.60192	-3.88406	-1.81399	-1.07426	-7.98944	-1.53839	-7.97376
StyblinskiTang_20D_50	-8421.55	-4.07232	-1.81361	-1.58494	-0.36025	-11.0167	-1.67938	-2.42728
Ackley_15D_50	-0.37482	-1461.6	-349.337	-366.021	-0.39687	-0.64413	-8.1502	-40.6806
Ackley_20D_50	-0.6389	-3341.96	-774.914	-1069.69	-0.71247	-3.71819	-12.4876	-19.9989
Levy_15D_50	-900.773	-3.55876	-1.75725	-0.57354	-0.59877	-0.15819	-1.6137	-0.7892
Levy_20D_50	-1289.68	-6.13656	-1.96066	-1.08188	-1.02164	-0.26444	-1.80867	-1.35183

Griewank_15D_50	-20393.8	-9.96287	-1.26968	-0.62846	-0.49188	-0.14246	-1.44313	0.058305
Griewank_20D_50	-4549.34	-10.4029	-2.10484	-0.89054	-0.51384	-0.36233	-1.38469	-0.00836
Schwefel_15D_50	-360.005	-1.55833	-1.84166	-0.88437	-1.64773	-0.49821	-1.48415	-0.75612
Schwefel_20D_50	-975.97	-1.17584	-4.39523	-1.69883	-1.39663	-0.49422	-1.48156	-0.69417
Rastrigin_15D_50	-1606.7	-18.8848	-3.29329	-2.39242	-0.0058	-0.18398	-1.5193	-1.10371
Rastrigin_20D_50	-1549.54	-45.6624	-5.20567	-6.08304	-0.00687	-0.39529	-1.35988	-1.22331
Ellipsoid_15D_50	-23340.3	-5.51581	-1.22467	-0.07337	-0.30575	-0.0926	-1.21187	-0.06556
Ellipsoid_20D_50	-573.937	-5.17341	-1.62009	-1.1181	-0.88182	-0.27916	-1.36751	-0.08129
Rosenbrock_15D_50	-520.629	-1.99403	-3.95239	-0.29097	-0.36454	-0.23029	-0.48699	-0.41526
Rosenbrock_20D_50	-2506.72	-3.20402	-1.48249	-0.86472	-0.68147	-0.3882	-0.83075	-0.77982
Dixon_Price_15D_50	-424.303	-4.30997	-2.28008	-0.10867	-0.43866	-0.147	-1.37387	-0.52721
Dixon_Price_20D_50	-1286.49	-6.4999	-2.37561	-0.51435	-40.4408	-0.31098	-1.34138	-1.8528
Sphere_2D_100	1	0.99965	0.999666	0.999969	1	0.997929	0.80736	0.982342
Sphere_4D_100	1	0.981112	0.717242	0.999956	0.981313	0.902683	0.100553	0.956661
Sphere_6D_100	1	0.927373	-0.04109	0.999821	0.956236	0.777296	-0.27995	0.886685
Sphere_8D_100	1	-0.16667	-0.55563	0.999466	0.921064	0.6317	-0.56573	0.78665
Sphere_10D_100	1	-0.16258	-0.43226	0.412037	0.876366	0.402262	-0.83289	0.5819
Ellipsoid_2D_100	1	0.99982	0.999772	0.999972	1	0.9982	0.784364	0.974431

Ellipsoid_4D_100	1	0.993186	0.865404	0.999956	0.99719	0.917747	0.396185	0.96394
Ellipsoid_6D_100	1	0.941299	-0.01079	0.999786	0.98161	0.730255	0.054482	0.886975
Ellipsoid_8D_100	1	-0.23135	-0.48541	0.953564	0.966779	0.493607	-0.28148	0.816487
Ellipsoid_10D_100	1	-0.38757	-0.4214	0.758963	0.945661	0.343529	-0.46984	0.552875
Sum of Different Powers_2D_100	0.964339	0.999726	0.999159	0.999856	0.999216	0.998201	0.784405	0.980066
Sum of Different Powers_4D_100	-0.8705	0.351187	0.701029	0.980536	0.971888	0.824231	0.213477	0.90923
Sum of Different Powers_6D_100	-1.5	-1.05405	0.006733	0.633103	0.882001	0.624486	-0.23639	0.756212
Sum of Different Powers_8D_100	-1.5	-1.5	-0.38393	0.481344	0.681914	0.550618	-0.35191	0.540724
Sum of Different Powers_10D_100	-1.5	-1.5	-0.49091	0.244909	0.271795	0.409834	-0.59584	-0.352
Sum of Squares_2D_100	1	0.997403	0.999615	0.999971	1	0.997465	0.799499	0.974431
Sum of Squares_4D_100	1	0.991355	0.84042	0.999952	0.993305	0.88549	0.383976	0.95756
Sum of Squares_6D_100	1	0.918967	-1.38269	0.999735	0.978326	0.757515	0.119054	0.914608
Sum of Squares_8D_100	1	0.199777	-0.34657	0.991573	0.927089	0.575758	-0.41417	0.819993

Sum of Squares_10D_100	1	-1.2358	-1.5	0.805907	0.961139	0.420671	-0.64533	0.008161
Trid_2D_100	1	0.999862	0.998768	0.999971	1	0.998915	0.882016	0.953138
Trid_4D_100	1	0.99392	0.765222	0.999968	0.946347	0.950204	0.100484	0.923903
Trid_6D_100	1	0.91078	-0.40977	0.999851	0.86273	0.819821	-0.42306	0.916463
Trid_8D_100	1	-0.353	-1.45304	0.999146	0.534371	0.500224	-0.78055	0.850779
Trid_10D_100	1	-0.76942	-0.51227	0.810362	-0.57863	0.379894	-1.00689	0.662435
Perm_2D_100	0.999977	0.354091	0.999767	0.999977	0.99957	0.998208	0.911485	0.966209
Perm_4D_100	0.998969	-0.5962	0.871317	-0.03338	0.996971	0.884642	-0.14143	-1.5
Perm_6D_100	0.999747	-0.63426	0.660174	-1.5	0.716814	0.539328	-0.65706	-1.5
Perm_8D_100	0.99968	-0.69584	-0.06751	-1.5	-0.51805	-0.03305	-1.23201	0.61802
Perm_10D_100	-1.5	-0.58043	-1.5	-1.5	-0.08866	-0.1162	-1.70249	0.54859
Bohachevsky_2D_100	1	0.99955	0.999615	0.999971	1	0.997465	0.798219	0.974427
Dixon_Price_2D_100	0.962144	0.999376	0.999505	0.999708	0.998908	0.991053	0.9432	0.797246
Dixon_Price_4D_100	0.942686	0.906834	0.571424	0.975191	0.985931	0.734701	0.433498	0.833402
Dixon_Price_6D_100	0.896997	-0.496	-0.31681	0.752149	0.949484	0.668013	0.146407	0.803235
Dixon_Price_8D_100	0.87422	-0.373	-0.31058	0.675906	0.844348	0.539473	-0.25009	0.605305
Dixon_Price_10D_100	-1.5	-0.60645	-0.41276	0.558593	0.75461	0.403837	-0.4097	0.220653
Rosenbrock_2D_100	0.99338	0.998946	0.999183	0.999674	0.998255	0.998734	0.953702	0.937547

Rosenbrock_4D_100	0.98977	0.994866	0.748634	0.989613	0.966956	0.802436	0.789707	0.763109
Rosenbrock_6D_100	0.983894	0.14025	0.468232	0.867305	0.969937	0.649575	0.526087	0.427453
Rosenbrock_8D_100	0.926319	-0.29036	0.083712	0.688164	0.903634	0.561604	0.389686	0.332927
Rosenbrock_10D_100	0.405181	-0.48691	0.00463	0.6338	0.864206	0.412254	0.056163	0.258153
3Hump_2D_100	0.992026	0.999273	0.997813	0.999372	0.998238	0.949417	0.94009	0.586264
6Hump_2D_100	0.808768	0.987471	0.99506	0.994517	0.99115	0.889815	0.76585	0.33991
Booth_2D_100	1	0.999339	0.996546	0.999911	0.999056	0.997493	0.845422	0.912401
Matyas_2D_100	1	0.999711	0.998958	0.999968	0.999052	0.998752	0.776753	0.971712
McCormick_2D_100	0.993771	0.99954	0.998953	0.99898	0.992643	0.997725	0.850223	0.935825
Zakharov_2D_100	0.976602	0.999248	0.999783	0.99962	0.98898	0.996763	0.798448	0.876816
Zakharov_4D_100	0.871351	0.942508	0.822502	0.963485	0.875454	0.699907	0.042634	0.490279
Zakharov_6D_100	0.834467	0.991816	0.380813	0.931822	0.632164	0.178995	-0.38604	0.151404
Zakharov_8D_100	0.541029	0.966513	-0.30276	0.90437	0.414211	0.019252	-1.2343	-0.1179
Zakharov_10D_100	-1.5	0.991791	-1.24256	0.812326	0.387415	-0.13516	-1.20447	-0.97747
PowerSum_4D_100	0.701677	0.926294	0.484722	0.923678	0.737328	0.683421	0.103222	0.450718
Beale_2D_100	0.890003	0.828599	0.890697	0.934007	0.619134	0.873895	-0.30611	0.497737
Branin_2D_100	0.965501	0.996191	0.991954	0.999137	0.987719	0.984671	0.75726	-1.5
GoldsteinPrice_2D_100	0.763113	0.978624	0.979587	0.986968	0.925294	0.968991	0.641087	0.815855

StyblinskiTang_2D_100	0.992399	0.6237	0.9892	0.996846	0.976875	0.983588	0.715191	-0.38387
StyblinskiTang_4D_100	0.984849	0.185946	-0.04802	0.360619	0.909603	-0.01087	0.226074	-1.10544
StyblinskiTang_6D_100	0.961058	-0.53195	-0.18489	0.035917	0.709176	-0.01325	-0.37195	-1.5
StyblinskiTang_8D_100	-1.5	-0.69141	-0.32279	-0.10375	0.364614	0.03226	-0.33692	-1.5
StyblinskiTang_10D_100	-1.5	-1.42988	-1.5	-0.22812	0.057046	-0.01888	-0.6637	-1.27769
Perm_db_2D_100	0.964041	0.992518	0.998844	0.997823	0.946315	0.997372	0.89644	0.9262
Perm_db_4D_100	0.937608	0.937164	-0.17734	0.970156	0.907385	0.244771	0.788168	-0.28141
Perm_db_6D_100	0.858219	-0.36116	-1.20282	0.93014	0.747633	-0.13163	0.608378	-0.58496
Perm_db_8D_100	0.710783	-0.66132	-0.72571	0.934984	0.809493	-0.22716	0.418545	-0.64362
Perm_db_10D_100	-1.5	-1.5	-0.98414	0.907809	0.687365	-0.21244	0.225983	-0.95269
Shekel_4D_100	-0.02928	0.125881	0.11039	0.164864	0.14177	0.404813	-0.32742	-1.5
Colville_4D_100	0.942485	0.867208	-0.03021	0.995901	0.985041	0.702282	0.639157	0.781973
Powell_4D_100	0.90228	0.957928	-0.4955	0.993968	0.846287	0.629944	-0.04009	0.825404
Powell_8D_100	0.732423	0.001133	-1.0269	0.821976	0.530489	0.156038	-0.59074	0.599838
Hartmann_6D_100	0.227834	0.251753	-0.68461	0.366989	0.108299	0.171043	-0.91136	-0.38435
Ackley_2D_100	0.425334	0.70875	0.510033	-1.44669	0.239919	0.643927	0.264013	0.644505
Ackley_4D_100	-0.35862	-1.5	-0.13132	-1.5	-0.93236	-0.5471	-2.08849	0.17329
Ackley_6D_100	-0.01372	-1.5	-1.5	-1.5	-1.5	-1.5	-3.98183	-0.66062

