Computers and Chemical Engineering 140 (2020) 106847

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Surrogate-based optimization for mixed-integer nonlinear problems

Sun Hye Kim, Fani Boukouvala[*]

School of Chemical & Biomolecular Engineering, Georgia Institute of Technology, Atlanta, GA, USA

ARTICLE INFO

Article history:
Received 1 July 2019
Revised 27 March 2020
Accepted 30 March 2020
Available online 19 May 2020

ABSTRACT

Simulation-based optimization using surrogate models enables decision-making through the exchange of data from high-fidelity models and development of approximations. Many chemical engineering optimization problems, such as process design and synthesis, rely on simulations and contain both discrete and continuous decision variables. Surrogate-based optimization with continuous variables has been studied extensively; however, there are many open challenges for the case of mixed-variable inputs. In this work, we propose an algorithm for mixed-integer nonlinear simulation-based problems that uses adaptive sampling and surrogate modeling with one-hot encoding. We propose techniques for the design of experiments for mixed-variable problems, surrogate modeling for mixed-variable response surfaces, and iterative approximation-optimization procedure that leads to optimal solutions. Results show that one-hot encoding leads to accurate and robust mixed-variable Gaussian Process and Neural Network models that are effective surrogates for optimization. The proposed algorithm is tested on mixed-integer nonlinear benchmark problems and a chemical process synthesis case study.

1. Introduction

Many optimization problems today depend on highly complicated computer simulations, which provide accurate and useful data that represent complex physical phenomena (Amaran et al., 2016; Bhosekar and Ierapetritou, 2018; Boukouvala et al., 2016; McBride and Sundmacher, 2019; Rios and Sahinidis, 2013). These simulations are typically comprised of a large system of equations, such as partial differential and ordinary differential equations, to model processes and systems accurately. In certain cases, due to the large size and complexity of the simulation or the presence of discontinuities caused by periodic boundary conditions, a simulation-based optimization approach may be the most practical and efficient way to optimize these problems (Amaran et al., 2016; Bhosekar and Ierapetritou, 2018; Boukouvala et al., 2017; Cozad et al., 2014; McBride and Sundmacher, 2019). In chemical engineering, one can observe a significant growth on the interest in simulation-based, or black/gray-box, or derivative-free optimization (Balasubramanian et al., 2018; Beykal et al., 2020; Bhosekar and Ierapetritou, 2018; Boukouvala et al., 2017; Cozad et al., 2015; Davis and Ierapetritou, 2008; Davis et al., 2018; Dias and Ierapetritou, 2020; Garud et al., 2019, 2018; Graciano and Le Roux, 2013; Keßler et al., 2019; Kim and Boukouvala, 2019; McBride and Sundmacher, 2019; Mencarelli et al., 2020; Schweidtmann et al., 2019; Schweidtmann and Mitsos, 2018; Tso et al., 2019; Wilson and Sahinidis, 2019; Zantye et al., 2019). In this work, we will interchangeably use the term “black-box” or “surrogate-based” optimization to refer to problems that rely on simulation input-output data and the derivatives of the original model are not directly used by the optimization solver (also known as derivative-free optimization) (Boukouvala et al., 2016; Conn et al., 2009; Rios and Sahinidis, 2013).

Black-box optimization heavily relies on data generated from complex simulations instead of first principle models consisting of explicit analytical equations. Existing black-box optimization algorithms proposed in the literature can be divided broadly into three categories: sampling-based, surrogate-based, and stochastic or evolutionary methods (Bhosekar and Ierapetritou, 2018; Boukouvala et al., 2016; Rios and Sahinidis, 2013). These three methods do not involve direct computation of the derivatives of the objective or constraints of the simulation, but they differ in how the simulation data is used to find the optimal solution. Sampling-based methodologies involve generating a set of points to guide the search, and different algorithms use unique metrics to choose the next sampling locations in order to either refine the solution or explore other areas of the search space (Hooke and Jeeves, 1961; Nelder and Mead, 1965; Reeves, 1997). Surrogate-based algorithms involve constructing an approximation model that relates the sampled input-output data, which is then optimized directly. These approximation models are also known as meta- or reduced-order models (Bhosekar and Ierapetri-

[*] Corresponding author.
E-mail address: fani.boukouvala@chbe.gatech.edu (F. Boukouvala).

https://doi.org/10.1016/j.compchemeng.2020.106847
0098-1354/© 2020 Published by Elsevier Ltd.

tou, 2018; Boukouvala and Floudas, 2017; Cozad et al., 2014; Davis et al., 2018; Forrester and Keane, 2009; Garud et al., 2018; Keßler et al., 2019; Kim and Boukouvala, 2019; McBride and Sundmacher, 2019; Queipo et al., 2005). Stochastic or evolutionary-type algorithms solely rely on sampling large populations, but they differ from direct-search methods due to the presence of stochastic criteria to generate populations of samples scattered in the entire search space. Most popular examples of such algorithms are Genetic Algorithms (Holland, 1992), Particle Swarm Optimization (Eberhart and Kennedy, 1995; Kennedy and Eberhart, 1995), and Simulated Annealing (Romeo and Sangiovanni-Vincentelli, 1991). Of the three aforementioned categories, the surrogate-based optimization literature has attracted significant attention lately, and this is undoubtedly linked to the recent developments in Machine Learning (ML) (Hastie et al., 2009). Many researchers from diverse fields have observed that surrogate-based optimization is a very promising method (Booker et al., 1999; Boukouvala and Floudas, 2017; Yondo et al., 2018; Zhang et al., 2019), and this is mainly due to the ability of the surrogate models to expedite the search for global optima and reduce the sampling requirements.

In all of the aforementioned literature, the majority of contributions is motivated by nonlinear optimization problems (NLPs) with only continuous input or decision variables (Boukouvala and Floudas, 2017; Cozad et al., 2014; Zhai and Boukouvala, 2019). However, many chemical engineering problems contain both continuous and discrete (integer or binary) decision variables. For example, the design of a distillation column involves continuous variables for operating conditions and discrete variables for the number of stages. Similarly, a superstructure synthesis optimization problem contains binary variables to represent design configurations, while nonlinear relationships represent phenomena within the processes (Caballero and Grossmann, 2008; Davis and Ierapetritou, 2008; Graciano and Le Roux, 2013; Henao and Maravelias, 2011; Sangbum et al., 2003). In (Larson et al., 2019), a case study on the design of solar plants is discussed, in which the discrete decisions are embedded in the simulation. This leads to a simulation-based optimization problem that cannot be relaxed or decoupled with respect to discrete and continuous variables.

There are a few black-box mixed integer nonlinear programming (bb-MINLP) optimization algorithms proposed in the direct-search literature (Abramson et al., 2008; Deep et al., 2009; Liuzzi et al., 2012; Liuzzi et al., 2015) and in the surrogate-based literature (Holmström et al., 2008; Müller, 2016, 2013; Rashid et al., 2013), which are described in detail in the next section. Nevertheless, the optimization of bb-MINLPs is still a difficult problem due to several open challenges that are intrinsic to MINLP (Müller, 2016). The first challenge is obtaining a representative, tractable, and balanced sample set when both discrete and continuous variables are present. When all of the decision variables are continuous, space-filling sample designs (e.g., Latin hypercube (McKay et al., 1979), orthogonal arrays (Owen, 1992), and Sobol sequences (Sobol, 1967)) are used to generate balanced sample sets, and the simulation is inquired at these points. These sampling methods cannot be directly applied for bb-MINLP problems because the simulation may not provide output values at non-integral values of the discrete variables (Müller et al., 2013). Another challenge is surrogate model fitting in the case of mixed-variable inputs. Existing surrogate modeling algorithms for bb-MINLP (Holmström et al., 2008; Müller, 2016; Müller et al., 2013; Rashid et al., 2013) assume all variables are continuous in order to obtain a smooth and continuous surrogate model. At the same time, surrogate models assume that all input variables are ordinal, which means that a higher value corresponds to higher intensity, such as temperature or pressure levels. Binary variables do not satisfy this assumption as "0" and "1" usually represent different choices, as opposed to intensity. While this limitation can be overcome by using multiple surrogates and patching them at discontinuities (e.g., piecewise functions), this could complicate the surrogate-based optimization formulation significantly (Swiler et al., 2014).

A mixed-variable response surface introduced in (Swiler et al., 2014) will be used here to demonstrate the aforementioned challenges (Fig. 1). This response surface has one continuous variable (x) and one discrete variable (y) with three possible levels $y = [-2, 0, 2]$. When plotting the response surface for different levels of y, one can observe that the behavior of the output is very different (Fig. 1a). There are multiple ways to approximate and subsequently optimize this problem, such as (a) treat each level of y as an independent problem by fitting and optimizing separate surrogate models, or (b) assume continuity in all variables and fit a single continuous response surface with sparse sampling in the y direction. The first approach is possible in low dimensions but would become intractable as the number of discrete variables and levels increases. On the other hand, if continuity in all variables is assumed, this could lead to inaccurate surrogate models since there will be no samples in between non-integral values of the discrete variables. For the same example, if this black-box input-output relationship is assumed to be a 2D continuous function (Fig. 1b), the complexity of the surface becomes apparent. More specifically, at the middle level of y, the function has a very sudden and steep change in response; thus, assuming continuity when fitting this response surface may lead to inaccurate surrogate models. Most importantly, if the level values of y do not represent an intensity, then the assumption of continuity in y is problematic. In this work, we study various techniques to obtain a representative set of samples and fit appropriate surrogate models for mixed-variable optimization problems. The presence of nonlinear constraints, both inequality and equality, and non-convexity of the problems all pose further challenges for the solution of bb-MINLP problems (Boukouvala et al., 2016).

In this work, we aim to develop an algorithm to solve the following black-/gray-box MINLP (P1):

$$
\begin{aligned}
&\min f(\boldsymbol{x},\boldsymbol{y})\\
&s.t.\quad g_{IB}(\boldsymbol{x},\boldsymbol{y}) \le 0\\
&g_{IK}(\boldsymbol{x},\boldsymbol{y}) \le 0\\
&h_{EB}(\boldsymbol{x},\boldsymbol{y}) = 0\\
&h_{EK}(\boldsymbol{x},\boldsymbol{y}) = 0\\
&\boldsymbol{x}^l \le \boldsymbol{x} \le \boldsymbol{x}^u,\ \ \boldsymbol{y} \in \{0,1\}^{k_2}\\
&\boldsymbol{x} \in \mathbb{R}^{k_1},\ \ k = k_1 + k_2
\end{aligned}
\tag{P1}
$$

where x represents continuous variables, y represents binary variables, x^l and x^u represent the lower and upper bounds of the continuous variables, k_1 and k_2 represent the dimensions of continuous and binary variables respectively, $f(\cdot)$ represents black-box objective, $g_{IB}(\cdot)$ and $g_{IK}(\cdot)$ represent inequality constraints that are unknown (black-box) and known, respectively. Similarly, and $h_{EB}(\cdot)$ and $h_{EK}(\cdot)$ represent equality constraints that may be unknown and known, respectively. Sets IB and EB represent the set of black-box inequality and equality constraints, respectively. Similarly, sets IK and EK denote the sets of known inequality and equality constraints, respectively. If all constraints and objective are unknown (i.e., $IK = \emptyset$, $EK = \emptyset$), the problem will be referred as a black-box MINLP (bb-MINLP). If some constraints are known, the problem will be referred as a gray-box MINLP (gb-MINLP). $g_{IK}(x, y)$ and $h_{EK}(x, y)$ represent the known inequality and equality constraints, and these can be handled directly without constructing surrogate models.

Through this work, we aim to answer several key questions related to bb-MINLP and propose a new algorithm that can solve bb/gb-MINLP problems of moderate sizes (i.e., up to 15 variables

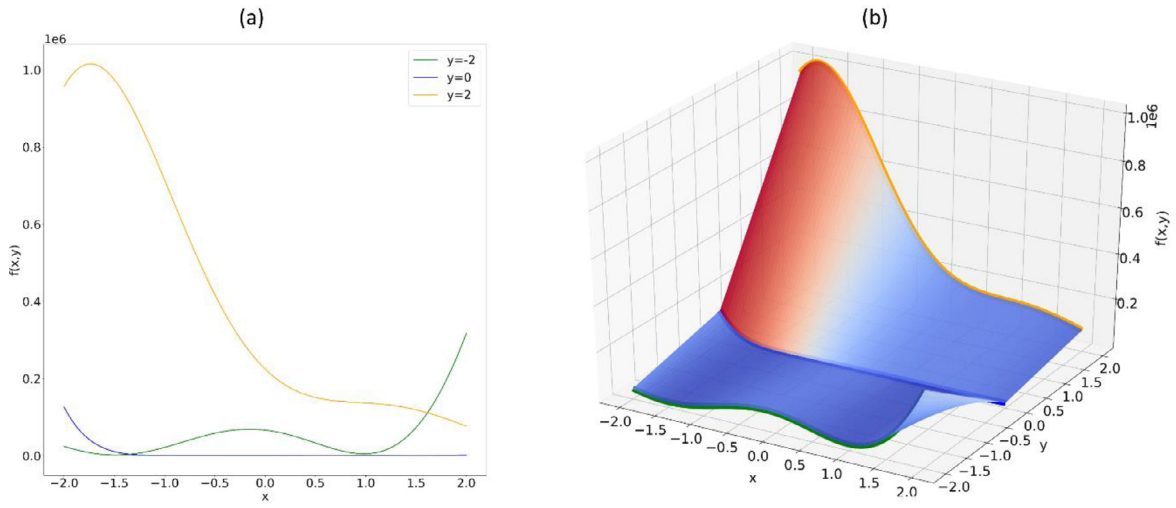


Fig. 1. Goldstein price function adapted from (Swiler et al., 2014). For each value of discrete variable y, the function exhibits a drastically different behavior. In (a) different y level values are plotted separately, in (b) the 2D surface is plotted.

and 23 constraints). First, to solve bb/gb- MINLP problems, we propose the use of mixed-integer surrogate models that can handle discrete variables directly, rather than relaxing the integrality constraints. Our framework utilizes, compares, and combines two types of surrogate models, namely Artificial Neural Network (ANN) and Gaussian Process (GP) models. A data-preprocessing technique, one-hot encoding, is used to address the modeling of mixed-variable problems, and the optimization problem is reformulated to reflect this transformation. In addition, we study the performance of three different sampling strategies for mixed-integer problems and propose the most appropriate method that balances solution accuracy and sampling requirements. Finally, we develop an algorithm that can solve both black- and gray-box formulations of (P1) using a hybrid combination of ANN and GP models for the MINLP and NLP stages, respectively. The performance of the algorithm is analyzed with respect to solution accuracy, sampling requirements, and computational efficiency and compared to those of two competing existing algorithms for bb-MINLP.

This paper is organized as follows. Section 2 introduces the necessary background on surrogate-based optimization and reviews some existing work on bb-MINLP. In Section 3, sampling, data-preprocessing, and surrogate modeling strategies for bb-MINLP are presented in detail. The overall proposed algorithm is described in Section 4 to illustrate how these strategies are integrated into the overall framework. Section 5 presents a comprehensive comparison of the proposed methodology on a set of benchmark problems. Finally, Section 6 introduces the surrogate formulation for a case study on superstructure optimization. We demonstrate how the MINLP problem can be decoupled into a gray-box problem and report the performance of our algorithm. A discussion of the findings is provided before the conclusions and future perspectives. The detailed formulations of the process synthesis case study are provided in the Appendix and detailed tables of all the results shown in this paper are provided as Supplementary Material.

2. Overview of surrogate-based optimization

2.1. Existing literature on derivative-free MINLP optimization

Existing work on derivative-free MINLP optimization algorithms can be divided into three broad categories: sampling-based/direct-search, model-based, and stochastic or evolutionary methods (Rios and Sahinidis, 2013). Direct-search bb-MINLP algorithms have been proposed as extensions to existing NLP direct-search methods (Abramson et al., 2008; Audet and Dennis, 2001; Cocchi et al., 2019; Larson et al., 2019; Giampaolo Liuzzi et al., 2015). One of the most popular existing software for constrained bb-MINLP is NOMAD (Nonsmooth Optimization by Mesh Adaptive Direct Search) adapted from (Audet and Dennis, 2006).

Due to their purely sampling-based nature, evolutionary-type methods (i.e., Genetic Algorithms (Reeves, 1997), Particle Swarm Optimization (Eberhart and Kennedy, 1995), and Simulated-Annealing (Romeo and Sangiovanni-Vincentelli, 1991)) could also be applied to solve (P1). The most widely used algorithm for bb-MINLP problems under this category is developed in a MATLAB “global optimization” toolbox, which employs a genetic algorithm (Deb, 2000; Deep et al., 2009). In addition, MEIGO (Egea et al., 2014) is an open source software tool that uses enhanced scatter search for global optimization of NLP and MINLP formulations.

