Optimization Letters
https://doi.org/10.1007/s11590-019-01428-7

ORIGINAL PAPER


# Machine learning-based surrogate modeling for data-driven optimization: a comparison of subset selection for regression techniques

**Sun Hye Kim**[1] · **Fani Boukouvala**[1]

Received: 26 July 2018 / Accepted: 18 April 2019


**Abstract**
Optimization of simulation-based or data-driven systems is a challenging task, which has attracted significant attention in the recent literature. A very efficient approach for optimizing systems without analytical expressions is through fitting surrogate models. Due to their increased flexibility, nonlinear interpolating functions, such as radial basis functions and Kriging, have been predominantly used as surrogates for data-driven optimization; however, these methods lead to complex nonconvex formulations. Alternatively, commonly used regression-based surrogates lead to simpler formulations, but they are less flexible and inaccurate if the form is not known a priori. In this work, we investigate the efficiency of subset selection regression techniques for developing surrogate functions that balance both accuracy and complexity. Subset selection creates sparse regression models by selecting only a subset of original features, which are linearly combined to generate a diverse set of surrogate models. Five different subset selection techniques are compared with commonly used nonlinear interpolating surrogate functions with respect to optimization solution accuracy, computation time, sampling requirements, and model sparsity. Our results indicate that subset selection-based regression functions exhibit promising performance when the dimensionality is low, while interpolation performs better for higher dimensional problems.

**Keywords** Machine Learning · Surrogate modeling · Black-box optimization · Data-driven optimization · Subset selection for regression

✉ Fani Boukouvala
fani.boukouvala@chbe.gatech.edu

[1] School of Chemical & Biomolecular Engineering, Georgia Institute of Technology, Atlanta, USA

Published online: 09 May 2019

## 1 Introduction

Many real-life engineering problems involve complex computer simulations, which allow us to obtain more accurate and high-fidelity information about complex, multi-scale, multiphase, and/or distributed systems. However, these often involve proprietary codes, if–then operators, or numerical integrators in order to describe phenomena that cannot be explicitly captured by physics-based algebraic equations [1]. Consequently, the algebraic model of the system and the derivatives are either absent or too complicated to obtain; thus, the system cannot be directly optimized using derivative-based optimization solvers. Such problems are known as "black-box" systems since the constraints and the objective of the problem cannot be obtained as closed-form equations [1–7]. In this work, we specifically address the following black-box problem (Problem 1), in which both the forms of objective $f(x)$ and constraints $g_c(x)$ are not known explicitly.

$$\min_x f(x),\ s.t. g_c(x) \leq 0\ \forall\ c \in \{1, \ldots, C\},\ x \in \mathbb{R}^M \qquad \text{(Problem 1)}$$

In Problem 1, $M$ represents the number of continuous variables with known bounds $\left[x_i^L, x_i^U\right]$, where $i = 1, \ldots, M$ is the set of continuous variables, and $C$ represents the number of constraints. In certain cases, the optimization problem is not entirely a black-box, because constraints and/or the objective can be represented algebraically, and this is known as a gray-box or hybrid optimization problem. These formulations have many applications in engineering, such as parameter estimation for simulation-based systems [3, 4, 8], flowsheet synthesis [9], supply chain optimization [10, 11], oilfield operations [12–15], and protein structure prediction [16–18].

One approach to black-box optimization problems is surrogate-based modeling and optimization. A surrogate model, also known as a metamodel or reduced-order model (ROM), is an approximation of the input–output data obtained by the simulation. Once a surrogate model is trained using input–output data, analytical representations of the constraints and the objective of a black-box problem become available and these are computationally cheaper to evaluate and compute derivatives for [5]. The surrogate function can then be optimized by any deterministic optimization solver of choice.

Existing surrogate models can be divided into two broad categories: non-interpolating and interpolating [19]. Non-interpolating models, such as linear, quadratic, polynomial, or generalized regression [20], minimize the sum of squared errors between some predetermined functional form and the sampled data points. While these may lead to simple and interpretable functional forms, they may not be flexible enough to sufficiently capture highly nonlinear correlations [21]. Alternatively, interpolating methods, such as Kriging [21] and radial basis functions (RBF) [5, 7, 8], exhibit increased flexibility by incorporating different basis functions (or kernels), which are built to exactly predict the training points [19]. Due to model flexibility and accuracy, they have been successfully applied in many areas, such as steady-state flowsheet simulation [22], modeling of pharmaceutical processes [9, 23], and aerodynamic design problems [24], to name a few. However, these models tend to have an increased number of parameters and nonlinear, nonconvex terms that cannot be easily globally optimized [1].

One promising approach to overcome the aforementioned limitations of existing surrogate-based optimization is subset selection for regression (SSR) [2]. Also known as a sparse representation or sparse coding, SSR involves selecting the most informative subset from a large set of variables or features, which are then linearly combined to generate a final model. This approach can lower the computational cost by reducing the number of variables or predictors and increase the prediction accuracy by eliminating uninformative features [25]. SSR has been widely applied in numerous fields, such as signal processing [26], gene selection for cancer classification [27], text classification [28], and face recognition [29], but only recently has it been considered for surrogate modeling for optimization [2, 30, 31].

Surrogate modeling is usually coupled with adaptive sampling to iteratively improve the approximation. Two broad categories of adaptive sampling currently exist. The first approach involves using adaptive sampling to identify the best approximation of a certain unknown correlation. Once the best approximation is found, it is directly optimized in one-stage using derivative-based optimization solvers [2]. The second approach does not seek to generate the best approximation. Instead, adaptive sampling is used to determine the location of new samples in promising regions with the aim of finding better solutions to Problem 1. Therefore, the criterion for adaptive sampling seeks to maintain a balance between diversity in sampling and optimization of the unknown objective function, and the surrogate model is used only as an intermediate step to guide the search toward better directions.

This work employs the latter adaptive sampling approach for surrogate-based optimization for the following reasons. This approach can be especially useful for problems where an accurate simulation exists; thus, the development of a surrogate model is simply an intermediate step that leads to the final goal of optimization. This has been found to be efficient in previous works [1, 32, 33], and it is especially suitable for high-dimensional problems, or when the sampling is computationally expensive. Reduced sampling is achieved because rather than focusing on covering the entire feature space to identify the universally best approximation, this paradigm focuses on areas where the global optimum is likely to be located.

In this work we explore various SSR techniques for surrogate modeling for optimization. A large set of basis functions or "features" is first created, and only a subset of those features is selected to generate a sparse model. This allows us to obtain a low-complexity surrogate model that is computationally cheaper and easier to optimize. Several existing SSR algorithms are compared and tested on both unconstrained and constrained benchmark problems. The novel aspects of this work are (a) the comprehensive comparison of the performance of various SSR techniques for surrogate modeling with traditionally used interpolating techniques, and (b) the integration of linear SSR techniques for building nonlinear surrogate functions within a surrogate based optimization framework.

The remainder of this paper is organized as follows. Section 2 illustrates the proposed methodology using a simple motivating example. Section 3 includes an extensive review on all SSR techniques used in this work, followed by Sect. 4, which introduces the overall algorithm and how each SSR method is implemented. Finally, Sect. 5 includes the numerical test problems and the comparison of the performance of SSR in surrogate modeling.

## 2 Motivating example

In this section, we introduce the two main concepts of this work by using a simple motivating example. First, the idea of using SSR for surrogate modeling is described. Subsequently, the optimization-based adaptive sampling technique will be introduced. By combining these two strategies, we aim to locate the global optimum with minimum sampling requirements and computation cost.

