Preprint Cambridge Centre for Computational Chemical Engineering ISSN 1473 – 4273

Learning based Evolutionary Assistive Paradigm for Surrogate Selection (LEAPS2)

Sushant S Garud [1], Iftekhar A Karimi [1], Markus Kraft [2,3]

released: March 2018

[1] Department of Chemical and Biomolecular Engineering
National University of Singapore
4 Engineering Drive 4
Singapore, 117576
Singapore
E-mail: cheiak@nus.edu.sg

[2] Department of Chemical Engineering and Biotechnology
University of Cambridge
New Museums Site, Pembroke Street
Cambridge, CB2 3RA
United Kingdom
E-mail: mk306@cam.ac.uk

[3] School of Chemical and Biomedical Engineering
Nanyang Technological University
62 Nanyang Drive
Singapore, 637459
Singapore

Preprint No. 197

UNIVERSITY OF CAMBRIDGE

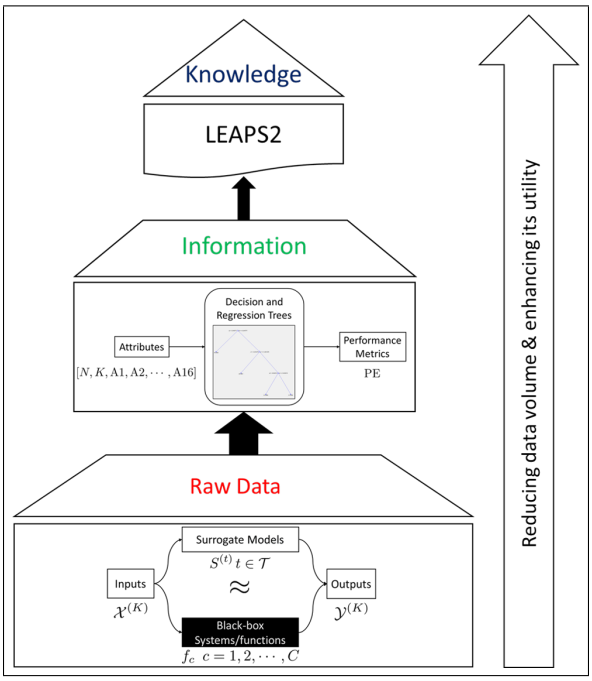
Keywords: Surrogate model, Meta-model, System attributes, Knowledge pyramid, Machine learning.

Edited by
Computational Modelling Group
Department of Chemical Engineering and Biotechnology
University of Cambridge
New Museums Site
Pembroke Street
Cambridge CB2 3RA
United Kingdom

Fax: + 44 (0)1223 334796
E-Mail: c4e@cam.ac.uk
World Wide Web: http://como.ceb.cam.ac.uk/

Abstract

We propose a learning-based paradigm (LEAPS2) to recommend the best surrogate/s with minimal computational effort using the input-output data of a complex physico-numerical system. Emulating the knowledge pyramid, LEAPS2 uses several attributes to extract system information from the data, correlates them with surrogate performances, stores this attribute-surrogate knowledge in a regression tree ensemble, and uses the ensemble to recommend surrogates for unknown systems. We implement LEAPS2 using data from 66 diverse analytical functions, 18 attributes, and 25 surrogates. By progressively adding data, we demonstrate that LEAPS2 learns to improve computational efficiency and functional accuracy. Besides, the architecture of LEAPS2 enables its evolution via more attributes and surrogates. We employ LEAPS2 to recommend surrogates for estimating the bubble and dew point temperatures of LNG. Interestingly, our assistive tool suggests a different surrogate for each temperature, and hints that DPT may be harder to approximate than BPT.



Highlights:

- A learning-based data-driven paradigm (LEAPS2) is proposed for selecting the best surrogate/s to approximate complex systems.
- The paradigm can evolve along three dimensions viz. data sets, system attributes, and surrogates.
- The evolution of LEAPS2 over data sets is demonstrated via five step progressive learning approach.
- The practicality of LEAPS2 is shown by employing it to estimate the VLE properties of LNG.

Contents

1 Introduction

Complex systems are typically studied via physical experiments, computer experiments or their combinations. Physical experiments involve conducting laboratory trials, making field observations, *etc.*, and usually require time and cost. Besides, some experiments may not even be feasible in practice. In such cases, computer experiments may be preferred over physical experiments. These involve experimenting on a rigorous first-principles or physics-based model instead of a real system [12]. Revolutionary advances in algorithmic and computing technologies over the last two decades have empowered researchers to incorporate greater details and accuracy into such models. However, this comes at the expense of larger model size and greater computational burden. Repetitive evaluations of such high-fidelity models for tasks such as sensitivity analysis and optimization often prove uneconomical. Commercial simulators, which are essentially black-box in nature, are often used for developing and simulating these high-fidelity models. Additionally, the development and implementation of such models demand deep domain expertise, which makes them inaccessible to users from other disciplines. Therefore, it is beneficial to replace high-fidelity models (in case of computer experiments) or real physical systems (in case of physical experiments) by computationally cheaper surrogate models that offer a simpler overall picture of the underlying system.

A surrogate model, also known as a meta-model or a response surface, is an empirical expression (analytical or numerical) quantifying the relationships among the most important or relevant inputs and outputs of a system. It is computationally much cheaper alternative to a physical system or its high-fidelity model, and is easily comprehensible to users with little domain knowledge. Its construction requires one to obtain the system/model response at several sample points via experiments or simulations. Then, one needs to choose a form and technique for the surrogate model development. The form is a mathematical relationship between the inputs and outputs, while the technique is a procedure to derive the best parameters of that form for the sampled input-output data. These two steps together yield a final model that we call the surrogate model. Henceforth, we will use "modelling technique" to mean the surrogate modelling technique, and "surrogate" to mean the final surrogate model.

The literature offers a variety of modelling techniques such as polynomial response surface models (PRSM) [15], support vector regression (SVR) [9], kriging [34], radial basis functions (RBF) [37], multivariate adaptive regression splines (MARS) [11], and artificial neural networks (ANN) [19]. Many techniques (*e.g.* kriging and RBF) offer a variety of functional forms, hence can yield several surrogates. Therefore, selecting the best modelling technique and the final surrogate are non-trivial tasks that have largely been done based on trial and error or intuition. Clearly, both modelling technique and sample data for building a surrogate are vital in achieving a good approximation to the real system. The task of obtaining sample data is well studied and discussed in our previous works [12–14] that present a smart sampling algorithm. In this work, we focus on the choice of the modelling technique, corresponding functional form, and the factors that impact these decisions. To this end, our work addresses the following two key questions that a typical user may ask while developing a surrogate from a given set of input-output data.

- Which is the best surrogate for approximating my system/data set?

- Is there a systematic, automated, user-friendly, efficient, and/or reliable procedure for selecting a few surrogates that are likely to be the best?

Although several previous works have attempted to answer the above questions, most are straightforward and enumerative benchmarking studies [3, 48]. They numerically compare and rank the surrogates for some well-known test functions [23, 32]. Inferences from such studies are obviously limited and their extrapolation to unknown systems may not be reliable. Naturally, this has inspired many researchers to explore alternate approaches [7, 31, 44, 49, 50]. Their key idea is to use a set of simple basis functions (*e.g.* $\sin$, $\cos$, $\exp$, *etc.*), and then iteratively evolve their best mix by solving a series regression problems. In other words, this approach is more adaptive and more system-targeted than the earlier benchmarking studies. It is further enhanced by some researchers by employing a variety of more complex surrogates in place of simple analytical functions. For example, Goel *et al.* [16] and Sanchez *et al.* [42] propose a framework that generates an optimal ensemble of surrogates by solving an optimization problem. Recently, Gorissen *et al.* [17] presented an evolutionary approach for automatically selecting one or more surrogates. Their approach also solves a generalized optimization problem using genetic algorithm to select suitable surrogates. Despite their generalized appeal, all these approaches consider surrogate selection as a stand-alone task. For every system, a user needs to solve an optimization problem that guides surrogate selection. Solving such optimization problems is not only arduous but also compute-intensive; and defeats the underlying incentive for using surrogates.

Cui *et al.* [8] attempted to change this conventional framework by employing a features-based meta-learning approach to select the best modelling technique. Their scheme guides modelling technique selection by capturing system (data set) characteristics in terms of several quantitative meta-features. While their work presents some interesting and novel ideas, it has some limitations. First, their numerical evaluation was limited to 10-dimensional functions only. Second, their algorithm neither incorporates nor addresses the effects of dimensionality and sample size, which are the key factors in surrogate construction. Third, they trained their recommendation scheme on 98% of the test functions (observations) and tested/validated it on the remaining 2%. Thus, the effects of training set size on the schemes performance remain unclear. Finally, their scheme recommends only the best modelling technique but fails to address the choice of associated functional form.

In this work, we develop an evolving knowledge-based framework that defines and computes prior information from sampled data of several systems to aid surrogate selection for

Table 1: *Qualitative comparison between LEAPS2 and CRS recommendation scheme based on various features.*

Recommendation scheme features	LEAPS2	CRS
Surrogate modelling technique	✓	✓
Surrogate model form	✓	✗
Effect of dimensionality	✓	✗
Effect of sample size	✓	✗
Prediction error based selection	✓	✓

other systems. We call this Learning based Evolutionary Assistive Paradigm for Surrogate Selection (LEAPS2). Table 1 shows its novelty over the scheme of Cui *et al.* [8] (CRS) and the key features of LEAPS2 are as follows:

1. It relies on the most commonly employed modelling techniques and their corresponding functional forms.
2. It defines several system attributes and performance metrics to characterize and quantify information about surrogate selections for various systems.
3. It utilizes this prior information to derive knowledge that drives our surrogate selection paradigm with minimal surrogate construction efforts.
4. It can perform surrogate selections over wide ranges of dimensions and sample sizes; thus making it practically useful.
5. It enables users to add new modelling techniques as well as system attributes without disturbing the existing architecture of the framework.

This article is structured as follows. Section 2 presents the necessary background for LEAPS2 that includes key definitions, notation, and a brief discussion of various sampling and modelling techniques. The problem statement is provided in section 3, and section 4 discusses the proposed guidance scheme and its algorithmic framework. In section 5, we present our detailed numerical evaluation of LEAPS2 followed by its application to a case study in section 6.

2 Background

2.1 Definitions and notation

The system features that are varied during physical or computer experiments to study a system response/s are called design/input variables or factors (commonly used by statisticians). Let $x = \{x_n \,|\, n = 1, 2, \cdots, N,\ x_n^L \leq x_n \leq x_n^U\} \in \mathbb{R}^N$ denote an N-dimensional vector of design variables. The space defined by the bounds $x^L \leq x \leq x^U$ is called the domain $\mathcal{D}$ which is typically scaled as $[-1, 1]^N$ or $[0, 1]^N$ to avoid numerical ill-conditioning. Therefore, with no loss of generality, we take $\mathcal{D} = [-1, 1]^N$. A sample or a sample point is a specific instance of $x \in \mathcal{D}$. Let the collection of sample points, $\mathcal{X}^{(K)} = \left\{x^{(k)} \,|\, k = 1, 2, \cdots, K\right\}$ denote a sample set of size K. The method used to generate $\mathcal{X}^{(K)}$ is called the sampling technique. Let the system response at x be described by M output variables in $y = \{y_m \,|\, m = 1, 2, \cdots, M\} \in \mathbb{R}^M$. The collection of the responses for a sample set $\mathcal{X}^{(K)}$ is a response set, $\mathcal{Y}^{(K)} = \left\{y^{(k)} = f\left(x^{(k)}\right) \,|\, k = 1, 2, \cdots, K\right\}$. Then, the surrogate approximation constructed using $\mathcal{X}^{(K)}$ and $\mathcal{Y}^{(K)}$ is given by $S(x) \approx f(x),\ \ S : \mathcal{D} \to \mathbb{R}^M$.

2.2 Sampling and surrogate modelling techniques

In a recent review, Garud *et al.* [12] discuss and evaluate several commonly employed sampling techniques like random or Monte Carlo sampling, Latin hypercube sampling, orthogonal arrays, Hammersley points, Halton sampling, Sobol sampling (QS). Their numerical evaluation shows that QS performs consistently better than the other techniques over a wide range of dimensions. Hence, we employ QS for generating input-output data sets from underlying systems which are then used for surrogate construction as discussed later in Section 4.

The literature is replete with the works [3, 48] on various modelling techniques, their applications, benefits and challenges. Here, we present the basics of six surrogate modelling techniques relevant to this work.