There have also been a few developments for solving bb-MINLP problems in the surrogate-based optimization literature. Existing surrogate-based MINLP algorithms start with relaxing the discrete variables to create smooth surrogate functions. For example, SO-MI introduced in (Müller et al., 2013) and MI-SO (Müller, 2016) use a cubic radial basis function (RBF) model to solve expensive black-box problems. An RBF-based algorithm for mixed-integer nonlinear constrained optimization has been proposed in (Rashid et al., 2013) and (Holmström et al., 2008). Both methods have been shown to perform well for problems up to 8 binary and 4 continuous variables and less than 10 constraints. All of these aforementioned surrogate-based MINLP optimization algorithms do not handle discrete variables directly. Instead, the integrality constraint is relaxed to construct a smooth surrogate model, even if the simulation cannot be inquired at non-integral locations of the discrete variables. Recently, the use of gradient-boosted tree has been proposed for mixed-integer convex nonlinear optimization (Mistry et al., 2018). While this method handles discrete variables directly, the resulting gradient-boosted tree model is discontinuous. Additional relevant work from the process systems engineering community involve optimization of MINLP formulations with embedded surrogate functions (Caballero and Grossmann, 2008; Davis and Ierapetritou, 2008; Henao and Maravelias, 2011). However, in these contributions, the discrete variables are decoupled from the surrogate models. All surrogate models are only a function of continuous variables, and the trained surrogate models are embedded in the overall MINLP formulation.

In the approximation literature, several efforts have been made to study mixed-variable surrogate models that can directly han-

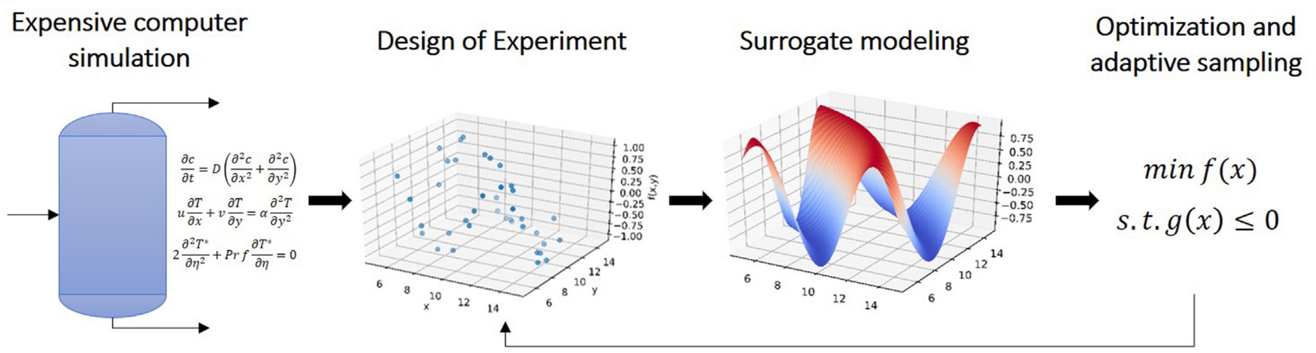

Expensive computer simulation
$\frac{\partial c}{\partial t} = D\left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2}\right)$
$u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} = \alpha\frac{\partial^2 T}{\partial y^2}$
$2\frac{\partial^2 T^*}{\partial \eta^2} + Pr\, f\frac{\partial T^*}{\partial \eta} = 0$
Design of Experiment
Surrogate modeling
Optimization and adaptive sampling
$min\, f(x)$
$s.t.\, g(x) \leq 0$

Fig. 2. General framework of adaptive surrogate-based optimization. Surrogate models approximate the data from expensive black-box simulations and are iteratively optimized and updated to find an optimal solution.

dle discrete variables (Gramacy and Lee, 2008; Qian et al., 2008; Swiler et al., 2014). These surrogate model techniques have been studied only with respect to approximation accuracy, and the optimization of these models has not been studied. This is very important because although certain functions might be accurate approximations, the formulation that needs to be embedded within the surrogate-based optimization problem may lead to intractable problems. For example, in (Qian et al., 2008), a Gaussian process with a special correlation function is proposed. This correlation function models interactions between discrete-discrete and discrete-continuous variables. We have found that optimizing this model requires many additional constraints to allow the selection of appropriate correlation coefficients, in addition to the number of constraints that increases proportionally with the number of samples that are used to construct the model. Thus, this leads to a very large surrogate bb-MINLP that is very challenging to optimize even with state-of-the-art deterministic optimization solvers. Another way to construct mixed-integer surrogate models is using regression trees as proposed in (Gramacy and Lee, 2008). Since regression tree methods involve dividing the search space into several partitions, this allows for a natural development of different regression functions for different realizations of discrete variables. For each partition, or a “node” of a tree, a Gaussian process model can be constructed. While this method is straightforward and easy to adapt, the resulting optimization problem is a piecewise function, which may require a generalized disjunctive programming formulation for its optimization. As these models were developed solely for the purpose of prediction, the optimization of these mixed-integer surrogate models may be infeasible or difficult. In this work, we have limited our study to methods that balance accuracy and tractability of formulation for optimization.

2.2. Overview of adaptive surrogate-based optimization

In this section, we provide an overview of the typical steps for adaptive surrogate-based optimization that will form the basis of our algorithm. The algorithmic steps will then be extended to handle specific challenges of bb-MINLP (Section 3). As computer simulations are becoming more and more computationally expensive, taking minutes, hours, or even days to generate one data point, an ideal black-box optimization algorithm should locate optimal solutions with a small number of function evaluations and within a reasonable computation time (Kim and Boukouvala, 2019). In order to locate an optimal solution, surrogate-based algorithms are based on the general notion that an initial limited set of samples can be used to develop an approximation model, and iterative optimization and resampling can be used to refine this model in promising locations. Surrogate-based algorithms generally consist of four steps: 1) initial sampling, 2) surrogate model construction, 3) optimization, and 4) adaptive sampling and optimization (Fig. 2).

First, the construction of a surrogate model begins with choosing an efficient sampling strategy to maximize the information gained while minimizing the number of samples. Samples should be uniformly distributed in the search space. Space-filling designs are based on the general concept that if we project the sample points onto each variable axis, no projections of the sample points will overlap. Latin Hypercube sampling (LHS) is one of the most commonly used type of non-collapsing space-filling design (McKay et al., 1979).

Next, a surrogate model type is selected to approximate the collected simulation data. Several types of surrogate models currently exist: Gaussian process (Boukouvala and Ierapetritou, 2013; Jones et al., 1998; Olofsson et al., 2018; Quirante et al., 2015; Rasmussen, 2004; Williams and Rasmussen, 1995), Radial Basis Functions (Chen et al., 1991; Müller et al., 2013; Regis and Shoemaker, 2005), Neural Networks (Henao and Maravelias, 2011; Schweidtmann and Mitsos, 2018; Specht, 1991), quadratic, and polynomial regression (Forrester and Keane, 2009; Hüllen et al., 2019), to name a few. The selection and parameter fitting of surrogate model are coupled with a *k*-fold cross-validation procedure to prevent overfitting. This procedure ensures that the surrogate model parameters are trained only on a subset of samples (training set), while the other subset of remaining samples is left out for validation (validation set). A prediction error on the validation set is calculated using the optimal model, and the procedure is repeated *k* times to allow the selection of the best model with the minimum cross-validation error to proceed to the optimization stage. Several approaches have been proposed for handling constraints, such as fitting a separate model for each constraint (Boukouvala and Floudas, 2017) or for a grouped penalty function (Ben-Tal and Zibulevsky, 1997; Deb, 2000). In this work, surrogate models are constructed for all black-box constraints and the objective function.

Finally, the surrogate formulation can be optimized using a derivative-based or deterministic optimization solver. However, regardless of the type of surrogate model used, it is unlikely that a highly accurate solution is obtained in just one iteration. This is mainly due to the limited number of samples collected in the first iteration. Hence, surrogate modeling is usually coupled with adaptive sampling to determine the location of next sampling points and to iteratively update the incumbent solution. An existing adaptive sampling approach determines the location of new samples in promising regions with the aim of improving the optimal solution, rather than constructing the best approximation over the entire search space (Boukouvala and Floudas, 2017; Jones et al., 1998; Kim and Boukouvala, 2019). This approach seeks to maintain a balance between diversity in sampling (i.e., exploration) and optimization (i.e., exploitation) of the feasible space and the objective function. As a result, the surrogate model is used only as an intermediate step to guide the search toward better directions and does not guarantee convergence (Boukouvala and Floudas, 2017; Jones et al., 1998; Kim and Boukouvala, 2019). In this work, we use the aforementioned adaptive sampling and optimization strategy (i.e., strategy that facilitates the search of optimum). Once the locations of new sample points have been determined, the simulation is re-inquired and the surrogate models are updated. The entire process repeats until certain convergence criteria are met.

3. Methods for surrogate-based bb-MINLP

3.1. Design of mixed-variable computer experiments

The accuracy of a surrogate model depends both on the number and the location of points. Hence, finding a good initial sampling design is an important step that highly affects the accuracy of the final solution (Bhosekar and Ierapetritou, 2018; Eason and Cremaschi, 2014; Forrester and Keane, 2009). When only continuous variables are present, LHS is typically used to generate an initial sampling design (Boukouvala and Floudas, 2017; Cozad et al., 2014; Kim and Boukouvala, 2019). However, for MINLPs, the standard way of obtaining a space-filling sampling design is an open question that we aim to study in this work.

To illustrate some existing sampling methodologies, let us define n_{lhs} to be the number of points selected for each LHS and m to be the total the number of discrete combinations. If all variables are continuous, the total number of LHS points is typically fixed to a number based on heuristics (e.g., $n_{lhs} = 10k_1 + 1$). If L_j represents the number of discrete levels for each discrete variable y_j (e.g., a binary variable y_j has two levels: $L_j = 2$), then $m = \prod_{j=1}^{k_2} L_j$. In (Müller, 2016; Müller et al., 2013), the LHS points corresponding to discrete variables are rounded to closest integers. In (Rashid et al., 2013) and (Holmström et al., 2008), an auxiliary problem is solved to eliminate infeasible samples. These approaches, however, do not guarantee that the same number of samples is collected for each discrete realization. Thus, this may lead to data imbalance: certain discrete combinations may contain more data points than the others, which may lead to inconsistency in the accuracy of the surrogate model for certain levels.

Swiler et al.,(2014) propose and compare different sampling techniques, such as standard Latin hypercube and k-Latin hypercube sampling for building accurate approximation models. When using the standard Latin hypercube approach, one LHS of size mn_{lhs} is generated in the continuous space, and the points are then randomly split into m groups of equal size sets, so that each group is assigned to a unique discrete level. In the k-Latin hypercube approach, a separate LHS of size n_{lhs} is generated for each discrete level. Both methods generate mn_{lhs} points. Although these methods have been compared for their approximation accuracy of low-dimensional functions, they have not been systematically compared with respect to their performance for surrogate-based optimization. In this work, we compare the performance of three different sampling strategies from the approximation and surrogate-based optimization literatures: k-Latin hypercube (Sampling Strategy 1), standard Latin hypercube (Sampling Strategy 2), and Latin hypercube sampling with simply rounding any discrete variables to their nearest integer value (Sampling strategy 3).

3.2. Surrogate modeling

While various surrogate models have been used in the literature, we only consider Artificial Neural Networks (ANN) and Gaussian Process (GP) models in this work. These two surrogate models have been widely used for surrogate modeling for NLP due to their accuracy and flexibility in representing a complicated nonlinear relationship between input and output data. A brief introduction to these two methods for continuous variables will be presented in this section, followed by a description on how these models can be modified to accommodate the presence of discrete variables.

3.2.1. Artificial neural network modeling

An Artificial Neural Network (ANN) is a nonlinear statistical model that has been used for both classification and regression (Hastie et al., 2009; Heaton, 2008). Following a standard ANN architecture, the input variable nodes represent the input layer, and the response variable nodes represent the output layer. The input and output layers are connected by hidden layers. The mathematical expression of an ANN with a single input node (x) and a single hidden layer can be expressed as follows:

$$\hat{f}_{NN}(x) = \sigma\left(\sum_l W_l^{(1)} \sigma\left(\sum_h W_h^{(0)} x + b^0\right) + b^{(1)}\right) \tag{1}$$

where h and l represent the number of nodes in hidden and output layers, respectively. The function σ is an activation function, which transfers the input of a node to an output, and $W^{(n)}$ and $b^{(n)}$ are the weights and biases for input-hidden ($W^{(0)}$ and $b^{(0)}$) and hidden-output ($W^{(1)}\, and\, b^{(1)}$) layers. The functional form of ANNs depends on the activation function, the number of hidden layers, and the number of nodes in each layer. One commonly used activation function is the hyperbolic tangent function ($\sigma(x) = tanh(x)$), while others have been proposed, including sigmoid and ReLU functions (Heaton, 2008). For the final layer, an identity activation function ($\sigma(x) = x$) is used for regression problems (Schweidtmann and Mitsos, 2018). One challenge in constructing a neural network model is hyperparameter optimization. Hyperparameters are variables that determine the structure and training of the network (e.g., number of hidden nodes, number of hidden layers), and these must be set before optimizing the weights and bias values of the neural network. Several strategies can be used to find the optimal hyperparameters, such as grid search, stochastic optimization using a genetic algorithm, and heuristics (Heaton, 2008). Depending on the desired accuracy of the ANN, we can choose different hyperparameter optimization strategies. After the optimal hyperparameters are determined, we can then optimize the weights and bias values of the neural network.

3.2.2. Gaussian process modeling

Gaussian process (GP) modeling, also known as Kriging, is an interpolating function that assumes that two points that are close to each other are likely to be correlated. This relationship is expressed using a correlation function, which depends on the distance between two points $\boldsymbol{x}^{(p)}$ and $\boldsymbol{x}^{(q)}$ (Jones et al., 1998):

$$cor\left(\epsilon\left(\boldsymbol{x}^{(p)}\right), \epsilon\left(\boldsymbol{x}^{(q)}\right)\right) = \exp\left[-\sum_{i=1}^{k} \theta_i \left(x_i^{(p)} - x_i^{(q)}\right)^2\right] \tag{2}$$

The correlation function in Eq. (2) captures the following: when two points are close to each other (i.e., distance is small), the correlation approaches to one (high correlation), while when the distance between two points is large, the correlation approaches zero. Using this correlation function, we can obtain the final functional form of GP models:

$$\hat{f}_{GP}(\boldsymbol{x}) = \mu + \sum_{n=1}^{N} c_n \exp\left[-\sum_{i=1}^{k} \theta_i \left(x_i - x_i^{(n)}\right)^2\right] \tag{3}$$

where N represents the number of data points used to train the model, k represents the dimension of the problem, θ_i and c_n represent correlation parameters, and μ is the estimated mean. The parameters can be found by using maximum likelihood estimation (MLE) (Jones et al., 1998). The final functional form is directly correlated with the number of data points used to train the model.

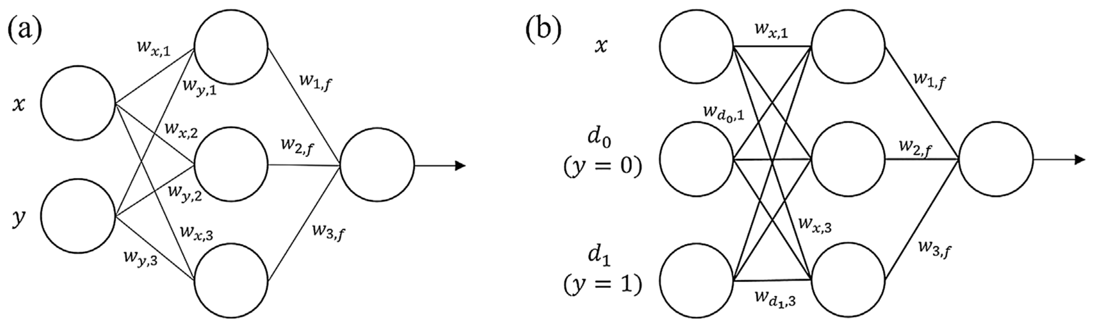


Fig. 3. Neural network without one-hot encoding (a) and with one-hot encoding (b).

