Data-driven Spatial Branch-and-bound Algorithm for Box-constrained Simulation-based Optimization

Jianyuan Zhai, Fani Boukouvala*

Georgia Institute of Technology, School of Chemical & Biomolecular Engineering, 311 Ferst Dr., Atlanta, GA, 30332, USA

Abstract

The ability to use complex computer simulations in quantitative analysis and decision-making is highly desired in science and engineering at the same rate as computation capabilities and first-principle knowledge advance. Due to the complexity of simulation models, direct embedding of equation-based optimization solvers may be impractical and data-driven optimization techniques are often needed. In this work, we present a novel data-driven spatial branch-and-bound algorithm for simulation-based optimization problems with box constraints, aiming for consistent globally convergent solutions. The main contribution of this paper is the introduction of the concept data-driven convex underestimators of data and surrogate functions, which are employed within a spatial branch-and-bound algorithm. The algorithm is showcased by an illustrative example and is then extensively studied via computational experiments on a large set of benchmark problems.

Keywords: black-box optimization, simulation-optimization, branch-and-bound, global optimization, convex underestimators

Declaration

Funding: National Science Foundation (NSF-1805724), RAPID and Georgia Institute of Technology Georgia Tech Startup Funding

Data-availability statement: My manuscript data will be available as supplementary materials

Conflicts of interest/Competing interests: None

*Corresponding author.
Email address: fani.boukouvala@chbe.gatech.edu (F. Boukouvala).

Introduction

Review of Black-Box Optimization and Challenges

Advances in computational capabilities have led to more frequent use of high-fidelity computer simulations for quantitative studies and decision-making in science and engineering in the recent decades[1,2]. The ability to solve design and optimization problems with embedded simulations is of high interest, often bridging the gap among multidisciplinary or multiscale knowledge. Simulations are often treated as "black-box" input-and-output data-generators within optimization frameworks, either because the details of the models are inaccessible under a proprietary license, or because the algebraic formulations are intractable due to transformations of systems of differential equations and if-then operators [3-5]. As a result, optimization of such black-box problems usually relies on the input-and-output data solely. Many recent contributions from the engineering literature aim to improve the performance of black-box optimization techniques for a wide variety of applications [6-11].

Existing black-box optimization methods can be categorized into two groups: a) sampling-based and b) model-based. Under the sampling-based category, there are deterministic and stochastic methods. Widely used examples of deterministic sampling-based methods include Nelder-Mead simplex algorithm [12], Pattern-Search [13,14], Mesh Adaptive Direct Search [15], the DIRECT algorithm [16,17] and Multilevel Coordinate Search (MCS) [18]. Among the aforementioned deterministic methods, DIRECT and MCS methods are global search methods that aim to approach the global optimum by progressively partitioning the search space. This approach can also be categorized as a deterministic Lipschitz optimization method with an unknown Lipschitz constant. Another class of sampling-based methods employ stochastic steps to randomize the search within bounded spaces, and popular examples are Simulated Annealing [19], Evolutionary Algorithms [20,21] (i.e., Genetic algorithms) and Particle Swarm Optimization Algorithms [22]. These methods have been widely adopted because of their global nature and user-friendly software implementations [1,23]. However, they typically require a significantly large number of samples to guarantee global convergence, and this may limit their application for certain simulation-

based case studies [24]. Sampling-based local search methods may converge to local stationary points and multistart approaches can be used to increase the chance of locating the global solution [23,25]. The literature is very rich and continuously growing in this area and we do not aim to perform a thorough review of these methods here. However, several excellent review articles have outlined the distinction between the these categories and provide a comprehensive list of methods and tools available [23,1,25-27].

Model-based methods, also referred to as surrogate-based methods, use surrogate models as mathematical approximations of the relationship between the input and output data. Local model-based methods, such as trust-region search (i.e., Powel's method) and implicit filtering, used surrogate models in a local search region to expedite the search. Global model-based methods construct surrogate models that represent the entire search space or subspaces generated by partitioning to find the global optimum. Sequential Design for Optimization (SDO) [28], Efficient Global Optimization (EGO) [29], and Stable Noisy Optimization by Branch-and-Fit (SNOBFIT) [30] are some of the most popular and high performing implementations of global model-based methods with search space partitioning. Earlier literature focused on linear or quadratic approximation models as the surrogates, but more recent work has been exploring more complex Machine Learning surrogate functions (i.e., Gaussian Process Models (GP), Artificial Neural Networks (ANN), Support Vector Regression (SVR), Generalized Linear Regression, Regression Trees (RT), etc.) [31,10,11,32,33].

Most of the current existing black-box solvers, shown in computational studies and reviews, exhibit varying performance that depends highly on the characteristics of the optimization problems and no single solver clearly outperforms all others for a wide range of problems [23,1,25]. Many authors have observed that the fitted surrogate models generally accelerate the search because they allow predictions to be made even in regions where samples are not collected [5,34]. There are also several drawbacks to using surrogate models, such as the added computational cost of fitting (optimization of parameters), tuning (hyperparameters) and validating these models, as well as challenges in optimizing them using deterministic

global optimization solvers [32,5,35]. Moreover, selecting the type of surrogate model and its training procedure given the available samples creates uncertainty in the outcome of each method run.

In this work, we adopt some of the characteristics of model-based methods due to their previously reported promising potential in exploring search spaces fast, but we propose a novel framework that aims to tackle some of the aforementioned challenges. Specifically, one common phenomenon is the inconsistency in solutions given different sample sets. Many efforts have been devoted to improve the accuracy of surrogate models in the sampling and modeling methodologies [9,36,32,11]. However, the best surrogate modeling and sampling strategy is still an open question and the answer depends heavily on the characteristics of the problem [5,34]. Moreover, most algorithms employ non-rigorous termination criteria, such as termination at sampling and/or computation limits, or no significant improvement in executive iterations. Therefore, it is usually impossible to know whether the solution would improve with more samples collected and no further information is provided at convergence regarding the quality of the incumbent solution. In the case where equations are available, deterministic derivative-based optimization solvers employ ways to search the space efficiently and provide upper and lower bounds on the final solution. In this work, we propose an algorithm that aims to provide better quality solutions and approximate bounds on these for problems that rely on input-output data. The current challenges and motivation of this work will be shown through a simple motivating example below.

Motivating Example

Many of the aforementioned surrogate-based optimization algorithms employ adaptive sampling techniques in an effort to limit sampling requirements. The typical steps are:(a) start with a low number of samples, (b) fit a surrogate model, (c) optimize surrogate formulation and identify new sampling locations using exploration and/or exploitation criteria and (d) repeat the process until convergence. These adaptive model-based algorithms have been shown to exhibit good performance, however, the issue of convergence to different solutions if initialized with different samples, or if a different surrogate model is used, always exists [32,5]. Another approach to solve black-box optimization problems (especially if sampling is not the

limiting factor) has been to collect many samples *a-priori* in order to train a sophisticated Machine Learning model that is then optimized globally in a single iteration. Recent promising work makes this possible due to development of customized global optimization algorithms specialized for artificial Neural Networks and Gaussian Process models [10]. However, even this approach may still lead to variation in the final optimal solution due to the uncertainty introduced by sampling and model fitting.

To show this challenge, we implement a surrogate-based algorithm that can train a support vector regression (SVR) model with radial basis function kernels via 10-fold cross-validation, and then embed it within a formulation which is optimized globally using equation-based global solver BARON [37]. The 'six-hump camel function' (6) is chosen as a motivating example and in this case we treat it as a black-box function. The test function contains only two input variables and the function is evaluated within $x_1 \in [-3, 3]$ and $x_2 \in [-2, 2]$.

$$f(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2 \qquad (6)$$

This problem has two global minima $f^* = -1.0316$ at $(0.0898, -0.7126)$ and $(-0.0898, 0.7126)$, which are marked with two red dots in the contour plot of the problem in Figure 1.