2.2.1 PRSM

Polynomial response surface model (PRSM) is one of the simplest modelling techniques. It uses a polynomial function of some known order ρ_{PRSM} as a surrogate model form. Most works employ the second order polynomial model shown in Eq. (1), which can easily be generalized to any order [43].

$$y \approx S_{\text{PRSM}} = \beta_0 + \sum_{n=1}^{N} \beta_n x_n + \sum_{n=1}^{N} \beta_{nn} x_n^2 + \sum_{n=1}^{N} \sum_{p=n+1}^{N} \beta_{np} x_n x_p \tag{1}$$

One major disadvantage of PRSM is that the number of its coefficients or parameters increases combinatorially with ρ_{PRSM} and N, thus demanding more and more sample data for model fitting. Hence, the higher order PRSMs ($\rho_{\text{PRSM}} \geq 3$) are typically avoided, especially for large N.

2.2.2 SVR

Support vector machine (SVM) is a well known technique for classifying categorical data sets [4]. It aims to locate a separating hyperplane such that the distance between the separated data sets is maximized. Drucker *et al.* [9] extended SVM to its regression version, known as support vector regression (SVR). Its simplest case is the linear SVR shown in Eq. (2).

$$y \approx S_{\text{SVR}} = \beta_0 + \sum_{n=1}^{N} \beta_n x_n \tag{2}$$

where $\beta = \{\beta_n \,|\, n = 1, 2, \cdots, N\}$ is a vector normal to the separating hyperplane. Here, the aim is to find a hyperplane that separates the closest data points as far as possible. This can be done by solving the following optimization problem:

$$\min \frac{1}{2} \beta^{\intercal} \beta \tag{3}$$

$$\text{s.t.} \quad \left| y^{(k)} - \left(\beta_0 + \sum_{n=1}^{N} \beta_n x_n^{(k)} \right) \right| \leq \epsilon \quad \forall \quad k = 1, 2, \cdots, K$$

where ϵ is a boundary parameter. The above problem is solved by employing the duality principle and the final SVR is expressed as follows:

$$y \approx S_{\text{SVR}} = \beta_0 + \sum_{k=1}^{K} \left(\alpha_k^* - \alpha_k \right) x^{\intercal} x^{(k)} \tag{4}$$

where α_k^* and α_k are the dual variables. With this understanding, the non-linear form of SVR is obtained by simply replacing $x^{\intercal} x^{(k)}$ in Eq. (4) by a kernel function f_k as shown below:

$$y \approx S_{\text{SVR}} = \beta_0 + \sum_{k=1}^{K} \left(\alpha_k^* - \alpha_k \right) f_k \left(x^{(k)}, x \right) \tag{5}$$

where f_k can be linear, Gaussian, or polynomial as in Table 2. Note that this is a simplified explanation of SVR and interested readers may refer to the rich dedicated literature for further details [6, 9, 46].

Table 2: *Kernel functions for SVR.*

Kernel type	**Function form**
Linear	$\sum_{n=1}^{N} x_n^{(j)} x_n^{(k)}$
Polynomial	$\left(1 + \sum_{n=1}^{N} x_n^{(j)} x_n^{(k)} \right)^{\rho_{\text{SVR}}} \quad \rho_{\text{SVR}} \in \{2, 3, 4\}$
Gaussian	$\exp\left(- \sum_{n=1}^{N} \left(x_n^{(j)} - x_n^{(k)} \right)^2 \right)$

2.2.3 Kriging

Kriging is an interpolating technique that describes a single scalar output to multiple inputs using a global model $(f_b(x))$ and its departure from responses. Eq. (6) shows the general form of kriging where $Z(x)$ is a random process with $\mathbb{E}(Z(x)) = 0$, variance σ^2_{KRG}, and non-zero covariance.

$$y \approx S_{\text{KRG}} = f_b(x) + Z(x) \tag{6}$$

Table 3: *Global polynomial functions for kriging.*

Global function type	**Function form** $(f_b(x))$
Zero order polynomial	1
First order polynomial	$1 + \sum_{n=1}^{N} x_n$
Second order polynomial	$1 + \sum_{n=1}^{N} x_n + \sum_{n=1}^{N} x_n^2 + \sum_{n=1}^{N} \sum_{p=n+1}^{N} x_n x_p$

Table 4: *Correlation functions for kriging.*

Correlation type	**Function form** $(R(\theta, x))$
Exponential	$\exp\left(-\sum_{n=1}^{N}\theta_n\left\lvert x_n^{(j)}-x_n^{(k)}\right\rvert\right)$
Gaussian	$\exp\left(-\sum_{n=1}^{N}\theta_n\left(x_n^{(j)}-x_n^{(k)}\right)^2\right)$
Linear	$\max\left\{0, 1-\theta_n\left\lvert x_n^{(j)}-x_n^{(k)}\right\rvert\right\}$
Spherical	$1-1.5\left(\min\left\{1,\theta_n\left\lvert x_n^{(j)}-x_n^{(k)}\right\rvert\right\}\right)+0.5\left(\min\left\{1,\theta_n\left\lvert x_n^{(j)}-x_n^{(k)}\right\rvert\right\}\right)^3$
Cubic spline	$1-3\left(\min\left\{1,\theta_n\left\lvert x_n^{(j)}-x_n^{(k)}\right\rvert\right\}\right)^2+2\left(\min\left\{1,\theta_n\left\lvert x_n^{(j)}-x_n^{(k)}\right\rvert\right\}\right)^3$

The covariance matrix of $Z(x)$ is given as follows:

$$\mathcal{COV}\left(Z\left(x^{(j)}\right), Z\left(x^{(k)}\right)\right) = \sigma^2_{\text{KRG}}\mathcal{R}\left(R\left(x^{(j)}, x^{(k)}\right)\right) \quad j,k = 1,2,\cdots,K \tag{7}$$

where $\mathcal{R}$ is a $K\times K$ symmetric correlation matrix with the diagonal of ones and $R\left(x^{(j)}, x^{(k)}\right)$ is a correlation function between any two sample points $x^{(j)}$ and $x^{(k)}$. The correlation function is typically specified by the user (Table 4) and several options exist in the literature [27]. The most commonly used correlation function for kriging is the Gaussian shown below:

$$R\left(x^{(j)}, x^{(k)}\right) = \exp\left(-\sum_{n=1}^{N}\theta_n\left(x_n^{(j)}-x_n^{(k)}\right)^2\right) \tag{8}$$

where $\theta_n, n = 1,2,\cdots,N$ are unknown correlation parameters. Further details regarding theory behind kriging can be found in the article by Kleijnen [26]

2.2.4 RBF

Radial basis functions (RBF) were proposed by Hardy [18] for interpolating high dimensional input-output data. They are the linear combination of radially symmetric functions based on the Euclidean distance from each sample point. The general form of RBF is as follows [37].

$$y \approx S_{\text{RBF}} = \sum_{k=1}^{K}\lambda_k\psi\left(\left|\left|x-x^{(k)}\right|\right|\right) + t_p(x) \tag{9}$$

where λ_k are the coefficients, ψ is a radially symmetric basis function, $||\cdot||$ is the Euclidean distance or L_2 norm, and t_p is a tail function also known as polynomial tail (Table 5).

Table 5: *Types of radial basis functions and associated tail functions.*

Type	Basis function $\left(\psi\left(\left\|\left\|x-x^{(k)}\right\|\right\|\right)\right)$	Tail function $(t_p(x))$
Linear/Biharmonic	$\left\|\left\|x-x^{(k)}\right\|\right\|$	$\beta_0+\sum_{n=1}^{N}\beta_n x_n$
Multiquadratic	$\left(\left\|\left\|x-x^{(k)}\right\|\right\|^2+\eta_m^2\right)^{0.5}$ and $\eta_m>0$	$\beta_0+\sum_{n=1}^{N}\beta_n x_n$
Inverse multiquadratic	$\frac{1}{\left(\left\|\left\|x-x^{(k)}\right\|\right\|^2+\eta_m^2\right)^{0.5}}$ and $\eta_m>0$	$\beta_0+\sum_{n=1}^{N}\beta_n x_n$
Polyharmonic/Thin plate	$\left\|\left\|x-x^{(k)}\right\|\right\|^2\log\left(\left\|\left\|x-x^{(k)}\right\|\right\|\right)$	$\beta_0+\sum_{n=1}^{N}\beta_n x_n$
Gaussian	$\exp\left(-\eta_g\left\|\left\|x-x^{(k)}\right\|\right\|^2\right)$ and $\eta_g>0$	0

2.2.5 MARS

In 1991, Friedman [11] proposed multivariate adaptive regression splines (MARS) that approximate multidimensional data by adaptively selecting the basis functions via forward or backward iterations. The general expression of a MARS model is as follows:

$$y \approx S_{\text{MARS}} = \beta_0 + \sum_{b=1}^{B}\beta_b h_b(x) \tag{10}$$

where β_0 and β_b are coefficients and the basis function is given as follows:

$$h_b(x) = \prod_{d=1}^{D_b}\left[s_{d,b}\left(x_{n(d,b)}-\nu_{n(d,b)}\right)\right]_+^{\rho_{\text{MARS}}} \tag{11}$$

where D_b is the number of splits, $s_{d,b}=\pm 1$ denotes the right or left sense of the step function, $x_{n(d,b)}$ is the n-th variable ($1\le n\le N$), and $\nu_{n(d,b)}$ is the location of a knot for the corresponding n-th variable. The subscript $+$ denotes that $h_b(x)$ is a truncated power function and the superscript ρ_{MARS} is the maximum order of the function as given in Eq. (12).

$$\left[s_{d,b}\left(x_{n(d,b)}-\nu_{n(d,b)}\right)\right]_+^{\rho_{\text{MARS}}} = \begin{cases}\left[s_{d,b}\left(x_{n(d,b)}-\nu_{n(d,b)}\right)\right]^{\rho_{\text{MARS}}}, & s_{d,b}\left(x_{n(d,b)}-\nu_{n(d,b)}\right)<0\\ 0, & \text{otherwise}\end{cases} \tag{12}$$

The major advantage of MARS is its lower computational cost compared to techniques like kriging [23].

2.2.6 ANN

Artificial neural network (ANN) is a modelling technique inspired by the human nervous system. It consists of neurons which are basically regression models composed of non-linear transfer functions. A typical ANN has an input layer, an output layer, and at least

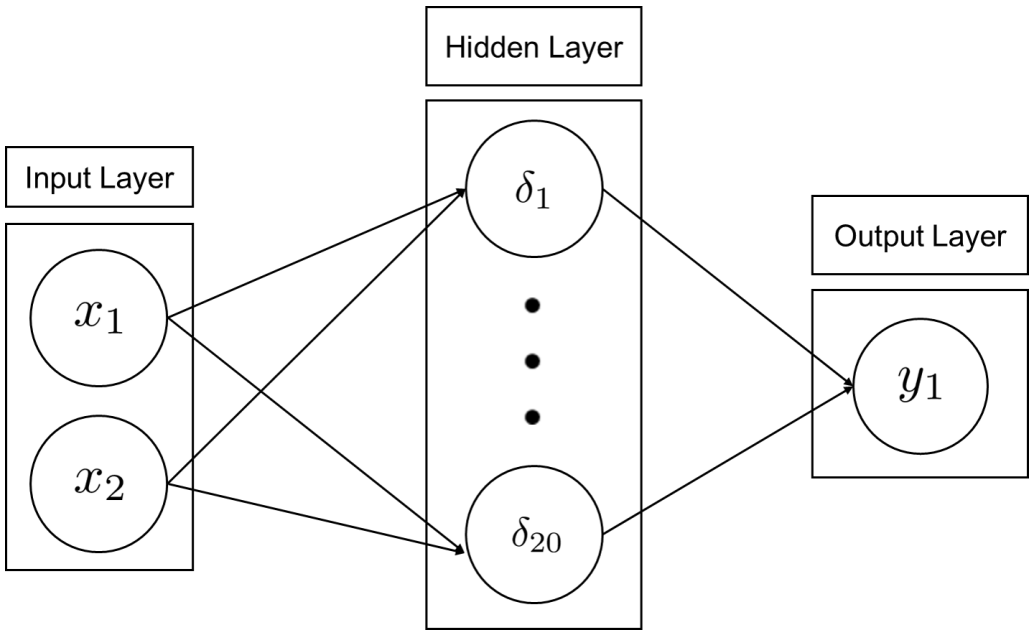


Figure 1: *Schematic of ANN with* 2 *neurons in the input layer,* 1 *output neuron, and* 20 *neurons in the hidden layer.*

one hidden layer. In this work, we limit ourselves to ANN with a single hidden layer, however, this can easily be generalized to an ANN with multiple hidden layers. A general architecture of ANN consists of L_I neurons in the input layer, L_H neurons in the hidden layer, a transfer function $\delta(\cdot)$ (typically a sigmoid function), and $M = 1$ neuron in the output layer corresponding to the response. Then, the network response is given as follows:

$$y \approx S_{\text{ANN}} = \left(\sum_{j=1}^{L_H} \omega_j \delta_j \left(\sum_{i=1}^{L_I} w_{ij} \delta_i(x) + \zeta_j \right) \right) + \zeta_0 + \varepsilon \tag{13}$$

where, w_{ij} is the weight factor for the connection between the ith input neuron and jth hidden neuron, ω_j is the weight factor for the connection between the jth hidden neuron and output neuron, ζ_j is the bias in the jth hidden neuron, ζ_0 is the bias in the output neuron, and ε is a random error with zero mean. Figure 1 shows the illustrative network for $L_I = 2$, $L_H = 20$, and $M = 1$. Note that the performance of ANN depends heavily on its parameters, hence parameter tuning is highly recommended [1, 36]. Interested readers may refer to a book by Haykin [19] for further discussion on ANN.