Ackley_8D_100	-0.05502	-1.5	-1.5	-1.5	-0.06417	-1.5	-4.46719	-1.5
Ackley_10D_100	-0.12562	-0.18864	-1.5	-1.5	-0.11932	-1.5	-7.77997	-1.5
Griewank_2D_100	0.999819	0.999631	0.999184	0.99979	0.999766	0.997979	0.794496	0.985984
Griewank_4D_100	0.999973	0.985266	0.870127	0.999928	0.969005	0.902469	0.098762	0.957777
Griewank_6D_100	0.999995	0.861048	0.10676	0.999833	0.943022	0.777748	-0.29115	0.889373
Griewank_8D_100	0.999997	-0.79957	-0.37811	0.99951	0.919213	0.632642	-0.63683	0.785604
Griewank_10D_100	0.999997	-1.5	-0.83193	0.410631	0.88907	0.427149	-0.8356	0.584749
Schwefel_2D_100	0.144511	0.024465	-0.01052	0.225827	0.289089	-0.16029	0.061892	-1.5
Schwefel_4D_100	0.065354	-0.05743	-0.10954	0.013203	-0.07005	-0.21493	-0.6459	-1.5
Schwefel_6D_100	0.037441	-0.03025	-0.24414	-0.04356	-0.20329	-0.47031	-0.98137	-1.5
Schwefel_8D_100	-0.47708	-1.14267	-0.78339	-0.44473	-0.03922	-0.43071	-1.07352	-1.5
Schwefel_10D_100	-1.03086	-0.7016	-0.56705	-0.48638	-0.30925	-0.35524	-1.09425	-1.5
Rastrigin_2D_100	0.344002	-0.04728	0.257224	0.170687	0.479212	0.122237	0.081978	-1.5
Rastrigin_4D_100	0.225433	-0.03575	0.009374	-0.01592	0.276753	-0.15787	-0.65448	-0.48445
Rastrigin_6D_100	-0.13438	-1.5	-0.38352	-0.01314	0.059962	-0.02456	-0.9365	-0.57165
Rastrigin_8D_100	-0.98236	-1.5	-0.75364	-0.30245	-0.2052	0.029005	-1.13397	-1.34883
Rastrigin_10D_100	-1.17346	-1.5	-0.59062	-0.53266	-0.64684	0.113615	-1.23337	-1.22309
Levy_2D_100	0.678392	0.649477	0.077183	0.991821	0.943592	0.471336	0.812526	0.177742

Levy_4D_100	0.508912	0.171952	-0.02092	0.364377	0.548382	0.303737	0.042945	-0.42958
Levy_6D_100	0.373461	-0.82096	-0.24866	0.185089	0.313556	0.240446	-0.40888	-0.7518
Levy_8D_100	-0.17147	-1.097	-0.69295	0.122398	-0.03262	0.288655	-0.63977	-0.85862
Levy_10D_100	-1.5	-1.5	-0.30288	0.09767	-0.19465	0.218847	-1.08598	-0.52653
Cross_IT_2D_100	0.355755	0.226267	0.266149	-0.55916	0.051767	0.15522	-0.05831	-0.56527
Drop_Wave_2D_100	0.228013	0.208136	0.125932	-1.5	0.150408	0.1039	-0.98652	-0.79842
Eggholder_2D_100	-0.02658	-0.32071	-0.32999	0.342489	-0.06172	-0.80445	-1.88996	-1.5
Holder_2D_100	0.377521	0.300838	0.114274	0.399522	0.376571	0.125071	-0.38402	-0.20563
Sphere_15D_100	1	-4.61891	-0.76974	-0.07296	0.318527	-0.06355	-1.20445	0.380956
Sphere_20D_100	-135633	-6.65182	-0.82295	-0.39472	-0.31862	-0.29499	-1.28239	0.175048
Sum of Squares_15D_100	1	-3.32874	-0.49349	0.343819	0.458818	0.021935	-0.87706	-0.18465
Sum of Squares_20D_100	-1633.07	-5.26714	-1.02498	0.050193	-30.2109	-0.21613	-1.0208	-0.06819
Sum of Different Powers_15D_100	-370266	-1.66984	-0.68563	-0.04739	-0.07944	-0.06124	-1.0003	-0.64449
Sum of Different Powers_20D_100	-10074.5	-2.86243	-0.92506	-0.24678	-0.18088	-0.2572	-1.24724	-0.82174
Trid_15D_100	-413.514	-2.01178	-0.56875	0.213448	-0.3442	-0.13084	-1.22784	-0.35138
Trid_20D_100	-939602	-3.68791	-1.78983	-0.08366	-0.77087	-0.38906	-1.49069	-0.59027

Zakharov_15D_100	-391.276	0.124606	-5.39704	0.694233	-0.7969	-0.317	-1.16995	-0.66209
Zakharov_20D_100	-431.496	-0.56637	-16.083	0.540396	-0.16921	-0.33212	-1.02751	-0.62253
StyblinskiTang_15D_100	-170.639	-2.51585	-0.70288	-0.62874	-0.78387	-8.33725	-1.05208	-1.6366
StyblinskiTang_20D_100	-5286.51	-2.48305	-0.49842	-0.84771	-0.6966	-11.4528	-1.35812	-2.46108
Ackley_15D_100	-0.3605	-838.892	-153.952	-76.6771	-0.24146	-2.72688	-6.0916	-10.797
Ackley_20D_100	-0.62184	-1920.41	-326.582	-651.912	-0.40764	-1.57843	-7.22221	-13.7508
Levy_15D_100	-530.085	-2.66509	-1.22043	-0.17959	-0.68693	-0.07477	-1.34705	-0.90935
Levy_20D_100	-35768.1	-3.74071	-0.75929	-0.18584	-0.78689	-0.2388	-1.54952	-0.50568
Griewank_15D_100	0.993283	-4.64028	-0.5374	-0.07431	0.385209	-0.0629	-1.23211	0.380775
Griewank_20D_100	0.87662	-7.50464	-0.82432	-0.40076	-0.28119	-0.29473	-1.30121	0.175041
Schwefel_15D_100	-674.401	-1.21062	-1.04949	-0.66133	-1.20287	-0.4779	-1.42391	-0.65559
Schwefel_20D_100	-882.447	-2.21223	-3.08766	-1.07284	-1.04636	-0.54277	-1.46753	-1.23604
Rastrigin_15D_100	-1349.92	-6.80422	-1.13365	-1.24732	-0.26674	-0.17617	-1.48785	-0.30097
Rastrigin_20D_100	-2682.37	-17.3632	-1.41755	-1.89806	-408.412	-0.38405	-1.45162	-0.34758
Ellipsoid_15D_100	1	-3.9619	-1.55759	0.326778	0.849871	-0.02959	-0.83311	0.055438
Ellipsoid_20D_100	1	-6.2128	-0.47332	-0.20845	-0.29098	-0.24084	-1.18017	-0.15193
Rosenbrock_15D_100	-390.441	-0.40476	-0.12977	0.592427	0.284645	-0.10759	-0.24556	0.016212
Rosenbrock_20D_100	-1109.34	-1.3916	-0.49045	0.249173	-0.30072	-0.37114	-0.63169	-0.16587

Dixon_Price_15D_100	-628.462	-1.62503	-0.61525	0.40711	0.143907	-0.07602	-0.88208	-0.66442
Dixon_Price_20D_100	-12405	-3.7154	-0.89822	0.062394	-67.4029	-0.25691	-1.00926	-0.92123
Sphere_2D_200	1	0.999818	0.99983	0.999977	1	0.99959	0.896055	0.983602
Sphere_4D_200	1	0.992352	0.878889	0.999981	1	0.978514	0.418393	0.971384
Sphere_6D_200	1	0.985016	0.632009	0.999953	0.998001	0.827003	-0.00538	0.951962
Sphere_8D_200	1	0.639883	0.051673	0.999892	0.96563	0.73601	-0.23206	0.918747
Sphere_10D_200	1	-1.5	-0.2005	0.999677	0.958004	0.573536	-0.49236	0.843856
Ellipsoid_2D_200	1	0.99989	0.999883	0.999978	1	0.998617	0.844945	0.975094
Ellipsoid_4D_200	1	0.997384	0.983302	0.999981	0.999218	0.971204	0.562183	0.970839
Ellipsoid_6D_200	1	0.973605	0.492198	0.999948	0.999715	0.845852	0.299883	0.95285
Ellipsoid_8D_200	1	0.892473	-0.03616	0.994167	0.992324	0.658037	0.056029	0.924889
Ellipsoid_10D_200	1	-0.01884	-0.08647	0.980247	0.955159	0.524508	-0.17972	0.8411
Sum of Different Powers_2D_200	0.965078	0.99976	0.999343	0.999892	0.998936	0.997917	0.851457	0.975448
Sum of Different Powers_4D_200	0.723677	0.423029	0.862193	0.996386	0.998446	0.876706	0.451397	0.923519
Sum of Different Powers_6D_200	-1.5	0.76884	0.394734	0.75132	0.982926	0.694949	0.098486	0.831234

Sum of Different Powers_8D_200	-1.5	-0.1117	0.021265	0.620664	0.901513	0.631853	-0.16226	0.700068
Sum of Different Powers_10D_200	-1.5	-0.90163	-0.08803	0.483529	0.810007	0.501085	-0.39864	0.408003
Sum of Squares_2D_200	1	0.999186	0.999806	0.999978	1	0.998696	0.871759	0.975094
Sum of Squares_4D_200	1	0.99688	0.977416	0.99998	0.999198	0.979738	0.601109	0.973801
Sum of Squares_6D_200	1	0.965016	0.665232	0.999941	0.998623	0.813232	0.289942	0.962057
Sum of Squares_8D_200	1	0.955805	0.02825	0.993986	0.996697	0.74067	0.073749	0.926106
Sum of Squares_10D_200	1	0.376399	-0.19035	0.99625	0.975702	0.551819	-0.15301	0.860892
Trid_2D_200	1	0.99943	0.999066	0.99999	1	0.999442	0.947466	0.954512
Trid_4D_200	1	0.994666	0.990421	0.999989	0.993572	0.984814	0.401353	0.943189
Trid_6D_200	1	0.957988	0.784977	0.999959	0.97316	0.882979	-0.15961	0.938718
Trid_8D_200	1	0.957383	0.321243	0.999898	0.921074	0.780188	-0.52382	0.900433
Trid_10D_200	1	0.293496	0.05587	0.999724	0.853658	0.571388	-0.94069	0.86727
Perm_2D_200	0.999978	0.433105	0.999818	0.999994	1	0.999277	0.967012	0.983126
Perm_4D_200	0.999135	-0.57932	0.984532	0.005877	0.999229	0.963326	0.270847	0.03539
Perm_6D_200	0.999822	-0.62142	0.842975	-1.5	0.752068	0.716287	-0.42759	-1.5
Perm_8D_200	0.999872	-0.61487	0.300041	-1.5	0.563159	0.300169	-1.02087	0.76686

Perm_10D_200	0.999922	-0.55692	-0.92345	-1.5	-0.14928	-0.03254	-1.57566	0.853605
Bohachevsky_2D_200	1	0.999387	0.999806	0.999978	1	0.998696	0.869326	0.974771
Dixon_Price_2D_200	0.966565	0.999637	0.999702	0.999897	0.999024	0.998887	0.968579	0.929645
Dixon_Price_4D_200	0.956467	0.980038	0.743902	0.996741	0.989611	0.808083	0.661836	0.86546
Dixon_Price_6D_200	0.943106	0.744308	0.332012	0.950206	0.99005	0.735041	0.333325	0.854972
Dixon_Price_8D_200	0.949429	0.004946	0.051139	0.769367	0.970354	0.670239	0.085456	0.815164
Dixon_Price_10D_200	0.916377	0.485476	-0.41575	0.703027	0.917587	0.533467	-0.02653	0.722816
Rosenbrock_2D_200	0.995028	0.999036	0.999501	0.9999	0.999579	0.998472	0.985012	0.955934
Rosenbrock_4D_200	0.993392	0.997879	0.909323	0.997595	0.993458	0.940803	0.872683	0.845712
Rosenbrock_6D_200	0.990655	0.978474	0.578007	0.974566	0.991723	0.732214	0.627812	0.685778
Rosenbrock_8D_200	0.987478	0.437981	0.388264	0.819124	0.977928	0.645638	0.458684	0.561884
Rosenbrock_10D_200	0.97955	0.347104	0.380649	0.710717	0.968348	0.572787	0.261818	0.516381
3Hump_2D_200	0.989037	0.999141	0.999416	0.999603	0.998821	0.996371	0.989557	0.768982
6Hump_2D_200	0.672994	0.956045	0.998738	0.998826	0.999012	0.994254	0.838314	0.782116
Booth_2D_200	1	0.999631	0.999558	0.999977	1	0.998794	0.949072	0.938488
Matyas_2D_200	1	0.999832	0.999349	0.999988	1	0.99822	0.913293	0.972474
McCormick_2D_200	0.994541	0.999484	0.999142	0.99927	0.99292	0.997904	0.928821	0.945781
Zakharov_2D_200	0.977709	0.999693	0.997582	0.999826	0.991445	0.998108	0.945739	0.893762