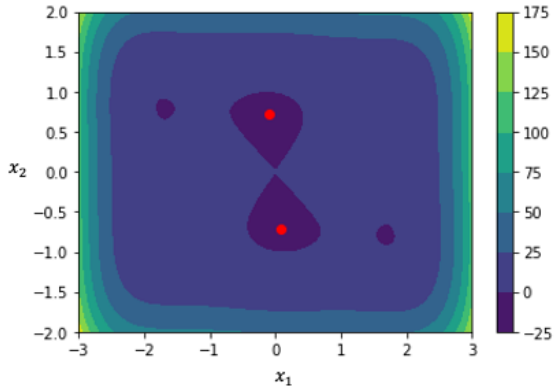

Figure 1 Contour plot of six-hump camel function (Global optima (●))

To examine the effect of sampling and fitting uncertainty, we optimize the test function after the SVR model is trained with varying sizes of Latin Hypercube sampling (LHS) sets[38]. The size of the LHS designs ranges from 100, to 500 and to 1000 samples and for each case we repeat the process 10 times. For each of the replications, a different SVR function is fitted with a different randomly created LHS design of the same size, and the optimal SVR model parameters are trained using the same k-fold cross-validation procedure. This simple test is used to assess the variability in identified optimal solutions, even when a large number of samples is available a-priori, the surrogate model is trained using thorough training-validation practices and the surrogate function is then globally optimized. The distribution of the solutions found with respect to the number of samples collected is plotted in Figure 2, where it can be observed that the variation in globally identified solutions decreases with increasing number of samples used. However, variation is still observed in the solutions found by globally optimizing the SVR models trained with 1000 samples. This variability in final solutions, even when a deterministic global optimization solver is applied, may come mainly from the fact that slight changes in training samples lead to differences in the parameters and eventually the global optima of these SVR models. This variability may be slightly higher or lower if a different surrogate model was used, but we have observed this behavior with other types of surrogate models, including Neural Networks and Kriging models[39]. It is this uncertainty that has led partially to lack of adoption of surrogate-based optimization techniques and mistrust of these methods. It must also be noted that we have just showed the variability observed with the singe-stage optimization approach and not the adaptive sampling-fitting procedure that many algorithms use. However, variability in obtained solutions has been reported in many previous works, for the same reasons of different initialization of samples or surrogate model selection and fitting even when an adaptive approach is used [32,34,36].

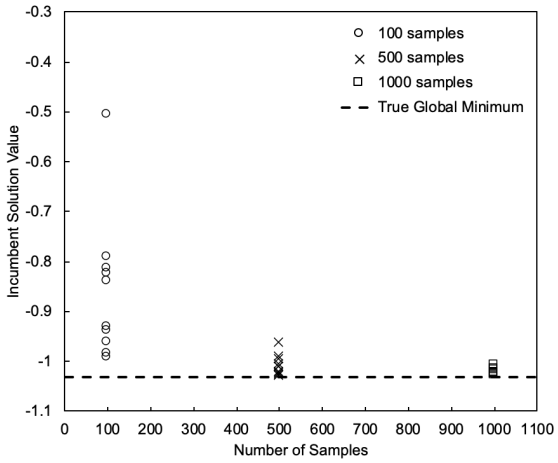

Figure 2 10 repetitions of incumbent solutions found by globally optimize SVR models trained with sample sets with 100, 500, and 1000 points

Problem Formulation and Novelty of Proposed Work

In this work, we propose a novel black-box optimization algorithm aiming to provide consistent solutions and indicate the potential improvement of the solution for box-constrained problems (F1).

$$(F) \quad \min_{x} f(\boldsymbol{x})$$

$$s.t. \boldsymbol{x_{lo}} \leq \boldsymbol{x} \leq \boldsymbol{x_{up}}$$

$$\boldsymbol{x} \in \mathbb{R}^n$$

Where $f(x)$ is the black-box objective function and variable x is bounded by an n-dimensional box $[\boldsymbol{x_{lb}}, \boldsymbol{x_{up}}]$. The key novelty of our approach is the adoption of the structure of deterministic spatial branch-and-bound (SBB) algorithms. Deterministic SBB algorithms are implemented in many equation-based global solvers for mix-integer nonlinear programming problems (MINLP) such as ANTIGONE [40], αBB [41], BARON [37]. As shown in Figure 3(a), with a known mathematical formulation of the optimization problem, a deterministic SBB algorithm first identifies a feasible solution as the upper bound

(UB) of the unknown global optima. Then, a convex underestimator (f_{lb}) for the nonconvex objective function is derived and its minimum serves as the lower bound (LB) of the global optima [40,37,42,25]. Next, the algorithm will progressively partition the search space, and repeat the local search and underestimating process. Eventually, the deterministic SBB algorithm converges when the gap between the lower bound and the upper bound becomes smaller than some tolerance value ε [40,37,42,25]. By comparing the lower bound in each subspace to the incumbent solution, some search spaces can also be pruned. For example, subspace S_1 is pruned because LB_1 is higher than UB_2 inferring that the global optimum is not in S_1.

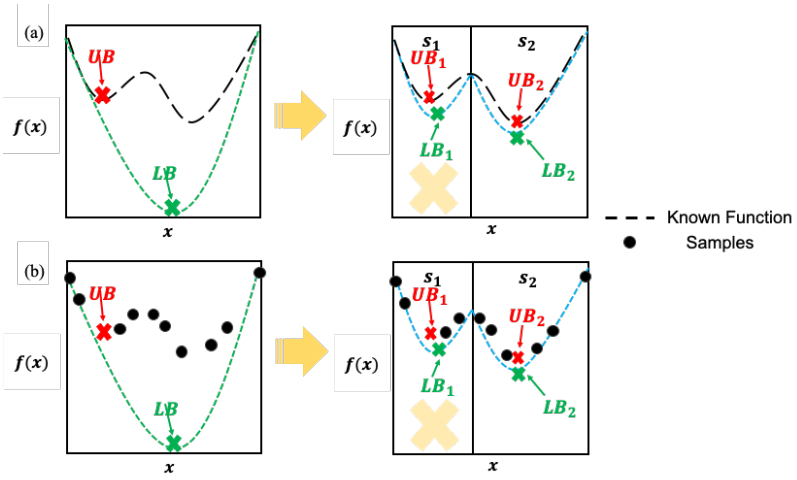

Figure 3 Schemes of (a) deterministic spatial branch-and-bound algorithm and (b) data-driven spatial branch-and-bound algorithm

In this work, we adopt the algorithmic structure of deterministic spatial branch-and-bound algorithms (SBB) and develop a data-driven equivalent. The key novelty of the framework is the use of data-driven convex underestimators (DDCU) with similar functions as the deterministic equation-based underestimators as in Figure 3(b). Data-driven convex underestimators are trained purely from the data in a way that we guarantee convexity and boundness of all current samples from below. We achieve this by solving a linear programing problem. By subdividing the search space, the data-driven convex

underestimator is expected to become closer to the unknown function and provide tighter lower bounds. Through the comparison of lower bounds and the incumbent solution, the search space can be reduced so that more samples can be collected in the areas that are more promising to find the global optima. Our method differs from Convex Relaxation Regression (CoRR) proposed by Azar et. al. [43] because CoRR focuses on training a global convex regression function for optimization while DDSBB-DDCU trains many local convex underestimators as it is partitioning the search space. In addition, we use various ML techniques to improve the convergence of the algorithm. Specifically, ML models are trained with high-fidelity samples collected from the black-box function to provide cheap low-fidelity samples. These ML models are also used to select the variables to be branched on.

Although some methods in the literature also incorporate partition strategies and the idea of underestimators, DDSBB-DDCU distinguishes itself in the way that the underestimators are derived from data solely and are practically used to guide the search and the pruning of the search spaces. In contrast, empirical stochastic branch-and-bound algorithm (ESBB) [44] employs stochastic sampling-based bounds to guide the branch and the search for the optimal solution. However, the lower bound and upper bounds of all the subregions are assumed to be equal. Some methods require additional information on the derivatives of the black-box simulation in order to derive the understimators. The SmoothD [45] algorithm derives smooth convex auxiliary underestimating functions assuming the objective function values and derivatives of the objective function can be obtained as black-box simulations simultaneously. Nonetheless, the subspaces are never pruned by comparing the lower bound with the incumbent solution but are prioritized by the lower bounds. A branch-and-bound approach is also proposed by Bajaj et al. [7], but the implementation requires the knowledge of a bound of the Hessian matrix of the black-box function to develop concave understimators.