Hence, the complexity of the functional form increases as more data points are used to construct the model (Jones et al., 1998).

3.3. Mixed-integer surrogate model construction via one-hot encoding

The simplest way to build surrogate models for mixed-variable functions is to assume that all k_2 discrete variables are continuous and proceed with training an ANN or a GP model with k inputs. This approach has been used in the model-based surrogate optimization literature thus far (Holmström et al., 2008; Müller, 2016; Müller et al., 2013; Rashid et al., 2013). This approach is attractive due to its simplicity, but it assumes that all inputs are continuous and ordinal. Instead, we propose an alternative way of constructing a mixed-variable surrogate model without relaxing the integrality constraint through the use of one-hot encoding (Brownlee, 2017). One-hot encoding involves converting binary or integer variables to dummy variables. Through one-hot encoding, we can make sure that regardless of the values of the discrete variables, the effect of this input on the output prediction does not diminish (McCaffrey, 2013). For example, Fig. 3 shows a simple ANN with one hidden layer with 3 nodes and 1 output node. The original problem has one continuous (x) variable and one binary (y) variable. If we are using a standard ANN, the functional form will be:

$$h_l = \sigma\left(w_{x,l}x + w_{y,l}y + b_l\right),\ l = 1, 2, 3 \tag{4}$$

$$\hat{f}_{NN} = \sigma\left(w_{1,f}h_1 + w_{2,f}h_2 + w_{3,f}h_3 + b_{out}\right) \tag{5}$$

where h_l represents the three nodes in the hidden layer, $w_{s,d}$ represents the weight of the ANN (s = source and d = destination), and b_l and b_{out} are bias values of the hidden and the output layers, respectively. If $y = 0$, then all signals from the binary variable node y will become zero and all of the terms $w_{y,\,l}y$ in Eq. (4) will be equal to zero. On the other hand, if $y = 1$, then all signals from node y will now become one. This is problematic as {0,1} does not represent an intensity, and $y = 0$ might represent a certain effect on the output that must be captured (e.g., the presence of a process unit or not).

One-hot encoding can be used to overcome this problem and improve model accuracy in the case of mixed-variable problems. One-hot encoding converts the original binary variable into two dummy variables, each representing a distinct value of the original binary variable. Consequently, the structural complexity of the model (i.e., number of input nodes in the neural network) increases due to additional dummy variables. If we have one binary variable ($y = \{0, 1\}$), two dummy variables are created as follows:

$$d_0 = \begin{cases} 1\ if\ y = 0 \\ 0\ if\ y = 1 \end{cases},\ d_1 = \begin{cases} 0\ if\ y = 0 \\ 1\ if\ y = 1 \end{cases} \tag{6}$$

The resulting functional form of the ANN is:

$$h_l = \sigma\left(w_{x,l}x + w_{d_0,l}d_0 + w_{d_1,l}d_1 + b_l\right),\ l = 1, 2, 3 \tag{7}$$

$$\hat{f}_{NN} = \sigma\left(w_{1,f}h_1 + w_{2,f}h_2 + w_{3,f}h_3 + b_{out}\right) \tag{8}$$

Thus, depending on the value of y, only one of d_0 or d_1 is active. While the dummy variables are still discrete, one-hot encoding allows that the effect of variable y on of h_l is represented evenly regardless of the value of y by allowing either d_0 or d_1 to be always equal to one. As a result, the overall signal from the binary node remains undiminished regardless of the value of the binary variable. For the case of integer variables, the surrogate model could be constructed without one-hot encoding if integer variables represent ordinal relationships. The integer values can then be scaled between 0 and 1 before constructing a surrogate model (Müller et al., 2013). If integer variables do not represent ordinal relationships, one-hot encoding can be used to represent the different integer variable levels, or the integer variables can be transformed to binary variables. Subsequently, Eq (6) can be used directly (Rall et al., 2019).

To optimize a model with one-hot encoding, we need to add an additional constraint to make sure only one of the dummy variables is selected (i.e., $d_0 + d_1 = 1$). Throughout this paper, the surrogate model generated using one-hot encoding will be noted as "mixed-integer (MI)" surrogate; the one generated without one-hot encoding (i.e., relaxing the integrality constraint) will be noted as "relaxed (RE)" surrogate. One-hot encoding is performed during the data-processing stage, where the dataset for binary variable is transformed into dummy variables. For example, if the original dataset is [$\boldsymbol{X}$, $\boldsymbol{Y}$], where $\boldsymbol{X}$ represents a set of data for continuous inputs and $\boldsymbol{Y}$ represents a set of data for binary inputs, the transformed dataset is [$\boldsymbol{X}, \boldsymbol{D_0}$, $\boldsymbol{D_1}$], where $\boldsymbol{D_0}$ and $\boldsymbol{D_1}$ represent each dummy inputs created for each binary value. After this transformation, a MI surrogate model is constructed using either ANN or GP using the transformed, augmented dataset.

4. Proposed algorithm

In order to solve (P1), the MI and RE surrogate models are integrated into a black-/gray-box optimization framework described previously. The overall algorithm can be decomposed into two main steps: 1) MINLP search, and 2) NLP search. Surrogate models are constructed in both search steps for all black-box constraints, and both the MINLP and NLP search steps are illustrated in detail in Tables 2 and 3. First, all black-box equality constraints h_{EB} are transformed into two inequalities and are added to set IB. The MINLP search is first performed to find a solution with respect to all variables (i.e., both continuous and discrete variables). The NLP search is then performed by fixing discrete variables at optimal values determined from the MINLP step and optimizing only with respect to continuous variables. The NLP search step allows

Table 1
Three sampling strategies for initial design of experiment.

```
Algorithm 1: Initial Design of Experiment
Input: problem dimension $k$, continuous dimension $k_1$, binary dimension $k_2$, total number of levels $m = \prod_{j=1}^{k_2} L_j$
Output: Latin hypercube design $\boldsymbol{S_{lhs}}$
Initialization:
$n_{lhs} = \max(5, [\frac{10k+1}{m}])$

Sampling Strategy 1 (SS1):
        $\boldsymbol{S_{lhs}} \leftarrow []$
        for $i = 1$ to $m$ do
            Generate a Latin hypercube design $\boldsymbol{X}$ of size $n_{lhs} \times k_1$ for $\boldsymbol{x}$
            Construct a dataset $\boldsymbol{Y}$ of size $n_{lhs} \times k_2$, where all rows represent a single unique combination of binary variables
            $\boldsymbol{S_{lhs}} \leftarrow [\boldsymbol{X}, \boldsymbol{Y}]$
        End

Sampling Strategy 2 (SS2):
        Generate a Latin hypercube design $\boldsymbol{X}$ of size $mn_{lhs} \times k_1$ for $\boldsymbol{x}$
        for $i = 1$ to $m$ do
            Randomly select $n_{lhs}$ rows from $\boldsymbol{X}$
            Construct a dataset $\boldsymbol{Y}$ of size $n_{lhs} \times k_2$, where all rows represent a single unique combination of binary variables
            $\boldsymbol{S_{lhs}} \leftarrow [\boldsymbol{X}, \boldsymbol{Y}]$
        End

Sampling Strategy 3 (SS3):
        Generate a Latin hypercube design $\boldsymbol{S} = [\boldsymbol{X}, \boldsymbol{Y}]$ of size $mn_{lhs} \times k$
        For $k_2$ columns, round the values to the closest integer: $\boldsymbol{S_{lhs}} \leftarrow [\boldsymbol{X}, ]$
return $\boldsymbol{S_{lhs}}$
```

the algorithm to further reduce constraint violations and refine the incumbent solution. The overall algorithmic steps for both the MINLP and NLP search stages consist of three main steps: 1) initial sampling, 2) surrogate modeling, and 3) optimization and adaptive sampling. The main differences between the MINLP and NLP step are: (a) whether one-hot encoding is used to construct a mixed-integer surrogate model or not, and (b) how the incumbent solution is selected at each iteration. The general framework is written in Python and the optimization is performed via an in-house Python-GAMS interface (Figure 4 and 5).

4.1. Initial sample design

For simulation-dependent optimization problems, choosing an initial sampling design is important since the simulation may fail to converge if a continuous value is used instead of a discrete value. Three sampling strategies are compared in this work as described earlier. For all sampling strategies, we generate a total of $m \times \max(5, \frac{10(k_1+k_2)+1}{m})$ points. The minimum number of 5 points is a heuristic on the minimum number of points that must be included in each level to ensure that at least this many points are sampled from each level. All collected samples are scaled between 0 and 1 using x_i^l and x_i^u before proceeding to fit any surrogate models. The three sampling strategies (SS1, SS2, SS3) are illustrated in Table 1.

4.2. Surrogate model construction

During this stage, surrogate models are developed in the scaled domain using either an ANN or GP to represent each of the outputs (i.e., objective function and unknown constraints). For mixed-integer surrogates, one-hot encoding is performed to convert the original binary variables to dummy variables. 10-fold cross-validation is used to find the best model during each iteration of the overall algorithm. While 10-fold cross-validation allows the algorithm to construct a model that generalizes well to a new set of data, one disadvantage of k-fold cross validation is the increased computational cost due to training k models at each iteration. In our work, we have observed that the CPU time for model construction is negligible relative to the optimization CPU time. Nevertheless, when model construction becomes computationally more expensive, surrogate fitting could happen in parallel using multiple processors to reduce the CPU time.

For the ANN, both the objective and all constraints are modeled simultaneously by using multiple output nodes (Multiple Input – Multiple Output ANN). In our work, we use a hyperbolic tangent function as an activation function; for the final layer, a linear activation function is used. As our goal is to locate a global optimum rather than finding a perfect surrogate representation, we are not necessarily interested in finding a good approximation in regions of low interest (i.e., areas far away from global optimum). Thus, the balance between model accuracy and sampling requirement is achieved by keeping the overall model complexity low while maintaining high accuracy in regions of high interest (i.e., areas where the global optimum is likely to be located). Instead of using an extensive search methodology to find the number of hidden layers and hidden nodes, we use a simple heuristic to determine the number of nodes in a hidden layer. Specifically, only one hidden layer is used and the number of nodes is 2/3 of the number of input nodes plus the number of output nodes (i.e., total number of constraints and the objective) (Heaton, 2008). This strategy allows us to locate a good optimum within a reasonable computation time. While methods such as grid search and stochastic optimization may lead to a more accurate ANN, it can be computationally too expensive for surrogate-modeling, especially when several iterations are required to converge to a solution. After the optimal hyperparameters are determined, we then compute optimal weight and bias values for the network using back propagation. For GP models, the training procedure does not require the selection of hyperparameters, but just the optimization of the surrogate model parameters. Another distinction is that all constraints and the objective are fitted separately using a multiple-input single output approach.

4.3. Surrogate model optimization and adaptive sampling

After constructing surrogate models for both the constraints and objective, an optimization problem is formulated. For ANN, the hyperbolic tangent function needs to be reformulated since optimization solvers cannot handle hyperbolic tangent functions. As suggested in (Schweidtmann and Mitsos, 2018), the hyperbolic tan-

Table 2
bb-MINLP optimization algorithm.

```
Algorithm 2. Bb-MINLP optimization

Initialization: Initial Sampling
1. Create an initial LHS $S_{lhd} = [X, Y]$ using the selected sampling strategy
2. Inquire simulation at $S_0 = [X, Y]$ to compute $Z_0 = f_{eval}(S_0)$. Assume one function evaluation provides the values of all constraints and the objective.

Data pre-processing
1. Scale $S_0$ and $Z_0$ between 0 and 1 and obtain $S_0'$ and $Z_0'$.
2. If fitting type = 'MI', perform one-hot encoding so that the binary variables $y_j$ are transformed into dummy variables $d_{j,0}$ and $d_{j,1}$.

Surrogate model construction and optimization
Initialization: $S' \leftarrow S_0', Z' \leftarrow Z_0'$
1. Use the chosen surrogate type to construct a surrogate model for all constraints and objective. Use 10-fold cross validation to find the best model.
2. Formulate the surrogate optimization problem and solve using both global and local solvers. If infeasible, solve infeasibility problem (P2).
3. Obtain $p$ local and global solutions $S'_{new} = [x_{new}, y_{new}]$.
4. Compute Euclidean distance between $S'_{new}$ and existing sample set $S'$:
   $dist_p = \frac{1}{k}\sqrt{\sum_{i=1}^{k_1}(X_{n,i} - x_{new,i})^2 + \sum_{j=1}^{k_2}(Y_{n,j} - y_{new,j})^2}$
5. If $dist_p \geq 1e^{-10}$, unscale $S'_{new}$ to the original bound and inquire simulation at $S_{new}$ and compute $Z_{new} = f_{eval}(S_{new})$ and $\upsilon$. Else, remove the solution from $S_{new}$.
6. Compute the solution score for all collected intermediate solutions:
   a. $S_{con} = rank(\upsilon)$, $S_{obj} = rank(f_{new})$
   b. $S = \frac{(S_{con}+S_{obj})}{2}$
7. $f_{min} = argmin(S)$
8. If one of the convergence criteria is met, end iteration
   Else, $S' \leftarrow S_{new}$ and $Z' \leftarrow Z_{new}$; repeat steps 1–2 for data preprocessing and steps 1–7 for surrogate model construction and optimization.
return $x^*$, $y^*$
```

gent function is reformulated as $\tanh(x) = 1 - \frac{2}{e^{2x}+1}$ since it was shown to outperform other reformulations. When a mixed-integer surrogate model is used, an additional constraint is needed to allow the selection of only one dummy variable for each binary variable. This can be formulated into a simple linear constraint: $d_0 + d_1 = 1$. For a gray-box problem, where we can assume certain constraints are known, we need to formulate and scale the gray-box constraints accordingly (see Section 6). A diverse set of local and global solutions are collected using global and multistart local optimization using BARON (Tawarmalani and Sahinidis, 2005) and DICOPT (Grossmann et al., 2002) solvers, respectively. This approach aims to find a balance between exploration and exploitation and avoids premature convergence to a local optimum.

In some instances, surrogate models might have failed to accurately approximate the constraints within the entire search space. As a result, the resulting surrogate optimization formulation is infeasible, even though this does not immediately imply that the original problem is infeasible. When this occurs, the algorithm then solves an infeasibility problem to locate the most feasible solution with the least constraint violation. Instead of minimizing the surrogate objective ($\hat{f}(\boldsymbol{x}, \boldsymbol{y})$) subject to surrogate constraints ($\hat{g}(\boldsymbol{x}, \boldsymbol{y})$), we minimize the sum of slack variables s_c, as shown in (P2).

$$\begin{aligned} &\min \sum_{c=1}^{C} s_c \\ &s.t.\ \hat{g}_c(\boldsymbol{x}, \boldsymbol{y}) - s_c \leq 0,\ c = 1, \ldots, C \\ &0 \leq s_c \leq 0.1,\ c = 1, \ldots, C \end{aligned} \tag{P2}$$

In the above formulation, set c represents unknown inequality constraints and $\hat{g}_c$ represents the surrogate approximations of all unknown inequality constraints.

All local and global solutions are added to the sampling set and the best incumbent solution is found at each iteration by calculating a score. The solutions are ranked in ascending order based on the value of the objective function value (after sampling the simulation) $f(\boldsymbol{x}^*, \boldsymbol{y}^*)$ and the total constraint violation v. Consequently, among a set of all local and global solutions, the solution with the smallest objective function value gets the lowest objective function score ($\boldsymbol{S_{obj}}$); the solution with the smallest constraint violation gets the lowest constraint violation score ($\boldsymbol{S_{con}}$). Each solution is characterized by two scores, which is equal to their rank with respect to feasibility $\boldsymbol{S_{con}}$ and objective function value $\boldsymbol{S_{obj}}$. The overall score is computed by averaging these two scores: $\boldsymbol{S} = \frac{(\boldsymbol{S_{con}}+\boldsymbol{S_{obj}})}{2}$. The solution with the lowest $\boldsymbol{S}$ score is chosen and added to the intermediate solution set. By considering both $\boldsymbol{S_{obj}}$ and $\boldsymbol{S_{con}}$, we hypothesize that we can achieve a balance between finding a global solution and finding a feasible solution when assessing the best solution found during each iteration.

These steps are repeated until one of the following termination criteria is met: 1) negligible constraint violation and model error (both $\leq 1e^{-5}$), 2) no improvement in the objective value over ten consecutive iterations, and 3) maximum number of samples is reached.