3 Problem statement

As discussed in the previous section, the literature offers a range of surrogates for various numerical paradigms like system approximation, prediction, optimization, visualization *etc.* Since these surrogates and their corresponding modelling techniques have different mathematical bases, it is not a trivial task to determine the best modelling technique and associated functional form for a given system. Although the "exhaustive" approach can

yield the best surrogate for a given system, it is not efficient and can become computationally expensive for higher dimensions, larger samples, more surrogate parameters, more complex systems, *etc*. Therefore, we wish to develop a computationally inexpensive automated surrogate selection scheme *viz.* LEAPS2 that would solve the following problem:

Given:

- A computationally expensive system f with N input variables and M output responses.
- An input data set (sample set) $\mathcal{X}^{(K)} = \left\{x^{(k)} | k = 1, 2, \cdots, K\right\}$ and the corresponding response set $\mathcal{Y}^{(K)} = \left\{y^{(k)} | k = 1, 2, \cdots, K\right\}$.
- A complete list of surrogates for the system approximation.

Obtain:

- Surrogate recommendation that may be the best for approximating f with minimal computational efforts.

Note that we would like to design LEAPS2 in such a way that it can be evolved dynamically as and when new system-surrogate data become available.

4 LEAPS2

LEAPS2 relies on the concept of knowledge pyramid (see Figure 2) to select surrogates based on the input-output data alone and with minimal surrogate fitting efforts. The knowledge pyramid involves three phases: (1) Data collection, (2) Information extraction, and (3) Knowledge derivation. The evolution of LEAPS2 begins with the data collection phase involving the following steps:

(D1) Select 66 test functions as representative systems (see Appendix A).

(D2) Select 25 surrogates resulting from six modelling techniques discussed earlier (see Table 6).

(D3) Use Sobol sampling (QS) to generate four input-output data sets for each test function.

(D4) Construct 25 surrogates for each input-output data set.

This yields a rich set of system-surrogate data which can be updated dynamically as and when more data become available. We then process these data to extract information.

(I1) Compute system attributes defined in Section 4.2.1.

(I2) Compute performance metrics defined in Section 4.2.2.

From the information computed in steps (I1) and (I2), we quantify and embed knowledge into LEAPS2 using regression tree ensemble.

(K1) Discard correlated attributes via the attribute selection algorithm outlined in Section 4.3.1.

(K2) Correlate the selected system attributes with surrogate performance metrics using regression tree ensemble. This serves as the brain of LEAPS2 for future surrogate selections.

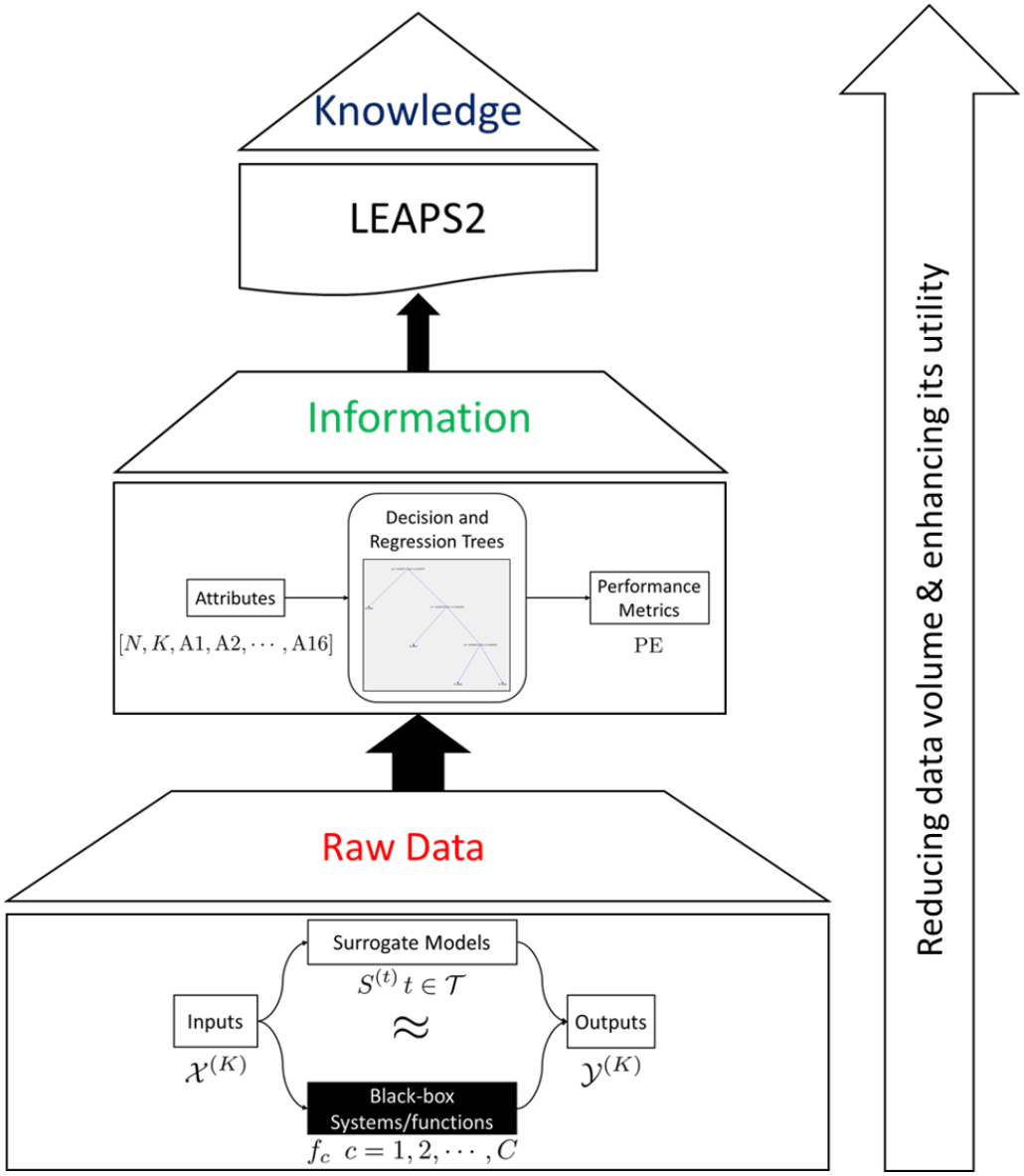


Figure 2: *The core philosophy of LEAPS2 based on the knowledge pyramid.*

Figure 3 shows the flowchart of the above discussed steps and their hierarchical interlinking. We call this the learning protocol of LEAPS2. Now, we discuss each phase of the learning protocol in detail.

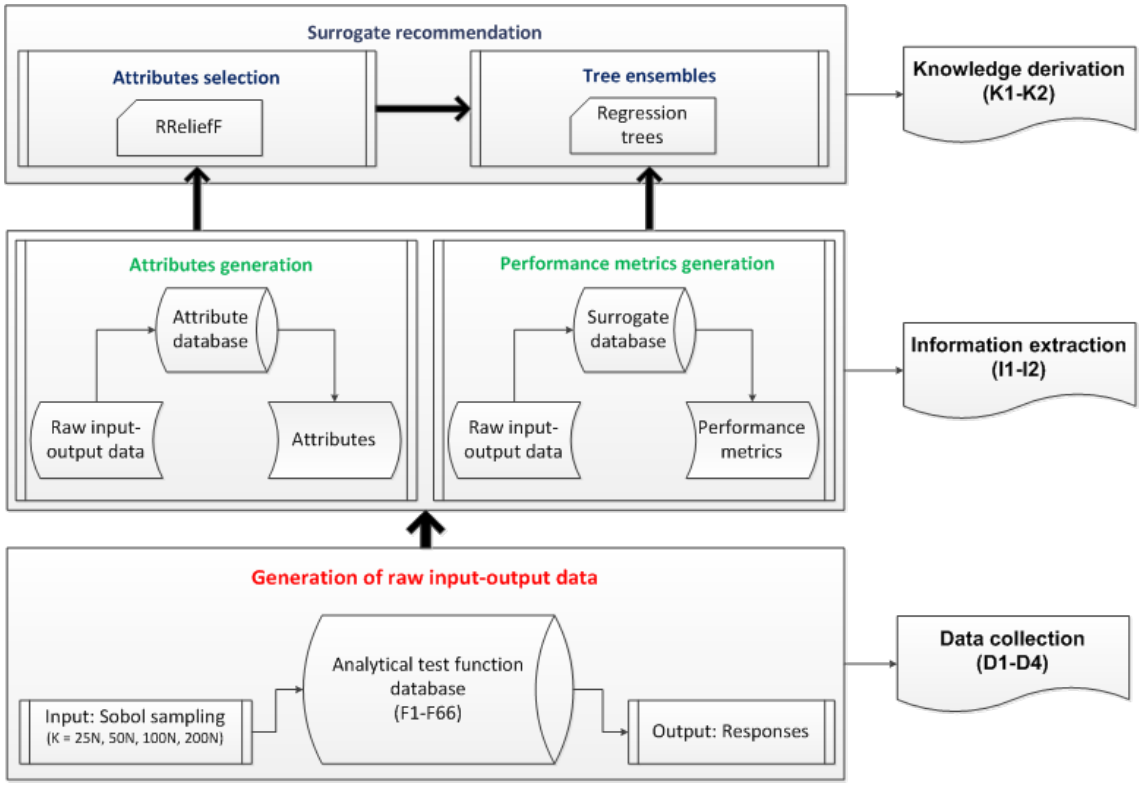


Figure 3: *A flowchart describing the learning protocol of LEAPS2.*

4.1 Data collection

The ability of LEAPS2 to recommend surrogates obviously depends on the extent of its learning. Thus, we need data sets from many diverse systems. Hence, our initial system directory consists of 66 analytical functions with wide ranges of dimensions ($2 \leq N \leq 20$), domain sizes, and key characteristics given in Appendix A (Tables A-I-A-IV). We evaluate each function for four sample sets generated using QS: $K = \{25N, 50N, 100N, 200N\}$ where N is the number of input/design variables. We use QS since it is a robust sampling technique that generates samples using a quasi-random, low discrepancy sequence and and assures uniform space-filling over a range of dimensions [12]. Overall, this amounts to $66 \times 4 = 264$ input-output data sets. The Sobol sampling [24] within MoDS toolkit [35] is used for the data generation. We then propose the following 25 different surrogates based on the six modelling techniques discussed earlier.

$$\mathcal{T} = \{\text{P1}, \text{P2}, \text{K0e}, \text{K1e}, \text{K2e}, \text{K0g}, \text{K1g}, \text{K2g}, \text{K0l}, \text{K1l}, \text{K2l}, \text{K0s},$$
$$\text{K1s}, \text{K2s}, \text{K0c}, \text{K1c}, \text{K2c}, \text{Rb}, \text{Rm}, \text{Ri}, \text{Rt}, \text{Rg}, \text{M}, \text{A}, \text{S}\}$$

Table 6 lists the surrogates associated with each label in $\mathcal{T}$. Finally, we construct 25 surrogates in $\mathcal{T}$ for each data set resulting in total $264 \times 25 = 6600$ surrogates.