Methods

Data-driven Convex Underestimators

Convex underestimators are one of the most important components of a spatial branch-and-bound algorithm. Convex relaxations of known functional forms are well-studied in the literature, such as McCormick relaxations [46] and outer approximation [47]. The underestimator guarantees convexity and underestimation of the original non-convex function. In the absence of the mathematical formulation of the black-box function, the derivation of the convex underestimator has to be purely data-driven. As a result, we do not claim any guarantees of building a convex underestimator of the true unknown black-box function, but we can claim to find a valid convex underestimator of all of our samples. To do so, we propose a linear programming formulation (F2) to obtain a data-driven convex underestimator that bounds all samples from below.

$$(\text{F2})\ \min \sum_i^N \left(f(\mathbf{x_i}) - f_{lb}(\mathbf{x_i})\right) \quad (1)$$

$$\text{s.t.}\ \ f(\mathbf{x_i}) - f_{lb}(\mathbf{x_i}) \geq 0 \quad \forall\, i = 1 \text{ to } N \quad (2)$$

$$f_{lb}(\mathbf{x_i}) = \mathbf{a}\mathbf{x_i^2} + \mathbf{b}\mathbf{x_i} + c \quad \forall\, i = 1 \text{ to } N \quad (3)$$

$$\mathbf{a} \geq 0 \quad (4)$$

$$\mathbf{a}, \mathbf{b} \in \mathbb{R}^D, c \in \mathbb{R} \quad (5)$$

where D is the dimension of the input space and *N* is the total number of samples collected. The objective function (1) is to minimize the distance of all sample points x_i within a search subspace X_k. The first constraint (2) guarantees that the underestimator f_{lb} bounds all sample points from below. The underestimator is convex by enforcing parameter **a** for the squared term to be non-negative in constraints (3 - 4). Noticeably, formulation (F2) is a linear programming problem and can be solved using linear solvers without adding significant computational cost. This approach is different from any similar approach in the literature which tries to employ underestimators [45,7] in the sense that DDCU does not require any derivative information. After the formulation of $f_{lb}(\mathbf{x_i})$ is obtained, the lower bound (LB_k) of the local

search space can be found analytically as the minimum of f_{lb}. The minimum of the samples $f(\mathbf{x_i})$ then serves as the upper bound (UB_k) of the local search space.

Returning to our motivating example, in Figure 4, we plot the DDCU obtained using 23 LHS samples for the 'six hump camel' function. The convex underestimator is able to bound all samples from below; however, it does not bound the black-box function in all of the search space. In this example, with only 23 samples the upper bound obtained is -0.041 and the lower bound is –12.766 which successfully underestimates the true global optimum (-1.0316). In order to quantify the overestimation of the unknown black-box function, we simulated 10,000 grid points in the 2D space and found that 81.8% of those are bounded by the underestimator trained with only 23 points. This core concept of data-driven convex underestimators forms the basis of our algorithm. However, the algorithm is comprised of many other components that aim to continuously improve the validity of underestimators, prune subspaces, and converge to the global optimum with high probability.

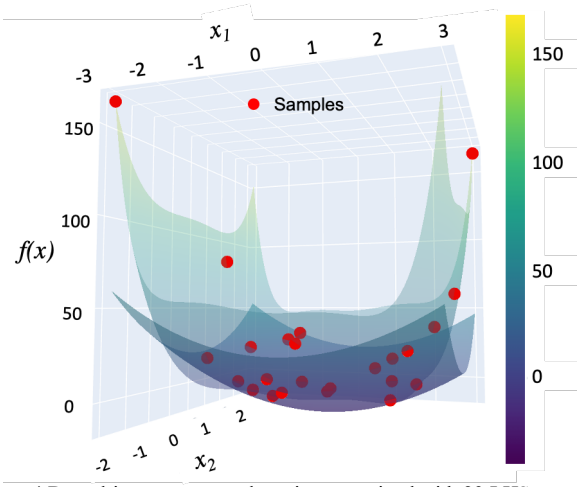

Figure 4 Data-driven convex underestimator trained with 23 LHS samples

One promising approach to improve the validity of the data-driven convex underestimators with respect to the underlying unknown black-box function is to add more samples. The hypothesis is that with increasing samples collected, the convex underestimator will be able to bound from below more regions

that have not yet been sampled. To validate this, we repeat the above experiment by adaptively collecting more samples to train the convex underestimator and the percentage of the 10,000 grid points bounded by these convex underestimators are shown in Table 1. As shown in Table 1, more percentage of validation points are bounded by the data-driven convex underestimator trained with more samples. It must also be mentioned that we are able to bound 98.8% of the samples with 300 samples, which is lower than the samples used to fit a single model in the motivating example and still observe variability. However, in many applications, collecting high-fidelity samples from the black-box simulation can be very expensive.

Table 1. Percentage of 10,000 validation points bounded by DDCU trained with increasing samples

Number of Samples	23	50	75	100	125	150	300
Percentage Bounded	81.8%	81.9%	94.3%	94.3%	96.9%	96.8%	98.8%

To tackle the potential limit on high-fidelity samples, we propose a multi-fidelity data approach (MF) which combines the high-fidelity samples and some low-fidelity samples collected using a ML model. Specifically, SVR with radial basis function kernel (RBF) [48,49] models are trained in the subregions using the high-fidelity samples. Although we select this specific type of ML model in this example, the type of ML model used to fit the data and produce the low-fidelity samples can be any regression or interpolating or ensembles of models, including Neural Networks, Gaussian Process Models, Generalized Regression Models, and many more. The enhanced formulation (F3) incorporates additional constraints (9,11) to ensure the trained convex underestimator also bounds the "cheap" low-fidelity samples from below, where x_m represent the inputs to collect the low-fidelity samples $f_{lf}(\mathbf{x_m})$ and M is the number of low-fidelity samples collected. More importantly, the objective function is revised to minimize the distance between the low-fidelity samples and the lower bound.

$$\text{(F3)}\ \min \sum_1^N \left(f(\mathbf{x_i}) - f_{lb}(\mathbf{x_i})\right) + \sum_1^M \left(f(\mathbf{x_m}) - f_{lb}(\mathbf{x_m})\right) \quad (7)$$

$$\text{s.t.}\ \ f(\mathbf{x_i}) - f_{lb}(\mathbf{x_i}) \geq 0 \quad \forall\, i = 1 \text{ to } N \quad (8)$$

$$f_{lf}(\mathbf{x_m}) - f_{lb}(\mathbf{x_m}) \geq 0 \quad \forall\, m = 1 \text{ to } M \tag{9}$$

$$f_{lb}(\mathbf{x_i}) = \mathbf{a}\mathbf{x_i^2} + \mathbf{b}\mathbf{x_i} + c \quad \forall\, i = 1 \text{ to } N \tag{10}$$

$$f_{lb}(\mathbf{x_m}) = \mathbf{a}\mathbf{x_m^2} + \mathbf{b}\mathbf{x_m} + c \quad \forall\, m = 1 \text{ to } M \tag{11}$$

$$\mathbf{a} \geq 0 \tag{12}$$

$$\mathbf{a}, \mathbf{b} \in \mathbb{R}^D, c \in \mathbb{R} \tag{13}$$

An example of the convex underestimator obtained using the same set of high-fidelity samples and 100 low-fidelity points using random sample strategy is shown in Figure 5. In this case, the upper bound found is the same as in the previous approach, while the lower bound becomes -11.203. The percentage of 10,000 validation points bounded by the underestimator increases from 81.8% to 87.3%, which indicates the underestimator trained with multi-fidelity samples is more conservative than the one trained with high-fidelity samples only. With adaptively collecting more high-fidelity samples to train the SVR model, we

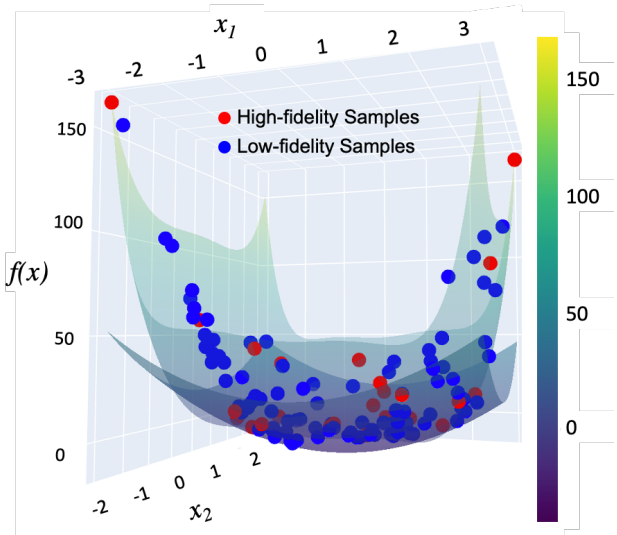

Figure 5 Data-driven convex underestimator trained with 25 high-fidelity samples and 100 low-fidelity samples

also observe that the percentage of validation points bounded increases from 87.3% to 99.6%, as shown in Table 2.