4.4. NLP search

After the MINLP search step is complete, the algorithm proceeds to the NLP search to refine the best solution ($\boldsymbol{x^*}, \boldsymbol{y^*}$) found during the MINLP search step (Table 3). This step also allows us to further reduce constraint violations, a crucial step when many equality constraints are present. The discrete values are fixed at y^* and the solution is refined with only respect to the continuous variables. The overall algorithm for NLP step is similar to that of the MINLP search step, except that one-hot encoding is not performed and CONOPT is used as a local solver (Drud, 1994). In the final step, the algorithm reduces the bounds of all continuous variables to $\pm$ 1% of the best solution found so far to further refine the solution. Assuming that the algorithm has already found an approximate solution, we only consider the constraint violation v to evaluate the solution quality during this final stage. The termination criterion of the NLP stage is identical to that of the MINLP stage.

5. Results

The performance of the proposed bb-MINLP algorithm is first tested on a set of benchmark problems obtained from MINLPLib ("MINLPLib: A library of mixed-integer and continuous nonlinear programming instances," 2019). In order to evaluate the performance of the algorithm on the most difficult possible scenario (i.e.,

Table 3
bb-NLP Algorithm.

```
Algorithm 3: bb-NLP optimization
Input: Best solution found from MINLP search step $(x^*, y^*)$, variable bounds $(x_i^l,\ x_i^u)$
Initialization: LHS sampling
1. Check if previously sampled points $S'$ can be reused. If yes, add to the sampling set.
2. Create an initial LHS $S_{0'} = [x'_{lhd}, y^*]$ of size $10k_1 + 1$ only for continuous variables.
3. Un-scale initial LHS: $x_{lhd,i} = x'_{lhd,i}(x_i^u - x_i^l) + x_i^l$
4. Inquire simulation at $S_0 = [x_{lhd}, y_{lhd}]$ to compute $Z_0 = f_{eval}(S_0)$. Assume one function evaluation provides the values of all constraints and objective.
Data-Preprocessing
1. Scale $S_0$ and $Z_0$ between 0 and 1 and obtain $S'_0$ and $Z_0{}'$.
Surrogate model construction and optimization
Initialization: $S' \leftarrow S'_0$, $Z' \leftarrow Z'_0$
1. Use the chosen surrogate type to construct a surrogate model for all constraints and objective. Use 10-fold cross validation to find the best model.
2. Formulate the surrogate optimization problem and solve using both global and local solvers. If infeasible, solve infeasibility problem (P2).
3. Obtain $p$ local and global solutions $S'_{new} = [x_{new}, y^*]$.
4. Compute Euclidean distance between $S'_{new}$ and existing sample set $S'$:
       $dist_p = \frac{1}{k}\sqrt{\sum_{i=1}^{k_1}(X_{n,i} - x_{new,i})^2 + \sum_{j=1}^{k_2}(Y_{n,j} - y_{new,j})^2}$
5. If $dist_p \geq 1e^{-10}$, unscale $S'_{new}$ to original bound and inquire simulation at $S_{new}$ and compute $Z_{new} = f_{eval}(S_{new})$ and $\upsilon$. Else, remove the solution from $S_{new}$.
6. Compute the solution score for all collected intermediate solutions:
$S_p = rank(\upsilon)$
7. $f_{min} = argmin(S_p)$
8. If one of the convergence criteria is met, end iteration
    Else, $S' \leftarrow S_{new}$ and $Z' \leftarrow Z_{new}$; repeat step 1 for data preprocessing and steps 1–7 for surrogate model construction and optimization.
return $x^*$, $y^*$
```

Table 4
Names and descriptions of the MINLP test problems from MINLPLib ("MINLPLib: A library of mixed-integer and continuous nonlinear programming instances," 2019). The equality constraints are transformed into two inequalities.

Problem	k_1	k_2	n_{leq}	n_{eq}
alan	4	4	5	2
ex1221	2	3	3	2
ex1222	2	1	3	0
ex1223	7	4	9	4
ex1223a	3	4	9	0
ex1224	3	8	5	2
ex1225	2	6	8	2
ex1226	2	3	4	1
fuel	12	3	9	6
gbd	1	3	4	0
gkocis	8	3	3	5
oaer	6	3	4	3
procsel	7	3	3	4
st_e13	1	1	2	0
st_e14	7	4	9	4
st_e15	2	3	3	2
st_e27	2	2	6	0
st_e29	3	8	5	2
synthes1	3	3	6	0

Table 5
Description of 6 surrogate model settings that are tested in Results. MI represents a mixed-integer surrogate model *with* one-hot encoding used in the MINLP stage, and RE represents a surrogate model with all variables assumed to be continuous.

Name	MINLP surrogate	NLP surrogate
ANNMI	ANN (w. one-hot encoding)	ANN
ANNRE	ANN	ANN
hyMI	ANN (w. one-hot encoding)	GP
hyRE	ANN	GP
GPMI	GP (w. one-hot encoding)	GP
GPRE	GP	GP

unknown objective function, equality and inequality constraints), we treat all problems as purely black-box systems. Table 4 shows the characteristics of the MINLP problems that are used as benchmarks in this work. Columns k_1 and k_2 represent the number of continuous and binary variables, and n_{leq} and n_{eq} are the number of inequality and equality constraints, respectively. The dimensionality of the problems ranges from 1-12 continuous variables ($k_1 \leq 12$) and 1-8 binary variables ($k_2 \leq 8$). The problems also have a varying amount of equality and inequality constraints, with the most challenging problem having 9 inequality constraints and 6 equality constraints.

Through the results shown in this work, we aim to study the performance of the proposed algorithm with respect to (a) the selection of a surrogate model type (i.e., ANN versus GP), (b) the use of one-hot encoding versus relaxation of integrality, and (c) the sampling strategy. A selection of the surrogate modeling type must be made for both the MINLP search stage and the NLP search stage of the algorithm. To compare different surrogate models, we have performed an analysis between a purely ANN-based and a purely GP-based versions of the proposed algorithm. However, based on findings described in the next section, we have also proposed a hybrid approach (i.e., the use of ANN models for MINLP search and GP models for NLP search). Thus, there are overall six approaches that are tested in this work with respect to the selection of the surrogate model as well as the use of one-hot encoding (Table 5). Finally, we compare the performance of the proposed approach with two other existing algorithms for bb-MINLP. Specifically, a Mesh-Adaptive Direct Search algorithm (NOMAD (Audet and Dennis, 2006)) and a Genetic Algorithm (Deep et al., 2009) are compared at their default settings.

The algorithm is tested 3 times for each setting, and the best, average, and standard deviation of the solutions obtained are re-

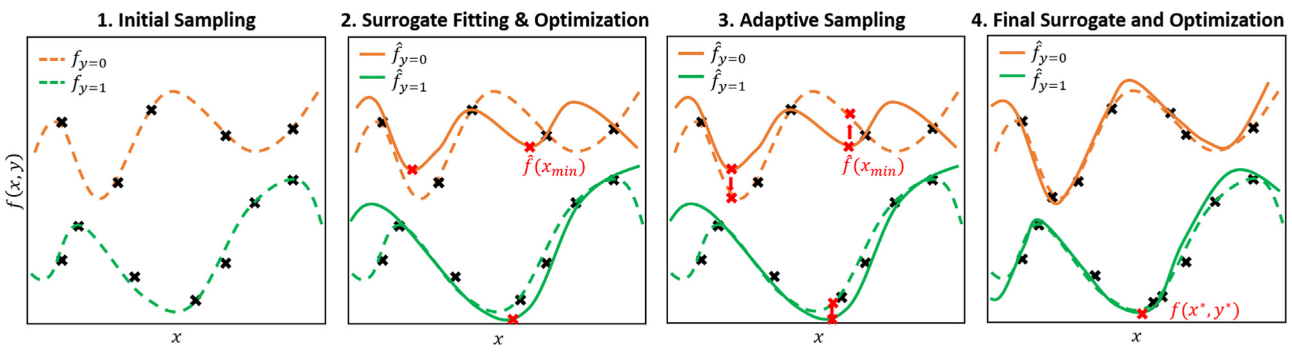


Fig. 4. Illustration of adaptive sampling for a mixed-integer problem with one continuous x and one binary y variable. The true global optimum occurs when $y = 1$. For each case of y, the true model exhibits a different behavior. At each iteration, all unique local and global optima of a surrogate model is collected, and the simulation is re-inquired at these points. The final surrogate model can accurately approximate the global optimum $f(x^*, y^*)$, which occurs when $y = 1$.

ported. The maximum number of samples allowed for each problem is 4000, since sampling is expensive in most simulation-based optimization studies and thus must be limited. Comparing both the best and the average results allows us to evaluate the performance of the algorithm and its consistency in locating global optimum. A problem is assumed to be solved if the relative error between the actual and predicted optimum is less than an error tolerance ($\epsilon = 0.01$), where: $\varepsilon_{\text{obj}} = \| \frac{f_{actual} - f^*}{f_{actual}} \leq \epsilon$. A very small constraint violation is allowed (i.e., $\nu \leq 1e^{-5}$), and this is mainly due to the difficulty in exact satisfaction of equality constraints in a black-box setting .

5.1. Selection of surrogate modeling type

First, the performance of mixed-integer and relaxed surrogate models is compared for both ANN and GP (ANNMI, ANNRE, GPMI, and GPRE). In order to eliminate the effect of the sampling strategy for these experiments, we only use the k-LHS sampling approach (Sampling Strategy 1). In the proposed algorithm, the MINLP search step aims mainly to locate the optimal discrete solution, while the NLP search step aims to refine the solution with only respect to the continuous variables. Consequently, the performance of the MINLP search step is important, since the algorithm will converge to a local solution during the NLP step if an incorrect binary solution is found during the MINLP step. Fig. 6 shows the percentage of correct binary solutions found by each method during the MINLP search step. For both ANN and GP surrogate models, the mixed-integer approach (i.e., the use of one-hot-encoding) allows the algorithm to more accurately locate binary solutions. This is due to the increased prediction accuracy of the MI model. For all of benchmark problems, the discrete variables are binary variables, and these are not ordinal. The results indicate that one-hot encoding enables the algorithm to search more efficiently through the use of more accurate surrogate models that capture the effect of both the "0" and the "1" cases. Hence, even if the resulting optimization model is more complex (i.e., increased problem dimension due to additional dummy variables and additional constraints relating new variables), the MI approach still outperforms the relaxation approach for the sizes of problems we are solving in this work. If the number of binary variables further increases, then tractability issues may arise, a potential limitation of our proposed methodology.

After the MINLP step terminates, the proposed algorithm proceeds to the NLP step. The goal of NLP step is to refine the solution obtained during the MINLP stage and locate the optimum while further reducing existing constraint violations. Fig. 7 shows the performance profiles of the obtained best result (out of 3 repetitions) with respect to the number of samples and CPU time. The

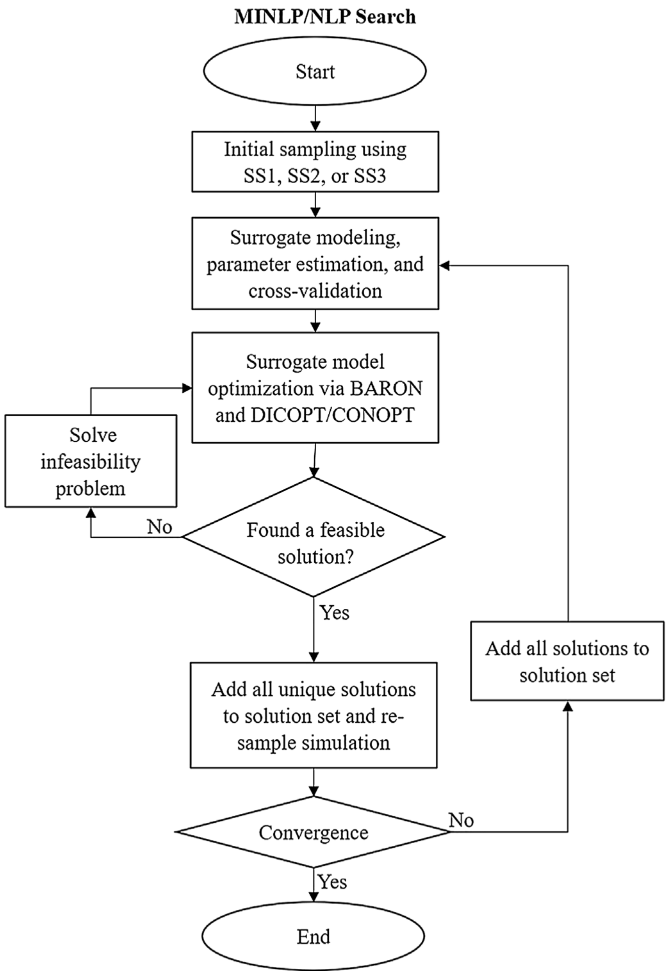

MINLP/NLP Search
Start
Initial sampling using SS1, SS2, or SS3
Surrogate modeling, parameter estimation, and cross-validation
Surrogate model optimization via BARON and DICOPT/CONOPT
Solve infeasibility problem
Found a feasible solution?
No
Yes
Add all unique solutions to solution set and re-sample simulation
Add all solutions to solution set
Convergence
No
Yes
End

Fig. 5. Overall algorithm for MINLP and NLP search steps.

results show that GPMI outperforms the three other methods, followed by GPRE. However, both GPMI and GPRE require more CPU time to converge compared to ANNMI and ANNRE. In addition, while ANNMI is able to better locate the correct discrete solution during the MINLP search step than ANNRE, Fig. 7 shows that ANNMI and ANNRE solve the same number of problems. This implies that even when a correct discrete solution has been identified, the algorithm can still fail to locate a globally optimal solution during the NLP search step. This indicates that both the MINLP and NLP search steps are crucial in locating a global solution.

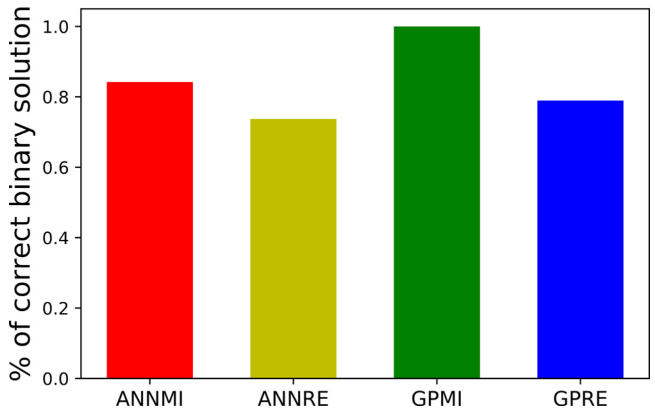


Fig. 6. Performance of algorithm with respect to the capability to accurately locate correct discrete solution during the MINLP search step.

In addition to the best obtained result, it is important to assess the consistency of the methods. In Fig. 8, the average and standard deviation of the three repetitions are reported with respect to solution accuracy and computational efficiency. When both the best and the average results are considered, it becomes clearer that the MI approaches on average perform better than RE approaches. This is attributed to the fact that the MI approaches locate the global binary solutions more consistently. For both ANNMI and GPMI, the average objective errors are smaller than those of ANNRE and GPRE, while the difference is much clearer for the ANN case. However, it is notable that while ANNMI has a smaller average objective error than that of GPMI (Fig. 8), GPMI overall solves more problems (Fig. 7). As we used strict criteria to generate a performance profile in Fig. 7 ($\epsilon = 0.01$ and $\nu = 1e^{-5}$), this result indicates that GP is better at accurately locating the solution and reducing constraint violations due to its interpolating nature. On the other hand, ANN models are good at finding the approximate location of the solution faster, but ANN models do not manage to continue improving the obtained solution and may converge to a local or infeasible solution. In addition, note that approximately 30% of the problems are not solved by any of the proposed methodologies. We observed that most of these unsolved problems have several equality constraints. Thus, the exact satisfaction of constraint violation criterion ($\nu = 1e^{-5}$) becomes increasingly challenging. Nevertheless, we can test the limit of our algorithm by including these challenging problems in our benchmark problem set.