### 2.1 Surrogate model construction using SSR

Data-driven optimization is challenging due to the lack of algebraic expressions of the objective and/or constraints. In this work, we seek to maximize model accuracy and minimize model complexity by using SSR. In order to use subset selection, we first need to generate a superset of features. Conventionally, SSR is used when a large set of original variables exists in order to select a subset of important variables by removing redundant variables. Instead, in this work, we expand the original variable set through nonlinear transformations to obtain a diverse feature set. For instance, for a two-dimensional problem with input variables $x_1$ and $x_2$, we can first generate a feature set, such as $x_1$, $x_2$, $x_1^2$, $x_2^2$, $exp(x_1)$, and $exp(x_2)$. We can then use SSR to choose only a subset of those features to construct a surrogate model (Fig. 1) [2].

We will illustrate this idea by using a simple test function: $f(x) = 2x^4 - 3x^2 + x$. The actual functional form of this test problem is assumed to be unknown, and 10 initial data points are collected by Latin Hypercube Design (LHD). Three different methods—linear, kriging, and SSR—are used, and the initial basis set for SSR consists of $x^5$, $x^4$, $x^3$, $x^2$, $x$, and $exp(x)$. The resulting surrogate models and their functional forms are shown in Fig. 2.

As expected, linear regression leads to an inaccurate model, because it cannot capture the nonlinearity of the actual function. While Kriging constructs a highly accurate model, its functional form is complicated (i.e., the number of terms is equal to the number of samples used). A balance between model accuracy and interpretability is achieved by SSR, which selects only a subset of basis functions. As a result, SSR is able to generate a surrogate model with a functional form almost identical to that of the actual test function.

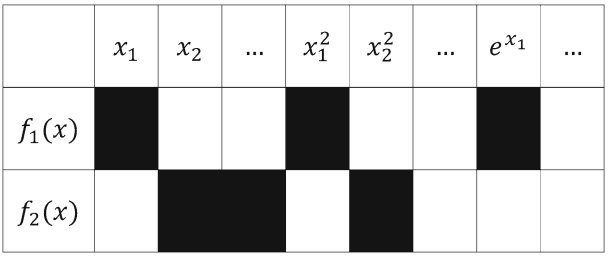
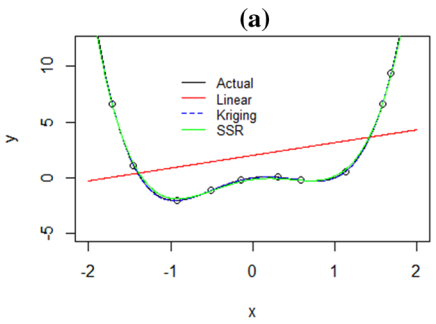



**Fig. 1** Illustration of SSR for surrogate modeling




**(b)**

| Type | Functional Form |
|---|---|
| Actual | $f(x) = 2x^4 - 3x^2 + x$ |
| Linear | $y = 1.144x + 2.016$ |
| Kriging | $y = 51.989 + \sum_{n=1}^{10} c_n \exp[-3.041\left(x_i - x_i^{(n)}\right)^2]$ |
| SSR | $y = 1.845x^4 - 2.535x^2 + 0.988x - 0.151$ |

**Fig. 2 a** Graphical representation of test problem $f(x) = 2x^4 - 3x^2 + x$, and **b** the resulting functional forms of surrogate models fitted by linear, Kriging, and SSR

### 2.2 Optimization-based adaptive sampling

One of the most important challenges of surrogate-based optimization is locating the global optimum, while simultaneously minimizing sampling requirements. When data collection is computationally expensive or time-consuming, data points should be collected such that the total number of samples collected at each iteration is minimized, while maximizing the information gained from these sampled points [34]. Since the ultimate goal of surrogate modeling for data-driven optimization is to locate the global optimum, points sampled near the optimum provide more valuable information than points collected far from the actual optimum. Furthermore, generating a surrogate model that fits all points perfectly, while ideal, would be an inefficient strategy, especially when the number of available samples is limited.

Motivated by this idea, we use optimization-based adaptive sampling in this work (Fig. 3). First, an initial sample set is generated using a space-filling experimental design (Latin Hypercube Design), and the first surrogate model is constructed. This initial surrogate model is optimized to global optimality using BARON [35], and the incumbent solutions (i.e., both local and global solutions) are then used as the next sampling point(s). The initial surrogate model is then updated, and this process repeats until a convergence criterion is satisfied. This sampling strategy has been shown to efficiently focus in areas that are more likely to contain the global minimum. It is important to emphasize that while we are interested in generating a good surrogate model that can accurately predict the optimum, we are not interested in constructing a "perfect" surrogate model that fits all the points exactly. Therefore, as shown in Fig. 3, the final surrogate model is allowed to be imperfect in regions of low interest (i.e., near the boundary or areas far from the minimum) but still accurately locates a global minimum.

## 3 Subset selection for regression

In a typical generalized linear regression setting, we are given a set of training data and we want to generate the following model:

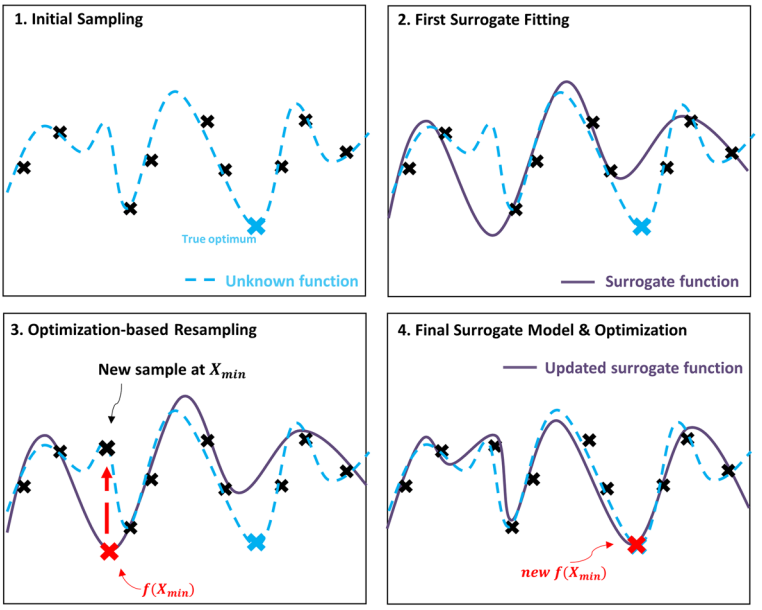



**Fig. 3** Optimization-based adaptive sampling

$$y = \beta_0 + \beta_1 X_1 + \cdots + \beta_P X_P = \beta_0 + \sum_{p=1}^{P} \beta_p X_p, \quad p = 1, \ldots, P \tag{1}$$

where $X = (X_1, \ldots, X_P)$ is a $P$-dimensional vector of features or predictors, $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_P)$ is the vector of regression coefficients, $y$ is the response variable. Following an ordinary least squares (OLS) approach, one would minimize the residual sum of squares of (1) using all of the $P$ features. This is often not desirable, because it is prone to overfitting and does not eliminate redundant features. Therefore, the model interpretability and accuracy can be improved if we locate the best and sparsest model, which includes only the predictors that explain the true variance and effects of the problem [36].

In this work, $X_p$ are the basis functions or the "features" unless mentioned otherwise. These are both linear and non-linear algebraic terms generated by transformations of original variables. Each problem consists of $P$ number of features and $n$ number of samples, and we want to choose only a small subset of the features, specifically $q$ features ($q \leq P$), to create an accurate and sparse model.