Zakharov_4D_200	0.916217	0.993713	0.989421	0.994302	0.895699	0.956234	0.446596	0.781601
Zakharov_6D_200	0.882411	0.994098	0.834732	0.963392	0.815855	0.360756	0.077539	0.503316
Zakharov_8D_200	0.838451	0.997028	0.455427	0.957258	0.436355	0.166781	-0.56145	0.129526
Zakharov_10D_200	0.78002	0.992723	-0.68585	0.876926	0.485268	-0.00617	-0.66176	-0.23254
PowerSum_4D_200	0.943419	0.786341	0.838609	0.964849	0.876552	0.892585	0.357408	0.649295
Beale_2D_200	0.899173	0.947187	0.996983	0.993066	0.303733	0.981536	-0.66078	0.695728
Branin_2D_200	0.972976	0.998292	0.999166	0.999661	0.993968	0.997903	0.903917	-1.5
GoldsteinPrice_2D_200	0.787703	0.938125	0.997334	0.99733	0.972962	0.993561	0.836872	0.877464
StyblinskiTang_2D_200	0.994238	0.666087	0.998323	0.998586	0.994841	0.998531	0.733281	-0.21106
StyblinskiTang_4D_200	0.991802	0.264073	0.143279	0.985166	0.978323	0.280935	0.336063	-0.50997
StyblinskiTang_6D_200	0.985479	-0.39652	-0.17671	0.304134	0.949769	0.064957	0.11936	-0.84185
StyblinskiTang_8D_200	0.985361	-0.19044	-0.21085	0.231946	0.808326	0.063083	-0.16754	-0.96863
StyblinskiTang_10D_200	0.984592	-0.28757	-0.35884	0.059154	0.724754	0.016931	-0.44823	-1.45684
Perm_db_2D_200	0.968134	0.997902	0.999226	0.99924	0.957421	0.997422	0.952989	0.946884
Perm_db_4D_200	0.946653	0.949768	0.325678	0.98258	0.941125	0.351398	0.87463	0.242627
Perm_db_6D_200	0.914079	0.94552	-0.09466	0.96457	0.940424	-0.04198	0.863859	0.108007
Perm_db_8D_200	0.790035	0.17447	-0.38519	0.963132	0.925096	-0.19855	0.783105	-0.06803
Perm_db_10D_200	0.63291	0.369778	-1.16754	0.957299	0.909499	-0.20046	0.672092	-0.30095

Shekel_4D_200	0.417063	0.033955	0.390318	0.320211	0.389425	0.585199	-0.12365	-1.5
Colville_4D_200	0.949572	0.998708	0.765646	0.999031	0.991599	0.797875	0.730587	0.798635
Powell_4D_200	0.922037	0.95819	0.779326	0.998152	0.950146	0.898562	0.212824	0.864881
Powell_8D_200	0.882435	0.198146	-0.09135	0.891319	0.818271	0.427284	-0.04207	0.805865
Hartmann_6D_200	0.329122	0.223876	0.106812	0.641357	0.285149	0.352589	-0.6833	0.457387
Ackley_2D_200	0.493708	0.80172	0.736576	0.857395	0.376523	0.617753	0.265979	0.705883
Ackley_4D_200	0.407019	-1.5	0.29503	-1.5	-0.69921	0.212316	-0.97231	0.273438
Ackley_6D_200	0.103835	-0.02729	-0.51483	-1.5	-0.9343	-0.87327	-1.7859	-0.33636
Ackley_8D_200	-0.04964	-1.5	-1.5	-1.5	-0.0354	-1.01	-2.7704	-1.5
Ackley_10D_200	-0.11988	-1.5	-1.5	-1.5	-1.5	-1.5	-3.86258	-1.5
Griewank_2D_200	0.999822	0.999577	0.999601	0.999802	0.999798	0.999404	0.896036	0.986915
Griewank_4D_200	0.999978	0.996132	0.975012	0.999957	0.999888	0.979562	0.410807	0.970943
Griewank_6D_200	0.999996	0.958977	0.608579	0.999953	0.990021	0.827324	0.010822	0.951963
Griewank_8D_200	0.999998	-0.53794	-0.055	0.999898	0.962566	0.736309	-0.26132	0.918594
Griewank_10D_200	0.999999	-1.5	-0.20168	0.99969	0.964544	0.574013	-0.50294	0.843237
Schwefel_2D_200	0.157636	0.161605	0.249702	0.159997	0.773878	0.353142	0.547145	-1.5
Schwefel_4D_200	0.128724	-1.5	0.014504	0.084389	0.261998	-0.31978	-0.32484	-1.5
Schwefel_6D_200	0.115043	-0.0617	0.014838	-0.04853	0.140859	-0.39478	-0.63846	-1.5

Schwefel_8D_200	0.038252	-0.43704	-0.18768	-0.25101	-0.12761	-0.45651	-0.83624	-1.5
Schwefel_10D_200	0.011067	-0.34575	-0.18644	-0.35639	-0.40137	-0.33184	-1.03906	-1.5
Rastrigin_2D_200	0.504087	0.490546	0.485987	0.501632	0.486768	0.19725	0.372325	-1.5
Rastrigin_4D_200	0.369426	0.320015	0.191642	0.059452	0.360863	0.140567	-0.46681	-0.4099
Rastrigin_6D_200	0.297453	-0.04104	0.008786	-0.06146	0.379445	0.140009	-0.70525	-0.26962
Rastrigin_8D_200	0.243487	-0.27912	-0.2752	-0.087	0.178034	0.10258	-0.91924	-0.36923
Rastrigin_10D_200	0.047222	-0.22323	-0.41667	-0.22127	0.054926	0.102172	-1.03388	-0.17789
Levy_2D_200	0.685197	0.665926	0.574143	0.993064	0.98647	0.63668	0.935363	0.175422
Levy_4D_200	0.546534	0.300702	0.267294	0.406376	0.852546	0.334979	0.457725	-0.2586
Levy_6D_200	0.529975	-0.14428	0.023929	0.401692	0.679021	0.275967	0.006299	-0.27814
Levy_8D_200	0.472022	-0.06115	-0.07859	0.244809	0.627604	0.312658	-0.31273	-0.51179
Levy_10D_200	0.402505	-0.27931	-0.21296	0.229599	0.324578	0.273611	-0.70559	-0.5306
Cross_IT_2D_200	0.399461	0.202927	0.358555	-0.18468	0.469212	0.11974	0.310156	-0.52542
Drop_Wave_2D_200	0.260811	0.25545	0.189965	-0.8102	0.112988	0.075107	-0.98535	-0.85436
Eggholder_2D_200	-0.03686	-0.01009	-0.07603	0.599	-0.00106	-0.15554	-1.39607	-1.5
Holder_2D_200	0.411098	0.473432	0.408782	0.729855	0.438439	0.395668	-0.32582	-0.19592
Sphere_15D_200	1	-1.10596	-0.18012	0.103611	0.885795	0.009449	-0.79551	0.609669
Sphere_20D_200	1	-2.94319	-0.58441	-0.07959	0.806231	-0.25571	-1.03455	0.536625

Sum of Squares_15D_200	1	-1.60616	-0.13131	0.853978	0.908208	0.081034	-0.55054	0.153405
Sum of Squares_20D_200	1	-2.64931	-0.50699	0.553723	0.723246	-0.17554	-0.76779	0.27881
Sum of Different Powers_15D_200	-6.11834	-1.35967	-0.19991	0.314967	0.414152	0.05269	-0.67896	-0.3395
Sum of Different Powers_20D_200	-16997.5	-1.49528	-0.89793	0.175209	0.049481	-0.26377	-0.89108	-0.53989
Trid_15D_200	1	-1.1554	-0.32189	0.72585	-0.03593	0.028136	-1.07258	0.57528
Trid_20D_200	-8.94887	-1.48695	-0.41392	0.31508	-0.12746	-0.3015	-1.29148	-0.09315
Zakharov_15D_200	-1.5917	0.767038	-3.12102	0.889578	0.271402	-0.2764	-0.88782	-0.16009
Zakharov_20D_200	-1.09594	-0.25067	-7.55503	0.755626	0.302948	-0.33	-0.90208	-0.66452
StyblinskiTang_15D_200	0.778489	-2.21122	-0.36875	-0.24801	0.100488	-8.42738	-0.698	-1.17408
StyblinskiTang_20D_200	0.248396	-1.9166	-0.36496	-0.31676	-0.19603	-11.5435	-0.83135	-1.08219
Ackley_15D_200	-0.35356	-712.02	-78.6763	-63.8104	-0.10715	-2.23492	-3.45269	-8.69487
Ackley_20D_200	-0.61356	-11.8529	-144.962	-1062.87	-0.16098	0.070974	-5.36489	-12.0536
Levy_15D_200	0.050259	-1.0422	-0.28584	0.193641	-0.23163	0.005206	-0.96598	-0.36653
Levy_20D_200	-4.27687	-2.57086	-0.49569	0.039353	-0.48955	-0.2251	-1.23707	-0.47902
Griewank_15D_200	0.999998	-1.09822	-0.18208	0.102262	0.8984	0.010028	-0.79658	0.611477
Griewank_20D_200	0.999997	-3.4306	-0.32981	-0.0731	0.81817	-0.25546	-1.04434	0.536445

Schwefel_15D_200	-0.35354	-1.24923	-0.63553	-0.45386	-0.18729	-0.50494	-1.26614	-1.00282
Schwefel_20D_200	-11.5228	-1.2842	-1.19632	-0.73198	-1.12964	-0.53657	-1.33332	-0.85648
Rastrigin_15D_200	-1.1761	-5.33882	-0.44525	-0.61836	-0.40686	-0.18305	-1.18889	-0.09932
Rastrigin_20D_200	-14.0494	-7.76826	-1.40439	-1.0837	-612.801	-0.35365	-1.28682	-0.1821
Ellipsoid_15D_200	1	-1.20502	-0.23038	0.798292	0.868753	0.087065	-0.54179	0.305774
Ellipsoid_20D_200	1	-1.18086	-0.21676	0.504959	0.790475	-0.18597	-0.84286	0.219933
Rosenbrock_15D_200	0.550824	-0.41237	0.227697	0.643653	0.886534	0.06032	-0.113	0.358381
Rosenbrock_20D_200	0.168131	-0.53004	0.004604	0.593593	0.789566	-0.31194	-0.34811	0.304644
Dixon_Price_15D_200	0.676807	-0.98356	-0.21912	0.511586	0.836783	0.055484	-0.43084	-0.16074
Dixon_Price_20D_200	-1.79887	-1.82254	-0.41316	0.36561	0.7026	-0.23679	-0.69579	-0.3843
Sphere_2D_400	1	0.999696	0.999852	0.999981	1	0.999398	0.958331	0.983727
Sphere_4D_400	1	0.990807	0.997655	0.999987	1	0.994541	0.624083	0.97561
Sphere_6D_400	1	0.974552	0.787676	0.999984	1	0.943437	0.248207	0.965773
Sphere_8D_400	1	0.931611	0.456962	0.999966	0.999438	0.812765	-0.05483	0.954387
Sphere_10D_400	1	-0.01913	0.04905	0.999927	0.999304	0.687485	-0.25671	0.923238
Ellipsoid_2D_400	1	0.999675	0.999898	0.999982	1	0.998506	0.959942	0.976304
Ellipsoid_4D_400	1	0.99092	0.998921	0.999987	0.999126	0.990759	0.740812	0.975165
Ellipsoid_6D_400	1	0.985747	0.690979	0.999981	0.999192	0.927103	0.484208	0.967774

Ellipsoid_8D_400	1	0.979106	0.124041	0.999955	0.999266	0.774351	0.253268	0.958376
Ellipsoid_10D_400	1	0.949353	0.061668	0.996781	0.997227	0.666721	0.056368	0.941112
Sum of Different Powers_2D_400	0.965434	0.99972	0.999457	0.999908	0.999348	0.99916	0.949303	0.976871
Sum of Different Powers_4D_400	0.238831	0.908122	0.876977	0.998709	0.999153	0.979588	0.635574	0.930242
Sum of Different Powers_6D_400	-1.5	0.812363	0.62638	0.980641	0.996563	0.790557	0.303425	0.850679
Sum of Different Powers_8D_400	-1.5	0.724127	0.256014	0.698747	0.994067	0.670084	0.051733	0.750378
Sum of Different Powers_10D_400	-1.5	0.131086	0.057524	0.596232	0.990881	0.586906	-0.19008	0.622271
Sum of Squares_2D_400	1	0.998844	0.999831	0.999982	1	0.998905	0.959102	0.976068
Sum of Squares_4D_400	1	0.992556	0.992203	0.999987	0.998947	0.992331	0.731073	0.973699
Sum of Squares_6D_400	1	0.989861	0.697368	0.999981	0.999161	0.942406	0.477936	0.969618
Sum of Squares_8D_400	1	0.972782	0.2991	0.999951	0.999211	0.810689	0.218584	0.960407
Sum of Squares_10D_400	1	0.425936	0.041325	0.997051	0.998038	0.661833	0.042835	0.933858
Trid_2D_400	1	0.999432	0.999278	0.999994	1	0.999051	0.981543	0.963012