One notable difference between the performance of ANN and GP is the CPU time for model construction and optimization. The most time-consuming step when using GP is optimization, but its model construction CPU time is negligible. In contrast to GP, the majority of the computation time for ANN is attributed to model construction (i.e., optimization of parameters). Therefore, GP

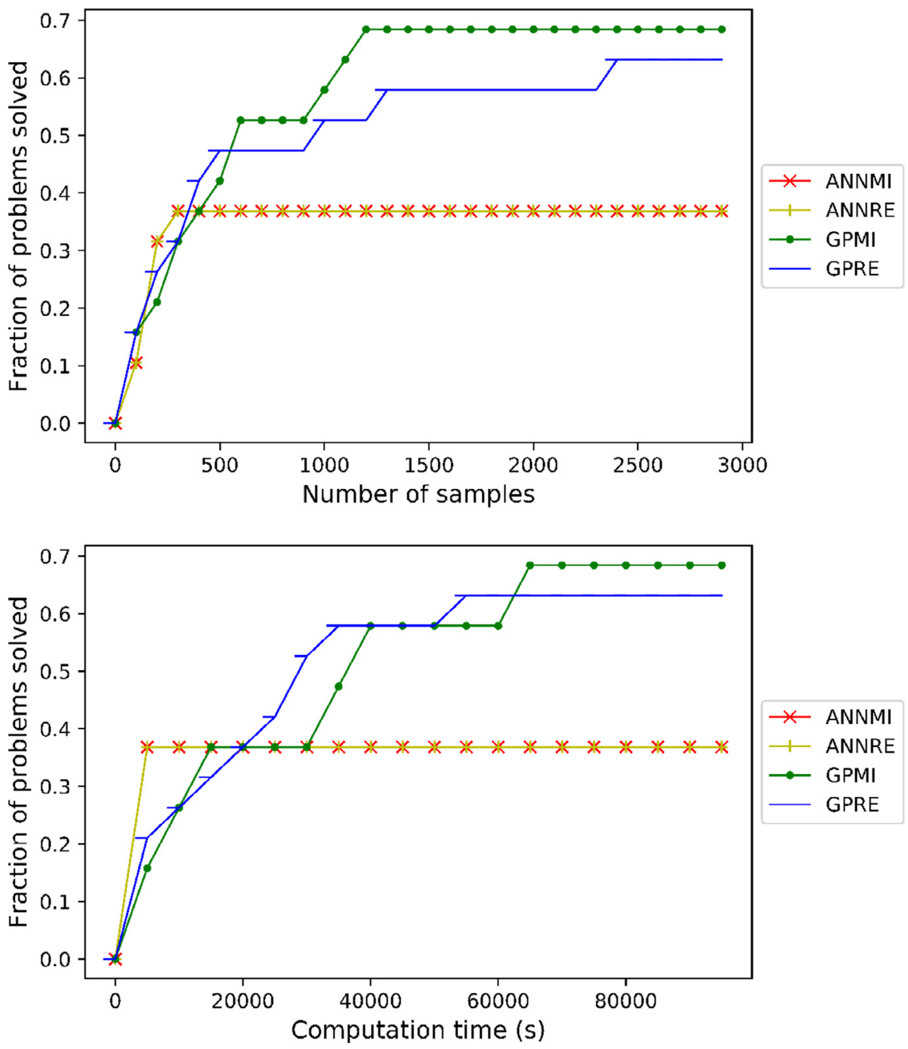


Fig. 7. Performance profile of the best run for $\epsilon = 0.01$ and $\nu = 1e^{-5}$. The model types are described in Table 4.

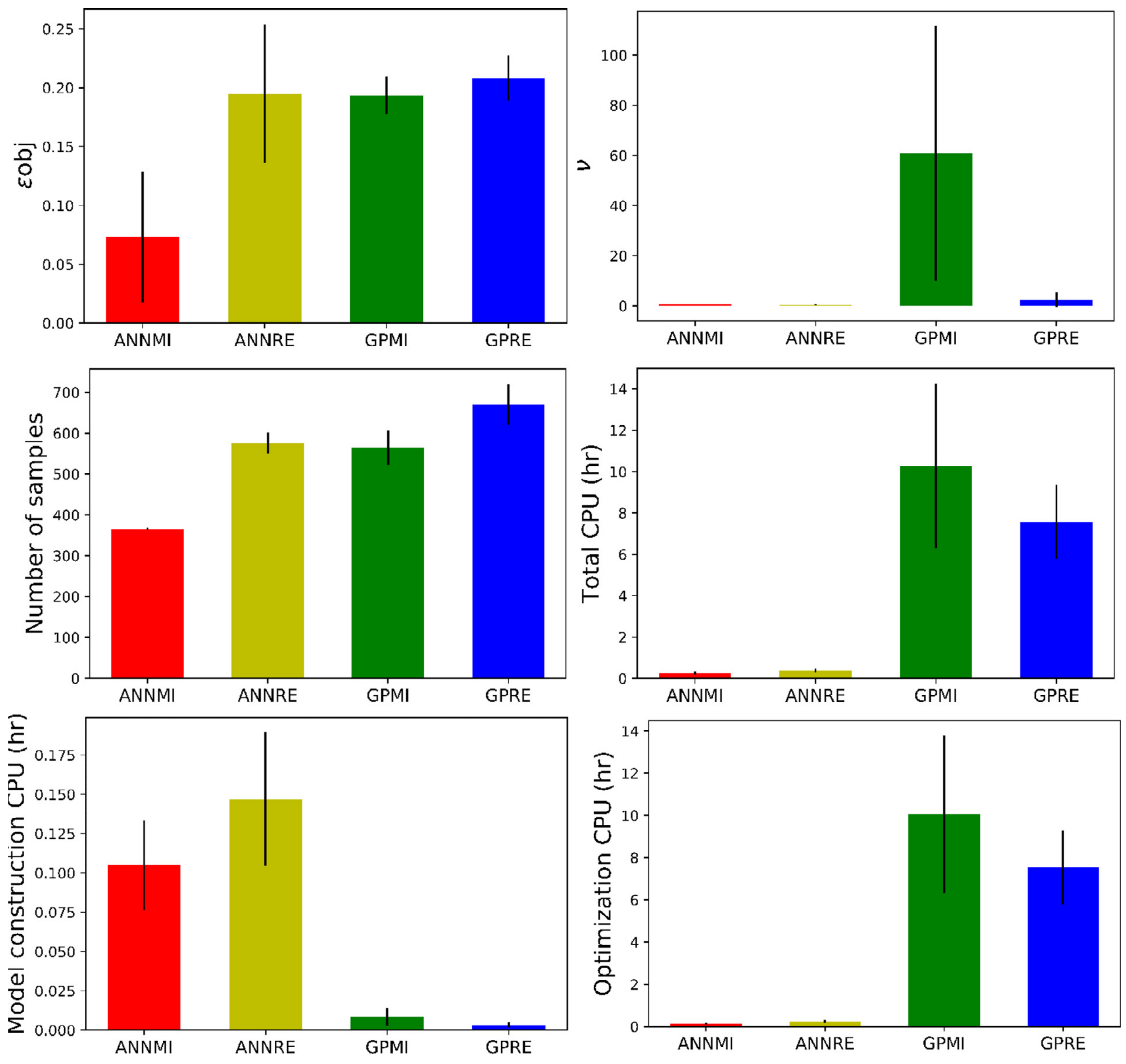
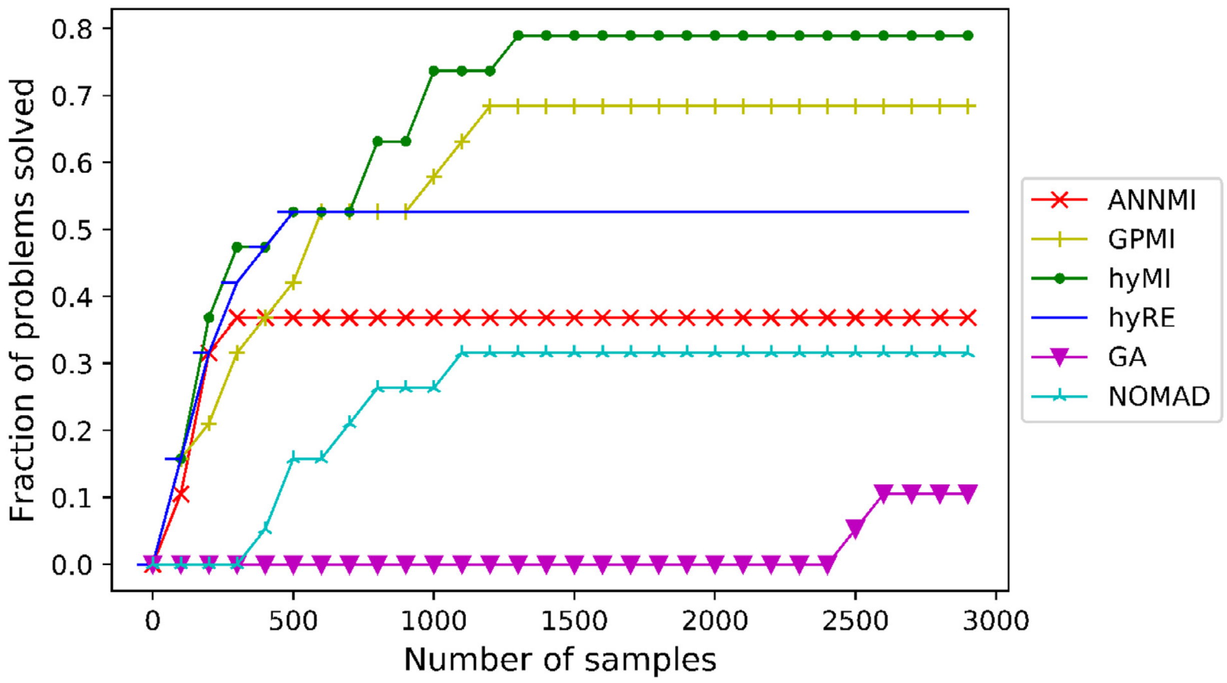


Fig. 8. Statistics of three runs for MI and RE models. The standard deviation is plotted with the average to show the consistency of each method.

and ANN exhibit a very contrasting trend: GP models are easy to construct but difficult to optimize, while ANN models are time-consuming to construct but require less cost to optimize. This difference is due to their algebraic expressions: for ANN, the model expression depends only on the number of hidden nodes and layers, while for GP, the model expression is dependent on the number of data points used to construct the model. As a result, as the algorithm collects more samples during each iteration, the complexity of a GP model increases, and the CPU time for globally optimizing the model increases significantly, especially for the MINLP search step. Our results have confirmed that the majority of optimization CPU for the GP cases is resulting from the global optimization of the MINLP model. One potential suggestion to circumvent this problem is through the sole use of multi-start local optimization of the surrogate models. We performed this test and observed that the performance of the algorithm deteriorated; thus, we have concluded that global optimization of the surrogate models is important, because it leads to the location of new and more informative sampling solutions (i.e., better exploitation of the search space).

5.2. Hybrid surrogate modeling approach

Based on the results obtained in the previous section, we have observed that ANN models have certain advantages over GP models due to their simpler functional form, which results in faster model optimization. On the other hand, all formulations with embedded GP models lead to more accurate approximations but are very difficult to optimize. These observations led to the idea of exploiting the advantages of ANN and GP by creating a hybrid model, which combines ANN and GP. In particular, the algorithm uses ANNs for the MINLP search step, followed by GP for the NLP search step. Using ANN models initially allows the algorithm to expedite the MINLP search step and obtain a somewhat accurate solution, especially with respect to the discrete variables. Subsequently, since the aim of the NLP search is refinement, it is important to improve the accuracy of the final solution, which can be accomplished better using GP models. During the NLP stage, we have fixed the values of the discrete variables, so the dimensionality has been reduced and the algorithm suffers less from the computational expense of GP optimization. It is important to note that



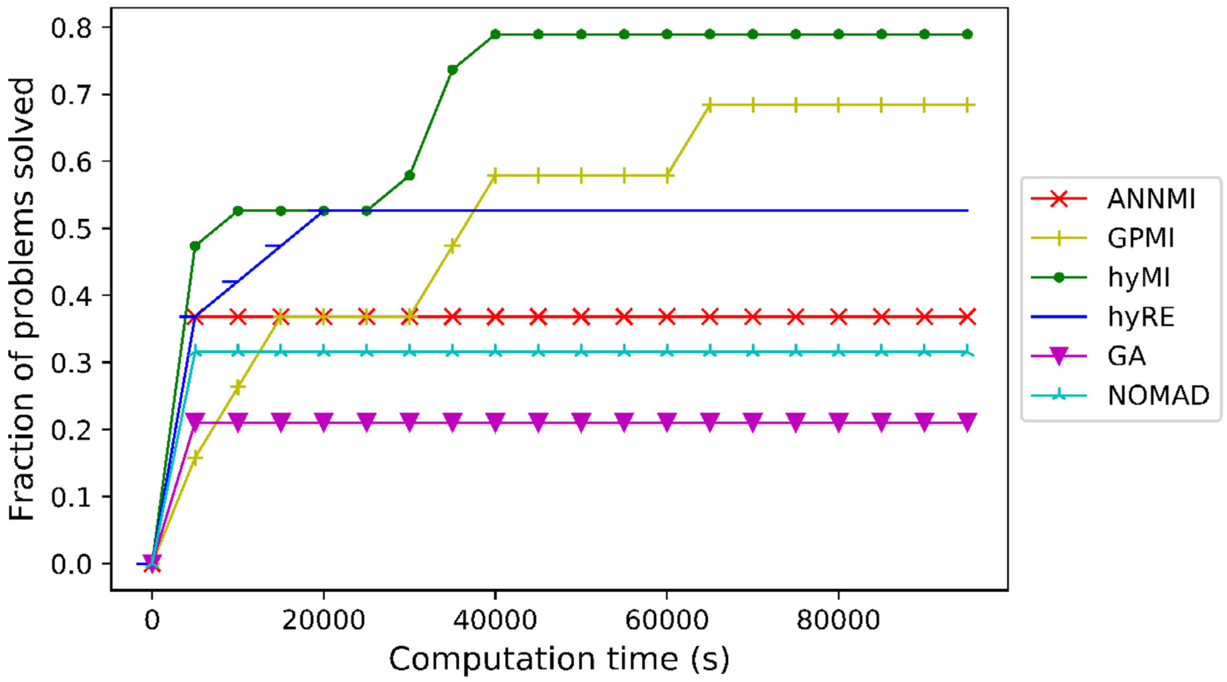


Fig. 9. Performance profile of 4 surrogate types compared with existing bb-MINLP algorithms.

our algorithm has the capability to keep multiple potential discrete solutions, which will further be refined through the NLP search. However, in this work we have only shown its performance for the case where the best single solution is kept at the end of the MINLP search step. If a mixed-integer model with one-hot encoding is used for the MINLP step, the model is referred as "hyMI"; if not, we refer the model as "hyRE". In this work, we have not looked into the reverse case, where GP is used for MINLP step and ANN is used for NLP step, because this approach is expected to deteriorate solution accuracy and/or make optimization computationally more demanding.