One way to achieve this is by using Subset Selection for Regression. Subset selection, also known as feature selection, can be achieved by two methods—convex optimization and greedy algorithms. Traditionally, SSR is used to create sparse models, overcome the risk of overfitting, and improve model interpretability [37]. In this work, our main motivation is to investigate whether existing SSR techniques can be embedded within an adaptive sampling surrogate-based optimization framework, in order to create simple and tractable surrogate functions that retain accuracy, yet

remain easily optimizable by existing solvers. In addition, we want to compare the performance of SSR-based surrogate functions with that of commonly used complex interpolating functions with respect to computational cost and accuracy in locating the global optimal solution.

### 3.1 Subset selection using convex optimization

A subset selection problem, when formulated as a convex optimization problem, leads to one-step feature selection. This approach usually involves solving an optimization problem in the presence of a constraint that leads to a sparse model. In particular, imposing a *L1*-norm constraint has become a popular approach for automatic feature selection [37]. Since the resulting optimization problem is convex, it can be efficiently solved by several optimization solvers. In this section, $N$ represents the total number of data points ($n = 1, \ldots, N$); $\hat{\boldsymbol{X}}$ is a $[N \times P]$ matrix of features created from the original input variables $x_i$ sampled at $N$ points; $\hat{\boldsymbol{y}}$ is the vector of the response collected at the samples.

#### 3.1.1 Lasso and elastic net

One of the most commonly used types of a sparse generalized linear model is Lasso. Lasso (Least Absolute Shrinkage and Selection Operator), originally introduced in [38], is based on the idea of using a *L1*-penalty of the regression coefficient. A sparse linear model is generated by solving the following optimization problem, where $\lambda$ is a regularization parameter:

$$\underset{\beta \in \mathbb{R}^P}{minimize} \left\{ \left\| \hat{\boldsymbol{y}} - \hat{\boldsymbol{X}}\boldsymbol{\beta} \right\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \right\} \tag{2}$$

Due to the presence of the *L1*-penalty term, Lasso can perform automatic variable selection by shrinking some coefficients completely to zero. Therefore, it is advantageous over Ridge regression, which minimizes the *L2* penalty on the regression coefficients, because Ridge regression only shrinks the coefficients toward zero but does not enforce them to be equal to zero [39]. While Lasso has been successfully applied in many cases, it has some drawbacks. First, when the number of variables is greater than the number of samples (i.e., $P > N$), Lasso can only select at most $N$ variables. In addition, if the variables are highly correlated, then Lasso tends to only select one variable from a set of strongly correlated variables and ignores the grouping effect [34].

Elastic Net overcomes these limitations by combining the *L2* and *L1* norm penalty terms. The presence of the *L2* norm makes Elastic Net penalty strictly convex for all $\alpha > 0$, and this strict convexity guarantees a grouping effect and overcomes the limitation on the number of variables selected in the $P > N$ case. Hence, a group of highly correlated predictors has approximately the same coefficients, whereas Lasso only selects one of the predictors due to its non-strictly convex penalty term

[39]. The generalized form of the Elastic Net optimization problem is as follows [40]:

$$\underset{\beta \in \mathbb{R}^P}{minimize} \left\{ \frac{1}{N} \left\| \hat{\boldsymbol{y}} - \hat{\boldsymbol{X}} \boldsymbol{\beta} \right\|_2^2 + \lambda \left[ \alpha \|\boldsymbol{\beta}\|_1 + \frac{1}{2}(1-\alpha) \|\boldsymbol{\beta}\|_2^2 \right] \right\} \tag{3}$$

The last two terms are the Elastic Net penalty, which is a combination of both Lasso and Ridge penalties. The Elastic Net penalty is controlled by $\alpha$: if $\alpha = 1$, the Elastic Net penalty is equivalent to Lasso regression; if $\alpha = 0$, it becomes a simple Ridge regression. The tuning parameter $\lambda$ controls the overall strength of the penalty and the degree of regularization.

### 3.1.2 Sparse principal component regression

Principal component regression (PCR) is a two-stage procedure that performs Principal Component Analysis (PCA) followed by ordinary least squares (OLS) regression. In particular, the regression is performed by using principal components as new explanatory variables instead of original variables, and the accuracy of the model can be controlled by varying the number of principal components included in the model [41]. However, since the principal components are linear combinations of all original variables, feature selection is not directly feasible via PCA. Consequently, a sparse model cannot be created solely by PCR.

To overcome this limitation, Zou et al. [41] proposed a new method called Sparse Principal Component Analysis (SPCA), which imposes the Elastic Net penalty on the regression coefficients. This method generates principal components with sparse loadings, which can be combined with OLS to create a sparse regression model. The sparsity of principal components is controlled by the Elastic Net penalty $\lambda$. This method will be noted as "sPCR1" in this work. However, one disadvantage of sPCR1 is that it is an unsupervised learning technique. The principal components are selected without utilizing information on the response variable, which could potentially degrade the performance of the regression model.

Kawano et al. proposed a supervised, one-stage approach for principal component regression in [42], which performs sparse PCA and regression simultaneously. It obtains sparse principal components that are related to the response variable. The model sparsity is obtained by imposing a penalty onto the regression coefficients using two regularization parameters ($\lambda_\beta$ and $\lambda_\gamma$). This method will be referred as "sPCR2".

### 3.1.3 Sparse partial least squares regression

Partial least squares (PLS) regression has been widely used as an alternative to OLS and PCR because of its robustness. Specifically, it has been found that model parameters do not drastically change as new samples are taken from the total population [43]. Unlike PCA, PLS is a supervised dimensionality reduction technique. However, similar to PCA, PLS does not lead to automatic feature selection as the final direction vectors are linear combinations of all of the original predictors. To address this problem, Chun et al. [44] have proposed a sparse partial least squares regression (sPLS).

This method imposes a *L1*-constraint on the dimension reduction stage of PLS and constructs a regression model by using only a subset of sPLS components as new explanatory variables [44]. Note that even though both sPLS and sPCR2 are supervised dimensionality reduction techniques, sPLS imposes the Elastic net penalty onto a surrogate of the direction vector instead of the regression coefficients [42, 44].

### 3.2 Subset selection using a greedy approach

Greedy algorithms involve iteratively adding or removing features, which can be achieved by forward selection or backward elimination. In forward selection, a model is generated by progressively incorporating variables and generating a larger subset, whereas in backward elimination, the model starts with all predictors and iteratively eliminates the least promising ones. Once the models are trained, the most predictive features are chosen by criteria that depend on measures such as the root mean squared error (RMSE), the cross-validation error, or the value of model coefficients (weights) [45]. In this work, support vector regression (SVR) is chosen as a regression strategy. Only backward elimination is considered in this work, because forward selection has been reported to find weaker subsets [45]. This behavior can be explained since the importance of variables is not assessed in the presence of other variables that are not included.

#### 3.2.1 Support vector machine–recursive feature elimination (SVM–RFE)

Support vector machines (SVM) seek to find a regularized function $f(\boldsymbol{X})$ that separates the data with at most $\varepsilon$ deviation from the actually observed data $y_n$, while making sure $f(\boldsymbol{X})$ is as flat as possible. If a data set is "linearly separable" (i.e., a linear function can separate a set of data without error), $f(\boldsymbol{X})$ takes the form:

$$f(\boldsymbol{X}) = \boldsymbol{w} \cdot \boldsymbol{X} + b, \quad \boldsymbol{w} \in \mathbb{R}^P, \; b \in \mathbb{R} \tag{4}$$

where $\boldsymbol{w}$ is a $P$-dimensional weight vector and $b$ is a bias value. In this work, we train a linear SVR model; however the final functional form is nonlinear because we use the augmented set of nonlinear features ($\boldsymbol{X}$).