Table 2. Percentage of 10,000 validation points bounded by DDCU trained with increasing high-fidelity samples and fixed number of low-fidelity samples

Number of Samples	23	50	75	100	125	150	300
Percentage Bounded	87.3%	82.8%	94.2%	93.2%	94.3%	98.5%	99.6%

There are two key observations to be made here. First, as expected, increasing the samples leads to more conservative and more "valid" under-estimators. More importantly, the effect of low-fidelity samples is more significant when high-fidelity samples are scarce, and these low-fidelity samples are "cheap" to evaluate.

To summarize, we see that in this motivating example with only 23 high-fidelity samples and no assumption about the nature of the underlying black-box function, the DDCU does not bound the function entirely. However, we have not used adaptive sampling or branching in this example. These initial "global" underestimators showed in the above examples are trained in the head node, where no pruning is happening. Our hypothesis is that as more samples are collected, and the search-space is reduced (using branching) the under-estimators will become more accurate. As a result, the aim of this work is to explore if the overall approach of DDCU coupled with branch-and-bound leads to better and more consistent performance than previously proposed approaches. However, no guarantees of global convergence in the deterministic sense are made, as it is not possible unless assumptions regarding the form of the "black-box" function or its derivatives are made *a-priori*, which will be the focus of future work.

Data-driven Spatial Branch-and-bound Algorithm

The data-driven spatial branch-and-bound algorithm is an iterative procedure that progressively partitions the search space and uses the underestimating method to fathom some search subspaces. As shown in Figure 6(a), the algorithm consists of three main blocks (solid-line) and one alternative block

(dashed-line). The pathway following the solid arrows is the "essential" flow of the algorithm. By activating the "alternative" pathway, surrogate models are added to improve the performance of the algorithm by providing additional information, such as variable ranking and/or multi-fidelity sampling. The algorithm starts from the root node and as the full search space subdivides, each node branches into two child nodes and eventually develops a tree structure similar to the example shown in Figure 6(b). The algorithm is implemented in Python with dependent packages: numpy[50,51], pyomo [52], pyDOE [53] and scikit-learn [54].

Essential Path of DDSBB

To initialize the algorithm, the initial search space $[\mathbf{x_{lo}^0}, \mathbf{x_{up}^0}]$ must be provided to the first algorithmic block that represents sampling. Latin Hypercube Sampling (LHS) is implemented for initial sampling in the root node and adaptive sampling to ensure that at least $\min\left(\left\lfloor\frac{min(10D,250)}{l}\right\rfloor + 1, \lfloor 2D \rfloor + 1\right)$ samples are in the child node k, where l is the level of the tree. Since LHS designs result in samples that are never located on the boundaries of the search space and this is important for convex underestimators, we also sample the corner points $\mathbf{x_{lo}^k}$ and $\mathbf{x_{up}^k}$ in each node. This is a sampling heuristic that allows us to add corner points without significantly increasing sampling requirements.

The second algorithmic block of the algorithm represents the process of deriving the convex underestimators and applying the DDCU method to obtain UB_k, LB_k and $\mathbf{x_{lb}^k}$ for each subregion k. Additionally, the minimizer $\mathbf{x_{lb}^k}$ of the convex underestimator f_{lb}^k of subspace k is validated by obtaining $f\left(\mathbf{x_{lb}^k}\right)$ from the black-box simulation. We also ensure $\mathbf{x_{lb}^k}$ is not resampled if $\mathbf{x_{lb}^k}$ is already in the sample set. Note that the training process of DDCU in each node is iterative, because the new sample point $f\left(\mathbf{x_{lb}^k}\right)$ may be higher than the current UB_k when validated by inquiring the high-fidelity simulation. Therefore, the UB_k is updated with the new samples until LB_k is lower bounding the local minimum.

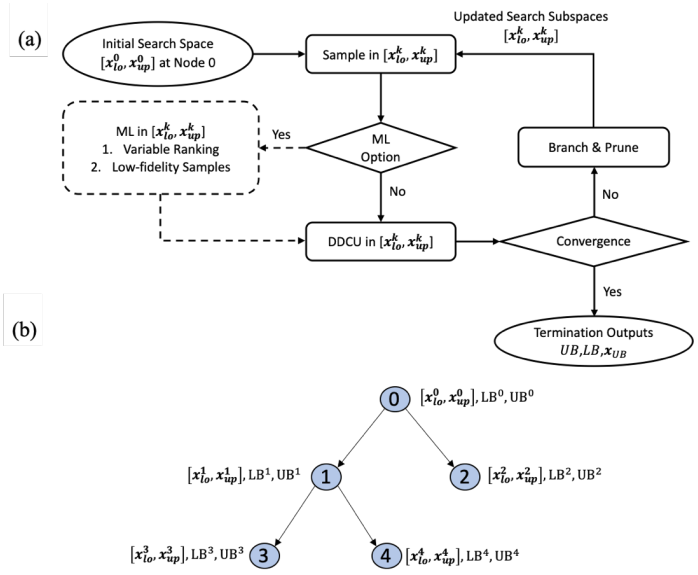

Figure 6 (a) Overview of the data-driven spatial branch-and-bound algorithm and (b) scheme of a branch-and-bound tree

After searching all active nodes, the upper bounds and lower bounds are compared across all active nodes and the minima among all local upper bounds and lower bounds are logged as the global upper bound (UB) and the global lower bound (**LB**), respectively. Next, the "Branch & Prune" block prunes some nodes if the lower bound is higher than the global upper bound and then determines the search region $[\mathbf{x_{lo}^k}, \mathbf{x_{up}^k}]$ for the next level by bisecting the selected dimension. The simplest rule to follow is to select the dimension d that has the maximum distance between $x_{lo_d}^k$ and $x_{up_d}^k$. The algorithm converges when the absolution gap $|UB - LB| \leq 0.05$ or the relative gap $\frac{|UB-LB|}{|LB|} \leq 0.001$. More stopping criteria are imposed to terminate the algorithm when the number of samples and CPU exceed a certain budget, however, to test the convergence behavior of our algorithm we set those criteria to very large specifications. The algorithm also stops if the maximum bound distance on the active subspace is smaller than an absolute tolerance of 0.05, which implies that the search space is very small and as a result this would lead to unnecessary sampling

and overfitting of surrogate models. Since decision-making depends on the high-fidelity samples from the black-box simulation in the essential pathway, this pathway is also referred to as high-fidelity (HF) approach in the rest of the paper.

Alternative Path of DDSBB

The alternative block applies ML techniques to improve the validity of the convex underestimators and the accuracy of the incumbent solution using the multi-fidelity approach. This block can be an add-on to provide many cheap low-fidelity samples to train the underestimator using formulation (F3). In addition to the minimizer of the f_{lb}^{k}, the minimizer $\mathbf{x}^{k}$ of the low-fidelity samples in node k will also be validated by evaluating $f(\mathbf{x}^{k})$ from the black-box function. Before validating any new points, the algorithm makes sure $\mathbf{x}^{k}$ does not already exist in the sample set.

Moreover, the SVR model trained to provide low-fidelity samples, can also help the algorithm determine the variable to be branched on. In our previous work [55], we presented an SVR-based variable selection algorithm ranks the input variables by a sensitivity-based criterion implying their relative importance to the training accuracy of the SVR model [56,55]. It is hypothesized that branching on the most important variable is a more informative heuristic that could expedite the convergence of the algorithm. These two ML-based features (ML-based branching and ML-regression for multi-fidelity sampling) do not have to always be used in tandem. In other words, one could use SVR-based variable selection for branching, but not turn on the multi-fidelity sampling feature, and we will explore different combinations of such features in the following sections.

Results

Motivating Example

Figure 7 shows the visualization of the branch-and-bound processes of four variations to solve the motivating example with 23 initial LHS samples: (a) the high-fidelity approach without variable selection for branching (HF), (b) the high-fidelity approach with variable selection for branching (HF_VS), (c) the

multi-fidelity approach without variable selection for branching (MF) , and (d) the multi-fidelity approach with variable selection for branching (MF_VS). In all subplots of Figure 7, the search spaces are shaded with grey colors in different gradients; the lighter the color, the earlier the subspace gets pruned in the process. We observe that larger subregions that are less promising to find the global optimum are pruned faster in the process with variable selection (7 (b) and (d)) than in those without variable selection (7(a) and (c)). All variations of the algorithm are able to locate at least one of the global optima of the test function. While the high-fidelity approaches are subject to higher risk of pruning regions that contain one of the global optima, MF and MF_VS are able to concentrate the search in the regions containing both global optima with the help of the low-fidelity samples.