We present the hybrid results (hyMI and hyRE) compared to those of ANNMI and GPMI, since the MI surrogate models were previously shown to perform better than the relaxed models. Fig. 9 shows the performance profiles of the hybrid approach compared to ANNMI, GPMI, and existing solvers (GA and NOMAD). The same criteria are used to generate the performance profile: $\epsilon = 0.01$ and $\nu = 1e^{-5}$. As these methods heavily rely on sampling, they tend to require a large number of samples to converge. However, GA and NOMAD algorithms require less CPU time than surrogate-based algorithms, because GA and NOMAD do not require the construction and optimization of surrogate models. However, when sampling becomes computationally expensive, GA and NOMAD are likely to be more time-consuming than surrogate-based approaches, since GA and NOMAD tend to require many function evaluations. It must also be mentioned that both NOMAD and GA contain a set of pa-

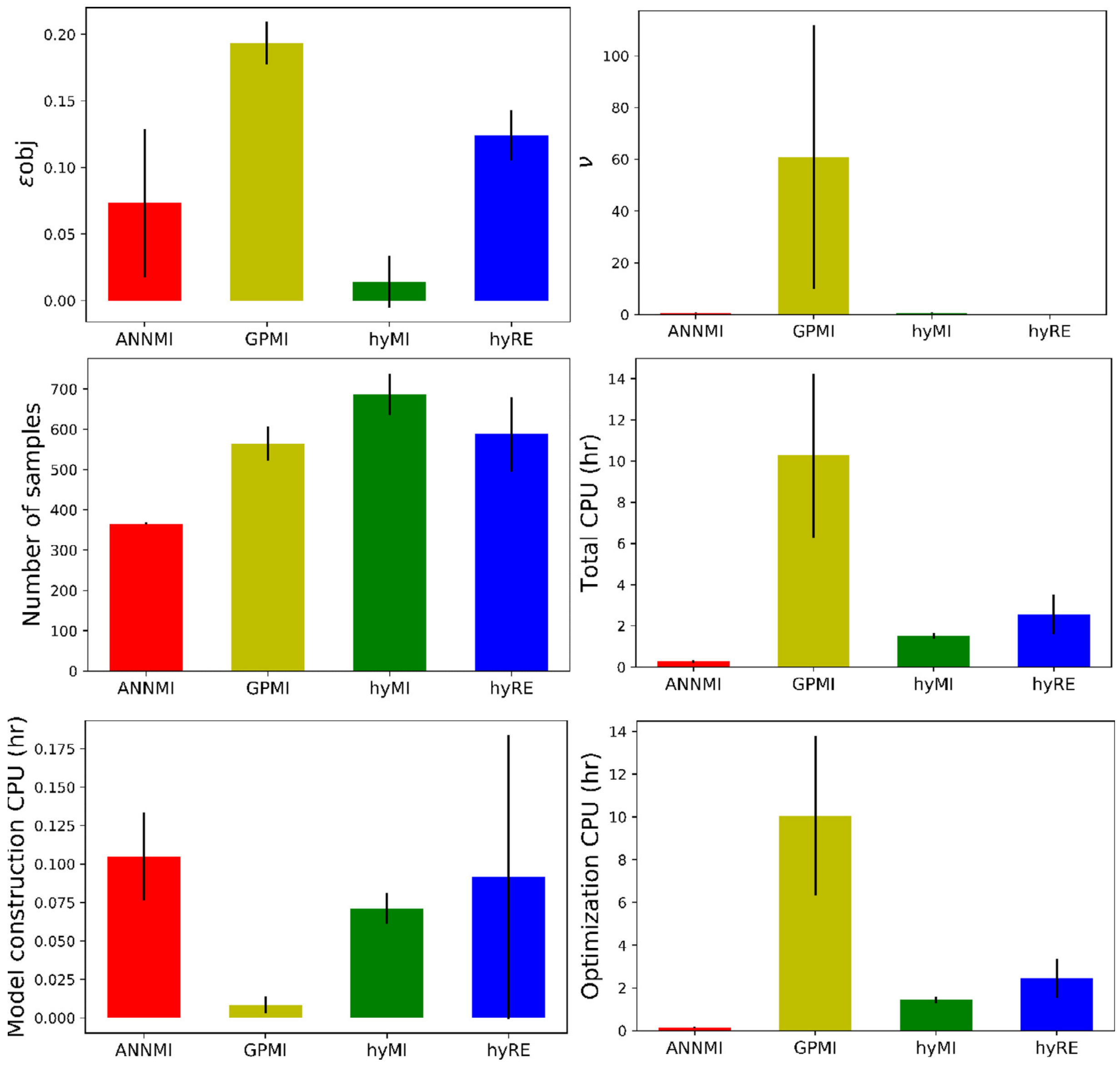


Fig. 10. Average result of three runs for MI and RE models. The standard deviation is plotted with the average to show the consistency of each method. Hybrid MI approach shows the most consistent performance and solves the most problem.

rameters that can be tuned to affect the performance of the algorithm, while in this work we compared all algorithms with their default settings.

When only the hybrid models are compared, hyMI outperforms hyRE. This is due to the capability of MI models to accurately locate binary solutions, which allows the algorithm to find a globally optimal as well as a feasible solution. After the correct binary solution is determined, the algorithm uses GP models to further refine the solution. The interpolating characteristic of GP is advantageous, particularly when the problem has several black-box equality constraints, because a good approximation of an equality constraint is crucial to find a feasible solution. Since the RE approach cannot find the correct binary solution as consistently as the MI approach, the hyRE model often converges to a local solution. As a result, the hyMI version of our algorithm combines all of the desirable characteristics and solves the most problems (about 80%), followed by GPMI and hyRE.

Fig. 10 shows the average results of three runs with their associated standard deviations to illustrate the performance as well as the consistency of the algorithms. The hyMI approach overall performs the best with the smallest average objective error and constraint violation. It is also one of the most consistent methods suggested by its small standard deviation. Compared to ANNMI, hyMI exhibits a significant improvement in solution accuracy. When the computation time is analyzed, hyMI can achieve a good balance between model construction and optimization CPU. Consequently, the optimization CPU for hyMI is significantly less than that of GPMI, which allows the algorithm to go through more iterations within a given time limit and further enhance the solution. The hyRE model performs better than ANNMI; however, it is outperformed by GPMI and hyMI since MI allows the algorithm to more accurately locate binary solutions. These results overall indicate that the MI approach using one-hot encoding outperforms the relaxation approach. Note that the hybrid approach has been proposed as a practical way of efficiently locating the solution within a reasonable CPU time. When no limit on computation resources exists, using GP for both the MINLP and NLP search steps may further improve the performance of the algorithm.

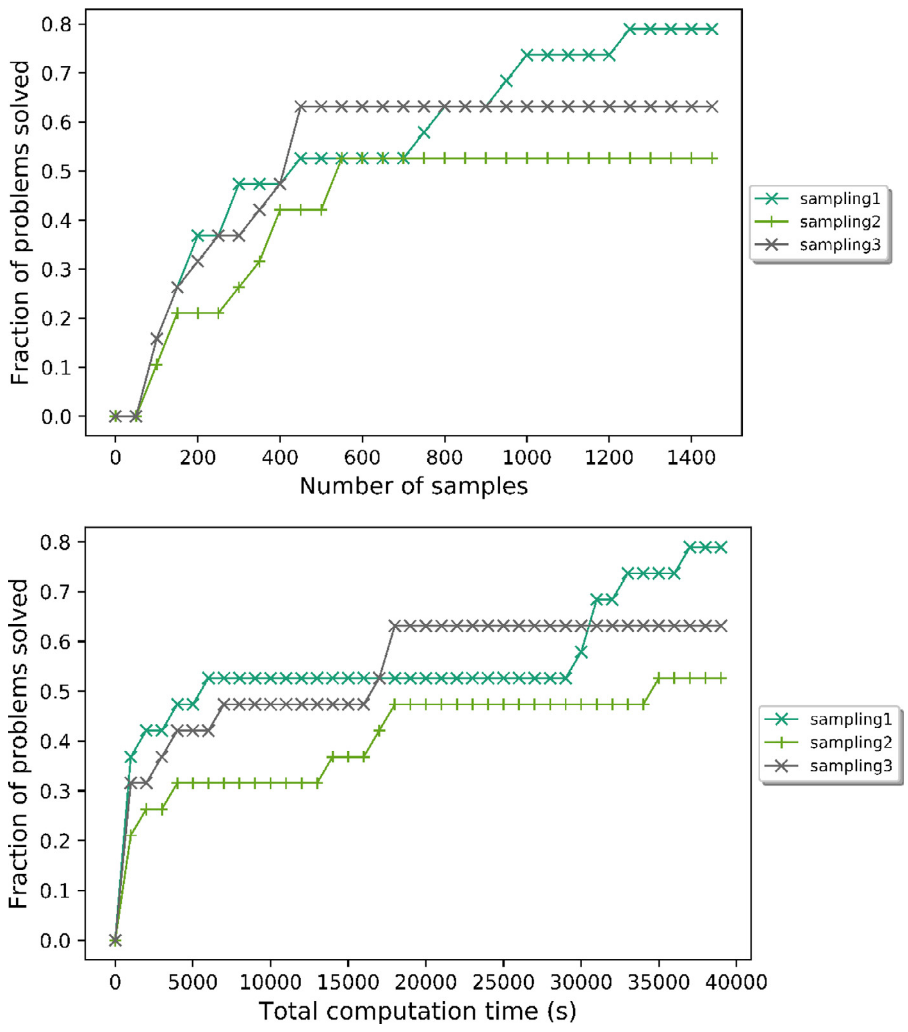


Fig. 11. Performance curve of the best out of three runs for three sampling strategies for $\epsilon = 0.01$ and $\nu = 1e\text{-}5$.

5.3. Sampling strategies for MINLP

Three sampling strategies are compared in this section while fixing all other algorithmic parameters. As we previously have determined that hyMI outperforms all other methods, we use the hyMI to test three sampling strategies (SS1, SS2, and SS3). Each sampling strategy is repeated 3 times and the best as well as the average results are calculated to evaluate its performance. Fig. 11 shows a performance curve of the best result with respect to both the number of samples and the total computation time for $\epsilon \leq 0.01$ and $\nu \leq 1e^{-5}$. When only the best result is considered out of all runs, SS1 solves about 80% of the problems and outperforms other sampling strategies. Fig. 12 shows the average result of three runs with respect to solution accuracy (ε and ν), the number of samples, and the computation time. Both the average and the best result indicate that SS1 outperforms all other sampling methods. Unlike SS2 and SS3, SS1 covers the entire search space evenly by generating a LHS of size n_{lhs} for all discrete levels. For SS2 and SS3, the points are randomly distributed into each discrete level, and thus the entire search space might not be represented evenly. Thus, even though all sampling strategies use the same number of samples, the ability to cover the entire search space evenly proves to be quite important for surrogate-based optimization. SS1 is also the most consistent approach, as shown by the small standard deviation values, while SS2 and SS3 are prone to data imbalance resulting from randomness.

6. Gray-box MINLP: a process synthesis case study

Up to this point, we have tested our algorithm for a black-box case, assuming all constraints and objective are unknown. In this section, we present a more challenging process synthesis case study and show how it can be solved as a gray-box MINLP formulation. When a problem has many constraints or variables, decoupling a problem into a gray-box could reduce the complexity and improve the solution accuracy. This is representative of a typical process synthesis problem because constraints that connect individual units, such as material balance and logical constraints to control process synthesis, will typically be known a-priori. As the surrogate models are constructed in a scaled space, these known constraints must be scaled properly. Furthermore, when a mixed-integer surrogate model with one-hot encoding is used, the resulting optimization formulation is now in terms of dummy variables, instead of original binary variables. We will illustrate how the original constraints can be modified accordingly.

6.1. Problem description

A superstructure optimization problem presented in (Duran and Grossmann, 1986) is used to demonstrate the proposed methodology. The objective is to determine the optimal structure and operating parameters for a process to minimize the sum of operating and capital costs. The binary variables are associated with

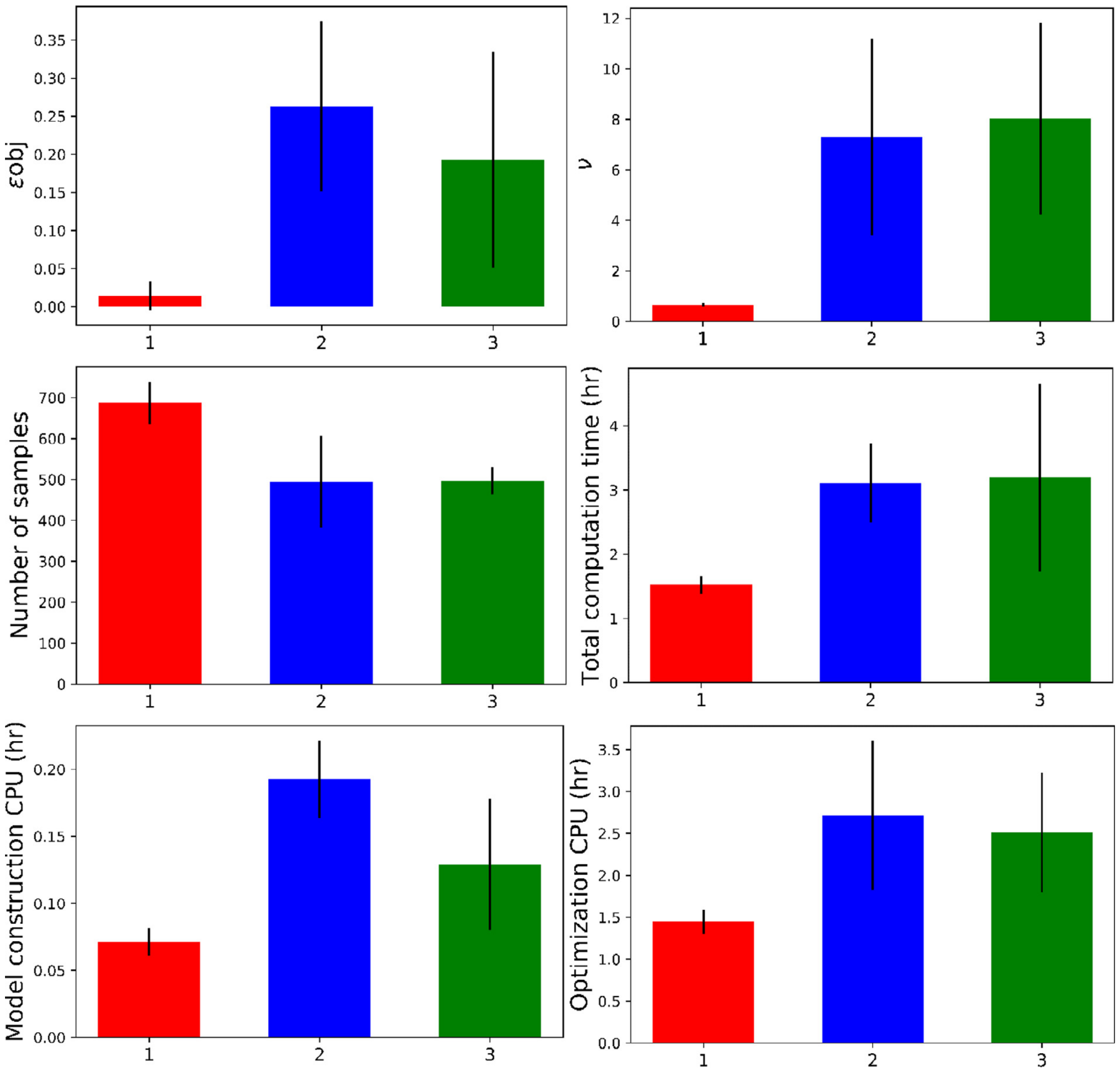


Fig. 12. Average result of three sampling strategies (SS1=1, SS2=2, SS3=3) with respect to solution accuracy and computational efficiency.

each process unit, and continuous variables represent total mass flowrate. The nonlinearities in the model are due to nonlinear input-output relationship of process units. Although this benchmark problem contains simplified nonlinear relationships to represent process units, in a real case study, these nonlinear relationships represent the underlying phenomena captured by expensive simulations. The original MINLP formulation has 9 continuous and 8 binary variables with 23 constraints, and the original problem is shown in Appendix A. The superstructure is shown in Fig. 13.

For process synthesis problems, we can safely assume that certain constraints related to mass conservation and process configuration will be known explicitly. For example, $y_1 + y_2 = 1$ enforces the selection of only one process unit; $1.25x_9 - 10y_3 \leq 0$ is a big-M constraint, which makes sure that a flow is set to zero when a process unit is not selected. These gray-box constraints can be handled explicitly (marked as (G) in Appendix A). Appendix B shows the re-formulation of the original problem, which is explained in the subsequent sections.

6.2. Reformulation and scaling of gray-box constraints

In order to generate an accurate surrogate model, normalization of input and output data is required. In this work, all input and output variables are scaled between 0 and 1 to construct a surrogate model. As a result, gray-box constraints need to be scaled properly so that optimization can now be performed in a scaled space. This is simple as all continuous variables embedded within any known constraints can be scaled using the following transformation: $x_i = x_i'(x_i^u - x_i^l) + x_i^l$, where x_i is a continuous variable in the original domain, x_i' is a continuous variable in 0–1 scaled domain, and x_i^l and x_i^u are lower and upper bounds of the variable, respectively. No scaling is required for binary variables as they are already scaled between 0 and 1.