The values of $\boldsymbol{w}$ and $b$ can then be found by solving the following optimization problem [46]:

$$minimize \;\; \tfrac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{n=1}^{N} \left(\xi_n + \xi_n^*\right) \tag{5}$$

$$subject\ to \begin{cases} y_n - \left\langle \boldsymbol{w}, \hat{\boldsymbol{X}}_n \right\rangle - b \le \varepsilon + \xi_n & n = 1, \ldots, N \\ \left\langle \boldsymbol{w}, \hat{\boldsymbol{X}}_n \right\rangle + b - y_n \le \varepsilon + \xi_n^* & n = 1, \ldots, N \\ \xi_n, \xi_n^* \ge 0 & n = 1, \ldots, N \end{cases}$$

This formulation generates a model that is a linear combination of all original features; thus, none of the original input features can be discarded. To generate a sparse model using SVM, Guyon et al. [27] developed a pruning technique that eliminates

some of the original features to generate a subset of features that yield the best performance. Recursive feature elimination (RFE) is iteratively used to rank the features and remove less important features by performing three simple steps: (1) train a SVM model and obtain the weight vector $\boldsymbol{w}$, (2) compute the ranking criterion $\left(w_i^2\right)$ for all features, and (3) remove the feature with the smallest ranking criterion. The squared weights $w_i^2$ are used as a ranking criterion, because the magnitude of $w_i^2$ denotes the importance of a feature to the overall model.

When the number of original features is large, it is computationally inefficient to remove one feature in every iteration. Instead, several features can be removed at each iteration, but this has to be cleverly done to not sacrifice performance accuracy. In this case, the method provides a feature subset ranking instead of a feature ranking, such that $F_1 \subset F_2 \subset \cdots \subset F$. Hence, the features in a subset $F_m$ should be taken together to generate a model. In this work, we employ an adaptive multiple-feature removal strategy. Initially, when the number of remaining features is large, more features are removed to speed up the elimination process. When only a few features remain, fewer features are removed to more carefully explore synergistic effects between remaining features. This is achieved by using a heuristic rule that we have found efficient, which removes $1/(iter + 4)$ features at a time, where $iter$ represents the iteration number. Therefore, when $iter = 1$, one-fifth of the existing features are removed, whereas when $iter = 10$, less features (1/11 of remaining features) are removed.

The performance of SVM depends on the selection of two hyper-parameters: $C$ and $\varepsilon$. The cost parameter $C$ is a regularization term that represents the cost of constraints violation, which controls how many samples inside the margin contribute to the overall error. In addition, $\varepsilon$ defines a margin of tolerance, in which no error penalty is given to a point lying inside this margin. These two parameters in conjunction control the width and the flatness of the margin, and they are usually tuned by performing a grid search and a cross-validation procedure [46]. However, these commonly used approaches can be computationally expensive. Instead, we use the following equation to obtain $C$ directly from the training data [47]:

$$C = max\left(\left|\bar{y} + 3\sigma_y\right|, \left|\bar{y} - 3\sigma_y\right|\right) \tag{6}$$

where $y$ and $\sigma_y$ are the mean and standard deviation of the y values in the training set. After the model is trained, the features are ranked and removed according to the value of $w_i^2$ until the change in RMSE of two consecutive iterations is greater than 10%.

## 4 Overall algorithm

We assess the performance of five SSR techniques by integrating them into a framework that has been developed to optimize black-box simulation-based problems, ARGONAUT [1]. ARGONAUT involves subcomponents to perform sampling, surrogate function construction, global optimization of the surrogate-based formulations, and optimization-based adaptive sampling to accurately locate the global minimum [1, 48].

### 4.1 Basis function generation

One advantage of SSR in surrogate modeling is that simple yet diverse and flexible basis functions can be used instead of highly complicated Gaussian or radial basis functions (Table 1). The original variable set is transformed by simple nonlinear transformations, such as polynomial and multinomial transformations. Other types of basis functions can be easily integrated, if needed. More complicated terms, such as trigonometric functions, are not directly handled by several optimization solvers; thus, they were not included in this work. The basis types that are included in the initial superset of features are shown in Table 1, where $\tau = \{1, 2, 3, 4\}, \alpha = \{1, 2, 3\}, \beta = \{1, 2, 3\}, \gamma = 1, \eta = 1$, and indices $i, i', i''$ represent different variables of the original input space.

Another noteworthy advantage of SSR is that a priori knowledge of the system can be used to improve model performance. If the actual functional form is fully or partially known based on first-principles or heuristics, the user can selectively include or exclude certain basis functions. For example, if we want to generate a surrogate model for a second-order reaction $\left(\text{rate} = \text{k}[\text{A}]^2\right)$, we can include $x^2$ in the basis set to guide the selection of the basis function toward the term included in the actual rate equation. For a greedy approach (i.e., SVM–RFE), it is even possible to guarantee inclusion of this term in the final model by assigning an arbitrarily large weight to $x^2$ at every iteration.

### 4.2 Surrogate model construction

A surrogate model is generated by the selected SSR method or by fitting an interpolating function. All SSR methods require tuning of hyper-parameters for optimal performance, and the specifics of how each method is tuned can be found in Table 2. The initial sample set is divided into $k$ training and $k$ validation sets, and $k$ models are generated using all SSR methods. The best model is determined by calculating the root-mean-square-error (RMSE) of $k$ models on the validation set. The model with the smallest RMSE is chosen to proceed to the next stage. Similarly, the constraints are modeled following the same sequence of steps with the selected method.

### 4.3 Optimization-based adaptive sampling

Latin Hypercube Design is used for initial sampling. Based on traditionally used heuristics [1, 48], when the dimension of the problem is less than or equal to 20,

**Table 1** List of possible basis functions

| Type | Equation |
|---|---|
| Polynomial | $x_i^{\tau}$ |
| Multinomial | $x_i^{\alpha} \cdot x_{i'}^{\beta}$ and $x_i \cdot x_{i'} \cdot x_{i'''}$ |
| Exponential | $exp\left(\frac{x_i}{\gamma}\right)^{\eta}$ |
| Logarithmic | $\ln\left(\frac{x_i}{\gamma}\right)^{\eta}$ |

**Table 2** Algorithm parameters

| Method | R package | Parameters | Method of tuning |
|---|---|---|---|
| Elastic net | glmnet (`glmnet`) [49] * | $\alpha$ = controls Elastic Net penalty<br>$\lambda$ = controls the overall strength of the penalty | Grid search using tenfold cross validation |
| sPCR1 | elasticnet (`spca`) [50] | $K_{spcr1}$ = number of components<br>$sparse$ = controls the number of sparse loadings<br>$para$ = vector of 1-norm penalty parameter | $K_{spcr1}$ = number of basis functions<br>$sparse$ = "penalty"<br>$para = 0.01$ |
| sPCR2 | spcr (`cv.spcr`, `spcr`) [51] | $\lambda_\beta$ = nonnegative regularization parameters for regression coefficients<br>$\lambda_\gamma$ = nonnegative regularization parameter for regression intercepts | Grid search using cross validation |
| sPLR | spls (`cv.spls`, `spls`) [52] | $\eta$ = thresholding parameter<br>$K$ = number of hidden components | $\eta$ = tenfold cross-validation<br>$K$ = range $(1, min\{p, 0.9n\})$ [52] |
| SVMRFE | kernlab (`svmLinear`) [53] * | $C$ = controls margin softness<br>$\varepsilon$ = margin of error tolerance | $C = max(\lvert\bar{y} + 3\sigma_y\rvert, \lvert\bar{y} - 3\sigma_y\rvert)$<br>$\varepsilon = 0.1$ |