Trid_4D_400	1	0.990563	0.997581	0.999994	1	0.992804	0.639202	0.953992
Trid_6D_400	1	0.976504	0.847151	0.999987	0.999175	0.938965	0.174085	0.954467
Trid_8D_400	1	0.948949	0.500943	0.999969	0.999806	0.869218	-0.20218	0.942358
Trid_10D_400	1	0.917109	0.166941	0.999938	0.983842	0.751921	-0.53797	0.924982
Perm_2D_400	0.999978	0.348701	0.999846	0.999998	1	0.999565	0.985618	0.98505
Perm_4D_400	0.999205	-0.57087	0.99074	0.036406	0.99738	0.988854	0.54944	0.95127
Perm_6D_400	0.99978	-0.57686	0.874437	-1.5	0.998085	0.891335	-0.14072	0.076109
Perm_8D_400	0.999892	-0.58205	0.483222	-1.5	0.862808	0.553419	-0.81752	-1.5
Perm_10D_400	0.999943	-0.54654	0.242635	-1.5	0.775111	0.201629	-1.17368	0.895034
Bohachevsky_2D_400	1	0.999698	0.999831	0.999982	1	0.998905	0.958898	0.976274
Dixon_Price_2D_400	0.967389	0.999638	0.999789	0.999956	0.998725	0.998876	0.989791	0.958183
Dixon_Price_4D_400	0.964108	0.931252	0.837907	0.998917	0.992059	0.960689	0.771996	0.870512
Dixon_Price_6D_400	0.962117	0.822754	0.694676	0.982879	0.993315	0.786632	0.523054	0.864853
Dixon_Price_8D_400	0.960931	0.816301	0.338609	0.964569	0.990662	0.719541	0.287782	0.850422
Dixon_Price_10D_400	0.955984	0.360551	0.076941	0.798305	0.987988	0.643162	0.127146	0.832585
Rosenbrock_2D_400	0.995389	0.998358	0.999598	0.999948	0.998516	0.998756	0.994422	0.965431
Rosenbrock_4D_400	0.994281	0.996107	0.955987	0.999499	0.998044	0.985081	0.923099	0.913496
Rosenbrock_6D_400	0.99386	0.977201	0.732678	0.99091	0.995534	0.870376	0.804882	0.834822

Rosenbrock_8D_400	0.993231	0.963305	0.544857	0.979199	0.991135	0.730189	0.652874	0.735973
Rosenbrock_10D_400	0.991593	0.528498	0.429493	0.928389	0.982797	0.647834	0.466107	0.658344
3Hump_2D_400	0.991984	0.999284	0.999748	0.999683	0.998908	0.997991	0.997243	0.871867
6Hump_2D_400	0.647121	0.994537	0.994531	0.999387	0.998766	0.997446	0.901497	0.858026
Booth_2D_400	1	0.995523	0.999631	0.999991	1	0.998763	0.978486	0.946271
Matyas_2D_400	1	0.999219	0.999169	0.999992	1	0.999188	0.979341	0.972613
McCormick_2D_400	0.99477	0.998466	0.999285	0.99968	0.993858	0.998102	0.980557	0.946033
Zakharov_2D_400	0.979281	0.997563	0.997779	0.999882	0.993772	0.996988	0.978953	0.907078
Zakharov_4D_400	0.933203	0.99793	0.997297	0.996932	0.91102	0.982551	0.705563	0.839914
Zakharov_6D_400	0.898846	0.998627	0.905664	0.992169	0.851709	0.84628	0.336519	0.677488
Zakharov_8D_400	0.887197	0.989913	0.652272	0.97726	0.778605	0.531191	-0.01653	0.579384
Zakharov_10D_400	0.87287	0.993593	-0.30346	0.935007	0.570934	0.215758	-0.4233	-0.0788
PowerSum_4D_400	0.810685	0.882191	0.910182	0.98436	0.97293	0.955725	0.594382	0.755086
Beale_2D_400	0.915509	0.858335	0.997936	0.998449	0.653778	0.993924	0.635997	0.780049
Branin_2D_400	0.974496	0.996422	0.998411	0.999857	0.996614	0.997532	0.954371	-1.5
GoldsteinPrice_2D_400	0.80186	0.982137	0.997497	0.998944	0.972649	0.996983	0.924842	0.93824
StyblinskiTang_2D_400	0.991106	0.996601	0.998205	0.999223	0.999041	0.997211	0.832956	-0.069
StyblinskiTang_4D_400	0.991186	0.337742	0.177536	0.996256	0.996011	0.879504	0.506128	0.301686

StyblinskiTang_6D_400	0.987347	0.091775	0.170285	0.449357	0.987379	0.153848	0.20412	-0.56928
StyblinskiTang_8D_400	0.990072	0.036754	0.070966	0.327905	0.975825	0.124625	-0.03458	-0.62417
StyblinskiTang_10D_400	0.981687	0.049363	-0.04194	0.207912	0.957876	0.084627	-0.2459	-0.96057
Perm_db_2D_400	0.964197	0.922989	0.998373	0.999778	0.971939	0.998264	0.977431	0.95407
Perm_db_4D_400	0.961591	0.946405	0.343505	0.990606	0.96908	0.761173	0.906623	0.27649
Perm_db_6D_400	0.912716	0.946242	0.129327	0.971718	0.968993	0.048516	0.917208	0.124472
Perm_db_8D_400	0.795347	-0.01084	-0.0783	0.966607	0.958703	-0.12859	0.892649	0.057008
Perm_db_10D_400	0.716282	0.234653	-0.34817	0.962708	0.954217	-0.20351	0.850869	-0.00896
Shekel_4D_400	0.457505	0.487617	0.627718	0.477724	0.511997	0.62046	-0.02087	-1.5
Colville_4D_400	0.953732	0.999028	0.930896	0.999636	0.997799	0.96063	0.843357	0.815367
Powell_4D_400	0.930099	0.968926	0.972259	0.9991	0.961439	0.968332	0.817657	0.871711
Powell_8D_400	0.917297	0.804403	0.315575	0.975794	0.908758	0.580675	0.342768	0.836212
Hartmann_6D_400	0.417395	0.508502	0.43962	0.879648	0.437493	0.7125	-0.23017	0.630687
Ackley_2D_400	0.469666	0.79908	0.811102	0.901217	0.438699	0.850266	0.75906	0.751401
Ackley_4D_400	0.51339	0.236607	0.482497	-0.46963	0.199572	0.527553	0.141093	0.577106
Ackley_6D_400	0.090669	0.342441	0.189507	-1.5	-0.25624	-0.31729	-0.87297	0.076698
Ackley_8D_400	-0.20073	-0.01684	-1.5	-1.5	-1.5	-0.6906	-1.22147	-1.01751
Ackley_10D_400	-0.63751	-0.08892	-1.5	-1.5	-0.01415	-0.87486	-1.64499	-1.5

Griewank_2D_400	0.999824	0.999576	0.999662	0.999806	0.999814	0.999218	0.956695	0.987014
Griewank_4D_400	0.999978	0.98972	0.961691	0.999965	0.999976	0.993969	0.623304	0.97528
Griewank_6D_400	0.999996	0.97934	0.907866	0.999981	0.999615	0.943389	0.240772	0.966034
Griewank_8D_400	0.999999	0.272142	0.5455	0.999966	0.999146	0.830183	-0.06476	0.954368
Griewank_10D_400	1	-0.0191	0.04848	0.999929	0.98357	0.688482	-0.24661	0.92293
Schwefel_2D_400	0.165454	0.098348	0.17238	0.233023	0.978788	-0.30804	0.638808	-1.5
Schwefel_4D_400	0.161326	0.09383	0.079074	0.108493	0.662241	-0.32523	-0.03218	-1.5
Schwefel_6D_400	0.123198	0.058277	0.015877	0.034737	0.423985	-0.48978	-0.38049	-1.45821
Schwefel_8D_400	0.10546	0.090216	-0.03006	-0.17917	0.22287	-0.42893	-0.60447	-1.5
Schwefel_10D_400	0.086154	0.027753	-0.03709	-0.2343	-0.0906	-0.35884	-0.87152	-1.5
Rastrigin_2D_400	0.507857	0.501167	0.503043	0.507221	0.617343	0.191664	0.464774	-1.5
Rastrigin_4D_400	0.49367	0.483028	0.339621	0.26294	0.519471	0.236068	-0.20046	-0.4138
Rastrigin_6D_400	0.431782	0.203334	0.218506	-0.07496	0.405164	0.172878	-0.48635	-0.17929
Rastrigin_8D_400	0.331033	0.021847	-0.01861	-0.04221	0.319855	0.100079	-0.70487	-0.21937
Rastrigin_10D_400	0.323102	-0.13889	-0.11536	-0.11257	-0.32541	0.062619	-0.83206	-0.20476
Levy_2D_400	0.706053	0.591572	0.595593	0.995062	0.99394	0.850909	0.960002	0.190561
Levy_4D_400	0.668564	0.377159	0.405573	0.696856	0.98156	0.343301	0.596688	-0.02285
Levy_6D_400	0.636065	0.33969	0.279651	0.427	0.940845	0.301091	0.228069	-0.06933

Levy_8D_400	0.615938	0.192437	0.115524	0.390323	0.868198	0.308734	-0.04128	-0.10048
Levy_10D_400	0.610527	0.008512	0.041721	0.366779	0.724048	0.316246	-0.31395	-0.37539
Cross_IT_2D_400	0.426827	0.413657	0.243922	0.301718	0.81305	0.046041	0.429405	-0.40655
Drop_Wave_2D_400	0.310639	0.277461	0.274684	-0.68116	0.109621	0.025943	-0.84986	-0.79106
Eggholder_2D_400	-0.02089	-0.01866	-0.04154	0.769137	-0.02665	0.019367	-0.97831	-1.5
Holder_2D_400	0.426774	0.390387	0.417275	0.839348	0.613788	0.297453	-0.01575	-0.12281
Sphere_15D_400	1	-0.39515	-0.17359	0.999563	0.982789	0.142356	-0.66016	0.808444
Sphere_20D_400	1	-1.71286	-0.17757	0.764135	0.938563	-0.18741	-0.91913	0.650408
Sum of Squares_15D_400	1	-0.35178	-0.07159	0.985503	0.990151	0.179028	-0.30923	0.820917
Sum of Squares_20D_400	1	-1.42163	-0.12987	0.885886	0.922274	-0.13659	-0.60212	0.490962
Sum of Different Powers_15D_400	-25.5107	-0.83696	-0.07621	0.466401	0.951628	0.215312	-0.53003	0.145991
Sum of Different Powers_20D_400	-10.3584	-0.8304	-0.17332	0.351022	0.676435	-0.21671	-0.734	-0.25291
Trid_15D_400	1	0.46039	-0.14092	0.999646	0.971349	0.231814	-0.91808	0.842893
Trid_20D_400	1	-0.3656	-0.20582	0.996348	0.84833	-0.21211	-1.10528	0.678411
Zakharov_15D_400	0.888379	0.909721	-0.40636	0.924166	0.588008	-0.1957	-0.48322	0.274553
Zakharov_20D_400	0.834273	0.72447	-2.31618	0.815123	0.524755	-0.30904	-0.66483	-0.34217

StyblinskiTang_15D_400	0.989324	-1.63732	-0.09849	0.057818	0.935918	-8.38842	-0.50927	-0.90577
StyblinskiTang_20D_400	0.985883	-1.30034	-0.11558	-0.04851	0.562885	-11.5143	-0.73165	-0.57459
Ackley_15D_400	-0.35014	-3.76164	-36.41	-42.3464	-1.43489	-1.22365	-2.47968	-6.93365
Ackley_20D_400	-0.60949	-505.883	-81.8248	-166.815	-3.26805	0.140794	-3.39421	-9.09875
Levy_15D_400	0.530301	-1.03987	-0.06584	0.295718	0.344053	0.114607	-0.75986	-0.43323
Levy_20D_400	0.342689	-0.8057	-0.14717	0.198511	0.18219	-0.18206	-1.02348	-0.34723
Griewank_15D_400	1	-0.22538	-0.08968	0.999571	0.971593	0.128699	-0.66465	0.808057
Griewank_20D_400	1	-0.73509	-0.16961	0.764257	0.937457	-0.18754	-0.91938	0.652847
Schwefel_15D_400	-0.0411	-2.55782	-0.46389	-0.38843	-0.3911	-0.57183	-1.09658	-1.38973
Schwefel_20D_400	-0.12279	-1.93537	-0.67033	-0.64494	-0.83667	-0.5575	-1.20491	-1.04756
Rastrigin_15D_400	0.235833	-1.82491	-0.41566	-0.10557	0.04382	-0.24307	-1.13971	-0.11993
Rastrigin_20D_400	0.079399	-7.66259	-0.66893	-0.40423	-84.2518	-0.36489	-1.26429	-0.0328
Ellipsoid_15D_400	1	-0.09739	-0.03702	0.984169	0.99273	0.207683	-0.31201	0.855834
Ellipsoid_20D_400	1	-0.72592	-0.11263	0.888534	0.985395	-0.13196	-0.60202	0.444602
Rosenbrock_15D_400	0.990396	-0.13268	0.323862	0.710937	0.966199	0.274336	0.041192	0.588158
Rosenbrock_20D_400	0.983713	-0.0262	0.288651	0.658235	0.952656	-0.21321	-0.27961	0.483032
Dixon_Price_15D_400	0.935346	-0.53432	-0.08887	0.657532	0.962118	0.24154	-0.20369	0.719995
Dixon_Price_20D_400	0.923226	-0.76939	-0.10763	0.497107	0.885766	-0.17818	-0.50379	0.140447