Table 6: *List of surrogates used in the development of LEAPS2.*

Modelling technique	Toolbox/Implementation	Polynomial order	Correlation/ Basis function	Label
PRSM	DACE [33]	1	-	P1
		2	-	P2
Kriging	DACE [33]	0	Exponential	K0e
		1		K1e
		2		K2e
		0	Gaussian	K0g
		1		K1g
		2		K2g
		0	Linear	K0l
		1		K1l
		2		K2l
		0	Spherical	K0s
		1		K1s
		2		K2s
		0	Cubic spline	K0c
		1		K1c
		2		K2c
RBF	RBF [22]	1	Biharmonic	Rb
		1	Multi-quadratic	Rm
		1	Inverse multi-quadratic	Ri
		1	Thin plate	Rt
		0	Gaussian	Rg
MARS	ARESLab [21]	-	-	M
ANN	MATLAB: Neural Networks	-	-	A
SVR	MATLAB: Statistics and Machine Learning	-	-	S

4.2 Information extraction

At the end of the data collection phase, we get 264 input-output data sets of different sizes (N and K) and 25 surrogates for each. Each data set reflects an image of the underlying system and we abstract that image in terms of several system attributes that are defined next. Furthermore, to quantify the quality of each surrogate, we define some performance metrics.

4.2.1 System attributes

Any insight into a black-box system can only be achieved based on its input-output data. In other words, these data characterize the underlying system in the absence of its physical/analytical form. System characteristics (*e.g.* non-linearity) may be simpler to grasp and visualize for low dimensional data ($N \leq 3$). However, the same becomes difficult for large N and K. We circumvent this difficulty by defining several numerical attributes of a data set, which capture its system features. In simple words, these attributes distil the essence of the underlying system irrespective of its size, dimensions, *etc.* Subsequently, these attributes can be used in an inference-making process (in our case deriving knowledge) instead of the entire data set. Since our inferences *i.e.* selected surrogates are tailored for paradigms like system prediction, optimization, and sensitivity analysis, our attributes are based on the following system aspects *viz.* (a) response, (b) gradient, and (c) extrema.

Response-based attributes:
We define seven attributes to characterize the magnitude and variations of a system response.

A1 **Empirical mean:** This measures the average magnitude of a response.

$$\overline{y} = \frac{1}{K}\sum_{k=1}^{K} y^{(k)} \tag{14}$$

A2 **Geometric mean:** This measures the geometric mean of a response.

$$\overline{y}_g = \left\{\prod_{k=1}^{K} y^{(k)}\right\}^{\frac{1}{K}} \tag{15}$$

A3 **Harmonic mean:** This measures the harmonic mean (Pythagorean mean) of a response.

$$\overline{y}_h = \frac{K}{\sum\limits_{k=1}^{K}\left(\frac{1}{y^{(k)}}\right)} \tag{16}$$

A4 **Empirical standard deviation:** This gives a measure of variation in a response.

$$s_y = s\left(y^{(k)}\right) = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}\left(y^{(k)} - \overline{y}\right)^2} \tag{17}$$

A5 **Empirical coefficient of variation:** This is nothing but the ratio of standard deviation to mean.

$$C_v = \frac{s_y}{\overline{y}} \tag{18}$$

A6 **Fisher-Pearson coefficient of skewness:** This measures the lack of symmetry in a response.

$$\gamma_1(y) = \frac{\sum_{k=1}^{K} \left(y^{(k)} - \overline{y}\right)^3}{K \times \left(s_y\right)^3} \tag{19}$$

A7 **Kurtosis:** This measures the flatness of a response with respect to the normal distribution for which the Kurtosis is 3.

$$\gamma_2(y) = \frac{\sum_{k=1}^{K} \left(y^{(k)} - \overline{y}\right)^4}{K \times \left(s_y\right)^4} - 3 \tag{20}$$

Gradient-based attributes:
For any $x^{(k)}$, let $x^{(j)}$ be its nearest neighbour based on the Euclidean distance. Then, we can approximate the gradient of the response at $x^{(k)}$ as follows:

$$g^{(k)} = \left\{ g_n^{(k)} = \frac{\left(y^{(k)} - y^{(j)}\right) \operatorname{sgn}\left(x_n^{(k)} - x_n^{(j)}\right)}{\max\left[e, \left|x_n^{(k)} - x_n^{(j)}\right|\right]} \middle| n = 1, 2, \cdots, N \right\}$$
$$\forall j, k = 1, 2, \cdots, K, k \neq j \tag{21}$$

Using this, we can then estimate the Hessian at $x^{(k)}$ as follows:

$$\mathcal{H}^{(k)} = \begin{bmatrix} H_{11}^{(k)} & H_{12}^{(k)} & \cdots & H_{1N}^{(k)} \\ H_{21}^{(k)} & H_{22}^{(k)} & \cdots & H_{2N}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ H_{N1}^{(k)} & H_{N2}^{(k)} & \cdots & H_{NN}^{(k)} \end{bmatrix} \tag{22}$$

and

$$H_{np}^{(k)} = \frac{\left(g_n^{(k)} - g_p^{(j)}\right) \operatorname{sgn}\left(x_n^{(k)} - x_p^{(j)}\right)}{\max\left[e, \left|x_n^{(k)} - x_p^{(j)}\right|\right]} \;\; \forall \;\; n, p = 1, 2, \cdots N \tag{23}$$

where e is a user specified small positive number that depends on the number of significant digits in the design variables. Let the eigenvalues of $\mathcal{H}^{(k)}$ are $\upsilon_1^{(k)}, \upsilon_2^{(k)}, \cdots, \upsilon_N^{(k)}$. By using the characteristic polynomial of a matrix, the determinant of $\mathcal{H}^{(k)}$ is given by $\det\left(\mathcal{H}^{(k)}\right) = \prod_{n=1}^{N} \upsilon_n^{(k)}$ [29]. With this, we propose the following six attributes that quantify the gradient and curvature information of the system.

A8 **Empirical mean of gradient estimates:** This measures the average steepness of the underlying system across its dimensions.

$$\overline{g_n} = \frac{1}{K} \sum_{k=1}^{K} \left|g_n^{(k)}\right| \;\; n = 1, 2, \cdots, N \tag{24}$$

$$\overline{g} = \frac{1}{N}\sum_{n=1}^{N}\overline{g_n} \tag{25}$$

A9 **Empirical standard deviation in gradient estimates:** This measures the variation in the gradient estimates across all dimensions. This can be viewed as a measure of relative non-linearity among the dimensions *i.e.* the larger its value the greater the relative non-linearity.

$$s_{\overline{g}} = s(\overline{g_n}) = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}\left(\overline{g_n} - \overline{g}\right)^2} \tag{26}$$

A10 **Empirical mean of the standard deviations in gradient estimates**: This measures the average variation in the gradient estimates across all dimensions.

$$s_{g_n} = s\left(g_n^{(k)}\right) = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}\left(g_n^{(k)} - \overline{g_n}\right)^2} \quad n = 1, 2, \cdots, N \tag{27}$$

$$\overline{s_g} = \frac{1}{N}\sum_{n=1}^{N} s_{g_n} \tag{28}$$

A11 **Empirical standard deviation in the standard deviations in gradient estimates**: This measures the standard deviation in the variations of the gradient estimate in every dimension.

$$s_{s_g} = s(s_{g_n}) = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}\left(s_{g_n} - \overline{s_g}\right)^2} \tag{29}$$

A12 **Empirical mean of curvature:** This quantifies the mean curvature of the system using the eigenvalues of the Hessian matrices computed at the sample points.

$$\overline{\kappa} = \frac{1}{K}\sum_{k=1}^{K}\det\left(\mathcal{H}^{(k)}\right) \tag{30}$$

where,

$$\det\left(\mathcal{H}^{(k)}\right) = \prod_{n=1}^{N}\upsilon_n^{(k)} \quad k = 1, 2, \cdots, K \tag{31}$$

A13 **Empirical standard deviation in the curvature**: This measures the variation in the curvatures computed across the sample points.

$$s_\kappa = s\left(\det\left(\mathcal{H}^{(k)}\right)\right) = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}\left(\det\left(\mathcal{H}^{(k)}\right) - \overline{\kappa}\right)^2} \tag{32}$$

Extrema-based attributes:
Let $c^{(r)}, r = 1, 2, \cdots, R_s$ be some sample points in $\mathcal{D}$ generated using the Latin hypercube sampling. Define

$$d = \min_{1 \leq r,t \leq R_s,\, r \neq t} \left|\left| c^{(r)} - c^{(t)} \right|\right| \tag{33}$$

Construct a hyper-sphere $(\mathcal{D}_r)$ of diameter d around each sample point $c^{(r)}$ to get R_s mutually exclusive subsets of $\mathcal{D}$. Then, we define three attributes as follows.

A14 **Fluctuation over $\mathcal{D}$**: This measures the maximum fluctuation in the response over $\mathcal{D}$.

$$R_{\text{min/max},\mathcal{D}} = y^{\text{max},\mathcal{D}} - y^{\text{min},\mathcal{D}} \tag{34}$$

where, $y^{\text{min},\mathcal{D}} = \min_{1 \leq k \leq K} y^{(k)}$, and $y^{\text{max},\mathcal{D}} = \max_{1 \leq k \leq K} y^{(k)}$.

A15 **Empirical mean of fractional local fluctuations**: This gives the mean response fluctuation over R_s hyper-spheres as a fraction of the maximum fluctuation over $\mathcal{D}$. It measures the average bumpiness in the response that arises from the systems non-linearity.

$$\overline{R}_{\text{min/max}} = \frac{1}{R_s} \sum_{r=1}^{R_s} R_{\text{min/max},r} \quad r = 1, 2, \cdots, R_s \tag{35}$$

$$R_{\text{min/max},r} = \frac{y^{\text{max},r} - y^{\text{min},r}}{R_{\text{min/max},\mathcal{D}}} \quad r = 1, 2, \cdots, R_s \tag{36}$$

where, $y^{\text{min},r} = \min_{y^{(k)} \in \mathcal{D}_r} y^{(k)}$, $y^{\text{max},r} = \max_{y^{(k)} \in \mathcal{D}_r} y^{(k)}$.

A16 **Empirical standard deviation in fractional local fluctuations**: This measures the variations in local fluctuations *i.e.* variations in local non-linearity.

$$s(R_{\text{min/max}}) = \sqrt{\frac{1}{R_s - 1} \sum_{r=1}^{R_s} \left(R_{\text{min/max},r} - \overline{R}_{\text{min/max}}\right)^2} \tag{37}$$

So far, we defined 16 attributes. In addition, N and K are also crucial in surrogate selection. Thus, we have 18 attributes that extract system features.

4.2.2 Performance metrics

Typically, the quality of a surrogate is measured using error-based metrics. Two commonly used metrics are average absolute error (AAE), and root mean squared error (RMSE) given in Eqs. (38) and (39) respectively. AAE measures the average magnitude of the error while RMSE quantifies its distribution.

$$\text{AAE} = \frac{\sum_{q=1}^{Q} \left| y^{(q)} - S\left(x^{(q)}\right) \right|}{Q} \tag{38}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{q=1}^{Q} \left(y^{(q)} - S(x^{(q)})\right)^2}{Q}} \tag{39}$$

Garud *et al.* [13] combined these two metrics into a single performance metric called pooled error (PE). The lower the PE, the better the quality of the surrogate.

$$\text{PE} = \sqrt{\text{AAE} \times \text{RMSE}} \tag{40}$$

For assessing any surrogate, we evaluate these metrics using a uniformly distributed test set $\mathcal{Q} = \left\{ \left(x^{(q)}, y^{(q)}\right) \mid q = 1, 2, \cdots, Q = 0.25K \right\}$.

With this, each of the 264 data sets gives us 18 attributes, and the 25 surrogates associated with that data set gives us 25 PEs. We address this entire set of 264 pairs of 18 system attributes and 25 PEs as system-surrogate information.

4.3 Knowledge derivation

We now process the system-surrogate information to derive the knowledge for recommending the best surrogate for any given data set. We employ regression tree ensemble to store and later extract the derived knowledge. Before this, we eliminate attributes that may be correlated using RReliefF or regressional ReliefF [28, 41].

4.3.1 Attribute elimination

RReliefF can detect linear as well as non-linear dependencies among the attributes and eliminate any redundant ones. For this, it assigns a weight between -1 to 1 to each attribute. The large positive weight is assigned to the more important attribute. We use "*relieff*" function from MATLAB and Table 7 shows the weights for our 18 attributes. Based on these weights, we reject A2, A5, A6, and A7 as redundant, thus we use 14 attributes in our regression tree ensemble.