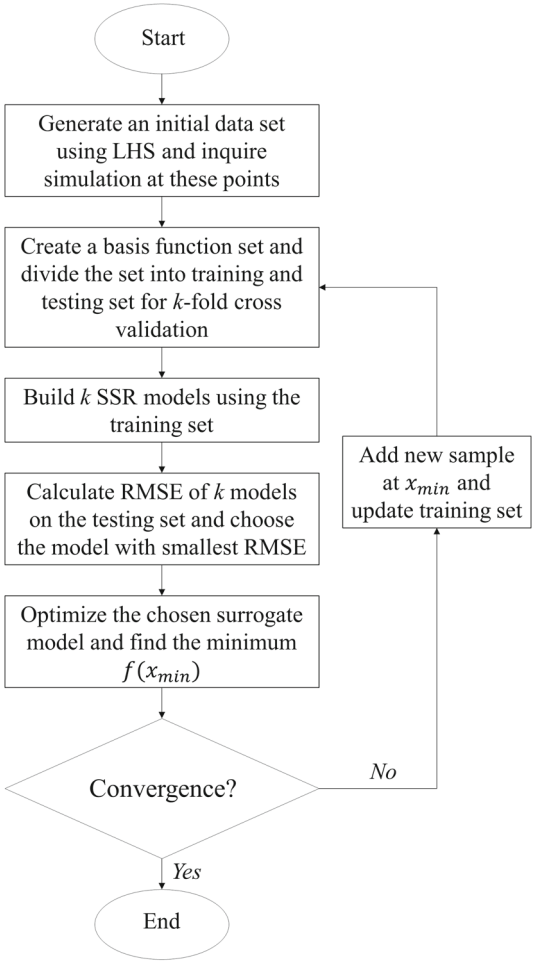
*The selected SSR method is coupled with R package caret [54] via `train` to perform cross-validation/grid search

$10M+1$ samples are used; when the dimension is greater than 20, a fixed number of 251 samples are collected. The simulation is inquired at these points, and the computation cost of sampling is reduced by parallelizing the procedure. After initial sampling, optimization-based adaptive sampling is used to update the surrogate models. The initial surrogate model is globally and locally optimized by using solvers BARON [35] and CONOPT [55], respectively. This allows us to find the global solution and a diverse set of local solutions. The simulation is then re-inquired at all of these incumbent solutions. The algorithm terminates if one of the following convergence criteria is met: (1) the incumbent solution does not improve over a consecutive set of iterations, (2) the maximum number of function calls has been reached, and (3) a feasible incumbent solution is found with a very low cross-validation RMSE [1]. The overall steps of the developed algorithm are shown in Fig. 4.

## 5 Computational studies

We test the five aforementioned SSR techniques on two sets of benchmark problems. The performance of SSR is compared to that of Kriging [32], a widely used interpolating surrogate model. Kriging is based on the assumption that two points that are close to each other are likely to be correlated, and the final functional form is:

**Fig. 4** Algorithmic flowchart




$$y(x) = \mu + \sum_{n=1}^{N} c_n \exp\left[ - \sum_{i=1}^{M} \theta_i \left( x_i - x_i^{(n)} \right)^2 \right] \tag{7}$$

where $\mu$, $\theta$, and $c_n$ are determined by maximum likelihood estimation. The best performance out of three runs is chosen to evaluate the performance of all algorithms. All methods are tuned according to Table 2 to find the best hyper-parameters; $k = 5$ is used for cross-validation. Both sets contain problems that are linear, nonlinear, convex, and/or nonconvex, and all problems have known bounds and known global minima. The functions contain a diverse set of algebraic terms, which may or may not be in the generated basis function set shown in Table 1. Furthermore, both the objective and constraints of the problems are assumed to be unknown, which is the definition of a black-box problem.

### 5.1 Test set A: unconstrained problems

The first test set consists of a subset of 191 test problems from Sahinidis library [56]. All problems are unconstrained with known bounds. The algorithm is parallelized using 2 processors [1]. The dimension and the number of problems in each set are shown in Table 3.

Model performance is evaluated by comparing the fraction of problems solved with the number of function calls and the total computation time. The number of function calls is important to evaluate model performance because black-box simulations can be computationally expensive, limiting the number of samples that are collected. The overall computation time is the total CPU time required to perform sampling, parameter estimation, and surrogate optimization. As our goal is to find a global minimum and not necessarily to find a surrogate model that fits all points perfectly, we evaluate the model performance based on the accuracy of the obtained optimum. Specifically, the error is normalized by the median of all samples so that the range of the search space is taken into account when evaluating the merit of a solution.

The performance profiles of all SSR techniques and Kriging are shown in Fig. 5. The problem is considered to be solved if the normalized error is within 1% ($\varepsilon = 0.01$) of the global solution. As shown in Fig. 5, Kriging solves the most number of problems and shows superior performance to SSR. However, it usually requires the most number of samples to solve the same fraction of problems. This implies that Kriging might not be a favorable choice when the number of samples is the most important limiting factor. sPLR does not solve as many problems as Kriging, but it solves more problems than Kriging if only up to 500 points are sampled. Furthermore, all other SSR techniques, such as Elastic net, sPCR1, sPCR2, and SVMRFE show relatively similar performance, with Elastic net slightly better than other three.

Next, we test the performance of all algorithms on solving the problem with a higher error tolerance (Fig. 6). The purpose of this test is motivated by the nature of data-driven applications, for which a 1% error tolerance might be too strict. For example, the simulation or data may contain uncertainty due to numerical or measurement errors; therefore, locating an optimal solution in the neighborhood of the optimum with the fewest samples possible may be sufficient. When the convergence criterion is relaxed to 10%, the fraction of problems solved increases drastically for all SSR methods. As expected, the performance profile for Kriging did not change significantly. This suggests that while all SSR methods are good at determining the approximate

**Table 3** Specifics of test set A

| Dimension of problems | Number of problems | Number of basis functions |
|---|---|---|
| 2 | 73 | 19 |
| 3–4 | 48 | 43–78 |
| 5–9 | 47 | 125–453 |
| 10–16 | 14 | 575–1720 |
| 20–30 | 9 | 2950–8155 |

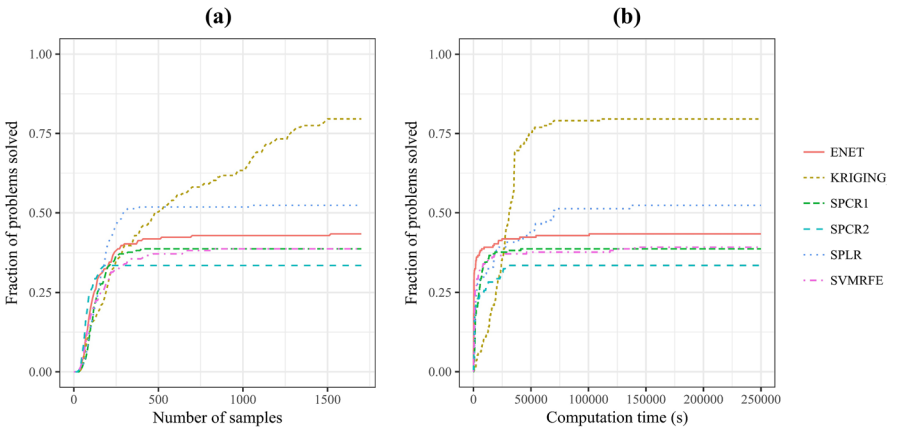



**Fig. 5** Performance profiles of test set A ($\varepsilon = 0.01$) with respect to **a** number of samples and **b** computation time