Sphere_2D_800	1	0.999649	0.999866	0.999984	1	0.999315	0.984919	0.983788
Sphere_4D_800	1	0.996345	0.918534	0.99999	1	0.996347	0.751833	0.977834
Sphere_6D_800	1	0.980497	0.778497	0.999991	1	0.989753	0.416637	0.976285
Sphere_8D_800	1	0.976502	0.523603	0.999987	1	0.921055	0.164623	0.963548
Sphere_10D_800	1	0.204806	0.160379	0.999977	1	0.806904	-0.07972	0.951555
Ellipsoid_2D_800	1	0.999633	0.999014	0.999985	1	0.999388	0.98744	0.976562
Ellipsoid_4D_800	1	0.997953	0.993742	0.99999	1	0.995478	0.820686	0.973825
Ellipsoid_6D_800	1	0.991022	0.755987	0.999991	0.999237	0.972368	0.619715	0.973765
Ellipsoid_8D_800	1	0.986339	0.492432	0.999984	0.999008	0.892048	0.425179	0.968309
Ellipsoid_10D_800	1	0.973478	0.185328	0.999965	0.998374	0.785316	0.241614	0.95789
Sum of Different Powers_2D_800	0.951177	0.99975	0.999489	0.999937	0.999017	0.998792	0.983815	0.977919
Sum of Different Powers_4D_800	-1.5	0.920497	0.844831	0.999233	0.998934	0.994941	0.752265	0.931517
Sum of Different Powers_6D_800	-1.5	0.843011	0.614043	0.992755	0.997736	0.873551	0.470851	0.850297
Sum of Different Powers_8D_800	-1.5	0.773035	0.436766	0.946521	0.995418	0.730218	0.236796	0.778148

Sum of Different Powers_10D_800	-1.5	0.235275	0.136504	0.651668	0.992843	0.610035	0.01198	0.687467
Sum of Squares_2D_800	1	0.999786	0.999847	0.999985	1	0.998468	0.987544	0.976567
Sum of Squares_4D_800	1	0.995003	0.998564	0.99999	1	0.996243	0.821441	0.97387
Sum of Squares_6D_800	1	0.987831	0.810437	0.99999	0.999247	0.971531	0.604048	0.974823
Sum of Squares_8D_800	1	0.985817	0.713596	0.999983	0.999145	0.908686	0.41664	0.967683
Sum of Squares_10D_800	1	0.971476	0.130568	0.999967	0.997438	0.799755	0.206827	0.955905
Trid_2D_800	1	0.998968	0.99932	0.999995	1	0.99934	0.992541	0.963655
Trid_4D_800	1	0.991945	0.984181	0.999996	1	0.993921	0.769476	0.959917
Trid_6D_800	1	0.98723	0.863952	0.999995	1	0.98217	0.406439	0.964588
Trid_8D_800	1	0.97361	0.590462	0.99999	0.998814	0.917003	0.090491	0.95372
Trid_10D_800	1	0.954639	0.400673	0.999982	0.993139	0.849136	-0.19266	0.947025
Perm_2D_800	0.999979	0.442545	0.999857	0.999998	0.998741	0.999656	0.995313	0.983808
Perm_4D_800	0.999234	-0.56722	0.996109	0.073024	0.997185	0.993667	0.741489	0.993635
Perm_6D_800	0.999845	-0.55994	0.922299	-1.5	0.997573	0.969982	0.148309	0.856017
Perm_8D_800	0.9999	-0.55049	0.780588	-1.5	0.996436	0.756681	-0.44806	-0.89604
Perm_10D_800	0.99995	-0.54103	0.624652	-1.5	0.991392	0.403884	-0.80881	-1.5
Bohachevsky_2D_800	1	0.999489	0.999847	0.999985	1	0.998468	0.987237	0.976496

Dixon_Price_2D_800	0.967807	0.9996	0.999558	0.999982	0.999104	0.998743	0.995589	0.969316
Dixon_Price_4D_800	0.966971	0.993729	0.746143	0.999776	0.993336	0.992959	0.862137	0.91106
Dixon_Price_6D_800	0.964688	0.864249	0.759508	0.995155	0.994732	0.867803	0.654457	0.871128
Dixon_Price_8D_800	0.964339	0.516214	0.450108	0.983799	0.991433	0.767606	0.487711	0.871072
Dixon_Price_10D_800	0.962498	0.791912	0.222745	0.967677	0.991298	0.715531	0.229437	0.865168
Rosenbrock_2D_800	0.995449	0.99886	0.999624	0.999974	0.998529	0.999159	0.997393	0.96862
Rosenbrock_4D_800	0.994908	0.996075	0.992708	0.999845	0.998479	0.99437	0.951219	0.944801
Rosenbrock_6D_800	0.994665	0.99793	0.79573	0.998366	0.995673	0.943811	0.871725	0.898812
Rosenbrock_8D_800	0.994383	0.985404	0.653817	0.985677	0.994597	0.823199	0.759118	0.827876
Rosenbrock_10D_800	0.994109	0.869777	0.541963	0.978959	0.992862	0.718157	0.599103	0.750844
3Hump_2D_800	0.993678	0.999317	0.999068	0.999713	0.998962	0.998874	0.998771	0.914966
6Hump_2D_800	0.945907	0.994543	0.999638	0.99991	0.998572	0.99732	0.954047	0.889285
Booth_2D_800	1	0.999143	0.999656	0.999997	1	0.999309	0.993461	0.949362
Matyas_2D_800	1	0.999405	0.999209	0.999993	1	0.998246	0.99189	0.972682
McCormick_2D_800	0.994816	0.998712	0.999342	0.999935	0.997025	0.998119	0.992583	0.95088
Zakharov_2D_800	0.980455	0.997141	0.997844	0.999934	0.993693	0.997958	0.990282	0.91859
Zakharov_4D_800	0.93612	0.997385	0.991192	0.998921	0.938258	0.988117	0.823905	0.872591
Zakharov_6D_800	0.91605	0.994979	0.949221	0.995432	0.906441	0.927555	0.588111	0.765993

Zakharov_8D_800	0.905471	0.989738	0.865079	0.986733	0.836814	0.779639	0.286487	0.673273
Zakharov_10D_800	0.905084	0.997025	0.216994	0.971029	0.750692	0.384723	0.016597	0.25278
PowerSum_4D_800	0.883708	0.967326	0.973294	0.992953	0.982277	0.977945	0.755931	0.849352
Beale_2D_800	0.920123	0.859493	0.998492	0.999484	0.955586	0.994087	0.926688	0.823584
Branin_2D_800	0.968544	0.996435	0.998522	0.999937	0.996403	0.997771	0.984677	-1.5
GoldsteinPrice_2D_800	0.797327	0.980308	0.997784	0.999493	0.971189	0.996138	0.970615	0.95344
StyblinskiTang_2D_800	0.990864	0.552167	0.99839	0.999484	0.997843	0.997702	0.931554	-0.02548
StyblinskiTang_4D_800	0.990898	0.378161	0.979697	0.998196	0.995545	0.973861	0.616359	0.68018
StyblinskiTang_6D_800	0.989185	0.250428	0.200768	0.993006	0.987903	0.371261	0.333382	-0.60406
StyblinskiTang_8D_800	0.990733	0.121192	0.108221	0.350386	0.984971	0.143272	0.129084	-0.55464
StyblinskiTang_10D_800	0.9896	0.056869	0.010532	0.273789	0.981113	0.109877	-0.07419	-0.73293
Perm_db_2D_800	0.997104	0.92452	0.998484	0.999905	0.995775	0.997797	0.992979	0.960585
Perm_db_4D_800	0.960004	0.963087	0.552993	0.994389	0.971292	0.868255	0.930373	0.535046
Perm_db_6D_800	0.934126	0.927128	0.222235	0.981746	0.970239	0.115384	0.917911	0.000391
Perm_db_8D_800	0.799285	0.298146	-0.05237	0.96935	0.962961	-0.14887	0.901609	0.093098
Perm_db_10D_800	0.724317	0.301476	-0.24882	0.96573	0.94425	-0.19475	0.880956	0.056266
Shekel_4D_800	0.475095	0.498823	0.684792	0.673321	0.581998	0.665099	0.219408	-1.5
Colville_4D_800	0.955943	0.9991	0.85155	0.999837	0.998253	0.991851	0.919449	0.906764

Powell_4D_800	0.936447	0.963273	0.972347	0.999556	0.980454	0.987691	0.884117	0.873389
Powell_8D_800	0.932427	0.833492	0.489379	0.991214	0.969286	0.679701	0.53831	0.87178
Hartmann_6D_800	-1.5	0.594109	0.545069	0.950477	0.587196	0.843477	0.156089	0.741242
Ackley_2D_800	0.510976	0.810404	0.886919	0.907675	0.524786	0.880991	0.781902	0.781825
Ackley_4D_800	0.503167	0.49379	0.65753	0.666274	0.011839	0.671646	0.366706	0.647319
Ackley_6D_800	0.43699	0.125312	0.333463	-1.09956	0.098607	0.347601	-0.17981	0.208862
Ackley_8D_800	-0.44824	0.042349	-1.2773	-1.5	-0.1231	-0.25906	-0.77608	-0.56006
Ackley_10D_800	-0.77566	-0.05559	-1.5	-1.5	-0.53844	-0.18848	-1.18839	-1.5
Griewank_2D_800	0.999825	0.999486	0.999684	0.99981	0.999819	0.998891	0.984598	0.987063
Griewank_4D_800	0.999978	0.992434	0.998005	0.999968	0.999976	0.996316	0.754225	0.977581
Griewank_6D_800	0.999996	0.985688	0.778379	0.999988	0.999656	0.990015	0.415838	0.976309
Griewank_8D_800	0.999999	0.983252	0.546786	0.999986	0.999643	0.919572	0.161867	0.963344
Griewank_10D_800	1	0.950707	0.159794	0.999978	0.999738	0.804125	-0.08059	0.951326
Schwefel_2D_800	0.184496	0.101924	0.176116	0.99661	0.989143	0.804108	0.781915	-1.5
Schwefel_4D_800	0.168521	0.102634	0.09625	0.248259	0.975974	-0.29233	0.170337	-1.5
Schwefel_6D_800	0.153837	0.082935	0.044283	0.085539	0.92391	-0.36446	-0.21597	-1.23252
Schwefel_8D_800	0.125663	0.083474	0.023397	-0.0716	0.827114	-0.44375	-0.48915	-1.5
Schwefel_10D_800	0.131119	0.06051	0.021626	-0.17812	0.618073	-0.34306	-0.68799	-1.5

Rastrigin_2D_800	0.510141	0.504614	0.505986	0.988263	0.981624	0.420917	0.521144	-0.99891
Rastrigin_4D_800	0.499892	0.499229	0.443242	0.345332	0.796458	0.232388	-0.09929	-0.2834
Rastrigin_6D_800	0.463228	0.447673	0.287806	0.172312	0.625954	0.133296	-0.40384	-0.16118
Rastrigin_8D_800	0.433782	0.167482	0.163565	0.065044	0.475903	0.157824	-0.55321	-0.07933
Rastrigin_10D_800	0.42862	0.122554	-0.03891	-0.03079	0.287762	0.110569	-0.69227	-0.06952
Levy_2D_800	0.712924	0.608963	0.686929	0.995738	0.994525	0.884631	0.972514	0.177774
Levy_4D_800	0.69841	0.576574	0.547446	0.733393	0.989743	0.417142	0.704943	0.037263
Levy_6D_800	0.682548	0.539384	0.377755	0.436707	0.985339	0.326949	0.387981	0.043375
Levy_8D_800	0.661423	0.49389	0.250213	0.421938	0.980663	0.319752	0.108836	-0.00803
Levy_10D_800	0.645834	0.044315	0.116493	0.399691	0.957476	0.322412	-0.14492	-0.06978
Cross_IT_2D_800	0.432157	0.421847	0.414561	0.359432	0.919904	0.318044	0.538974	-0.35785
Drop_Wave_2D_800	0.338852	0.314596	0.341525	-0.23341	0.072714	0.113093	-0.80949	-0.538
Eggholder_2D_800	-0.01377	-0.0094	-0.01471	0.885086	-0.06601	0.228336	-0.24962	-1.5
Holder_2D_800	0.495524	0.388077	0.426306	0.941644	0.805238	0.501174	0.590302	-0.27848
Sphere_15D_800	1	0.657866	-0.53982	0.999907	0.99246	0.305597	-0.43793	0.912988
Sphere_20D_800	1	-0.48043	-0.14448	0.999458	0.982466	-0.12448	-0.79721	0.803449
Sum of Squares_15D_800	1	0.673111	-0.21944	0.995407	0.993411	0.363532	-0.12227	0.919229
Sum of Squares_20D_800	1	-0.26976	-0.08094	0.987058	0.982568	-0.07073	-0.39439	0.826161