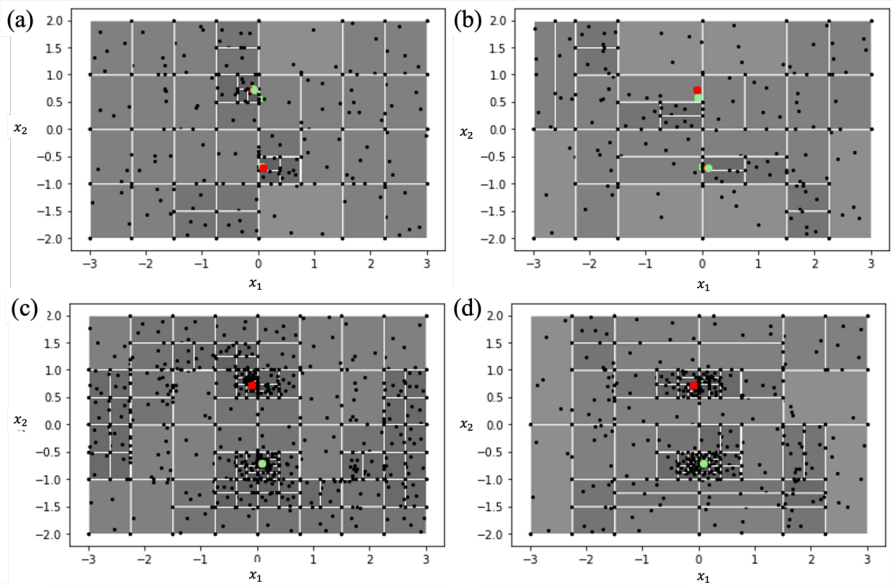

Figure 7 Illustration of data-driven spatial branch-and-bound algorithms: (a) HF, (b) HF_VS, (c) MF, and (d)MF_VS.

To address the issue on the variability introduced by different initial sample sets, we repeat the above computational experiments 10 times for each DDSBB algorithm variations. Due to the randomness in data-driven processes, the algorithms may converge with different number of samples and iterations, but

what is more important is that upon convergence all runs converge to a global optimum. In Figure 8, we align the results shown in Figure 1 with the upper bounds found with the number of samples collected in the progress of HF, HF_VS, MF and MF_VS. By comparing Figure 8 (a) and (c) with (b) and (d), we notice that variable selection is able to reduce the variation in the incumbent solution found after the third iteration across different runs. Even though the multi-fidelity approach requires more samples to converge eventually, the upper bounds approach the global minimum with less samples and the variability of the upper bounds as the algorithm progresses is smaller when compared to that of the high-fidelity approaches. Above all, all versions of the DDSBB algorithm are able to find consistent solutions in all 10 runs initialized with different sample sets and the variability in the incumbent solutions is much smaller than the solutions found by globally optimizing the SVR models with an equivalent number of samples. It is also important to note here that the DDSBB algorithm converged each time either due to closing the absolute or relative gap and not due to sampling or CPU limitations.

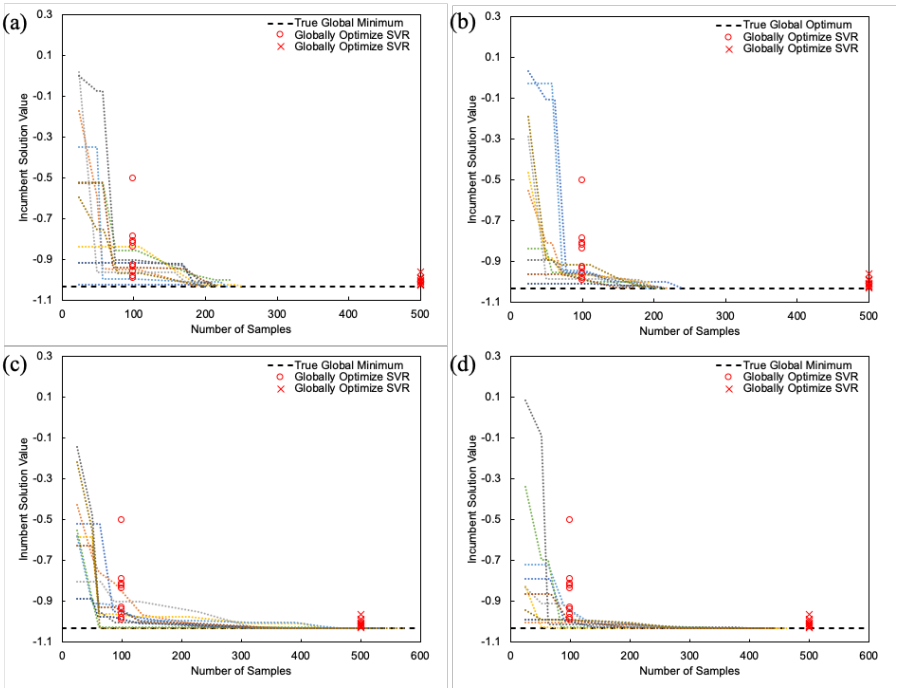

Figure 8 History of the upper bounds for 'six hump camel' function with increasing number of samples collected in 10 runs of data-driven spatial branch-and-bound algorithms: (a) HF, (b) HF_VS, (c) MF, and (d)MF_VS.

In addition to the variability of the upper bounds, the variability in the lower bounds under different initial samples is also crucial to the data-driven spatial branch-and-bound algorithm, because the lower bound controls the speed of convergence and the accuracy of pruning. The record of the lower bounds in the same 10 runs are shown in Figure 9. Due to the randomness in sampling and the lack of derivative information, the low bounds found by the data-driven convex underestimators do not follow a strict monotonous convergence behavior as those in the equation-based branch-and-bound algorithms in the first iterations. Noticeably, trained with the same number of high-fidelity samples, the lower bounds found by the multi-fidelity approach are more conservative; thus, the algorithm converges with more samples than those found by the high-fidelity approach. In the absence of derivative information, the increased conservativeness can help the algorithms prune the search space more cautiously. We also observe fewer

cases where the lower bound is higher than the true global optimum in the multi-fidelity approaches (Figure 9 (c,d)) than in the high-fidelity approaches (Figure 9 (a,b)). These results validate our hypothesis that adding more samples (even low-fidelity samples) the underestimators become more conservative, which eventually lead to more accurate solutions.

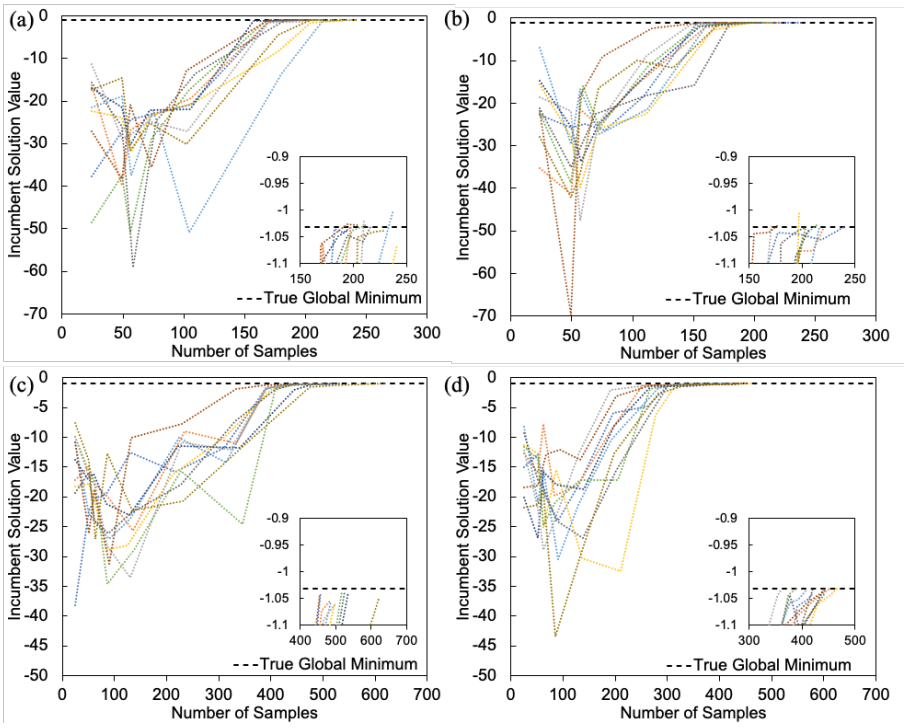

Figure 9 Lower bounds found for 'six hump camel' function with increasing number of samples collected in 10 runs of data-driven spatial branch-and-bound algorithms: (a) HF, (b) HF_VS, (c) MF, and (d)MF_VS.