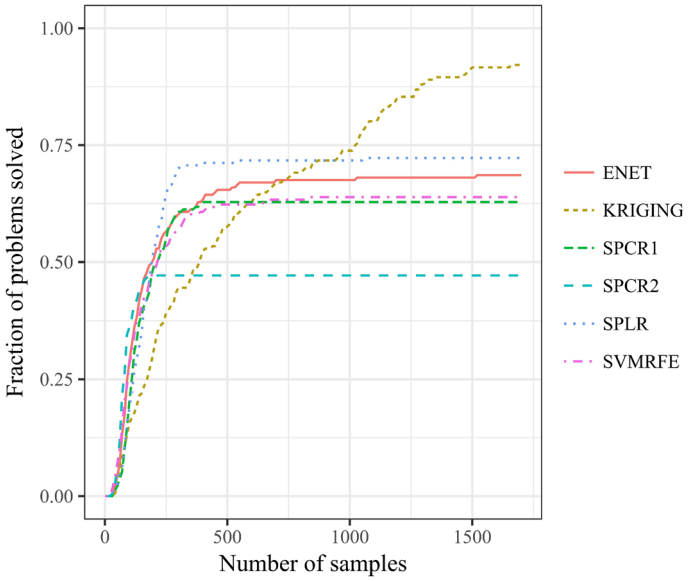



**Fig. 6** Performance profiles of test set A ($\varepsilon = 0.1$)

location of the optima, Kriging is better at locating a very precise solution. This can be rationalized by the fact that Kriging is an interpolating method, and hence the model is very flexible to represent highly nonlinear input–output relationships.

### 5.2 Test set B: constrained problems

The second test set is from GlobalLib [1], in which the problems have inequality constraints, known bounds and global minima (Table 4). As these problems have multiple constraints, different surrogate models are fitted for the objective and each

**Table 4** Specifics of test set B

| Dimension of problems | Number of constraints | Number of problems | Number of basis functions |
|---|---|---|---|
| 2–3 | 1–10 | 31 | 19–43 |
| 4–6 | 1–12 | 24 | 78–185 |
| 7–10 | 4–14 | 16 | 259–575 |
| 11–30 | 9–22 | 17 | 715–8155 |

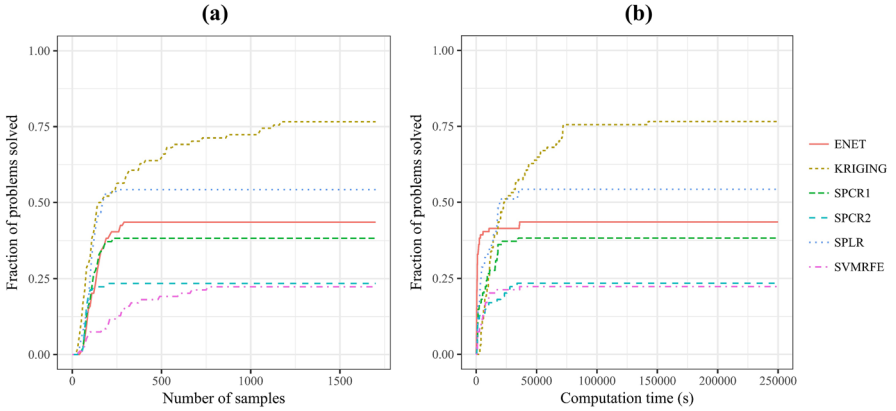



**Fig. 7** Performance profiles of test set B ($\varepsilon = 0.01$) with respect to **a** number of samples and **b** computation time

of the constraints. This procedure is performed in parallel. For all problems, none of the objective nor the constraints are assumed to be known to test the algorithm in the most challenging black-box case. The model performance is compared by using the same convergence criterion that is used for test set A. Both $\varepsilon = 0.01$ and $\varepsilon = 0.1$ are used to generate performance profiles.

Similar to test set A result, Fig. 7 shows that Kriging solves the most number of problems (~75%), followed by sPLR, which solved about 55% of the problems. When the error tolerance is increased to 10%, Fig. 8 shows that the fraction of problems solved increases drastically. Therefore, we can conclude that SSR is good at determining the approximate location of the global solution with less samples. Furthermore, we noticed that SPLR and Kriging exhibit similar performance when the problem dimensionality is low. In fact, for up to 5-dimensional problems, SPLR solves 91% of the problems, and Kriging solves 95% of the problems. This implies that SSR performs well when the dimensionality is low, but its performance degrades as the problem dimensionality increases. For high-dimensional problems, SSR-based surrogate models are not flexible enough to very precisely determine the global solution.

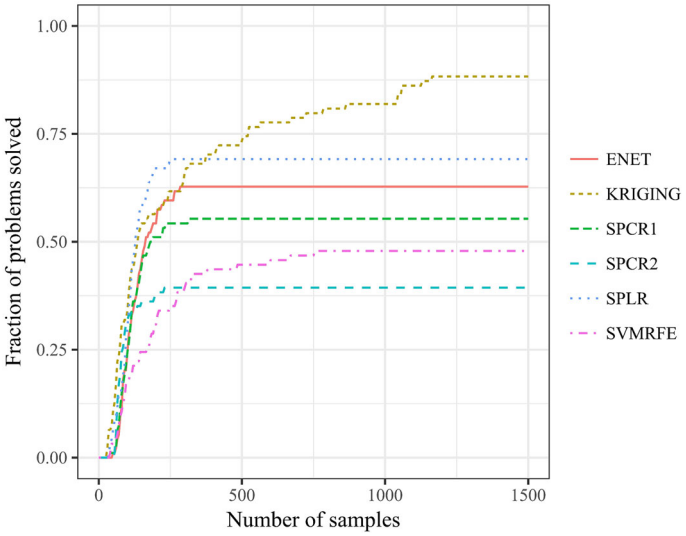



**Fig. 8** Performance profiles of test set B ($\varepsilon = 0.1$)

### 5.3 Computational time

One potential advantage of using SSR over Kriging is that SSR could lead to low-complexity models that are easier to globally optimize. Figure 9 shows the breakdown of CPU time for all problems that are solved within 1% error. The sampling stage represents the total computational time to run the simulation and collect the samples for both initial and adaptive sampling. The parameter estimation stage includes the total computation time to perform parameter estimation, cross-validation, and surrogate model construction for the objective and constraints with parallelization. This is usually the most computationally intensive step in the algorithm, especially when the problem involves multiple constraints. Lastly, the optimization stage involves local and global optimization of the surrogate model.

As expected, Kriging requires the highest computation time for optimization stage due to the high complexity of the surrogate model (Fig. 9c). The computation time required to optimize a Kriging model is on average 500 times greater than those of SSRs. However, Kriging is the most computationally efficient model with respect to the cost required for parameter estimation (Fig. 9b). Therefore, while Kriging leads to a more complicated model, this can be compensated by improved accuracy and a less computationally intensive parameter estimation stage of the model.