Sum of Different Powers_15D_800	-124.562	0.07512	-0.00308	0.535689	0.983834	0.368267	-0.32876	0.426821
Sum of Different Powers_20D_800	-7.72661	-1.06627	-0.07526	0.476968	0.972813	-0.10266	-0.57496	0.050868
Trid_15D_800	1	0.770758	0.037977	0.999922	0.978469	0.455395	-0.65442	0.910437
Trid_20D_800	1	0.397929	-0.12871	0.999611	0.96426	-0.07005	-0.98298	0.846118
Zakharov_15D_800	0.914532	0.952146	0.270889	0.956584	0.834539	-0.0637	-0.2389	0.443411
Zakharov_20D_800	0.905478	0.875101	-0.76126	0.913099	0.747555	-0.2836	-0.4285	-0.00685
StyblinskiTang_15D_800	0.987926	-0.22326	0.012594	0.176881	0.968053	-8.24043	-0.38653	-0.65058
StyblinskiTang_20D_800	0.990988	-1.24321	-0.02038	0.115512	0.949897	-11.4952	-0.61884	-0.3529
Ackley_15D_800	-0.43246	-0.36789	-7.80535	0.368023	-2.30029	-0.40615	-1.96011	-3.94765
Ackley_20D_800	-1.6957	-1.1361	-23.2863	-0.04556	-61152.4	0.23526	-4.65759	-10.7803
Levy_15D_800	0.613834	-0.27823	0.004534	0.344671	0.931967	0.20466	-0.56141	-0.20426
Levy_20D_800	0.582987	-1.20076	-0.03931	0.284471	0.625852	-0.06963	-0.87478	-0.17611
Griewank_15D_800	1	0.599557	-0.11264	0.999908	0.999208	0.300506	-0.43802	0.912674
Griewank_20D_800	1	-0.47896	-0.14515	0.999462	0.961822	-0.12427	-0.79258	0.803182
Schwefel_15D_800	0.081031	-0.64209	-0.18778	-0.29424	0.394892	-0.54571	-0.96847	-1.25502
Schwefel_20D_800	0.057362	-1.58782	-0.34063	-0.39696	-0.20466	-0.5454	-1.13768	-1.1366

Rastrigin_15D_800	0.344936	-0.45873	-0.27062	0.011878	0.264482	-0.2408	-0.98123	-0.13185
Rastrigin_20D_800	0.288826	-1.94727	-0.56312	-0.24276	-10658.4	-0.38467	-1.14618	-0.03046
Ellipsoid_15D_800	1	0.776812	-0.04454	0.995215	0.995794	0.331884	-0.09436	0.924636
Ellipsoid_20D_800	1	0.017878	-0.05659	0.993144	0.985905	-0.0623	-0.40292	0.845662
Rosenbrock_15D_800	0.994055	0.57657	0.440346	0.843398	0.978947	0.453325	0.236019	0.694309
Rosenbrock_20D_800	0.992139	-0.06995	0.305331	0.736882	0.964536	-0.04732	-0.027	0.653843
Dixon_Price_15D_800	0.957524	0.518144	0.029872	0.720863	0.984069	0.437569	-0.06128	0.833311
Dixon_Price_20D_800	0.945513	-0.53934	-0.06158	0.801769	0.98008	-0.0302	-0.29528	0.743968
Sphere_2D_1200	1	0.999804	0.999868	0.999986	1	0.99968	0.991896	0.983809
Sphere_4D_1200	1	0.995465	0.997863	0.999991	1	0.996857	0.789296	0.980703
Sphere_6D_1200	1	0.98862	0.957189	0.999993	1	0.992716	0.512863	0.978068
Sphere_8D_1200	1	0.983091	0.862573	0.999991	1	0.963926	0.250036	0.968389
Sphere_10D_1200	1	0.98138	0.166268	0.999987	1	0.86804	0.034364	0.961394
Ellipsoid_2D_1200	1	0.999379	0.999028	0.999987	1	0.998769	0.992952	0.976503
Ellipsoid_4D_1200	1	0.995257	0.998358	0.999991	1	0.99657	0.853445	0.975336
Ellipsoid_6D_1200	1	0.992173	0.762677	0.999993	0.999259	0.984398	0.674587	0.974657
Ellipsoid_8D_1200	1	0.98403	0.525662	0.99999	0.998998	0.920681	0.49189	0.969739
Ellipsoid_10D_1200	1	0.983116	0.243274	0.999982	0.998367	0.843771	0.323332	0.963591

Sum of Different Powers_2D_1200	0.951259	0.999754	0.999496	0.999955	0.999057	0.998504	0.991026	0.977813
Sum of Different Powers_4D_1200	-1.5	0.902279	0.954952	0.999354	0.999007	0.9957	0.806664	0.948255
Sum of Different Powers_6D_1200	-1.5	0.853277	0.694969	0.99425	0.996699	0.933423	0.54743	0.858081
Sum of Different Powers_8D_1200	-1.5	0.787268	0.414135	0.968456	0.994856	0.748471	0.309538	0.789219
Sum of Different Powers_10D_1200	-1.5	0.412786	0.195396	0.674248	0.992483	0.647988	0.12127	0.69653
Sum of Squares_2D_1200	1	0.999535	0.99985	0.999987	1	0.998769	0.992826	0.976493
Sum of Squares_4D_1200	1	0.997624	0.980627	0.999991	0.998844	0.996999	0.852476	0.969509
Sum of Squares_6D_1200	1	0.992021	0.779839	0.999993	0.999248	0.984304	0.662688	0.975558
Sum of Squares_8D_1200	1	0.986527	0.542723	0.999989	0.999077	0.931941	0.480943	0.968502
Sum of Squares_10D_1200	1	0.984888	0.23892	0.999982	0.998214	0.850591	0.289929	0.963546
Trid_2D_1200	1	0.998956	0.999328	0.999995	1	0.999506	0.99547	0.964665
Trid_4D_1200	1	0.993322	0.99711	0.999996	1	0.994876	0.816749	0.960935

Trid_6D_1200	1	0.991274	0.870457	0.999996	1	0.986977	0.473246	0.96949
Trid_8D_1200	1	0.984209	0.700002	0.999994	1	0.940813	0.172686	0.957454
Trid_10D_1200	1	0.963588	0.422292	0.99999	0.995352	0.881274	-0.06497	0.955147
Perm_2D_1200	0.999979	0.99973	0.999063	0.999998	1	0.999692	0.997454	0.964384
Perm_4D_1200	0.999243	0.988007	0.994869	0.097144	0.996865	0.994187	0.827253	0.998685
Perm_6D_1200	0.999792	-1.5	0.991689	-1.5	0.996703	0.982443	0.259953	0.991509
Perm_8D_1200	0.999902	-1.5	0.888386	-1.5	0.99738	0.86649	-0.25778	0.95744
Perm_10D_1200	0.999951	-1.5	0.605135	-1.5	0.978555	0.579159	-0.66151	0.883825
Bohachevsky_2D_1200	1	0.999535	0.99985	0.999987	1	0.998769	0.9928	0.976638
Dixon_Price_2D_1200	0.967951	0.999603	0.999566	0.99999	0.999112	0.999232	0.997864	0.974404
Dixon_Price_4D_1200	0.967535	0.993659	0.877039	0.999853	0.996448	0.995561	0.902566	0.933281
Dixon_Price_6D_1200	0.965313	0.861831	0.683022	0.995515	0.992993	0.927625	0.731914	0.875586
Dixon_Price_8D_1200	0.965089	0.844022	0.41486	0.985431	0.993031	0.787445	0.55962	0.879944
Dixon_Price_10D_1200	0.965538	0.817671	0.263543	0.972853	0.989621	0.742433	0.370267	0.875579
Rosenbrock_2D_1200	0.995513	0.998716	0.999629	0.999982	0.998522	0.999068	0.998507	0.969877
Rosenbrock_4D_1200	0.995018	0.997891	0.965676	0.999915	0.99856	0.995962	0.965422	0.955415
Rosenbrock_6D_1200	0.994695	0.977898	0.803876	0.999415	0.995348	0.964495	0.890636	0.926122
Rosenbrock_8D_1200	0.994611	0.716826	0.684812	0.99006	0.99426	0.885697	0.793471	0.870975

Rosenbrock_10D_1200	0.994176	0.900713	0.551694	0.982928	0.989478	0.764321	0.660186	0.808826
3Hump_2D_1200	0.992013	0.999316	0.999123	0.999983	0.998954	0.998996	0.999113	0.929183
6Hump_2D_1200	0.946035	0.756748	0.99966	0.999941	0.998586	0.997728	0.973184	0.903182
Booth_2D_1200	1	0.998117	0.999661	0.999997	1	0.99848	0.99592	0.949864
Matyas_2D_1200	1	0.998422	0.999217	0.999993	1	0.99844	0.994777	0.972705
McCormick_2D_1200	0.994717	0.998643	0.999353	0.999966	0.993983	0.998596	0.995425	0.952243
Zakharov_2D_1200	0.980713	0.996609	0.997862	0.999955	0.9932	0.997275	0.994144	0.920567
Zakharov_4D_1200	0.9378	0.997344	0.950434	0.999292	0.934034	0.989806	0.872663	0.886163
Zakharov_6D_1200	0.919756	0.998479	0.972684	0.996705	0.904299	0.96092	0.64586	0.799109
Zakharov_8D_1200	0.9102	0.989169	0.710278	0.989787	0.897625	0.816559	0.347985	0.703315
Zakharov_10D_1200	0.913455	0.997323	0.556858	0.981744	0.877402	0.437224	0.262649	0.389651
PowerSum_4D_1200	0.958115	0.955246	0.944064	0.995228	0.97949	0.981907	0.79577	0.85777
Beale_2D_1200	0.923556	0.833494	0.998552	0.999631	0.845799	0.997399	0.95272	0.85231
Branin_2D_1200	0.963631	0.996936	0.998554	0.99996	0.995337	0.997825	0.990577	-1.5
GoldsteinPrice_2D_1200	0.792883	0.970379	0.997847	0.999663	0.971153	0.996823	0.985152	0.960373
StyblinskiTang_2D_1200	0.993619	0.546254	0.998436	0.999897	0.998909	0.51137	0.960765	0.251555
StyblinskiTang_4D_1200	0.989871	0.397985	0.980386	0.998755	0.996516	0.991143	0.681022	0.795022
StyblinskiTang_6D_1200	0.988307	0.369957	0.277	0.996554	0.996038	0.730671	0.454327	0.093897

StyblinskiTang_8D_1200	0.986517	0.123046	0.190168	0.458691	0.984497	0.179468	0.228373	-0.45461
StyblinskiTang_10D_1200	0.991038	0.068462	0.072886	0.34547	0.978826	0.121122	0.055063	-0.5934
Perm_db_2D_1200	0.997656	0.989446	0.998508	0.99993	0.97425	0.997834	0.995487	0.963298
Perm_db_4D_1200	0.962984	0.962187	0.616974	0.995703	0.974212	0.959884	0.938935	0.629667
Perm_db_6D_1200	0.936889	0.92381	0.3082	0.988299	0.973339	0.378165	0.927315	0.149442
Perm_db_8D_1200	0.804331	0.658164	0.028068	0.971824	0.968977	-0.11364	0.914694	0.109094
Perm_db_10D_1200	0.730561	0.278253	-0.20555	0.968145	0.963613	-0.17891	0.901955	0.073579
Shekel_4D_1200	0.480465	0.468386	0.617649	0.698154	0.63936	0.678746	0.280929	-1.5
Colville_4D_1200	0.956407	0.999149	0.883472	0.999873	0.998195	0.995156	0.940176	0.93016
Powell_4D_1200	0.937058	0.975282	0.967716	0.999744	0.906455	0.991931	0.927013	0.906958
Powell_8D_1200	0.934814	0.885134	0.50485	0.996247	0.93189	0.755471	0.595892	0.874092
Hartmann_6D_1200	-1.5	0.553477	0.486336	0.968029	0.64439	0.901417	0.393026	0.781507
Ackley_2D_1200	0.517633	0.875124	0.906587	0.923063	0.670533	0.887768	0.813705	0.791955
Ackley_4D_1200	0.574568	0.502061	0.610009	0.767003	0.390493	0.696559	0.450808	0.666885
Ackley_6D_1200	0.610921	0.463932	0.482279	0.629409	0.288344	0.485581	-0.07236	0.23012
Ackley_8D_1200	0.540206	0.024194	0.116668	0.518813	-0.28374	0.009503	-0.55338	-0.52077
Ackley_10D_1200	-0.2229	0.105279	-1.5	0.257123	-0.89528	-0.10203	-0.8652	-1.39035
Griewank_2D_1200	0.999825	0.999622	0.999689	0.999812	0.999821	0.999504	0.991262	0.98708