For MI surrogate models with one-hot encoding, the surrogate models are in terms of continuous and dummy variables, instead of original binary variables. As a result, gray-box constraints also need to be in terms of dummy variables. For a binary variable y_j with

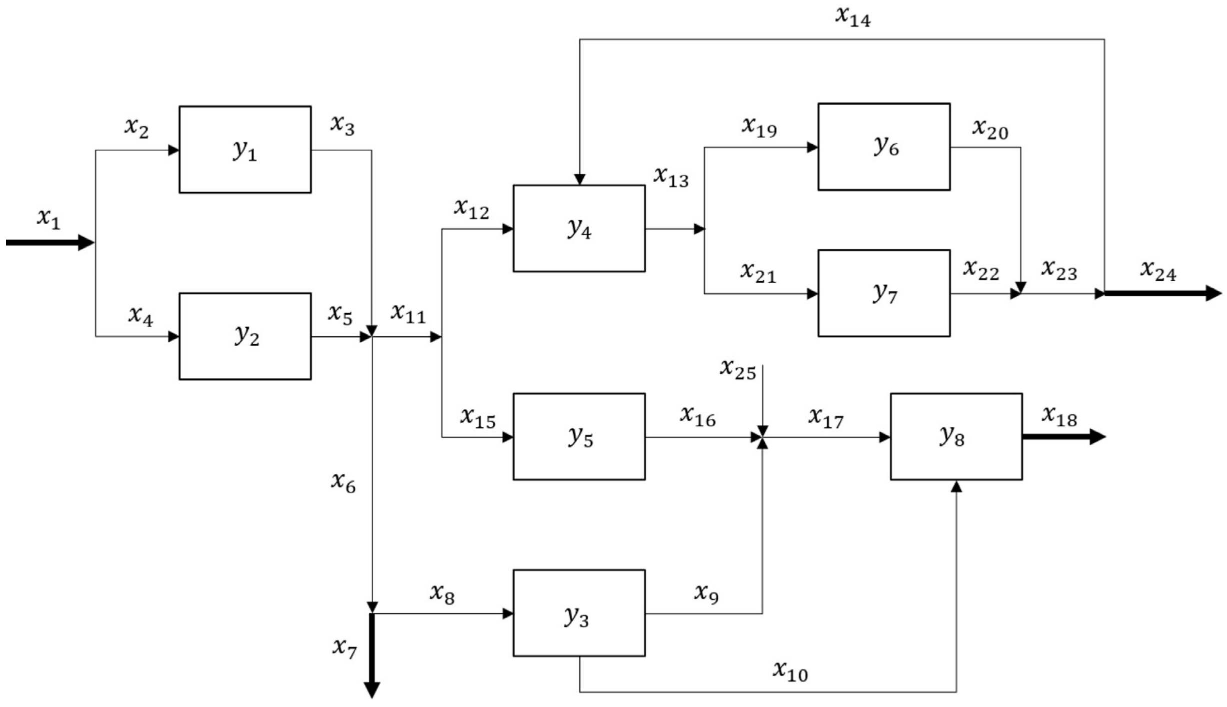


Fig. 13. Superstructure of a process synthesis case study from (Duran and Grossmann, 1986). Binary variables are used to represent 8 units, while continuous variables represent total mass flowrate.

dummy variables $d_{j,0}$ and $d_{j,1}$ where $d_{j,0} = \begin{cases} 1 \; if \; y_j = 0 \\ 0 \; if \; y_j = 1 \end{cases}$ and $d_{j,1} = \begin{cases} 0 \; if \; y_j = 0 \\ 1 \; if \; y_j = 1 \end{cases}$, the following transformation is required for gray-box constraints:

$$d_{j,0}\left(1 - y_j\right) + d_{j,1} y_j = 1$$

The original binary variable y_j can now be expressed as in terms of dummy variables:

$$y_j = \frac{1 - d_{j,0}}{d_{j,1} - d_{j,0}}$$

Furthermore, to allow the selection of only one dummy variable for each binary variable, an additional constraint is required for each y_j: $d_{j,0} + d_{j,1} = 1$. These are the overall set of steps that needs to be performed in order to re-formulate the original problem constraints, so that they are compatible with the surrogate-based formulation of our proposed algorithm.

6.3. Process synthesis case study results

For the initial sampling design, SS1 is used. At each discrete level L_j, a Latin hypercube design of size n_{lhd} is generated. The input and output datasets are both scaled between 0 and 1, and hyMI model is constructed. As the problem consists of 9 continuous and 8 binary variables, the resulting neural network has 25 input nodes (i.e., 9 nodes for continuous variables and 16 nodes for dummy variables) and 18 output nodes for the objective and black-box constraints. In total, 6 out of 23 constraints are assumed to be known, while the rest are assumed to be unknown. This categorization of unknown and known constraints was performed such that only the very simple constraints (activation of flowrates and unit selection) are considered as known. A different decomposition of known and unknown constraints may lead to different results, with the expectation that more known constraints will help the algorithm converge faster. All gray-box constraints are scaled and reformulated with respect to dummy variables, and the final formulation is shown in Appendix B. Both global and local solvers are used to collect a diverse set of intermediate solutions. The model is optimized, and the algorithm terminates when one of the termination criteria is met.

Table 6 shows the final solution obtained using the proposed methodology, and Fig. 14 shows the improvement in solution during each iteration. At the end of each iteration, we select the best solution based on both the objective function value and constraint violation v to find a solution that is not only globally optimal but also feasible. Note that at iteration 5, ε_{obj} temporarily increases because the algorithm found a more feasible solution with smaller constraint violation. During the next iteration, it quickly converges back to the actual solution. After 19 iterations, the algorithm finds a global solution with less than 1% error and $v = 5e^{-4}$. After 22 iterations, the algorithm converges to an exact global solution with $\varepsilon_{obj} = 0$ and $v = 0$. The computational cost of this run is high and this is mainly attributed to both the training of the surrogate models and their optimization. However, the total number of samples required to solve this 17-variable problem is very low, considering the state-of-the-art in surrogate-based optimization. It should be noted that this problem is quite challenging when treated as a bb-MINLP problem. In fact, our proposed algorithm could not solve this problem within the given sampling and CPU limitations, when all constraints were treated as unknown. However, known constraints and the ability to incorporate them together with surrogate models, significantly facilitates the performance of the algorithm.

Table 6
Optimization result of process synthesis case study. By using the gray-box approach, the algorithm is able to find a global optimum with $v = 0$.

f^*	f_{actual}	v	N	CPU (hr)
68.0072	68.0072	0	1492	7.4

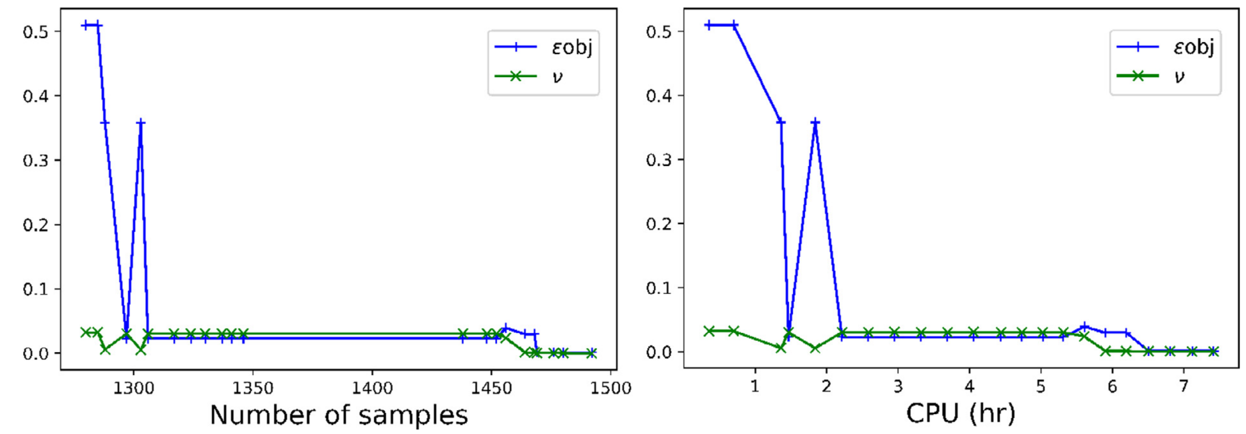


Fig. 14. Solution error and constraint violation vs. the number of samples and computation time. Each point represents a single iteration. The algorithm locates an optimal solution with less than 1% error and negligible constraint violation in just a few iterations.

7. Conclusions and future perspectives

In this work, we propose a data-dependent mixed-integer optimization algorithm for black-/gray- box problems. Unlike existing bb-MINLP algorithms, we do not relax the integrality constraint to construct a surrogate model. Instead, one-hot encoding is used to explicitly handle binary variables. Two surrogate types are considered in this work (ANN and GP) as well as a hybrid model (ANN+GP). These surrogate models are tested and compared with existing bb-MINLP solvers and the results indicate that mixed-integer surrogate models outperform relaxed surrogate models with respect to both solution quality and computational efficiency. We also demonstrate how known constraints can be explicitly incorporated within surrogate-based MINLP formulations through a process synthesis case study to facilitate the search of global optimum. Lastly, we compare different sampling strategies for bb-MINLP optimization and conclude that this has an important effect in the overall performance of the algorithm. The most effective sampling approach ensures that the sampling design is balanced in all combinations of the discrete variables. Our results indicate that when certain constraints are known *a-priori*, these should be directly incorporated within the surrogate-based formulation, because they will significantly limit the feasible search space and will allow the algorithm to focus the exploration and exploitation within feasible subspaces. In addition, we have found that satisfaction of equality constraints is exceptionally difficult in a black-box optimization setting, and this was quite effectively overcome by the incorporation of a surrogate-based feasibility sampling stage.

The proposed work can be applied to numerous simulation-based problems with both continuous and discrete variables embedded in the simulation. One specific example that is currently being studied is the synthesis of adsorption cycles. Adsorption processes contain different operating steps and cycle configurations that can be represented by binary variables. Decoupling these steps and the associated continuous and binary variables that are embedded in the simulation is not often possible. Using the proposed bb-MINLP algorithm, one can determine the optimal cycle design using input-output data from rigorous adsorption simulation models. Another MINLP case study that is currently being studied is the design of mixed-material, hybrid modular separation systems, for which discrete variables represent the selection of materials and units that are optimal for the separation of different gas mixtures.

Overall our algorithm shows promise for the solution of MINLP problems with moderate number of variables and constraints. For applications on problems with significantly more degrees of freedom, the current algorithmic implementation will require improvements to reduce its computational cost by taking advantage of parallel computing, heuristics, and recent exciting advances towards globally optimizing complex NN and GP surrogate formulations.

CRediT authorship contribution statement

Sun Hye Kim: Conceptualization, Methodology, Writing - original draft, Writing - review & editing, Conceptualization. **Fani Boukouvala:** Conceptualization, Methodology, Supervision, Funding acquisition, Writing - review & editing.

Acknowledgements

SHK and FB would like to acknowledge funding from DOE/AIChE through the RAPID Manufacturing Institute (102623; Synopsis project) and NSF-CBET(1805724).

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compchemeng.2020.106847.

Appendix

A. Process Synthesis Case Study: Original Problem Formulation adapted from (Duran and Grossmann, 1986)

$$\begin{aligned} \textit{minimize}\ z = {} & 5y_1 + 8y_2 + 6y_3 + 10y_4 + 6y_5 + 7y_6 + 4y_7 + 5y_8 \\ & -10x_3 - 15x_5 + 15x_{10} + 80x_{17} + 25x_{19} \\ & +35x_{21} - 40x_9 + 15x_{14} - 35x_{25} + \exp(x_3) + \exp\left(\tfrac{x_5}{1.2}\right) \\ & -6.5\ln(x_{10} + x_{17} + 1) \end{aligned}$$

$$s.t.\ -1.5\ln(x_{19}+1) - \ln(x_{21}+1) - x_{14} \le 0$$

$$-\ln(x_{10}+x_{17}+1) \le 0$$

$$-x_3 - x_5 + x_{10} + 2x_{17} + 0.8x_{19} + 0.8x_{24} - 0.5x_9 - x_{14} - 2x_{25} \le 0$$

$$-x_3 - x_5 + 2x_{17} + 0.8x_{19} + 0.8x_{21} - 2x_9 - x_{14} - 2x_{25} \le 0$$

$$-2x_{17} - 0.8x_{19} - 0.8x_{21} + 2x_9 + x_{14} + 2x_{25} \le 0$$

$$-0.8x_{19} - 0.8x_{21} + x_{14} \le 0$$

$$-x_{17} + x_9 + x_{25} \le 0$$

$$-0.4x_{14} - 0.4x_{21} + 1.5x_{14} \le 0$$

$$0.16x_{19} + 0.16x_{21} - 1.2x_{14} \le 0$$

$$x_{10} - 0.8x_{17} \le 0$$

$$-x_{10} + 0.4x_{17} \le 0$$

$$\exp(x_3) - 10y_1 \le 1$$

$$\exp\left(\frac{x_5}{1.2}\right) - 10y_2 \le 1$$

$$x_9 - 10y_3 \le 0\ (G)$$

$$0.8x_{19} + 0.8x_{21} - 10y_4 \le 0$$

$$2x_{17} - 2x_9 - 2x_{25} - 10y_5 \le 0$$

$$x_{19} - 10y_6 \le 0\ (G)$$

$$x_{21} - 10y_7 \le 0\ (G)$$

$$x_{10} + x_{17} - 10y_8 \le 0\ (G)$$

$$y_1 + y_2 = 1\ (G),\ y_4 + y_5 \le 1(G)$$

$$-y_4 + y_6 + y_7 = 0\ (G),\ y_3 - y_8 \le 0\ (G)$$

$$y \in \{0,1\}^8,\ a \le x \le b,$$
$$x = \left(x_j : j = 3, 5, 10, 17, 19, 21, 9, 14, 25\right) \in \mathbb{R}^9$$

$$a^T = \{0,0,0,0,0,0,0,0,0\},\ b^T = \{2,2,1,2,2,2,2,1,3\}$$

B. Process Synthesis Case Study: Gray-box Formulation in the scaled space

$$minimize\ \hat{f}\left(x'_i, d_j\right)$$
$$s.t.\ \hat{g}_c\left(x'_i, d_j\right) \le 0$$
$$2x'_9 - 10\left(\frac{1-d_{6,0}}{d_{6,1}-d_{6,0}}\right) \le 0$$

$$2x'_{19} - 10\left(\frac{1-d_{7,0}}{d_{7,1}-d_{7,0}}\right) \le 0$$
$$x'_{10} + 2x'_{17} - 10\left(\frac{1-d_{8,0}}{d_{8,1}-d_{8,0}}\right) \le 0$$

$$\left(\frac{1-d_{1,0}}{d_{1,1}-d_{1,0}}\right) + \left(\frac{1-d_{2,0}}{d_{2,1}-d_{2,0}}\right) = 1$$

$$\left(\frac{1-d_{4,0}}{d_{4,1}-d_{4,0}}\right) + \left(\frac{1-d_{5,0}}{d_{5,1}-d_{5,0}}\right) \le 1$$

$$-\left(\frac{1-d_{4,0}}{d_{4,1}-d_{4,0}}\right) + \left(\frac{1-d_{6,0}}{d_{6,1}-d_{6,0}}\right) + \left(\frac{1-d_{7,0}}{d_{7,1}-d_{7,0}}\right) \le 1$$

$$\left(\frac{1-d_{3,0}}{d_{3,1}-d_{3,0}}\right) - \left(\frac{1-d_{8,0}}{d_{8,1}-d_{8,0}}\right) \le 0$$

$$d_{j,0} + d_{j,1} = 1,\ j = 1, \dots, 8$$
$$d_j \in \{0,1\}^{16},\ 0 \le x'_i \le 1$$

References

Abramson, M.A., Audet, C., Chrissis, J.W., Walston, J.G., 2008. Mesh adaptive direct search algorithms for mixed variable optimization. Optim. Lett. 3 (1), 35. doi:10.1007/s11590-008-0089-2.

Amaran, S., Sahinidis, N.V., Sharda, B., Bury, S.J., 2016. Simulation optimization: a review of algorithms and applications. Ann. Oper. Res. 240 (1), 351–380. doi:10.1007/s10479-015-2019-x.

Audet, C., Dennis, J., 2001. Pattern search algorithms for mixed variable programming. SIAM J. Optim. 11 (3), 573–594. doi:10.1137/S1052623499352024.

Audet, C., Dennis, J., 2006. Mesh adaptive direct search algorithms for constrained optimization. SIAM J. Optim. 17 (1), 188–217. doi:10.1137/040603371.

Balasubramanian, P., Bajaj, I., Hasan, M.M.F., 2018. Simulation and optimization of reforming reactors for carbon dioxide utilization using both rigorous and reduced models. J. CO2 Utiliz. 23, 80–104 doi:10.1016/j.jcou.2017.10.014.

Ben-Tal, A., Zibulevsky, M., 1997. Penalty/Barrier multiplier methods for convex programming problems. SIAM J. Optim. 7 (2), 347–366. doi:10.1137/S1052623493259215.

Beykal, B., Avraamidou, S., Pistikopoulos, I.P.E., Onel, M., Pistikopoulos, E.N., 2020. DOMINO: data-driven Optimization of bi-level Mixed-Integer NOnlinear Problems. J. Global Optim. doi:10.1007/s10898-020-00890-3.

Bhosekar, A., Ierapetritou, M., 2018. Advances in surrogate based modeling, feasibility analysis, and optimization: a review. Comput. Chem. Eng. 108, 250–267 doi:10.1016/j.compchemeng.2017.09.017.