Table 7: *Importance weights for all the attributes computed using RReliefF.*

Attribute	Weight	Attribute	Weight	Attribute	Weight
N	0.046	A5	−0.001	A11	0.180
K	0.040	A6	−0.032	A12	0.037
A1	0.270	A7	−0.020	A13	0.062
A2	−0.001	A8	0.230	A14	0.253
A3	0.043	A9	0.244	A15	0.017
A4	0.266	A10	0.176	A16	0.012

4.3.2 Regression tree ensemble

A regression tree [2, 30, 38] employs recursive partitioning of the input domain to make a decision/prediction for a given set of input values. It is a connected graph denoting the

recursive partitions, where each node represents a partition cell with its decision. A regression tree is preferred over other machine learning approaches (*e.g.* nearest neighbour) since it offers following advantages:

- Its hierarchical architecture reveals the relative importance of various input variables.
- Predictions require a simple query of the tree without any elaborate computations.
- In the absence of full data, the decisions can still be based on a partial tree.

The performance of a regression tree can be further improved via a variety of modifications such as boosting, bagging, random forest *etc.* [5]. They typically formulate an ensemble of regression trees to arrive at the improved decisions. We employ regression tree ensemble in the knowledge derivation step of LEAPS2 to ascertain excellent performance. In our case, the attributes are the inputs, and the PEs are the outputs/decisions. Finally, we use the refined system-surrogate information (pairs of 14 attributes and 25 PEs for each of 264 data sets) to grow the regression tree ensemble which basically stores the derived knowledge.

We implement LEAPS2 using MATLAB 2016b and grow regression tree ensemble using "*fitrensemble*" in MATLAB. Since the performance of a tree ensemble can be improved by optimizing its hyper-parameters, we consider the following hyper-parameters: ensemble-aggregation method (*e.g.* bagging/random forest, least-squares boosting), number of ensemble learning cycles, minimum number of leaf node observations, maximum number of decision splits, and number of predictors for a split. Table 8 lists the settings used for this optimization and further details can be found in the MATLAB documentation [47].

Table 8: *Settings for optimizing hyper-parameters of regression tree ensemble.*

Legend	Optimization option	Value
Optimizer	Bayesian optimizer	-
Objective Function	Expected improvement plus	-
Termination criterion	Maximum evaluation	30
Cross-validation	K-fold cross validation	20

The fully grown ensemble then can be used to select the best possible surrogate for any given data set as discussed next.

4.4 Surrogate recommendation

For any given input-output data set, LEAPS2 selects the best surrogate based on the recommendation protocol comprising the following steps:

(R1) Normalize the input data as $[-1, 1]^N$ and compute the 14 system attributes.

(R2) Feed the attributes to the regression tree ensemble to predict the PEs of the 25 surrogates.

(R3) Arrange the surrogates in the ascending order based on their predicted PEs.

(R4) Select the top P^* surrogates where P^* is determined by striking a balance between the computational effort and recommendation performance as discussed next.

(R5) Construct and evaluate the P^* surrogates to recommend the best of them.

The goal of LEAPS2 is to identify the best surrogate for a given data set with minimal computational expenditure. Let us assume that we evaluate the top P surrogates from the regression tree ensemble to pick the best. This computational effort will be maximum for $P = P_{\text{max}} = 25$ while it will be zero for $P = 1$. On the other hand, the likelihood of identifying the best surrogate is 100% for $P = P_{\text{max}} = 25$ and it reduces with the P. Therefore, we extend the analogy of information function from Shannon entropy [40] to quantify the total computational savings achieved by LEAPS2 as follows.

$$\text{Coefficient of Computational Savings (CoCS)} = 1 - \frac{\ln(P)}{\ln(P_{\text{max}} = 25)} \tag{41}$$

If the recommendation from LEAPS2 based on the P surrogates matches the true best (obtained via exhaustive enumeration), then we consider LEAPS2 to be successful. To this end, we define a coefficient of success for LEAPS2 as the fraction of data sets for which LEAPS2 succeeds in identifying the true best surrogate.

$$\text{Coefficient of Success (CoS)} = \left.\frac{\text{Number of successful data sets}}{\text{Total number of data sets}}\right|_P \tag{42}$$

Naturally, a higher P would increase CoS and decrease CoCS. For $P = P_{\text{max}}$, LEAPS2 behaves like exhaustive enumeration. Therefore, an optimal P would be the one that trades-off between CoS and CoCS. To obtain this optimal P, we combine CoS and CoCS into a single objective called coefficient of reward (CoR) = CoS $\times$ CoCS and solve the following optimization problem.

$$\max_{1<P<P_{\text{max}}} \text{CoR} \tag{43}$$

The optimum P for the above problem gives us P^* which is used within the recommendation protocol (Step (R4)).

Although CoS is a good performance measure for LEAPS2, often the difference between the top few surrogates is marginal. Therefore, it is useful to assess the performance of LEAPS2 in terms of its degree of success (DoS) *i.e.* success in identifying either the first, second, or third best surrogate correctly. To this end, let DoS1 denote the fraction of data sets for which the recommendation from LEAPS2 matches the first true best surrogate. Similarly, DoS2 and DoS3 denote the fractions for which LEAPS2 correctly identifies the second and third true best surrogates. This distribution of DoS (DoS1, DoS2, and DoS3) quantifies the overall performance of LEAPS2 for an evaluation paradigm. Next, we present an extensive numerical evaluation of LEAPS2.

5 Training and evaluation

We use a progressive approach to train and evaluate LEAPS2 using the following steps:

(S1) Select 70% of the 264 data sets randomly and train LEAPS2. Evaluate its performance on the same sets.

(S2) Assess the performance of LEAPS2 from S1 on the remaining 30% data sets.

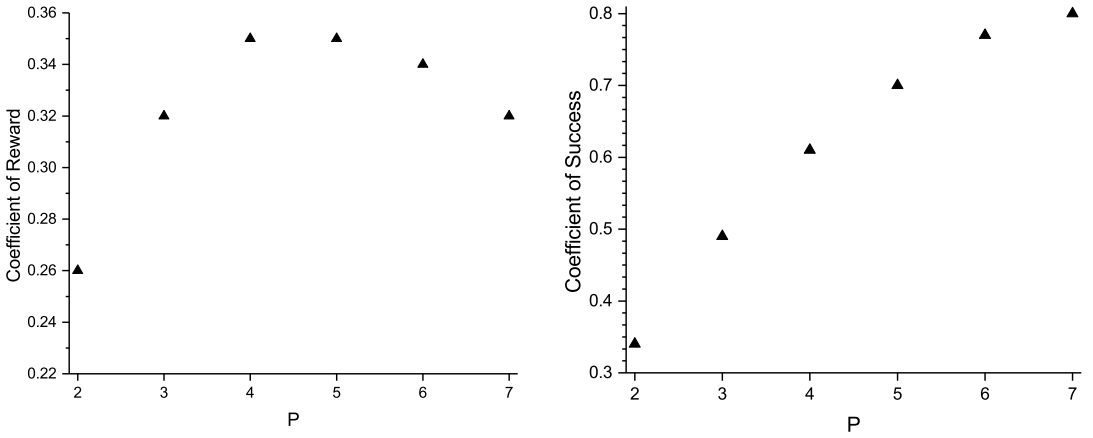
(S3) Train LEAPS2 on all 264 data sets and then evaluate its performance on the same data sets.

(S4) Generate 10 data sets each from test functions F1, F6, F11, F16, F36, F46, F51, F66 using Sobol sampling with different seeds. This results in total 80 data sets. Evaluate the performance of LEAPS2 from S3 on these data sets.

(S5) Expand the data set directory by adding the 80 data sets from S4 to the initial 264 sets and retrain LEAPS2. Evaluate its performance on all 344 data sets.

Note that we optimize P at each step to determine P^* for which LEAPS2 exhibits the best performance.

After training LEAPS2 in S1, the first task is to determine P^* for the recommendation protocol. For this, we graphically solve the optimization problem in Eq. (43) to get $P^* = 5$. Figure 4 illustrates the selection of P^* for S1 and we use the same approach for all the subsequent steps. We then evaluate the recommendation performance of LEAPS2 on the 185 data sets using $P^* = 5$ and it attains DoS1 $= 0.70$, DoS2 $= 0.20$, and DoS3 $= 0.04$. In other words, LEAPS2 is unable to recommend one of the top three true best surrogates for only 6% of the data sets. We now proceed to S2 where we test the performance of LEAPS2 on the remaining 79 unlearned data sets. We get $P^* = 6$ for the recommendation protocol and the distribution of DoS for S2 is: DoS1 $= 0.73$, DoS2 $= 0.15$, DoS3 $= 0.04$. Clearly, LEAPS2 performs very well on both the training and testing data sets separately.



(a) *CoR vs.* P **(b)** *CoS vs.* P

Figure 4: *Optimization of* P *for the recommendation protocol of LEAPS2 in S1.*

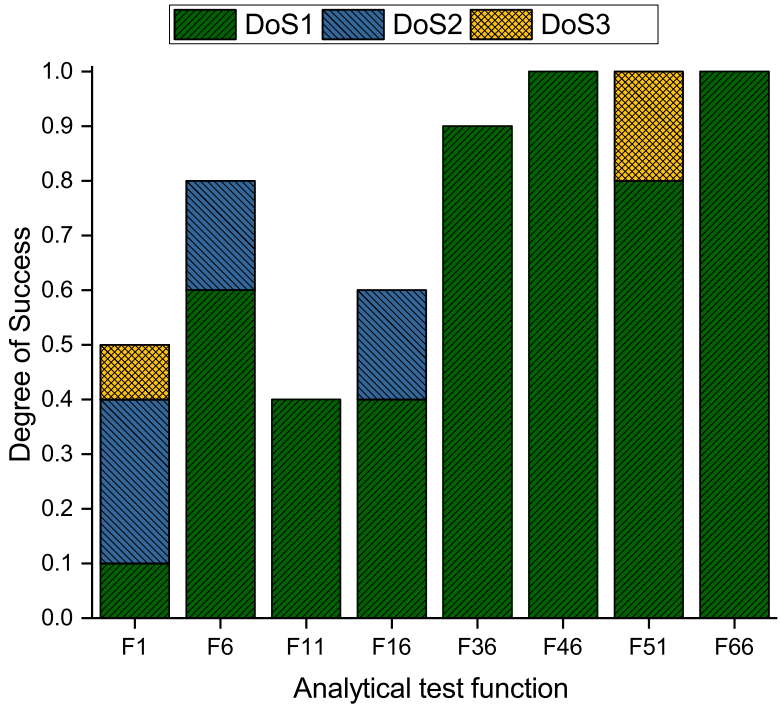


Figure 5: *Distribution of degree of success for the evaluation of LEAPS2 across various functions.*

In S3, we retrain LEAPS2 using all 264 data sets and evaluate its performance for $P^* = 5$. It shows an excellent performance with the total DoS (DoS1 + DoS2 + DoS3) of 0.96. We now evaluate the resilience of this fully trained LEAPS2 by evaluating its performance across multiple data sets from the same system as stated earlier in S4. Figure 5 shows its distribution of DoS for all the eight functions. It achieves an average DoS1 $= 0.65$, DoS2 $= 0.10$, and DoS3 $= 0.04$. In other words, LEAPS2 can successfully identify the first true best surrogate for 7 out of 10 data sets while the second true best surrogate for 1 out of 10 data sets from the same system. This highlights the resilient performance of LEAPS2 across the 8 test functions.

Our goal is to update LEAPS2 as when new data become available. Thus, we add the 80 data sets from S4 to the initial directory and retrain LEAPS2 with all 344 data sets. We get $P^* = 4$ and evaluate the performance of LEAPS2 across all the data sets. Table 9 summarises the training and evaluation performance of LEAPS2 in all the five steps. Overall, it achieves average DoS1 $= 0.71$, DoS2 $= 0.14$, and DoS3 $= 0.05$ across all the five steps.

The learning and evolution of LEAPS2 towards a powerful system modelling tool can occur along three dimensions, *viz.* data sets, attributes, and surrogates. Our progressive development of LEAPS2 in this work shows its evolution with respect to data sets. As we added more and more data sets progressively, LEAPS2 learnt. This is evident from the decreasing P^* through S1 to S5 as seen in Table 9. The lower value of P^* reveals that LEAPS2 is able to recommend the best surrogates with higher certainty and lower computational efforts.

Table 9: *Learning of LEAPS2 from S1 to S5.*

Description	S1	S2	S3	S4	S5
Training data sets	185	185	264	264	344
Evaluation data sets	185	79	264	80	344
P^*	5	6	5	3	4
DoS1	0.70	0.73	0.80	0.65	0.69
DoS2	0.20	0.15	0.10	0.10	0.17
DoS3	0.04	0.04	0.06	0.04	0.07

The evolution of LEAPS2 over attributes can be spawned by incorporating more attributes to the current set of 18 attributes. Similarly, its evolution over surrogates can be affected by adding more surrogates to its current database of 25 surrogates. This will require us to update our system-surrogate information by computing the new attributes and by constructing and evaluating the new surrogate for all its data sets (currently 344). Then, LEAPS2 must be retrained using the updated system-surrogate information. The addition of the new attributes will equip LEAPS2 for better system identification while the new surrogates will provide more options for system approximation. Overall, such learning will impart more versatility to LEAPS2 and extend its ability to handle more complex systems.