Griewank_4D_1200	0.999978	0.99287	0.998363	0.999969	0.999977	0.996601	0.790875	0.980748
Griewank_6D_1200	0.999996	0.988686	0.839765	0.99999	0.999996	0.992739	0.515443	0.978203
Griewank_8D_1200	0.999999	0.985109	0.575737	0.999991	0.999767	0.963938	0.245158	0.96822
Griewank_10D_1200	1	0.741827	0.116949	0.999988	1	0.862525	0.032799	0.96129
Schwefel_2D_1200	0.186711	0.103004	0.184327	0.998469	0.993704	0.771986	0.843496	-1.5
Schwefel_4D_1200	0.176148	0.10225	0.092047	0.286497	0.984236	-0.28688	0.250282	-1.48306
Schwefel_6D_1200	0.171342	0.098274	0.070012	0.104585	0.976188	-0.3935	-0.14133	-1.16865
Schwefel_8D_1200	0.166721	0.093453	0.045699	-0.03381	0.937153	-0.39026	-0.39148	-1.15338
Schwefel_10D_1200	0.164728	0.095537	0.013334	-0.14341	0.874635	-0.34476	-0.631	-1.5
Rastrigin_2D_1200	0.510756	0.506906	0.508642	0.998013	0.984083	0.431793	0.596395	-0.9254
Rastrigin_4D_1200	0.503941	0.497342	0.460745	0.351639	0.923752	0.231356	-0.00816	-0.28442
Rastrigin_6D_1200	0.478979	0.461771	0.359338	0.139169	0.694134	0.123802	-0.34836	-0.16263
Rastrigin_8D_1200	0.458138	0.174257	0.223343	0.13432	0.58387	0.159451	-0.49805	-0.04845
Rastrigin_10D_1200	0.439601	0.046008	-0.0133	0.045525	0.530829	0.130357	-0.63689	-0.07633
Levy_2D_1200	0.715123	0.61099	0.690448	0.995814	0.994877	0.887795	0.97932	0.197529
Levy_4D_1200	0.701008	0.572236	0.547986	0.719131	0.991957	0.47204	0.755537	0.068325
Levy_6D_1200	0.696474	0.550328	0.444852	0.524255	0.987626	0.349088	0.448391	0.054558
Levy_8D_1200	0.68212	0.245934	0.323118	0.431335	0.983495	0.331098	0.200383	0.04975

Levy_10D_1200	0.673929	0.479089	0.119744	0.436947	0.980303	0.342399	-0.0451	0.000574
Cross_IT_2D_1200	0.431432	0.412344	0.41881	0.372231	0.948717	0.344142	0.638731	-0.29203
Drop_Wave_2D_1200	0.343684	0.318479	0.348519	0.132209	0.207065	0.151096	-0.67213	-0.558
Eggholder_2D_1200	-0.00538	-0.00557	0.037781	0.885452	0.044063	0.244573	0.152157	-1.5
Holder_2D_1200	0.501875	0.39992	0.528125	0.97222	0.7588	0.493249	0.694686	-0.30239
Sphere_15D_1200	1	0.855494	0.137155	0.999954	1	0.460865	-0.34434	0.933559
Sphere_20D_1200	1	0.199602	-0.07322	0.999822	0.998696	0.084377	-0.64844	0.871431
Sum of Squares_15D_1200	1	0.915202	0.099506	0.999099	0.993978	0.491611	-0.01628	0.932133
Sum of Squares_20D_1200	1	0.809993	-0.04535	0.994572	0.993782	0.095875	-0.29204	0.886374
Sum of Different Powers_15D_1200	-14.5261	0.426108	0.062873	0.561389	0.986664	0.478143	-0.22905	0.466372
Sum of Different Powers_20D_1200	-14.5264	-0.22127	2.1E-05	0.496044	0.980595	0.104495	-0.46923	0.142235
Trid_15D_1200	1	0.80782	0.113752	0.999961	0.978234	0.584562	-0.58254	0.92831
Trid_20D_1200	1	0.725437	-0.07804	0.999885	0.974946	0.129998	-0.88571	0.899299
Zakharov_15D_1200	0.919822	0.96337	0.372835	0.971757	0.80155	0.103194	-0.14881	0.558712

Zakharov_20D_1200	0.913582	0.944549	-0.64348	0.936211	0.819971	-0.12497	-0.32238	0.147867
StyblinskiTang_15D_1200	0.990141	-0.01139	0.018484	0.222833	0.972978	-7.23099	-0.28803	-0.6335
StyblinskiTang_20D_1200	0.989557	-0.36905	0.01962	0.185759	0.971091	-10.0815	-0.52113	-0.40566
Ackley_15D_1200	0.317216	-0.40603	-6.0567	0.390101	-1.0037	0.007104	-2.34177	-3.57873
Ackley_20D_1200	-0.16331	-0.63947	-27.0515	-0.02801	-1.75898	0.364076	-2.75677	-9.85332
Levy_15D_1200	0.643462	0.097711	-0.0366	0.365	0.959871	0.318104	-0.4562	-0.23658
Levy_20D_1200	0.621659	-0.20776	-0.0166	0.32143	0.927098	0.096234	-0.75665	-0.13698
Griewank_15D_1200	1	0.855778	0.136854	0.999955	0.999157	0.465879	-0.34658	0.933584
Griewank_20D_1200	1	0.260972	-0.15638	0.999823	0.9868	0.07581	-0.64955	0.871268
Schwefel_15D_1200	0.12387	-0.34612	-0.11854	-0.27653	0.818164	-0.30441	-0.87241	-1.26346
Schwefel_20D_1200	0.084842	-0.739	-0.25684	-0.34648	0.173521	-0.28225	-1.01248	-1.15556
Rastrigin_15D_1200	0.400583	-0.0863	-0.18453	0.055653	-0.00479	-0.00963	-0.86386	-0.18179
Rastrigin_20D_1200	0.367451	-0.53631	-0.40856	0.101329	0.061813	-0.12125	-1.02063	-0.03264
Ellipsoid_15D_1200	1	0.961096	0.08432	0.999084	0.993827	0.489973	-0.00868	0.9405
Ellipsoid_20D_1200	1	0.495207	-0.03139	0.997564	0.993861	0.10559	-0.2552	0.896431
Rosenbrock_15D_1200	0.994454	0.768262	0.44624	0.913402	0.979099	0.569604	0.343784	0.728412
Rosenbrock_20D_1200	0.994096	0.35857	0.454618	0.815169	0.972966	0.171239	0.059372	0.718859
Dixon_Price_15D_1200	0.960611	0.734953	0.066343	0.906992	0.987618	0.546781	0.068516	0.855328

Dixon_Price_20D_1200	0.955546	0.354477	0.004097	0.812559	0.98259	0.168942	-0.20382	0.806583
Sphere_2D_1600	1	0.999828	0.999868	0.999987	1	0.999686	0.994559	0.983819
Sphere_4D_1600	1	0.997931	0.922767	0.999992	1	0.997218	0.821381	0.981974
Sphere_6D_1600	1	0.982198	0.845435	0.999994	1	0.993855	0.567365	0.97938
Sphere_8D_1600	1	0.977923	0.587821	0.999993	1	0.974199	0.307969	0.972291
Sphere_10D_1600	1	0.62307	0.184512	0.999991	1	0.905131	0.108962	0.965735
Ellipsoid_2D_1600	1	0.999692	0.999032	0.999988	1	0.998771	0.99522	0.976265
Ellipsoid_4D_1600	1	0.998371	0.974529	0.999992	1	0.997159	0.873116	0.97462
Ellipsoid_6D_1600	1	0.994919	0.765862	0.999994	0.999271	0.988425	0.709443	0.973956
Ellipsoid_8D_1600	1	0.979219	0.599677	0.999992	0.999022	0.941663	0.539774	0.973505
Ellipsoid_10D_1600	1	0.976207	0.269139	0.999987	0.998326	0.862591	0.383907	0.967958
Sum of Different Powers_2D_1600	0.9513	0.999779	0.9995	0.999965	0.999072	0.998576	0.993693	0.977268
Sum of Different Powers_4D_1600	-1.5	0.922374	0.932503	0.99939	0.999136	0.995956	0.839934	0.954576
Sum of Different Powers_6D_1600	-1.5	0.845503	0.700438	0.994947	0.998186	0.951579	0.611094	0.875177

Sum of Different Powers_8D_1600	-1.5	0.784877	0.472899	0.974915	0.996086	0.768785	0.379599	0.787604
Sum of Different Powers_10D_1600	-1.5	0.726781	0.25746	0.857156	0.993469	0.67081	0.199673	0.704517
Sum of Squares_2D_1600	1	0.999565	0.99985	0.999988	1	0.998771	0.995481	0.976348
Sum of Squares_4D_1600	1	0.996087	0.998112	0.999992	1	0.997292	0.872337	0.974323
Sum of Squares_6D_1600	1	0.9969	0.784233	0.999994	0.999216	0.989284	0.705925	0.976262
Sum of Squares_8D_1600	1	0.98498	0.62337	0.999992	0.999171	0.945974	0.496388	0.969937
Sum of Squares_10D_1600	1	0.981027	0.257619	0.999988	0.9983	0.879641	0.361293	0.966506
Trid_2D_1600	1	0.999064	0.999331	0.999995	1	0.999507	0.99697	0.965344
Trid_4D_1600	1	0.996125	0.997263	0.999997	1	0.995463	0.85134	0.95982
Trid_6D_1600	1	0.991991	0.872794	0.999997	1	0.988911	0.524246	0.970444
Trid_8D_1600	1	0.976739	0.674621	0.999996	1	0.959344	0.223598	0.962085
Trid_10D_1600	1	0.978083	0.532841	0.999993	0.999884	0.897638	0.017934	0.957893
Perm_2D_1600	0.999979	0.43648	0.99986	0.999998	0.998841	0.999647	0.998179	0.986919
Perm_4D_1600	0.999248	-0.56481	0.996957	0.117186	0.99653	0.994921	0.854886	0.998413
Perm_6D_1600	0.999848	-0.55488	0.9815	-1.5	0.996231	0.985217	0.320285	0.972447

Perm_8D_1600	0.999903	-0.54437	0.934985	-1.5	0.996802	0.898284	-0.20284	0.844817
Perm_10D_1600	0.99995	-0.53766	0.609655	-1.5	0.987473	0.643735	-0.50347	0.270371
Bohachevsky_2D_1600	1	0.999565	0.99985	0.999988	1	0.998771	0.995445	0.976207
Dixon_Price_2D_1600	0.968049	0.999606	0.99957	0.999994	0.999263	0.999701	0.998575	0.976717
Dixon_Price_4D_1600	0.967643	0.997098	0.940762	0.999877	0.99544	0.995916	0.92346	0.943459
Dixon_Price_6D_1600	0.966214	0.918917	0.687876	0.999505	0.993812	0.954047	0.760728	0.875671
Dixon_Price_8D_1600	0.96669	0.83528	0.499048	0.992997	0.99439	0.804635	0.594675	0.879581
Dixon_Price_10D_1600	0.966456	0.840258	0.325223	0.983548	0.994273	0.762298	0.452424	0.87943
Rosenbrock_2D_1600	0.995522	0.998594	0.999631	0.999986	0.998493	0.998932	0.998866	0.970229
Rosenbrock_4D_1600	0.995043	0.996116	0.997547	0.999939	0.99859	0.996235	0.973441	0.959506
Rosenbrock_6D_1600	0.994871	0.934071	0.806998	0.999684	0.995513	0.978193	0.910565	0.938707
Rosenbrock_8D_1600	0.994741	0.960944	0.693027	0.994194	0.995276	0.907133	0.82646	0.890408
Rosenbrock_10D_1600	0.994473	0.919337	0.596711	0.984259	0.993329	0.802786	0.697253	0.844418
3Hump_2D_1600	0.992259	0.999349	0.99914	0.999989	0.998974	0.999041	0.999304	0.939343
6Hump_2D_1600	0.946501	0.993993	0.997881	0.999949	0.998577	0.997703	0.979618	0.913514
Booth_2D_1600	1	0.9997	0.999663	0.999998	1	0.998846	0.997241	0.950234
Matyas_2D_1600	1	0.998436	0.999221	0.999993	1	0.998429	0.996272	0.972716
McCormick_2D_1600	0.994725	0.999332	0.999355	0.999975	0.99403	0.998045	0.996977	0.954171