Computational Studies on Benchmark Problems

Solution Accuracy

Besides the illustrative example, the performance of the DDSBB algorithms is further examined on a large set of continuous box-constrained benchmark problems with known global optima obtained from

Deleted: ¶

[57]. These benchmark problems have been divided into two groups depending on the dimensionality. The lower dimensional group contains 118 problems with 2-3 variables and the higher dimensional group contains 69 problems with 4-10 variables. The complete list of the test problems is available in the Appendix. The performance of the four variations of the data-driven spatial branch-and-bound algorithms is compared with two current state-of-art algorithms, DIRECT and SNOBFIT, under the same sampling and CPU limitations. DDSBB algorithms are initialized with $10N + 1$ initial samples and converge at $|UB - LB| \leq 0.05$ or the relative gap $\frac{|UB-LB|}{|LB|} \leq 0.001$. The commercially available solvers initialization and convergence criteria depend on the settings of the available implementations. DIRECT is set to stop when the absolute difference in the solutions found between two iterations is smaller than 0.05. SNOBFIT settings dictate that the number of initial samples and adaptive samples in each call are both $N + 6$, the minimum sampling requirement is 10N+1, and the algorithm stops when the solutions found in five consecutive iterations remain the same.

In order to make solution profiles for the fraction of problems solved by a specific solver with respect to sampling and CPU requirements, a proper performance criterion that can distinguish between the solvers is needed. We propose a performance criterion (14) to determine whether a problem is solved based on the incumbent solution.

$$f_{best} \leq \max\left(f^* + a, (1 + a)f^*\right) \quad (14)$$

where a is a scaling factor between 0 and 0.15. This criterion reflects whether the incumbent solution f_{best} is within a small distance scaled by a to the known global optimum f^*. By gradually increasing a, the fraction of problems solved is expected to increase as this criterion is being relaxed. In Figure 10 (a) and (b), the cumulative fraction of problems solved is plotted with increasing a for the lower dimensional and higher dimensional groups, respectively. As shown in Figure 10 (a), the fraction of problems solved plateaus for all six methods after a reaches 0.05 and MF_VS solves the most fraction of problems among all methods. In the higher dimensional group, the fraction of problems levels out when a is larger than 0.1,

and the differences in performances among the four variations of DDSBB algorithm gradually diminish as a increases. In both groups of test problems, DDSBB algorithms solve significantly more problems than SNOBFIT and DIRECT even with relaxed characterization criteria, which indicates SNOBFIT and DIRECT may terminate prematurely at local minima more easily than DDSBB algorithms.

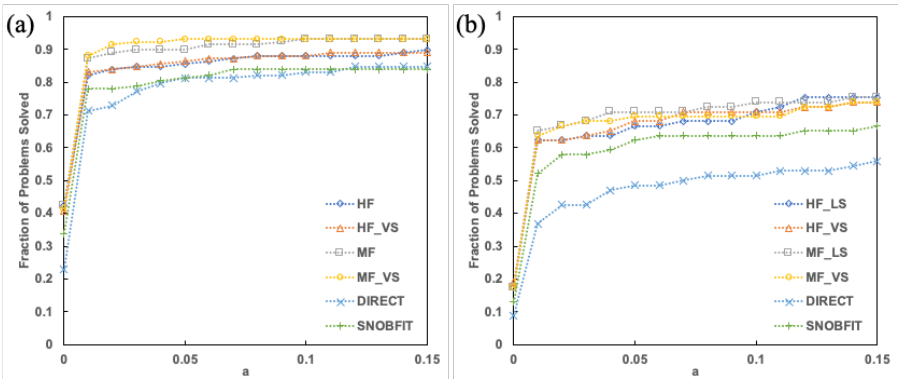

Figure 10 Fraction of problems solved with relaxed criterion (a) the lower dimensional group and (b) the higher dimensional group by HF, HF_VS, MF, MF_VS, DIRECT and SNOBFIT.

Sampling Requirements

Based on the results in Figure 10, at the conservative value of $a = 0.01$, the fractions of problems solved is distinguishable across six algorithms; thus, $a = 0.01$ is chosen to study the performance of the algorithms with respect to sampling and CPU requirements. Figure 11 (a) and (b) show the cumulative fraction of problems solved with the number of samples collected at termination for the two test problem groups. In both the lower and the higher dimensional group, DDSBB algorithms solve more problems but require more samples, which are needed to converge the lower bound and upper bounds. DIRECT solves a small fraction of problems with very little number of samples, because there is no stopping criterion on the minimum number of samples. In the lower dimensional group, we see that MF_VS converges and solves more problems than three other variations of DDSBB with the same number of samples. In the higher dimensional group, high-fidelity approaches converge with a smaller number of samples but solve less problems than the multi-fidelity approaches. Noticeably, HF_VS and MF_VS converges with less sampling

requirements than HF and MF, respectively, implying that variable selection plays an important role in accelerating the search in higher dimensional problems.

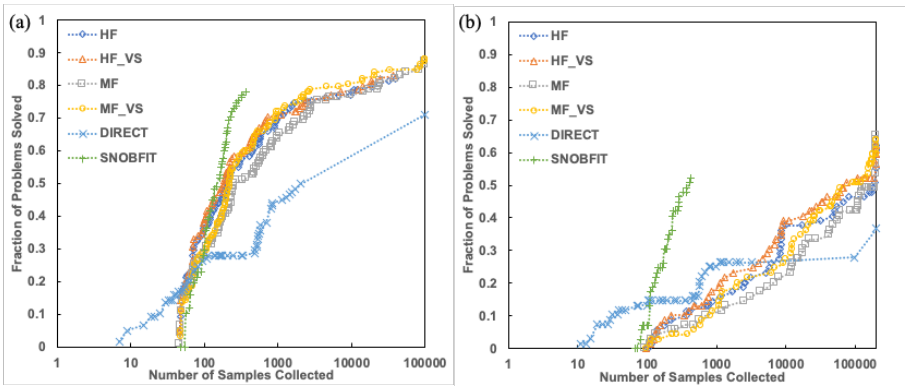

Figure 11 Fraction of problems solved with number of samples collected at termination in (a) the lower dimensional group and (b) the higher dimensional group by HF, HF_VS, MF, MF_VS, DIRECT and SNOBFIT.

CPU requirements

As for the CPU requirements, we plot the fraction of problems solved with increasing computational resources needed for the two groups of test problems in Figure 12 (a) and (b), respectively. Overall, we observe that DDSBB algorithms with embedded variable selection converge faster than the corresponding versions with branching on the longest dimension, which indicates the extra computation for the embedded variable selection algorithm is balanced off by the acceleration in search and pruning. Meanwhile, the employment of low-fidelity samples increases CPU requirements because the data-driven convex underestimator becomes more conservative and its training requires more CPU as well due to increasing number of constraints in (F3). Admittedly, DIRECT and SNOBFIT have significant advantages in sampling requirements over DDSBB algorithms, indicated by the CPU requirements at termination. However, if accuracy in solution is desired, we believe that the improvement in solution quality by DDSBB algorithms and the provided bounds on the solution outweigh the disadvantages in computational requirement in the cases when global optimization is the ultimate goal.

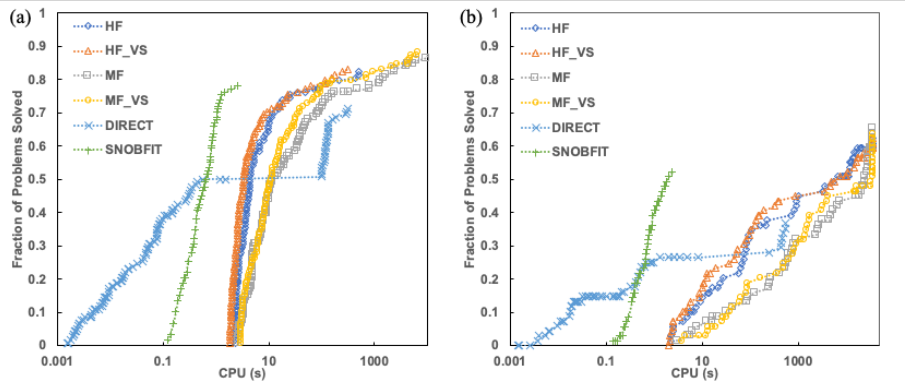

Figure 12 Fraction of problems solved with CPU requirement at termination in (a) the lower dimensional group and (b) the higher dimensional group by HF, HF_VS, MF, MF_VS, DIRECT and SNOBFIT.