Next, we demonstrate the practical utility of LEAPS2 on a real-life case study.

6 Surrogates for the VLE properties of LNG

Natural gas (NG), the cleanest fossil fuel available, has gained a lot of attention due to pressing environmental issues, need for energy sustainability, and rapidly increasing global competition [25]. Typically, NG is transported over long distances as liquefied natural gas (LNG). However, some of the LNG inevitably boils off during loading, unloading, and transportation, requiring a keen understanding of its vapour-liquid equilibrium (VLE) for modelling LNG systems. This understanding is crucial in the operational modelling and optimization of LNG systems [10, 20, 25, 39]. Typically, these works rely on process simulators (*e.g.* Aspen HYSYS) for thermodynamic properties and numerical software (*e.g.* MATLAB) for solving optimization problems. Their approach necessitates an inter-

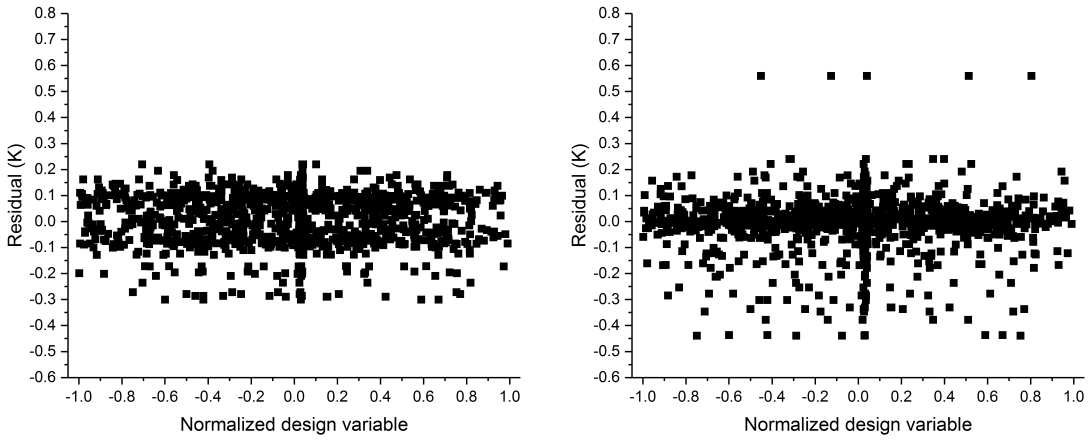
Table 10: *Design variables and their bounds for approximating the BPT and DPT of LNG.*

Design variables	Lower bound	Upper bound
CH_4 (Mole fraction)	0.7330	0.9440
C_2H_6 (Mole fraction)	0.0040	0.0930
C_3H_8 (Mole fraction)	0.0040	0.0940
N_2 (Mole fraction)	0.0200	0.0480
P (Pressure (kPa))	101.1	300

Table 11: *Performance metrics of the surrogates suggested by LEAPS2 for approximating the BPT and DPT of LNG based on a uniformly distributed set of 250 test points.*

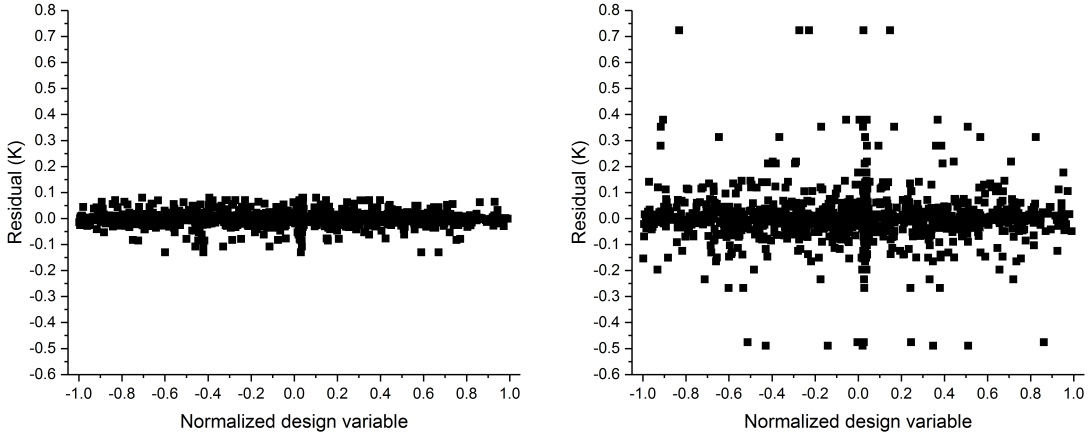
BPT				DPT			
Surrogates	**AAE**	**RMSE**	**PE**	**Surrogates**	**AAE**	**RMSE**	**PE**
P2	0.08	0.10	0.09	K1e	0.60	0.93	0.75
K1e	0.06	0.10	0.08	K0g	0.26	0.60	0.40
K2e	0.02	0.03	0.02	K1g	0.20	0.37	0.27
K0g	0.05	0.10	0.07	K2g	0.16	0.27	0.21

face between the black-box simulator and the numerical software, and repeated calls to the simulator for the physical properties. Such interfacing causes instability and much longer compute times. These issues can be ameliorated by employing surrogates to approximate the VLE properties of LNG within the numerical software. Therefore, we use LEAPS2 to recommend surrogates for approximating the VLE data of LNG obtained from Aspen



(a) *Residuals for P2*

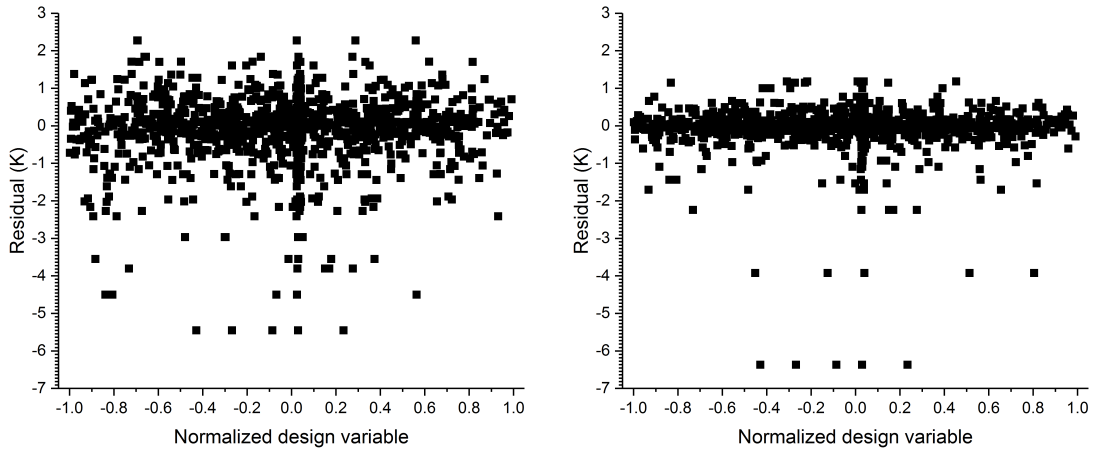
(b) *Residuals for K1e*



(c) *Residuals for K2e*

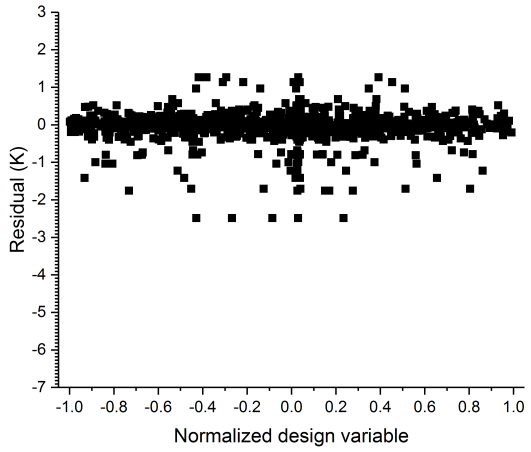
(d) *Residuals for K0g*

Figure 6: *Residuals for* 4 *surrogates recommended by LEAPS2 to approximate BPT.*

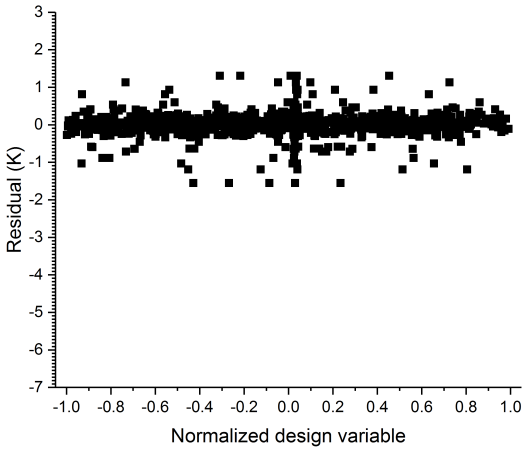


(a) *Residuals for K1e*

(b) *Residuals for K0g*



(c) *Residuals for K1g*



(d) *Residuals for K2g*

Figure 7: *Residuals for* 4 *surrogates recommended by LEAPS2 to approximate DPT.*

HYSYS.

LNG consists of CH_4, C_2H_6, C_3H_8, C_4H_{10}, and N_2. We wish to approximate its bubble point (BPT) and dew point temperatures (DPT) as functions of composition and pressure.

Table 12: *System attributes computed using input-output data set of* $K = 1000$ *for BPT and DPT of LNG.*

BPT				**DPT**			
N	5	A10	1856.65	N	5	A10	4595.72
K	1000	A11	2629.51	K	1000	A11	7636.78
A1	111.33	A12	-1.48E19	A1	221.67	A12	-3.44E21
A3	111.09	A13	4.40E20	A3	221.17	A13	7.32E22
A4	5.17	A14	22.98	A4	10.37	A14	51.97
A8	265.99	A15	0.32	A8	717.15	A15	0.30
A9	329.63	A16	0.30	A9	1103.39	A16	0.29

Table 10 shows the five design variables and their bounds derived from the typical LNG compositions. We then generate an input sample set of design variables and compute their BPTs and DPTs using Aspen HYSYS. For these data, LEAPS2 recommends the surrogates in Table 11 using $P^* = 4$. K2e is the best surrogate for BPT with PE = 0.02, and K2g is the best for DPT with PE = 0.21.

Interestingly, the best PE for DPT is much higher than that for BPT, which suggest that DPT is harder to approximate than BPT. A possible reason for the difference in PE could be the different magnitudes of BPT and DPT, as evidenced from their A1 values in Table 12. Another reason could be the higher A4 (10.4 vs. 5.2) for DPT than BPT. Finally, the higher values of A8 to A11 (Table 12) for DPT also suggest that it is much more non-linear than BPT.

For a visual comparison of various surrogates, we plot their residuals (Eq. (44)) shown in Figures 6 and 7.

$$res^{(q)} = y\left(x^{(q)}\right) - S\left(x^{(q)}\right) \; \forall \, q = 1, 2, \cdots Q \tag{44}$$

The residuals for each surrogate (Figures 6a-7d) are plotted with respect to all the five normalized design variables. The superior approximations of K2e and K2g are clear from their narrower spreads in Figure 6d and 7d respectively.

7 Comparison with CRS

Finally, we compare the performance of LEAPS2 with the three metalearning-based recommendation schemes (CRS-ANN, CRS-1 NN, and CRS-3 NN) of Cui et al. 8. As discussed earlier in Section 1, CRS in contrast to LEAPS2 recommends a surrogate modelling technique rather than a surrogate model. Furthermore, CRS schemes are based on total 44 ten dimensional data sets of which 98% were used for training and the remaining 2% for evaluation. On the contrary, LEAPS2 is based on 264 data sets with wide ranges of dimensions and sizes for training and validation. Therefore, a straightforward comparison is not entirely fair. However, we can still compare CRS and LEAPS2 based on their success in identifying the best modelling technique. To this end, we use the LEAPS2 from S3 (see Section 5) to recommend the best modelling techniques for all 264 data sets. LEAPS2 identifies the the true best for 236 data sets, thus attaining CoS $= 0.89$. Table 13 compares this CoS with those reported by Cui *et al.* for their three schemes. LEAPS2 offers higher CoS than the three CRS schemes showing superiority despite versatility.