Booker, A.J., Dennis, J.E., Frank, P.D., Serafini, D.B., Torczon, V., Trosset, M.W., 1999. A rigorous framework for optimization of expensive functions by surrogates. Struct. Optim. 17 (1), 1–13. doi:10.1007/BF01197708.

Boukouvala, F., Floudas, C.A., 2017. ARGONAUT: algoRithms for Global Optimization of coNstrAined grey-box compUTational problems. Optim. Lett. 11 (5), 895–913. doi:10.1007/s11590-016-1028-2.

Boukouvala, F., Hasan, M.M.F., Floudas, C.A., 2017. Global optimization of general constrained grey-box models: new method and its application to constrained PDEs for pressure swing adsorption. J. Global Optim. 67 (1), 3–42. doi:10.1007/s10898-015-0376-2.

Boukouvala, F., Ierapetritou, M.G., 2013. Surrogate-Based Optimization of Expensive Flowsheet Modeling for Continuous Pharmaceutical Manufacturing. J Pharm Innov 8 (2), 131–145. doi:10.1007/s12247-013-9154-1.

Boukouvala, F., Misener, R., Floudas, C.A., 2016. Global optimization advances in Mixed-Integer Nonlinear Programming, MINLP, and Constrained Derivative-Free Optimization, CDFO. Eur J Oper Res 252 (3), 701–727 doi:10.1016/j.ejor.2015.12.018.

Brownlee, J. (2017). Why One-Hot Encode Data in Machine Learning? Retrieved from https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/.

Caballero, J.A., Grossmann, I.E., 2008. An algorithm for the use of surrogate models in modular flowsheet optimization. AIChE J. 54 (10), 2633–2650. doi:10.1002/aic.11579.

Chen, S., Cowan, C.F.N., Grant, P.M., 1991. Orthogonal least squares learning algorithm for radial basis function networks. IEEE Trans. Neural Netw. 2 (2), 302–309. doi:10.1109/72.80341.

Cocchi, G., Pillo, G., Fasano, G., Liuzzi, G., Lucidi, S., Piccialli, V., Truemper, K. (2019). DFL - A Derivative-Free Library. Retrieved from http://www.iasi.cnr.it/~liuzzi/DFL/index.php/news-list.

Conn, A., Scheinberg, K., & Vicente, L. (2009). *Introduction to Derivative-Free Optimization*: Society for Industrial and Applied Mathematics.

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

Cozad, A., Sahinidis, N.V., Miller, D.C., 2015. A combined first-principles and data-driven approach to model building. Comput. Chem. Eng. 73, 116–127 doi:10.1016/j.compchemeng.2014.11.010.

Davis, E., Ierapetritou, M., 2008. A kriging-based approach to MINLP containing black-box models and noise. Ind. Eng. Chem. Res. 47 (16), 6101–6125.

Davis, S.E., Cremaschi, S., Eden, M.R., 2018. Efficient surrogate model development: impact of sample size and underlying model dimensions. In: Eden, M.R., Ierapetritou, M.G., Towler, G.P. (Eds.). In: Computer Aided Chemical Engineering, 44. Elsevier, pp. 979–984.

Deb, K., 2000. An efficient constraint handling method for genetic algorithms. Comput. Methods. Appl. Mech. Eng. 186 (2), 311–338 doi:10.1016/S0045-7825(99)00389-8.
Deep, K., Singh, K.P., Kansal, M.L., Mohan, C., 2009. A real coded genetic algorithm for solving integer and mixed integer optimization problems. Appl. Math. Comput. 212 (2), 505–518 doi:10.1016/j.amc.2009.02.044.
Dias, L.S., Ierapetritou, M.G., 2020. Integration of planning, scheduling and control problems using data-driven feasibility analysis and surrogate models. Comput. Chem. Eng. 134, 106714 doi:10.1016/j.compchemeng.2019.106714.
Drud, A.S., 1994. CONOPT—A Large-Scale GRG Code. ORSA J. Comput. 6 (2), 207–216. doi:10.1287/ijoc.6.2.207.
Duran, M.A., Grossmann, I.E., 1986. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Program. 36 (3), 307–339. doi:10.1007/BF02592064.
Eason, J., Cremaschi, S., 2014. Adaptive sequential sampling for surrogate model generation with artificial neural networks. Comput. Chem. Eng. 68, 220–232 doi:10.1016/j.compchemeng.2014.05.021.
Eberhart, R., Kennedy, J., 1995. A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science.
Egea, J.A., Henriques, D., Cokelaer, T., Villaverde, A.F., MacNamara, A., Danciu, D.-.P., Saez-Rodriguez, J., 2014. MEIGO: an open-source software suite based on metaheuristics for global optimization in systems biology and bioinformatics. BMC Bioinformatics 15 (1), 136. doi:10.1186/1471-2105-15-136.
Forrester, A.I.J., Keane, A.J., 2009. Recent advances in surrogate-based optimization. Prog. Aerosp. Sci. 45 (1), 50–79 doi:10.1016/j.paerosci.2008.11.001.
Garud, S.S., Karimi, I.A., Kraft, M., 2018. LEAPS2: learning based evolutionary assistive paradigm for surrogate selection. Comput. Chem. Eng. 119, 352–370 doi:10.1016/j.compchemeng.2018.09.008.
Garud, S.S., Mariappan, N., Karimi, I.A., Kraft, M., 2019. Surrogate-based black-box optimisation via domain exploration and smart placement. Comput. Chem. Eng. 130, 103–114 https://doi.org/10.1016/j.compchemeng.2019.106567.
Graciano, J.E.A., Le Roux, G.A.C., 2013. Improvements in surrogate models for process synthesis. Application to water network system design. Comput. Chem. Eng. 59, 197–210 doi:10.1016/j.compchemeng.2013.05.024.
Gramacy, R.B., Lee, H.K.H., 2008. Bayesian treed gaussian process models with an application to computer modeling. J. Am. Sta.t Assoc. 103 (483), 1119–1130. doi:10.1198/016214508000000689.
Grossmann, I., Viswanathan, J., Vecchietti, A., Raman, R., & Kalvelagen, E. (2002). *GAMS/DICOPT: A discrete continuous optimization package* (Vol. 11).
Hastie, T., Tibshirani, R., Friedman, J.H., 2009. The Elements of Statistical Learning, (2 ed.) Springer.
Heaton, J., 2008. Introduction to Neural Networks For JAVA, (2 ed.) Heaton Research, Inc.
Henao, C.A., Maravelias, C.T., 2011. Surrogate-based superstructure optimization framework. AIChE J. 57 (5), 1216–1232. doi:10.1002/aic.12341.
Holland, J.H., 1992. Adaptation in Natural and Artificial Systems. MIT Press.
Holmström, K., Quttineh, N.-.H., Edvall, M.M., 2008. An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization. Optim. Eng. 9 (4), 311–339. doi:10.1007/s11081-008-9037-3.
Hooke, R., Jeeves, T.A., 1961. “Direct search” solution of numerical and statistical problems. J. ACM 8, 212–229.
Hüllen, G., Zhai, J., Kim, S.H., Sinha, A., Realff, M.J., Boukouvala, F., 2019. Managing uncertainty in data-driven simulation-based optimization. Comput. Chem. Eng., 106519 doi:10.1016/j.compchemeng.2019.106519.
Jones, D.R., Schonlau, M., Welch, W.J., 1998. Efficient global optimization of expensive black-box functions. J. Global Optim. 13 (4), 455–492. doi:10.1023/A:1008306431147.
Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Paper presented at the Proceedings of ICNN'95 - International Conference on Neural Networks.
Keßler, T., Kunde, C., McBride, K., Mertens, N., Michaels, D., Sundmacher, K., Kienle, A., 2019. Global optimization of distillation columns using explicit and implicit surrogate models. Chem. Eng. Sci. 197, 235–245.
Kim, S.H., Boukouvala, F., 2019. Machine learning-based surrogate modeling for data-driven optimization: a comparison of subset selection for regression techniques. Optim. Lett. .
Larson, J., Leyffer, S., Palkar, P., & Wild, S.M. (2019). A method for convex black-box integer global optimization. *arXiv preprint* arXiv:1903.11366.
Liuzzi, G., Lucidi, S., Rinaldi, F., 2012. Derivative-free methods for bound constrained mixed-integer optimization. Comput. Optim. Appl. 53 (2), 505–526. doi:10.1007/s10589-011-9405-3.
Liuzzi, G., Lucidi, S., Rinaldi, F., 2015. Derivative-free methods for mixed-integer constrained optimization problems. J. Optim. Theory Appl. 164 (3), 933–965. doi:10.1007/s10957-014-0617-4.
McBride, K., Sundmacher, K., 2019. Overview of surrogate modeling in chemical process engineering. Chem. Ing. Tech. 91 (3), 228–239.
McCaffrey, J. (2013). Neural network data normalization and encoding. Retrieved from https://visualstudiomagazine.com/articles/2013/07/01/neural-network-data-normalization-and-encoding.aspx.
McKay, M.D., Beckman, R.J., Conover, W.J., 1979. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21 (2), 239–245. doi:10.2307/1268522.
Mencarelli, L., Chen, Q., Pagot, A., Grossmann, I.E., 2020a. A review on superstructure optimization approaches in process system engineering. Comput. Chem. Eng. 136, 106808 doi:10.1016/j.compchemeng.2020.106808.
Mencarelli, L., Pagot, A., Duchêne, P., 2020b. Surrogate-based modeling techniques with application to catalytic reforming and isomerization processes. Comput. Chem. Eng. 135, 106772 doi:10.1016/j.compchemeng.2020.106772.
MINLPLib: A library of mixed-integer and continuous nonlinear programming instances. (2019). Retrieved from http://www.minlplib.org/. Retrieved May 7, 2019 http://www.minlplib.org/.
Mistry, M., Letsios, D., Krennrich, G., Lee, R.M., & Misener, R. (2018). Mixed-Integer Convex Nonlinear Optimization with Gradient-Boosted Trees Embedded. *arXiv e-prints*. Retrieved from https://ui.adsabs.harvard.edu/abs/2018arXiv180300952M.
Müller, J., 2016. MISO: mixed-integer surrogate optimization framework. Optim.Eng. 17 (1), 177–203. doi:10.1007/s11081-015-9281-2.
Müller, J., Shoemaker, C., Piche, R., 2013. SO-MI: a surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. Comput. Oper. Res. 40 (5), 1383–1400. doi:10.1016/j.cor.2012.08.022.
Nelder, J.A., Mead, R., 1965. A simplex method for function minimization. Comput. J. 7 (4), 308–313. doi:10.1093/comjnl/7.4.308.
Olofsson, S., Deisenroth, M., Misener, R., 2018. Design of experiments for model discrimination hybridising analytical and data-driven approaches. In: Paper presented at the Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research.
Owen, A.B., 1992. Orthogonal arrays for computer experiments, integration and visualization. Stat. Sin. 2 (2), 439–452. Retrieved from http://www.jstor.org/stable/24304869 .
Qian, P.Z.G., Wu, H., Wu, C.F.J., 2008. Gaussian process models for computer experiments with qualitative and quantitative factors. Technometrics 50 (3), 383–396. doi:10.1198/004017008000000262.
Queipo, N.V., Haftka, R.T., Shyy, W., Goel, T., Vaidyanathan, R., Kevin Tucker, P., 2005. Surrogate-based analysis and optimization. Prog. Aerosp. Sci. 41 (1), 1–28 doi:10.1016/j.paerosci.2005.02.001.
Quirante, N., Javaloyes, J., Ruiz-Femenia, R., Caballero, J.A., 2015. Optimization of chemical processes using surrogate models based on a kriging interpolation. In: Gernaey, K.V., Huusom, J.K., Gani, R. (Eds.). In: Computer Aided Chem. Engineering, 37. Elsevier, pp. 179–184.
Rall, D., Menne, D., Schweidtmann, A.M., Kamp, J., von Kolzenberg, L., Mitsos, A., Wessling, M., 2019. Rational design of ion separation membranes. J. Memb. Sci. 569, 209–219 doi:10.1016/j.memsci.2018.10.013.
Rashid, K., Ambani, S., Cetinkaya, E., 2013. An adaptive multiquadric radial basis function method for expensive black-box mixed-integer nonlinear constrained optimization. Eng. Optim. 45 (2), 185–206. doi:10.1080/0305215X.2012.665450.
Rasmussen, C.E., 2004. Gaussian processes in machine learning. In: Bousquet, O., von Luxburg, U., Rätsch, G. (Eds.), Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures. Springer, Berlin Heidelberg, pp. 63–71 Berlin, Heidelberg.
Reeves, C. (1997). *Genetic Algorithms for the Operations Researcher* (Vol. 9).
Regis, R.G., Shoemaker, C.A., 2005. Constrained global optimization of expensive black box functions using radial basis functions. J. Global Optim. 31 (1), 153–171. doi:10.1007/s10898-004-0570-0.
Rios, L.M., Sahinidis, N.V., 2013. Derivative-free optimization: a review of algorithms and comparison of software implementations. J. Global Optim. 56 (3), 1247–1293. doi:10.1007/s10898-012-9951-y.
Romeo, F., Sangiovanni-Vincentelli, A., 1991. A theoretical framework for simulated annealing. Algorithmica 6 (1), 302. doi:10.1007/BF01759049.
Sangbum, L., En Sup, Y., Grossmann, I.E., 2003. Superstructure optimization of chemical process. Paper presented at the SICE 2003 Annual Conference. IEEE Cat. No. 03TH8734.
Schweidtmann, A.M., Huster, W.R., Lüthje, J.T., Mitsos, A., 2019. Deterministic global process optimization: accurate (single-species) properties via artificial neural networks. Comput. Chem. Eng. 121, 67–74 doi:10.1016/j.compchemeng.2018.10.007.
Schweidtmann, A.M., Mitsos, A., 2018. Deterministic global optimization with artificial neural networks embedded. J. Optim. Theory Appl. doi:10.1007/s10957-018-1396-0.
Sobol, I.M., 1967. On the distribution of points in a cube and the approximate evaluation of integrals. USSR Comput. Math. Math. Phys. 7 (4), 86–112 doi:10.1016/0041-5553(67)90144-9.
Specht, D.F., 1991. A general regression neural network. IEEE Trans. Neural Netw. 2 (6), 568–576. doi:10.1109/72.97934.
Swiler, L.P., Hough, P.D., Qian, P., Xu, X., Storlie, C., Lee, H., 2014. Surrogate models for mixed discrete-continuous variables. In: Ceberio, M., Kreinovich, V. (Eds.), Constraint Programming and Decision Making. Springer International Publishing, Cham, pp. 181–202.
Tawarmalani, M., Sahinidis, N.V., 2005. A polyhedral branch-and-cut approach to global optimization. Math. Program. 103 (2), 225–249. doi:10.1007/s10107-005-0581-8.
Tso, W.W., Demirhan, C.D., Floudas, C.A., Pistikopoulos, E.N., 2019. Multi-scale energy systems engineering for optimal natural gas utilization. Catal. Today doi:10.1016/j.cattod.2019.09.009.
Williams, C.K.I., Rasmussen, C.E., 1995. Gaussian processes for regression. In: Paper presented at the Proceedings of the 8th International Conference on Neural Information Processing Systems. Denver, Colorado.
Wilson, Z.T., Sahinidis, N.V., 2019. Automated learning of chemical reaction networks. Comput. Chem. Eng. 127, 88–98 doi:10.1016/j.compchemeng.2019.05.020.

Yondo, R., Andrés, E., Valero, E., 2018. A review on design of experiments and surrogate models in aircraft real-time and many-query aerodynamic analyses. Prog. Aerosp. Sci. 96, 23–61 doi:10.1016/j.paerosci.2017.11.003.

Zantye, M.S., Arora, A., Faruque Hasan, M.M., 2019. Operational power plant scheduling with flexible carbon capture: a multistage stochastic optimization approach. Comput. Chem. Eng. 130, 106544 doi:10.1016/j.compchemeng.2019.106544.

Zhai, J., Boukouvala, F., 2019. Nonlinear variable selection algorithms for surrogate modeling. AIChE J. 0. doi:10.1002/aic.16601, (ja), e16601.

Zhang, K.-.S., Han, Z.-.H., Gao, Z.-.J., Wang, Y., 2019. Constraint aggregation for large number of constraints in wing surrogate-based optimization. Struct. Multidiscipl. Optim. 59 (2), 421–438. doi:10.1007/s00158-018-2074-4.