Lastly, while the computation times required for the sampling stage (Fig. 9a) and optimization (Fig. 9c) exhibit no significant difference between all SSR methods, the computational cost for parameter estimation differs significantly (Fig. 9b). In general, Elastic Net is the most computationally efficient model, because the model only requires a simple optimization problem to be solved to create a sparse model. SVMRFE, while computationally intensive when only one feature is removed at a time, overcomes this limitation by removing multiple features at a time as previously

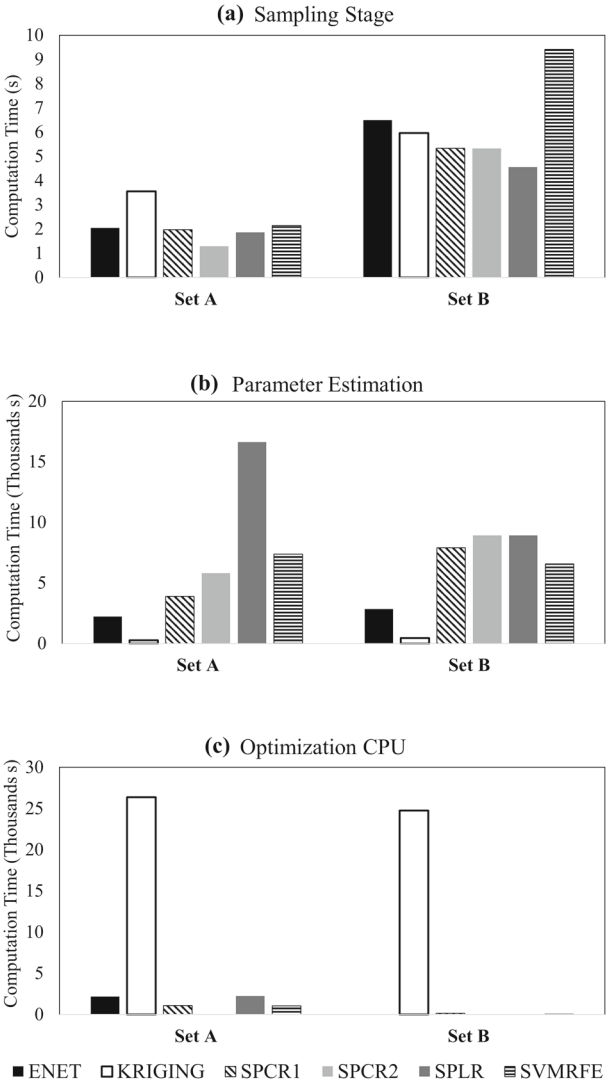



**Fig. 9** Breakdown of computational cost for all solved problems ($\varepsilon = 0.01$) for **a** sampling stage, **b** parameter estimation stage, and **c** optimization stage

discussed. All other methods, including sPCR1, sPCR2, and sPLR, exhibit slower parameter estimation performance, especially for high-dimensional problems.

### 5.4 Model complexity

In this section, we further explore our results with respect to model complexity of the resulting surrogate models. A method that generates a simpler model with fewer terms

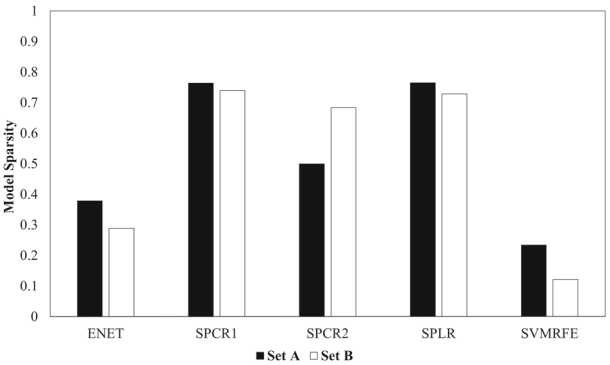



**Fig. 10** Model sparsity of final objective equation for all solved problems for $\varepsilon = 0.01$

is more desirable because a simpler model is generally easier to optimize. In this work, the model complexity is defined by computing the sparsity of the final model:

$$\text{Model sparsity} = \frac{\#\text{ of selected basis functions}}{\#\text{ of all possible basis functions}} \tag{8}$$

A model with sparsity close to 0 contains very few features, and a model with sparsity 1 contains all original features. The model sparsity of the final surrogate model of the objective is shown for both test set A and B (Fig. 10). ENET and SVMRFE usually lead to the sparsest final model, which only retains 20–30% of the original features. sPLR, sPCR1, and sPCR2, on the other hand, keep about 70% of the original features.

Model sparsity becomes more important as the dimension of the problem increases. For example, a 20-dimensional problem leads to 2950 initial basis terms. Therefore, if only 10% of the features are eliminated via SSR (model sparsity = 0.9), the resulting model is a linear combination of 2655 terms. Therefore, if we consider both the accuracy of the solution and model complexity, we can conclude that either ENET or sPLR exhibit the best performance. Since ENET leads to generally sparse solutions and sPLR generally leads to a higher accuracy solution, sPLR is preferable to ENET when the dimension of the problem is low, but ENET can offer a better and faster solution for higher-dimensional problems.

## 6 Conclusions

In this work, we present a comprehensive comparison of five different subset selection for regression techniques for surrogate modeling. We investigate the hypothesis of whether subset selection for generalized regression compared to complicated kernel-based interpolating surrogate functions is better for data-driven optimization. Different subset selection methods are tested over a large set of box-constrained and constrained benchmark problems with up to 30 dimensions, and their performance is compared to that of a popular interpolating surrogate modeling technique, Kriging. While a

Kriging-based approach solves the most number of problems, our results indicate that the computational time required to optimize a complex interpolating model is orders of magnitude greater than those of SSRs. Nevertheless, Kriging requires the least computational time for parameter estimation, which overall may justify the higher computational cost for the model optimization. In addition, our results indicate that when using regression surrogate functions, more problems are solved when sampling is very limited. All subset selection methods show promising performance, especially for low-dimensional problems; however, their performance degrades as the dimension of the problems increases in addition to their high computational cost for selection of features and identification of the model parameters.