Table 13: *Comparison between the performances of LEAPS2 and CRS using CoS for recommending modelling techniques.*

Recommendation scheme	CoS
LEAPS2	0.89 (236/264)
CRS-ANN	0.86 (38/44)
CRS-1 NN	0.82 (36/44)
CRS-3 NN	0.84 (37/44)

8 Conclusions and future works

In this work, we develop a surrogate recommendation paradigm namely LEAPS2 which selects the best surrogate/s for a underlying physico-numerical system simply based on its input-output data. The novel core philosophy of our paradigm embraces the concept of knowledge pyramid that hierarchically reduces data volume while enhancing its efficacy by exploiting data-information-knowledge inter-linkage. By progressively adding more data, we demonstrate that LEAPS2 learns to attain better computational efficiency and functional accuracy. Moreover, the architecture of LEAPS2 allows its evolution by including more attributes and surrogates. Besides, it performs better than the literature approaches in recommending modelling techniques despite its generalized philosophy. Finally, the practical utility of LEAPS2 is demonstrated by successfully employing it to recommend surrogates for estimating BPT and DPT of LNG system. Interestingly, LEAPS2 recommends a different surrogate for each temperature, and suggests that the latter may be more non-linear and thus harder to approximate than the former. Overall, LEAPS2 is a comprehensive, practical, data-driven surrogate recommendation scheme that performs very well for a wide range of systems.

This is just a first step towards developing an intelligent tool for system modelling and we aim to march on several fronts in the future. First, we aim to incorporate more analytical functions to the existing directory as LEAPS2 continuously evolves with the addition of new data. Second, we will develop and add more system attributes to LEAPS2 for better system identification. Lastly, we will integrate the complexity of surrogates to the existing performance metrics that will give an additional objective to users for surrogate selection. In a nutshell, these future tasks will further empower LEAPS2 to tackle more complex problems efficiently.

Acknowledgement

This publication is made possible by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

Nomenclature

Abbreviations

AAE: average absolute error
ANN: artificial neural network
BPT: bubble point temperature
CoCS: coefficient of computational savings
CoR: coefficient of reward
CoS: coefficient of success
CRS: Cui *et al.* recommendation scheme
DoS: degree of success
DPT: dew point temperature
LNG: liquefied natural gas
MARS: multivariate adaptive regression splines
NG: natural gas
PE: pooled error
PRSM: polynomial response surface model
QS: Sobol sampling
RBF: radial basis functions
RMSE: root mean squared error
SVM: support vector machine
SVR: support vector regression
VLE: vapour liquid equilibrium

Notation

Subscripts

b: index for basis functions in MARS
d: index for splits of hinge functions in MARS
m: index for elements of response/output variables' vector
n: index for elements of design/input variables' vector
p: index for input variables in PRSM expansion
r: index for elements in a set of subregions

Superscripts

j: index for elements of a set
k: index for elements of a set
q: index for elements in test set
L: lower bound
U: upper bound

Parameters

ϵ: threshold parameter in SVR
η_g: positive parameter in Gaussian RBF
η_m: positive parameter in multi-quadratic and inverse multi-quadratic RBF
θ_n: correlation parameter for n-th variable in kriging
ρ_{MARS}: maximum order of expansion for hinge function in MARS
ρ_{PRSM}: maximum order of expansion in PRSM
ρ_{SVR}: maximum order of polynomial kernel function in SVR
e: small positive number depending on the number of significant digits
D_b: total number of splits in b-th hinge function in MARS
K: total number of sample points in a sample set
L_H: number of neurons in the hidden layer of ANN
L_I: number of neurons in the input layer of ANN
M: total number of output/response variables
N: total number of input/design variables
P: internal parameter of LEAPS2
P_{max}: total number of surrogate options available for selection
Q: test set size
R_s: total number of subregions in $\mathcal{D}$

Continuous Variables

x: vector of input/design variables
y: vector of output/response variables

Symbols

α: vector of dual variables in SVR
α^*: vector of dual variables in SVR
β_0: constant term in polynomial expansion
β_b: coefficient of b-th hinge function in MARS
β_n: coefficient of the first order term of the n-th variable in polynomial expansion
β_{nn}: coefficient of the second order term of the n-th variable in polynomial expansion
β_{np}: coefficient of the second order interaction between n-th and p-th variables in polynomial expansion
γ_1: Fisher-Pearson coefficient of skewness
γ_2: Kurtosis of responses
δ: transfer function in ANN
ε: a random error with zeros mean
ζ_0: bias of the output neuron in ANN
ζ_j: bias of the j-th hidden neuron in ANN
$\overline{\kappa}$: empirical mean of curvature
λ: coefficients of basis functions in RBF
$\nu_{n(d,b)}$: location of the knot for n-th variable in d-th split of b-th hinge function in MARS

σ^2_{KRG}: variance of random process in kriging
$\upsilon_n^{(k)}$: n-th eigen value of $\mathcal{H}^{(k)}$
ψ: radial basis function
ω_{ij}: weight factor of the connection between i-th input neuron and j-th hidden neuron in ANN
ω_j: weight factor of the connection between j-th hidden neuron and the output neuron in ANN
c: centre of hyper-spherical sub-region $\mathcal{D}_r$
d: diameter of hyper-spherical sub-region $\mathcal{D}_r$
f: computationally expensive black-box system
f_b: global model in kriging
f_k: kernel function in SVR
$g^{(k)}$: estimate of a gradient at $x^{(k)}$
h: hinge function in expansion of MARS
$s_{d,b}$: sign function denoting the right/left sense of the step function in MARS
s_κ: empirical standard deviation in the curvature
s_{g_n}: empirical mean of the standard deviations in gradient estimates
s_{s_g}: empirical standard deviation in the standard deviations in gradient estimates
s_y: empirical standard deviation in response y
t_p: polynomial tail function in RBF
$\overline{g}$: empirical mean of the gradient estimates
$s_{\overline{g}}$: empirical standard deviation in gradient estimates
$\overline{y}$: empirical mean of the responses
$\overline{y}_g$: empirical geometric mean of the responses
$\overline{y}_h$: empirical harmonic mean of the responses
C_v: empirical coefficient of variation
R: correlation function in kriging
$R_{\text{min/max},\mathcal{D}}$: fluctuation in $\mathcal{D}$
$\overline{R}_{\text{min/max},\mathcal{D}_r}$: mean fractional fluctuation over all subregions
$R_{\text{min/max},\mathcal{D}_r}$: local fluctuation in subregion r
S: surrogate model
S_{ANN}: ANN surrogate
S_{HDMR}: HDMR surrogate
S_{KRG}: kriging surrogate
S_{MARS}: MARS surrogate
S_{PRSM}: PRSM surrogate
S_{RBF}: RBF surrogate
S_{SVR}: SVR surrogate
$Z(x)$: random process over x
$\mathbb{E}$: expectation
$\mathbb{N}$: set of natural numbers
$\mathbb{R}$: set of real numbers
$\mathcal{COV}$: covariance matrix of random process in kriging
$\mathcal{D}$: input domain
$\mathcal{D}_r$: r-th subregion in the domain
$\mathcal{H}^{(k)}$: estimate of a Hessian at $x^{(k)}$

$\mathcal{Q}$: uniformly distributed test set of size Q
$\mathcal{R}$: correlation matrix in kriging
$\mathcal{T}$: set of surrogate modeling techniques
$\mathcal{X}^{(K)}$: sample set of size K
$\mathcal{Y}^{(K)}$: response set of size K
A: ANN surrogate
DoS1: first degree of success
DoS2: second degree of success
DoS3: third degree of success
K0e: kriging surrogate with zeroth order global function and exponential correlation function
K1e: kriging surrogate with first order global function and exponential correlation function
K2e: kriging surrogate with second order global function and exponential correlation function
K0g: kriging surrogate with zeroth order global function and Gaussian correlation function
K1g: kriging surrogate with first order global function and Gaussian correlation function
K2g: kriging surrogate with second order global function and Gaussian correlation function
K0l: kriging surrogate with zeroth order global function and linear correlation function
K1l: kriging surrogate with first order global function and linear correlation function
K2l: kriging surrogate with second order global function and linear correlation function
K0s: kriging surrogate with zeroth order global function and spherical correlation function
K1s: kriging surrogate with first order global function and spherical correlation function
K2s: kriging surrogate with second order global function and spherical correlation function
K0c: kriging surrogate with zeroth order global function and cubic spline correlation function
K1c: kriging surrogate with first order global function and cubic spline correlation function
K2c: kriging surrogate with second order global function and cubic spline correlation function
M: MARS surrogate
P1: first order polynomial surrogate
P2: second order polynomial surrogate
Rb: RBF surrogate with biharmonic basis function
Rm: RBF surrogate with multi-quadratic basis function
Ri: RBF surrogate with inverse multi-quadratic basis function
Rt: RBF surrogate with thin plate basis function
Rg: RBF surrogate with Gaussian basis function
S: SVR surrogate

A Analytical test functions

This appendix provides the test functions used throughout the article for training and evaluation of LEAPS [45]. Tables A-I-A-IV list analytical test functions with following characteristics: (i) multi-modality, (ii) bowl-shaped, (iii) plate-shaped, and (iv) ridges-shaped. Note that we use the following parameters for function computation wherever applicable.

- F16[a]: The user must define $L \in \mathbb{N}$, c a vector of size N, and a matrix $\mathcal{A}$ of size $L \times N$.
- F17-F22[b]: $c_n = 1 + \frac{x_n - 1}{4}$
- F61-F66[c]: $L = 10$

Table A-I: *Analytical functions with multimodality for evaluation of LEAPS.*

Legend	N	Test Function	Domain Bounds
F1	3	$-20\exp\left(-0.20\sqrt{\frac{1}{N}\sum_{n=1}^{N}x_n^2}\right)$	$[-32.768, 32.768]^N$
F2	6		
F3	9		
F4	12		
F5	15		
F6	18		
F7	2	$-\frac{1+\cos\left(12\sqrt{x_1^2+x_2^2}\right)}{2+0.50\left(x_1^2x_2^2\right)}$	$[-5.12, 5.12]^2$
F8	2	$-(x_2+47)\sin\left(\sqrt{\left\|x_2+\frac{x_1}{2}+47\right\|}\right)-x_1\sin\left(\sqrt{\left\|x_1-(x_2+47)\right\|}\right)$	$[-512, 512]^2$
F9	2	$\left(\sum_{i=1}^{5} i\cos((i+1)x_1+1)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2+i)\right)$	$[-5.12, 5.12]^2$
F10	4	$1+\sum_{n=1}^{N}\frac{x_n^2}{4000}-\prod_{n=1}^{N}\cos\left(\frac{x_n}{\sqrt{n}}\right)$	$[-600, 600]^N$
F11	7		
F12	10		
F13	13		
F14	16		

Table A-I: *Analytical functions with multimodality for evaluation of LEAPS.*

Legend	N	Test Function	Domain Bounds
F15	2	$-0.0001\left(\left\lvert\sin(x_1)\sin(x_2)\exp\left(\left\lvert 100-\frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert\right)\right\rvert+1\right)^{0.10}$	$[-10,10]^2$
F16[a]	2	$\sum_{l=1}^{L} c_l \exp\left(-\frac{1}{\pi}\sum_{n=1}^{2}\left(x_n-\mathcal{A}_{ln}\right)^2\right)\times\cos\left(\pi\sum_{n=1}^{2}\left(x_n-\mathcal{A}_{ln}\right)^2\right)$	$[0,10]^2$
F17[b]	5	$(\sin(\pi c_1))^2+\sum_{n=1}^{N-1}(c_n-1)^2\left[1+10\times\sin\left(\pi x_n+1\right)^2\right]+(c_n-1)^2\left[1+(\sin(2\pi c_n))^2\right]$	$[-10,10]^N$
F18[b]	8		
F19[b]	11		
F20[b]	14		
F21[b]	17		
F22[b]	20		
F23	2	$(\sin(3\pi x_1))^2+(x_1-1)^2\left[1+(\sin(3\pi x_2))^2\right]+(x_2-1)^2\left[1+(\sin(2\pi x_2))^2\right]$	$[-10,10]^2$
F24	4	$10N+\sum_{n=1}^{N}\left[x_n^2-10\cos(2\pi x_n)\right]$	$[-5.12,5.12]^N$
F25	7		
F26	10		
F27	13		
F28	16		
F29	19		