Pre-Convergence Solution Quality

In many engineering applications where simulations are expensive, it is not always practical or feasible to allow global search algorithms, like DDSBB, to collect as many samples as needed. Therefore, the capability of approaching the global optimal with limited number of samples is an important aspect to evaluate. We record the minimum number of samples required to solve the problems before the algorithms terminate and plot the cumulative fraction of problems solved with the number of samples collected in Figure 13 (a) and (b) for all six algorithms. In both the lower and the higher dimensional groups, there is no significant difference in performance among the four DDSBB algorithms, which indicates the convergence behavior in Figure 11 (a) and (b) depends heavily on the conservativeness of the data-driven convex underestimators and on the branching rules. Overall, DDSBB algorithms are able to solve more problems with less samples compared to DIRECT. SNOBFIT, as discussed previously, is able to solve a relatively large portion of the problems with less samples than DDSBB algorithms and DIRECT but converges locally for the rest of the problems. Notably, DDSBB allows users to resume the search after it terminates with the preset limits after evaluating the potential gains with more samples invested, and after having pruned non promising regions entirely.

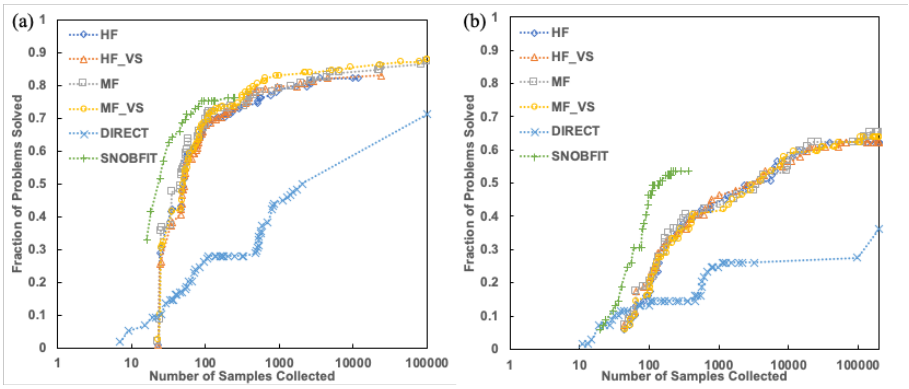

Figure 13 Fraction of problems solved with minimum number of samples collected in (a) the lower dimensional group and (b) the higher dimensional group by HF, HF_VS, MF, MF_VS, DIRECT and SNOBFIT.

Bounding Accuracy

Lastly, the validity of the lower bounds with respect to the known global optimum across different variations of DDSBB algorithms is investigated. Since the data-driven convex underestimators are derived with samples solely, it is misleading to claim the validity of the data driven convex underestimator under the same measure for the equation-based convex underestimators. However, some metrics on the gap between the global optimum and the lower bound from the equation-based global optimization literature would still be necessary and informative to assess the performance of our proposed approach, especially in the context of black-box optimization where no other method provides such information. Here, we calculate the absolute gap (14) between the known global optimum f^* and the best lower bound LB to character the validity.

$$Relative\ Gap = f^* - LB \tag{14}$$

Unlike the equation-based convex underestimators, which theoretically guarantee underestimating of the objective function, data-driven convex underestimators only guarantee underestimating all samples collected. Therefore, the proposed absolute gap may bear a negative value meaning that the lower bound fails to underestimate the global optimum, which is referred as a violation instance in the rest of the

discussion. In order to show the fraction of violation instances using different DDSBB algorithms, the absolute of solved and unsolved problems with various ranges of absolute gaps obtained at the termination of the algorithms are plotted in Figure 14. As shown in Figure 14 (a) and (b), the absolute gaps found mostly lies in the valid range of [0, 0.5), where above 98% and 95% of these problems in the lower and the higher dimensional groups, respectively. In the higher dimensional group, there are more problems with large absolute gaps above 100 than in the lower dimensional group because the algorithms terminate prematurely by hitting the sampling or CPU limit.

Most importantly, the summation of all violation instances makes up a relatively small portion of all test problems and the absolute gaps of most violation instances fall in the range of [-0.5,0), which indicates the lower bounds are very close to the global optimum at termination. Noticeably, majority of the slight violation instances are solved while severe violation instances with absolute gaps below -0.5 are never solved. Furthermore, violation instances happen more frequently when the high-fidelity approach is used, especially for the lower dimensional group, which agrees with the observations on the illustrative example that the introduction of low-fidelity samples improves the data-driven underestimators. The most conservative approach, MF, can provide a valid lower bound for above 90% problems in both the lower and the higher dimensional groups. Overall, the fractions of problems with a relative gap higher than -0.5 are above 90% by all four DDSBB algorithms regardless of dimensionality.

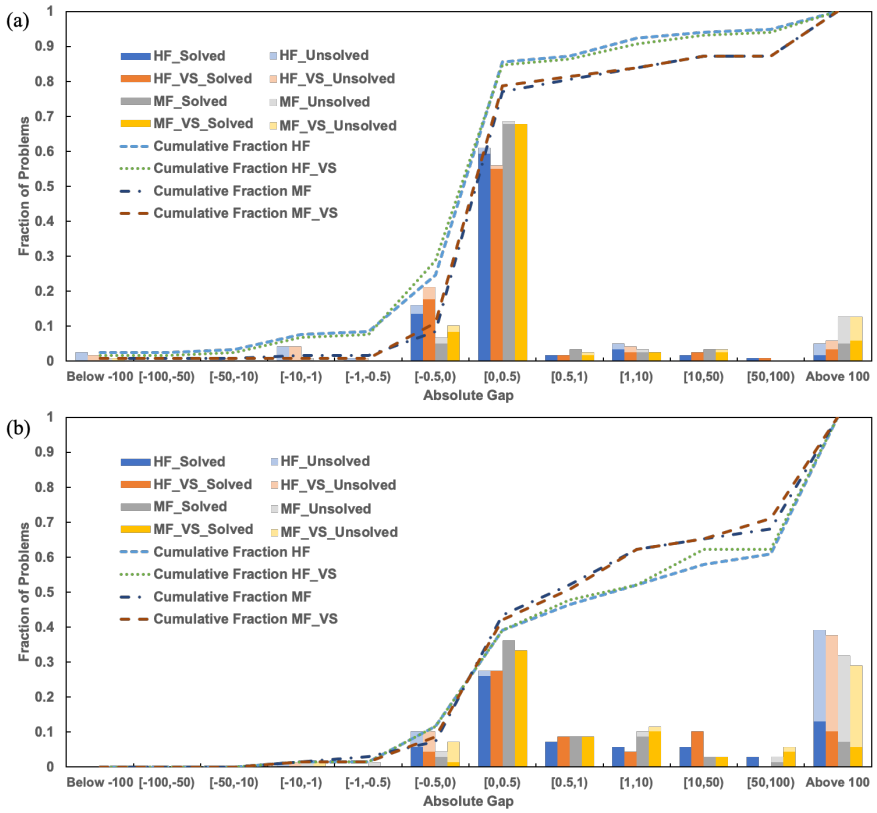

Figure 14 Distribution of absolute gaps of (a) lower dimensional group and (b) higher dimensional group at termination of HF, HF_VS, MF, and MF_VS

Discussion and Future Directions

In this paper, we present a novel data-driven spatial branch-and-bound algorithm (DDSBB) for simulation-based black-box optimization problems aiming to provide consistently global-convergent solutions. The DDSBB algorithm mimics the basic structure of traditional equation-based branch-and-bound algorithms but employs data-driven convex underestimators which are trained directly from samples and do not require any derivative information. The main purpose of this work is to introduce the new idea of the data-driven convex underestimators (DDCU) and showcase the capabilities of DDSBB algorithms

with DDCUs using data sources with different fidelities via a large set of benchmark problems. Although the data-driven convex underestimators are developed to underestimate all samples collected, we show that DDCU is able to provide a valid lower bound with respect to the true global optimum for the majority of the test problems. Among the cases that the lower bound is higher than the true global optimum, most of them are within a relatively gap of 0.5. The computational studies on the motivating example and the benchmark test problems suggest that DDSBB algorithms are capable of providing consistently globally convergent solutions to box-constrained black-box optimization problems. We also show that DDSBB algorithms with different features outperform current state-of-art solvers in certain aspects and show convergent behavior when one can afford to collect more samples.