## References


1. Boukouvala, F., Floudas, C.A.: ARGONAUT: AlgoRithms for Global Optimization of coNstrAined grey-box compUTational problems. Optim. Lett. **11**(5), 895–913 (2017)
2. Cozad, A., Sahinidis, N.V., Miller, D.C.: Learning surrogate models for simulation-based optimization. AIChE J. **60**(6), 2211–2227 (2014)
3. Amaran, S., et al.: Simulation optimization: a review of algorithms and applications. 4OR **12**(4), 301–333 (2014)
4. Tekin, E., Sabuncuoglu, I.: Simulation optimization: a comprehensive review on theory and applications. IIE Trans. **36**(11), 1067–1081 (2004)
5. Bhosekar, A., Ierapetritou, M.: Advances in surrogate based modeling, feasibility analysis, and optimization: a review. Comput. Chem. Eng. **108**, 250–267 (2018)
6. Bajaj, I., Iyer, S.S., Faruque Hasan, M.M.: A trust region-based two phase algorithm for constrained black-box and grey-box optimization with infeasible initial point. Comput. Chem. Eng. **116**, 306–321 (2017)
7. Forrester, A.I.J., Keane, A.J.: Recent advances in surrogate-based optimization. Prog. Aerosp. Sci. **45**(1), 50–79 (2009)
8. Jakobsson, S., et al.: A method for simulation based optimization using radial basis functions. Optim. Eng. **11**(4), 501–532 (2010)
9. Boukouvala, F., Muzzio, F.J., Ierapetritou, M.G.: Dynamic data-driven modeling of pharmaceutical processes. Ind. Eng. Chem. Res. **50**(11), 6743–6754 (2011)
10. Bittante, A., Pettersson, F., Saxén, H.: Optimization of a small-scale LNG supply chain. Energy **148**, 79–89 (2018)
11. Sampat, A.M., et al.: Optimization formulations for multi-product supply chain networks. Comput. Chem. Eng. **104**, 296–310 (2017)
12. Beykal, B., et al.: Global optimization of grey-box computational systems using surrogate functions and application to highly constrained oil-field operations. Comput. Chem. Eng. **114**, 99–110 (2018)
13. Ciaurri, D.E., Mukerji, T., Durlofsky, L.J.: Derivative-free optimization for oil field operations, in computational optimization and applications in engineering and industry. In: Yang, X.-S., Koziel, S. (eds.), pp. 19–55 Springer, Berlin (2011)
14. Jansen, J.D., Durlofsky, L.J.: Use of reduced-order models in well control optimization. Optim. Eng. **18**(1), 105–132 (2017)
15. Isebor, O.J., Durlofsky, L.J., Echeverría Ciaurri, D.: A derivative-free methodology with local and global search for the constrained joint optimization of well locations and controls. Comput. Geosci. **18**(3), 463–482 (2014)
16. Khoury, G.A., et al.: Princeton_TIGRESS 2.0: High refinement consistency and net gains through support vector machines and molecular dynamics in double-blind predictions during the CASP11 experiment. Proteins Struct. Funct. Bioinform. **85**(6): 1078–1098 (2017)
17. Liwo, A., et al.: Protein structure prediction by global optimization of a potential energy function. Proc. Natl. Acad. Sci. **96**(10), 5482 (1999)
18. DiMaio, F., et al.: Improved molecular replacement by density- and energy-guided protein structure optimization. Nature **473**, 540 (2011)

19. Wang, C., et al.: An evaluation of adaptive surrogate modeling based optimization with two benchmark problems. Environ. Model Softw. **60**, 167–179 (2014)
20. Fen, C.-S., Chan, C., Cheng, H.-C.: Assessing a response surface-based optimization approach for soil vapor extraction system design. J. Water Resour. Plan. Manag. **135**(3), 198–207 (2009)
21. Jones, D.R.: A taxonomy of global optimization methods based on response surfaces. J. Glob. Optim. **21**(4), 345–383 (2001)
22. Palmer, K., Realff, M.: Metamodeling approach to optimization of steady-state flowsheet simulations: model generation. Chem. Eng. Res. Des. **80**(7), 760–772 (2002)
23. Anand, P., Siva Prasad, B.V.N., Venkateswarlu, C.H.: Modeling and optimization of a pharmaceutical formulation system using radial basis function network. Int. J. Neural Syst. **19**(02), 127–136 (2009)
24. Jeong, S., Murayama, M., Yamamoto, K.: Efficient optimization design method using Kriging model. J. Aircr. **42**, 413–420 (2005)
25. Miller, A.J.: Selection of subsets of regression variables. J. R. Stat. Soc. Ser. A (General) **147**(3), 389–425 (1984)
26. Candès, E.J., Romberg, J.K., Tao, T.: Stable signal recovery from incomplete and inaccurate measurements. Commun. Pure Appl. Math. **59**(8), 1207–1223 (2006)
27. Guyon, I., et al.: Gene Selection for cancer classification using support vector machines. Mach. Learn. **46**(1), 389–422 (2002)
28. Feng, G., et al.: Feature subset selection using naive Bayes for text classification. Pattern Recogn. Lett. **65**, 109–115 (2015)
29. Wright, J., et al.: Robust face recognition via sparse representation. IEEE Trans. Pattern Anal. Mach. Intell. **31**(2), 210–227 (2009)
30. Sahinidis, N.: The ALAMO approach to machine learning. In: Kravanja, Z., Bogataj, M. (eds.) Computer Aided Chemical Engineering, p. 2410. Elsevier, Amsterdam (2016)
31. Cozad, A., Sahinidis, N., Miller, D.: A combined first-principles and data-driven approach to model building. Comput. Chem. Eng. **73**, 116–127 (2015)
32. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Glob. Optim. **13**(4), 455–492 (1998)
33. Regis, R.G., Shoemaker, C.A.: Constrained global optimization of expensive black box functions using radial basis functions. J. Glob. Optim. **31**(1), 153–171 (2005)
34. Gorissen, D., et al.: A surrogate modeling and adaptive sampling toolbox for computer based design. J. Mach. Learn. Res. **11**, 2051–2055 (2010)
35. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**(2), 225–249 (2005)
36. Hastie, T., Tibshirani, R., Wainwright, M.: Statistical Learning with Sparsity. Chapman and Hall, New York (2015)
37. Ren, H.: Greedy vs. L1 Convex Optimization in Sparse Coding: Comparative Study in Abnormal Event Detection (2015)
38. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Ser. B (Methodol.) **58**(1), 267–288 (1996)
39. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **67**(2), 301–320 (2005)
40. Hastie, T., Qian, J.: Glmnet Vignette (2014). [cited 2018; https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html]
41. Zou, H., Hastie, T., Tibshirani, R.: Sparse principal component analysis. J. Comput. Graph. Stat. **15**(2), 265–286 (2006)
42. Kawano, S., et al.: Sparse principal component regression with adaptive loading. Comput. Stat. Data Anal. **89**, 192–203 (2015)
43. Geladi, P., Kowalski, B.R.: Partial least-squares regression: a tutorial. Anal. Chim. Acta **185**, 1–17 (1986)
44. Chun, H., Keleş, S.: Sparse partial least squares regression for simultaneous dimension reduction and variable selection. J. R. Stat. Soc. Ser. B Stat. Methodol. **72**(1), 3–25 (2010)
45. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (2003)
46. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Stat. Comput. **14**(3), 199–222 (2004)

47. Cherkassky, V., Ma, Y.: Practical selection of SVM parameters and noise estimation for SVM regression. Neural Netw. **17**(1), 113–126 (2004)
48. Boukouvala, F., Hasan, M.M.F., Floudas, C.A.: Global optimization of general constrained grey-box models: new method and its application to constrained PDEs for pressure swing adsorption. J. Global Optim. **67**(1), 3–42 (2017)
49. Friedman, J.H., et al.: Package 'glmnet': lasso and elastic-net regularized generalized linear models (2018). https://cran.r-project.org/web/packages/glmnet/glmnet.pdf. Accessed 1 May 2018
50. Zou, H.: Package 'elasticnet': elastic-net for sparse estimation and sparse PCA (2015). https://cran.r-project.org/web/packages/elasticnet/elasticnet.pdf. Accessed 1 May 2018
51. Kawano, S.: Package 'spcr': sparse principal component regression (2016). https://cran.r-project.org/web/packages/spcr/spcr.pdf. Accessed 1 May 2018
52. Chung, D., Chun, H., Keleş, S.: An introduction to the 'spls' package, Version 1.0. (2018). https://cran.r-project.org/web/packages/spls/vignettes/spls-example.pdf. Accessed 1 May 2018
53. Karatzoglou, A., Smola, A.J., Hornik, K.: Package 'kernlab': kernel-based machine learning lab (2018). https://cran.r-project.org/web/packages/kernlab/kernlab.pdf. Accessed 1 May 2018
54. Kuhn, M.: Package 'caret': classification and regression training (2018). https://cran.r-project.org/web/packages/caret/caret.pdf. Accessed 1 May 2018
55. Drud, A.: CONOPT. [cited 2018; https://www.gams.com/latest/docs/S_CONOPT.html
56. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. J. Glob. Optim. **56**(3), 1247–1293 (2013)