Table A-I: *Analytical functions with multimodality for evaluation of LEAPS.*

Legend	N	Test Function	Domain Bounds
F30	2	$0.50 + \frac{\sin\left(x_1^2 - x_2^2\right)^2 - 0.50}{\left[1 + 0.001(x_1^2 + x_2^2)\right]^2}$	$[-100, 100]^2$
F31	2	$0.50 + \frac{\cos\left(\sin\left(\left\|x_1^2 - x_2^2\right\|\right)\right) - 0.50}{[1 + 0.001(x_1^2 + x_2^2)]^2}$	$[-100, 100]^2$
F32	4	$418.98 \times N - \sum_{n=1}^{N} x_n \sin\left(\sqrt{\lvert x_n \rvert}\right)$	$[-500, 500]^N$
F33	7		
F34	10		
F35	13		
F36	16		
F37	19		
F38	2	$-\left\lvert \sin(x_1)\cos(x_2)\exp\left(\left\lvert 1 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi} \right\rvert\right) \right\rvert$	$[-10, 10]^2$

Table A-II: *Bowl-shaped analytical functions for evaluation of LEAPS.*

Legend	N	Test Function	Domain Bounds
F39	2	$x_1^2 + 2x_2^2 - 0.30\cos(3\pi x_1) - 0.40\cos(4\pi x_2) - 0.70$	$[-100, 100]^2$
F40	2	$x_1^2 + 2x_2^2 - 0.30\cos(3\pi x_1) \times \cos(4\pi x_2) + 0.30$	$[-100, 100]^2$
F41	2	$x_1^2 + 2x_2^2 - 0.30\cos(3\pi x_1 + 4\pi x_2) + 0.30$	$[-100, 100]^2$
F42	2	$\sum_{n=1}^{2}\left(\sum_{p=1}^{2}(p+10)\left(x_p^n - \frac{1}{p^n}\right)\right)^2$	$[-2, 2]^N$

Table A-II: *Bowl-shaped analytical functions for evaluation of LEAPS.*

Legend	N	Test Function	Domain Bounds
F43	5	$\sum_{n=1}^{N} \sum_{p=1}^{n} x_p^2$	$[-65.54, 65.54]^N$
F44	8		
F45	11		
F46	14		
F47	17		
F48	20		
F49	5	$\sum_{n=1}^{N} x_n^2$	$[-5.12, 5.12]^N$
F50	8		
F51	11		
F52	14		
F53	17		
F54	20		

Table A-III: *Plate-shaped analytical functions for evaluation of LEAPS.*

Legend	N	Test Function	Domain Bounds
F55	2	$(x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	$[-10, 10]^2$
F56	2	$0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	$[-10, 10]^2$
F57	2	$\sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1$	$x_1 \in [-1.5, 4], x_2 \in [-3, 4]$

Table A-III: *Valley-shaped analytical functions for evaluation of LEAPS.*

Legend	N	Test Function	Domain Bounds
F58	2	$2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$	$x \in [-5, 5]^N$
F59	2	$\left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	$x_1 \in [-3, 3], x_2 \in [-2, 2]$

Table A-IV: *Ridges-shaped analytical functions for evaluation of LEAPS.*

Legend	N	Test Function	Domain Bounds
F60	2	$-\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2-(x_2-\pi)^2\right)$	$x \in [-100, 100]$
F61[c]	2	$-\sum_{n=1}^{N}\sin(x_n)\left(\sin\left(\frac{nx_n^2}{\pi}\right)\right)^{2L}$	$x \in [0, \pi]$
F62[c]	5		
F63[c]	8		
F64[c]	11		
F65[c]	14		
F66[c]	17		

References

[1] M. Bashiri and A. F. Geranmayeh. Tuning the parameters of an artificial neural network using central composite design and genetic algorithm. *Scientia Iranica*, 18 (6):1600–1608, 2011. doi:10.1016/j.scient.2011.08.031.

[2] R. A. Berk. Classification and regression trees (CART). In *Statistical learning from a regression perspective*, pages 129–186. Springer, 2016.

[3] A. Bhosekar and M. Ierapetritou. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering*, 108(Supplement C):250 – 267, 2018. ISSN 0098-1354. doi:https://doi.org/10.1016/j.compchemeng.2017.09.017.

[4] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM. ISBN 0-89791-497-X. doi:10.1145/130385.130401.

[5] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] P.-H. Chen, R.-E. Fan, and C.-J. Lin. A study on SMO-type decomposition methods for support vector machines. *IEEE Trans. Neural Networks*, 17(4):893–908, 2006. doi:10.1109/TNN.2006.875973.

[7] A. Cozad, N. V. Sahinidis, and D. C. Miller. Learning surrogate models for simulation-based optimization. *AIChE Journal*, 60(6):2211–2227, 2014. doi:10.1002/aic.14418.

[8] C. Cui, M. Hu, J. D. Weir, and T. Wu. A recommendation system for meta-modeling: A meta-learning based approach. *Expert Systems with Applications*, 46:33–44, 2016. doi:10.1016/j.eswa.2015.10.021.

[9] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[10] S. Effendy, M. S. Khan, S. Farooq, and I. A. Karimi. Dynamic modelling and optimization of an lng storage tank in a regasification terminal with semi-analytical solutions for N_2-free LNG. *Computers & Chemical Engineering*, 99:40–50, 2017. doi:doi.org/10.1016/j.compchemeng.2017.01.012.

[11] J. H. Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[12] S. S. Garud, I. A. Karimi, and M. Kraft. Design of computer experiments: A review. *Computers & Chemical Engineering*, 106:71 – 95, 2017. doi:10.1016/j.compchemeng.2017.05.010.

[13] S. S. Garud, I. A. Karimi, and M. Kraft. Smart sampling algorithm for surrogate model development. *Computers & Chemical Engineering*, 96:103–114, 2017. doi:10.1016/j.compchemeng.2016.10.006.

[14] S. S. Garud, I. A. Karimi, G. P. Brownbridge, and M. Kraft. Evaluating smart sampling for constructing multidimensional surrogate models. *Computers & Chemical Engineering*, 108(Supplement C):276 – 288, 2018. ISSN 0098-1354. doi:https://doi.org/10.1016/j.compchemeng.2017.09.016.

[15] J. Gergonne. The application of the method of least squares to the interpolation of sequences. *Historia Mathematica*, 1(4):439–447, 1974. doi:10.1016/0315-0860(74)90034-2.

[16] T. Goel, R. T. Haftka, W. Shyy, and N. V. Queipo. Ensemble of surrogates. *Structural and Multidisciplinary Optimization*, 33(3):199–216, 2007. doi:10.1007/s00158-006-0051-9.

[17] D. Gorissen, T. Dhaene, and F. D. Turck. Evolutionary model type selection for global surrogate modeling. *Journal of Machine Learning Research*, 10(Sep):2039–2078, 2009.

[18] R. L. Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research*, 76(8):1905–1915, 1971. ISSN 2156-2202. doi:10.1029/JB076i008p01905.

[19] S. S. Haykin. *Neural networks and learning machines*. Pearson Education, third edition, 2009.

[20] T. He, I. A. Karimi, and Y. Ju. Review on the design and optimization of natural gas liquefaction processes for onshore and offshore applications. *Chemical Engineering Research and Design*, 2018. ISSN 0263-8762. doi:https://doi.org/10.1016/j.cherd.2018.01.002.

[21] G. Jekabsons. Areslab: Adaptive regression splines toolbox for matlab/octave. `http://www.cs.rtu.lv/jekabsons`, 2016. 2017-07-6.

[22] G. Jekabsons. Rbf: Radial basis functions toolbox for matlab/octave. `http://www.cs.rtu.lv/jekabsons`, 2016. 2017-07-6.

[23] R. Jin, W. Chen, and T. Simpson. Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization*, 23(1):1–13, Dec 2001. ISSN 1615-1488. doi:10.1007/s00158-001-0160-4.

[24] S. Joe and F. Y. Kuo. Constructing sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing*, 30(5):2635–2654, 2008. doi:10.1137/070709359.

[25] M. S. Khan, I. Karimi, and D. A. Wood. Retrospective and future perspective of natural gas liquefaction and optimization technologies contributing to efficient lng supply: A review. *Journal of Natural Gas Science and Engineering*, 2017. doi:doi.org/10.1016/j.jngse.2017.04.035.

[26] J. P. Kleijnen. Kriging metamodeling in simulation: A review. *European journal of operational research*, 192(3):707–716, 2009. doi:10.1016/j.ejor.2007.10.013.

[27] J. Koehler and A. Owen. 9 Computer experiments. *Handbook of statistics*, 13: 261–308, 1996. doi:10.1016/S0169-7161(96)13011-X.

[28] I. Kononenko, E. Šimec, and M. Robnik-Šikonja. Overcoming the myopia of inductive learning algorithms with RELIEFF. *Applied Intelligence*, 7(1):39–55, 1997. doi:doi.org/10.1023/A:1008280620621.

[29] E. Kreyszig. *Advanced engineering mathematics*. John Wiley & Sons, 2010.

[30] R. L. Lawrence and A. Wright. Rule-based classification systems using classification and regression tree (CART) analysis. *Photogrammetric engineering and remote sensing*, 67(10):1137–1142, 2001.

[31] S. Lessmann, R. Stahlbock, and S. F. Crone. Genetic algorithms for support vector machine model selection. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 3063–3069. IEEE, 2006. doi:10.1109/IJCNN.2006.247266.

[32] D. Lim, Y.-S. Ong, Y. Jin, and B. Sendhoff. A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1288–1295. ACM, 2007. doi:10.1145/1276958.1277203.

[33] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. DACE - a matlab kriging toolbox, version 2.0, 2002. URL `http://www2.imm.dtu.dk/pubdb/p.php?3213`.

[34] G. Matheron. Principles of geostatistics. *Economic geology*, 58(8):1246–1266, 1963. doi:10.2113/gsecongeo.58.8.1246.

[35] MoDS. Description of the MoDS toolkit. `http://www.cmclinnovations.com/`, 2017. 2017-06-16.

[36] M. Packianather, P. Drake, and H. Rowlands. Optimizing the parameters of multilayered feedforward neural networks through taguchi design of experiments. *Quality and reliability engineering international*, 16(6):461–473, 2000. doi:10.1002/1099-1638(200011/12)16:6¡461::AID-QRE341¿3.0.CO;2-G.

[37] M. J. Powell. *The theory of radial basis function approximation*. University of Cambridge. Department of Applied Mathematics and Theoretical Physics, 1990.

[38] A. M. Prasad, L. R. Iverson, and A. Liaw. Newer classification and regression tree techniques: bagging and random forests for ecological prediction. *Ecosystems*, 9(2): 181–199, 2006. doi:10.1007/s10021-005-0054-1.

[39] H. N. Rao, K. H. Wong, and I. A. Karimi. Minimizing power consumption related to BOG reliquefaction in an LNG regasification terminal. *Industrial & Engineering Chemistry Research*, 55(27):7431–7445, 2016. doi:10.1021/acs.iecr.6b01341.

[40] A. Rényi. On measures of entropy and information. Technical report, HUNGARIAN ACADEMY OF SCIENCES Budapest Hungary, 1961.

[41] M. Robnik-Šikonja and I. Kononenko. An adaptation of relief for attribute estimation in regression. In *Machine Learning: Proceedings of the Fourteenth International Conference (ICML97)*, pages 296–304, 1997.

[42] E. Sanchez, S. Pintos, and N. V. Queipo. Toward an optimal ensemble of kernel-based approximations with engineering applications. *Structural and multidisciplinary optimization*, 36(3):247–261, 2008. doi:10.1007/s00158-007-0159-6.

[43] J. J. Sikorski, G. Brownbridge, S. S. Garud, S. Mosbach, I. A. Karimi, and M. Kraft. Parameterisation of a biodiesel plant process flow sheet model. *Computers & Chemical Engineering*, 95:108 – 122, 2016. doi:10.1016/j.compchemeng.2016.06.019.

[44] M. Streeter and L. A. Becker. Automated discovery of numerical approximation formulae via genetic programming. *Genetic Programming and Evolvable Machines*, 4(3):255–286, 2003. doi:10.1023/A:1025176407779.

[45] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved November 9, 2017, from `http://www.sfu.ca/~ssurjano`, 2017.

[46] J. A. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999. doi:doi.org/10.1023/A:101862860.

[47] The MathWorks Inc. Documentation of regression tree ensembles. `https://www.mathworks.com/help/stats/regression-trees.html`, 2017. 2017-10-20.

[48] G. G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical design*, 129(4):370–380, 2007. doi:10.1115/1.2429697.

[49] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999. doi:10.1109/5.784219.

[50] Y.-S. Yeun, W.-S. Ruy, Y.-S. Yang, and N.-J. Kim. Implementing linear models in genetic programming. *IEEE transactions on evolutionary computation*, 8(6):542–566, 2004. doi:10.1109/TEVC.2004.836818.