Zakharov_2D_1600	0.980755	0.998827	0.99787	0.999966	0.993992	0.996832	0.995203	0.920387
Zakharov_4D_1600	0.93835	0.997566	0.993911	0.999635	0.934548	0.990842	0.891078	0.897161
Zakharov_6D_1600	0.920823	0.996505	0.94864	0.997877	0.915924	0.970389	0.695441	0.81893
Zakharov_8D_1600	0.913674	0.990619	0.724309	0.992372	0.89996	0.861584	0.450044	0.733163
Zakharov_10D_1600	0.915704	0.997007	0.209359	0.985178	0.700378	0.6006	0.336001	0.446459
PowerSum_4D_1600	0.981346	0.750991	0.926173	0.996565	0.974082	0.985866	0.848407	0.887386
Beale_2D_1600	0.916901	0.494464	0.998553	0.999677	0.847354	0.997279	0.960734	0.863616
Branin_2D_1600	0.962439	0.998116	0.998565	0.999969	0.992994	0.998784	0.993549	-1.5
GoldsteinPrice_2D_1600	0.793274	0.974704	0.997876	0.999756	0.973306	0.996041	0.991387	0.963329
StyblinskiTang_2D_1600	0.993611	0.997976	0.998454	0.99994	0.998124	0.998048	0.96811	0.004283
StyblinskiTang_4D_1600	0.99292	0.381793	0.995988	0.998914	0.997632	0.992376	0.733756	0.839561
StyblinskiTang_6D_1600	0.987111	0.379544	0.230659	0.997452	0.994942	0.829417	0.502352	0.291248
StyblinskiTang_8D_1600	0.99132	0.159382	0.135524	0.981831	0.983618	0.210068	0.299261	-0.47286
StyblinskiTang_10D_1600	0.98966	0.091858	0.091378	0.355035	0.979663	0.137505	0.135526	-0.52983
Perm_db_2D_1600	0.994346	0.990804	0.998513	0.999935	0.974976	0.997934	0.997029	0.963539
Perm_db_4D_1600	0.976529	0.963924	0.576453	0.99763	0.973071	0.977747	0.952705	0.683085
Perm_db_6D_1600	0.937524	0.951428	0.340628	0.990112	0.973433	0.500445	0.935321	0.135585
Perm_db_8D_1600	0.805749	0.78722	0.106371	0.974433	0.970717	-0.09388	0.920898	0.11344

Perm_db_10D_1600	0.733381	0.292838	0.041721	0.966602	0.968385	-0.15861	0.906443	0.086711
Shekel_4D_1600	0.493072	0.510565	0.551382	0.71797	0.61829	0.685652	0.344164	-1.5
Colville_4D_1600	0.956388	0.865729	0.9549	0.999896	0.998575	0.996325	0.951837	0.942249
Powell_4D_1600	0.938107	0.968921	0.79249	0.999804	0.949139	0.992822	0.938953	0.914972
Powell_8D_1600	0.935955	0.837452	0.525128	0.997674	0.931742	0.815596	0.629576	0.874842
Hartmann_6D_1600	-1.5	0.619711	0.543734	0.973456	0.651593	0.914878	0.462117	0.811724
Ackley_2D_1600	0.518037	0.833376	0.897858	0.936409	0.555232	0.894789	0.854412	0.799454
Ackley_4D_1600	0.55065	0.490701	0.694313	0.796112	0.391323	0.697587	0.48952	0.685957
Ackley_6D_1600	0.644019	0.470064	0.448429	0.648907	-0.04238	0.519597	0.03528	0.283452
Ackley_8D_1600	0.58385	0.057882	-0.54151	0.493618	0.13989	0.203868	-0.36781	-0.43735
Ackley_10D_1600	-0.62817	-0.01582	-1.5	0.237318	-0.02828	-0.05504	-0.70117	-1.5
Griewank_2D_1600	0.999826	0.999649	0.99969	0.999813	0.999823	0.999511	0.993868	0.987088
Griewank_4D_1600	0.999978	0.997877	0.998443	0.99997	0.999977	0.997197	0.822975	0.983556
Griewank_6D_1600	0.999996	0.987767	0.872595	0.99999	0.999996	0.994125	0.56256	0.979573
Griewank_8D_1600	0.999999	0.987894	0.760094	0.999993	0.999999	0.973121	0.308186	0.972273
Griewank_10D_1600	1	0.264357	0.191931	0.999991	1	0.909567	0.098674	0.965722
Schwefel_2D_1600	0.187565	0.104978	0.187383	0.998864	0.991848	0.79858	0.871999	-1.5
Schwefel_4D_1600	0.18109	0.124823	0.136294	0.294608	0.987296	-0.29543	0.303803	-1.42619

Schwefel_6D_1600	0.181784	0.097304	0.082097	0.154571	0.975387	-0.37517	-0.06622	-1.04575
Schwefel_8D_1600	0.171062	0.095345	0.059787	-0.04794	0.965929	-0.41774	-0.32048	-1.00114
Schwefel_10D_1600	0.175609	0.096785	0.054854	-0.1324	0.948622	-0.35813	-0.55673	-1.49876
Rastrigin_2D_1600	0.511063	0.507791	0.510814	0.999002	0.979163	0.434392	0.60158	-0.82894
Rastrigin_4D_1600	0.517995	0.505014	0.480923	0.356363	0.963429	0.266445	0.057881	-0.27123
Rastrigin_6D_1600	0.488256	0.472387	0.372965	0.190648	0.802082	0.108524	-0.27896	-0.15136
Rastrigin_8D_1600	0.464325	0.180177	0.24416	0.153086	0.701661	0.163321	-0.46584	-0.0367
Rastrigin_10D_1600	0.461392	0.103407	-0.00772	0.045149	0.628042	0.149596	-0.60067	-0.04954
Levy_2D_1600	0.715903	0.62385	0.904335	0.99586	0.995375	0.8904	0.9843	0.308814
Levy_4D_1600	0.70349	0.579495	0.572019	0.931599	0.993511	0.597885	0.779861	0.15483
Levy_6D_1600	0.699211	0.554867	0.413489	0.556914	0.987081	0.33627	0.512224	0.044948
Levy_8D_1600	0.696073	0.542386	0.296813	0.491928	0.981225	0.321157	0.262914	0.048386
Levy_10D_1600	0.689446	0.158008	0.158055	0.44029	0.981037	0.359537	0.03735	0.024425
Cross_IT_2D_1600	0.432628	0.42928	0.420614	0.72311	0.963486	0.344913	0.679322	-0.30257
Drop_Wave_2D_1600	0.357069	0.360002	0.362662	0.823681	0.289076	0.201776	-0.45744	-0.48614
Eggholder_2D_1600	-0.00428	-0.00165	0.024226	0.930592	0.126356	0.281104	0.310172	-1.5
Holder_2D_1600	0.502776	0.392856	0.461777	0.97788	0.792675	0.504639	0.743412	-0.02665
Sphere_15D_1600	1	0.901505	0.20175	0.999969	1	0.559725	-0.28168	0.943338

Sphere_20D_1600	1	0.576528	-0.05209	0.999901	0.999906	0.186835	-0.54515	0.893752
Sum of Squares_15D_1600	1	0.923974	0.16058	0.999121	0.994226	0.562176	0.041534	0.943499
Sum of Squares_20D_1600	1	0.864634	-0.03381	0.998027	0.991448	0.1898	-0.20152	0.905749
Sum of Different Powers_15D_1600	-22.2867	0.532569	0.074018	0.581916	0.98842	0.530873	-0.18027	0.484168
Sum of Different Powers_20D_1600	-0.20763	0.095108	0.021095	0.531611	0.983104	0.197582	-0.40544	0.183406
Trid_15D_1600	1	0.917803	0.162417	0.999974	0.978065	0.641281	-0.57338	0.937721
Trid_20D_1600	1	0.823767	0.011106	0.999935	0.975215	0.223823	-0.82066	0.9183
Zakharov_15D_1600	0.923043	0.959522	0.490517	0.980954	0.850865	0.209839	-0.02704	0.623248
Zakharov_20D_1600	0.932479	0.964112	-0.30988	0.949076	0.84579	-0.05404	-0.20122	0.171923
StyblinskiTang_15D_1600	0.989944	0.03297	0.046734	0.245061	0.977066	-6.79218	-0.19647	-0.57231
StyblinskiTang_20D_1600	0.991372	-0.2237	0.003281	0.195755	0.970357	-9.46084	-0.45652	-0.35589
Ackley_15D_1600	0.466418	-0.15219	-5.4967	0.38639	-2.25907	0.124186	-1.66472	-3.3722
Ackley_20D_1600	0.296027	-0.4963	-18.9065	-0.01684	-0.9248	0.412572	-1.97712	-9.1835
Levy_15D_1600	0.657746	0.218874	0.050544	0.377196	0.968344	0.366444	-0.3843	-0.2493

Levy_20D_1600	0.638234	-0.15514	0.022997	0.344354	0.956147	0.178694	-0.65011	-0.13486
Griewank_15D_1600	1	0.826225	-0.08963	0.99997	0.999867	0.555825	-0.28171	0.943276
Griewank_20D_1600	1	0.5767	-0.06968	0.999901	0.989234	0.186971	-0.54701	0.89341
Schwefel_15D_1600	0.153291	-0.08762	-0.05846	-0.23824	0.896725	-0.20392	-0.79758	-1.34205
Schwefel_20D_1600	0.102122	-0.28392	-0.17068	-0.35337	0.559899	-0.17865	-0.97228	-1.17021
Rastrigin_15D_1600	0.422804	0.112539	-0.30215	0.07256	0.372763	0.113301	-0.81729	0.017335
Rastrigin_20D_1600	0.40394	-0.34557	-0.2763	0.129919	0.265411	-0.00175	-0.9605	-0.00222
Ellipsoid_15D_1600	1	0.933372	0.15557	0.999097	0.99489	0.547912	0.04905	0.949877
Ellipsoid_20D_1600	1	0.87625	-0.03465	0.997726	0.989704	0.19704	-0.19764	0.914573
Rosenbrock_15D_1600	0.994576	0.799546	0.557017	0.955473	0.977751	0.614582	0.389156	0.751673
Rosenbrock_20D_1600	0.994323	0.52986	0.436496	0.846009	0.972027	0.265218	0.121412	0.739971
Dixon_Price_15D_1600	0.963182	0.780149	0.101105	0.940525	0.989112	0.604457	0.13765	0.866524
Dixon_Price_20D_1600	0.959964	0.638323	0.042476	0.845173	0.983526	0.264793	-0.12912	0.843491

Appendix C – pyBOUND Test Problems

No.	Function Name	Dimensions	Bounds	Minimum Value	Minimum Location
1	Langermann	2	[0,10]	-5.1621259	[2.00299219, 1.006096]
2	Easom	N	[-2π, 2π]	-1	π
3	Bent_cigar	N	[-100, 100]	0	0
4	Alπne1	N	[-10, 10]	0	0
5	Alπne2	N	[0, 10]	2.808^D	7.917052698
6	Brown	N	[-1, 4]	0	0
7	Spring	N	[0, 10]	-1	5
8	Yang3	N	[-2π, 2π]	0	0
9	Sargan	N	[-100, 100]	0	0
10	Michalewicz	2	[0, π]	-1.8013	[2.20, 1.57]
11	Holzman	N	[-10, 10]	0	0
12	Saloman	N	[-100, 100]	0	0
13	Chichinadze	2	[-30, 30]	−42.9443870189909	[6.189866586965680, 0.5]
14	Leon	2	[-1.2, 1.2]	0	[1, 1]
15	Trig2	N	[-500, 500]	1	0.9
16	Mishra	3	[-10, 10]	0	[1, 2, 3]
17	Layeb02	N	[-10, 10]	0	1
18	Layeb01	N	[-100, 100]	0	1
19	Biggs3	3	[0, 20]	0	[1, 10, 5]
20	Biggs4	4	[0, 20]	0	[1, 10, 1, 5]
21	Biggs5	5	[0, 20]	0	[1, 10, 1, 5, 4]
22	Biggs6	6	[-20, 20]	0	[1, 10, 1, 5, 4, 3]
23	Chung_Reynolds	N	[-100, 100]	0	0
24	Cube	2	[-10, 10]	0	[1, 1]
25	El_Attar	2	[-500, 500]	1.712780355	[3.4091868222,-2.1714330361]
26	Hartmann3	3	[0, 1]	-3.862782	[0.114614, 0.555649, 0.852547]
27	Himmelblau	2	[-5, 5]	0	[3, 2]
28	Pathological	N	[-100, 100]	0	0
29	Periodic	2	[-10, 10]	0.9	[0, 0]
30	Quadratic	2	[-10, 10]	-3873.7243	[0.19388, 0.48513]