DDSBB algorithms, aiming for globally convergent solutions, may require a large number of samples to reduce the gap between the lower bound and the upper bound. There is a trade-off between obtaining an acceptable solution with small number of samples and locating a solution that is much closer to the global optimum at higher sampling cost. Meanwhile, we have showed that DDSBB algorithms are able to locate a good solution even before they converge or terminate by reaching other stopping criteria. At the same time, the biggest advantage of DDSBB is that the lower bound information is always available so that the users can assess the potential improvement and decide whether to allocate more computational resources for a better solution. Based on the results, we have observed that data-driven convex underestimators in test problems with very steep changes are highly conservative, which leads to very wide bounds and slow convergence. To tighten the bounds and eventually reduce the sampling requirements, we are currently working on exploring different types of data-driven convex underestimators which may include bilinear terms and polynomial terms. Future improvement of the algorithm will also be devoted to reducing the computational load by parallelization and algorithmic optimization.

Other limitations of DDSBB are the scalability with increased dimensionality of the input space. In order to improve the scalability with increasing dimensionality, techniques for decoupling the input space as well as dimensionality projection methods are under exploration. Nonetheless, we believe that the

general concept of data-driven convex underestimators is a promising approach to solve simulation-based optimization problems, because it circumvents the need to optimize highly complex surrogate models directly, which is a key computational challenge of many surrogate-based optimization solvers. Based on its globally convergent behavior, we foresee this algorithm to be highly applicable in simulation-based problems where global optimum is strongly desired; for example, parameter estimation and process optimization for dynamic models.

Acknowledgements

The authors acknowledge financial support from the National Science Foundation (NSF-1805724) (JZ, FB), RAPID (FB) and Georgia Institute of Technology Startup Funding (JZ, FB).

1

References

1. Amaran, S., Sahinidis, N.V., Sharda, B., Bury, S.J.: Simulation optimization: a review of algorithms and applications. 4OR **12**(4), 301-333 (2014). doi:10.1007/s10288-014-0275-2
2. Gosavi, A.: Background. In: Gosavi, A. (ed.) Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning. pp. 1-12. Springer US, Boston, MA (2015)
3. Caballero, J.A., Grossmann, I.E.: An algorithm for the use of surrogate models in modular flowsheet optimization. AIChE Journal **54**(10), 2633-2650 (2008). doi:doi:10.1002/aic.11579
4. Henao, C.A., Maravelias, C.T.: Surrogate-based superstructure optimization framework. AIChE Journal **57**(5), 1216-1232 (2011). doi:doi:10.1002/aic.12341
5. Bhosekar, A., Ierapetritou, M.: Advances in surrogate based modeling, feasibility analysis, and optimization: A review. Computers & Chemical Engineering **108**, 250-267 (2018). doi:https://doi.org/10.1016/j.compchemeng.2017.09.017
6. McBride, K., Sundmacher, K.: Overview of Surrogate Modeling in Chemical Process Engineering. Chemie Ingenieur Technik **91**(3), 228-239 (2019). doi:10.1002/cite.201800091
7. Bajaj, I., Hasan, M.M.F.: Deterministic global derivative-free optimization of black-box problems with bounded Hessian. Optimization Letters (2019). doi:10.1007/s11590-019-01421-0
8. Kieslich, C.A., Boukouvala, F., Floudas, C.A.: Optimization of black-box problems using Smolyak grids and polynomial approximations. Journal of Global Optimization (2018). doi:10.1007/s10898-018-0643-0
9. Cozad, A., Sahinidis, N.V., Miller, D.C.: Learning surrogate models for simulation-based optimization. AIChE Journal **60**(6), 2211-2227 (2014). doi:10.1002/aic.14418
10. Schweidtmann, A.M., Mitsos, A.: Deterministic Global Optimization with Artificial Neural Networks Embedded. Journal of Optimization Theory and Applications **180**(3), 925-948 (2019). doi:10.1007/s10957-018-1396-0
11. Garud, S.S., Mariappan, N., Karimi, I.A.: Surrogate-based black-box optimisation via domain exploration and smart placement. Computers & Chemical Engineering **130**, 106567 (2019). doi:https://doi.org/10.1016/j.compchemeng.2019.106567
12. Nelder, J.A., Mead, R.: A Simplex Method for Function Minimization. The Computer Journal **7**(4), 308-313 (1965). doi:10.1093/comjnl/7.4.308
13. Torczon, V.: On the Convergence of Pattern Search Algorithms. SIAM J. on Optimization **7**(1), 1-25 (1997). doi:10.1137/s1052623493250780
14. Lewis, R.M., Torczon, V.: Pattern Search Algorithms for Bound Constrained Minimization. SIAM Journal on Optimization **9**(4), 1082-1099 (1999). doi:10.1137/s1052623496300507
15. Audet, C., J. E. Dennis, J.: Mesh Adaptive Direct Search Algorithms for Constrained Optimization. SIAM Journal on Optimization **17**(1), 188-217 (2006). doi:10.1137/040603371
16. Jones, D.R.: Direct global optimization algorithmDirect Global Optimization Algorithm. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization. pp. 725-735. Springer US, Boston, MA (2009)
17. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. Journal of Optimization Theory and Applications **79**(1), 157-181 (1993). doi:10.1007/bf00941892
18. Huyer, W., Neumaier, A.: Global Optimization by Multilevel Coordinate Search. Journal of Global Optimization **14**(4), 331-355 (1999). doi:10.1023/a:1008382309369
19. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science **220**(4598), 671-680 (1983). doi:10.1126/science.220.4598.671
20. Reeves, C.R.: Feature Article—Genetic Algorithms for the Operations Researcher. INFORMS Journal on Computing **9**(3), 231-250 (1997). doi:10.1287/ijoc.9.3.231
21. Whitley, D.: A genetic algorithm tutorial. Statistics and Computing **4**(2), 65-85 (1994). doi:10.1007/bf00175354

22. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, 27 Nov.-1 Dec. 1995 1995, pp. 1942-1948 vol.1944
23. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. Journal of Global Optimization **56**(3), 1247-1293 (2013). doi:10.1007/s10898-012-9951-y
24. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-free Optimization. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), (2009)
25. Boukouvala, F., Misener, R., Floudas, C.A.: Global optimization advances in Mixed-Integer Nonlinear Programming, MINLP, and Constrained Derivative-Free Optimization, CDFO. European Journal of Operational Research **252**(3), 701-727 (2016). doi:https://doi.org/10.1016/j.ejor.2015.12.018
26. Larson, J., Menickelly, M., Wild, S.M.: Derivative-free optimization methods. Acta Numerica **28**, 287-404 (2019). doi:10.1017/S0962492919000060
27. Audet, C., Hare, W.: The Beginnings of DFO Algorithms. In: Derivative-Free and Blackbox Optimization. pp. 33-54. Springer International Publishing, Cham (2017)
28. Cox, D.D., John, S.: A statistical method for global optimization. In: [Proceedings] 1992 IEEE International Conference on Systems, Man, and Cybernetics, 18-21 Oct. 1992 1992, pp. 1241-1246 vol.1242
29. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient Global Optimization of Expensive Black-Box Functions. Journal of Global Optimization **13**(4), 455-492 (1998). doi:10.1023/A:1008306431147
30. Huyer, W., Neumaier, A.: SNOBFIT -- Stable Noisy Optimization by Branch and Fit. ACM Trans. Math. Softw. **35**(2), 1-25 (2008). doi:10.1145/1377612.1377613
31. Thebelt, A., Kronqvist, J., Mistry, M., Lee, R.M., Sudermann-Merx, N., Misener, R.: ENTMOOT: A Framework for Optimization over Ensemble Tree Models. arXiv e-prints (2020).
32. Boukouvala, F., Floudas, C.A.: ARGONAUT: AlgoRithms for Global Optimization of coNstrAined grey-box compUTational problems. Optimization Letters **11**(5), 895-913 (2017). doi:10.1007/s11590-016-1028-2
33. Müller, J., Park, J., Sahu, R., Varadharajan, C., Arora, B., Faybishenko, B., Agarwal, D.: Surrogate optimization of deep neural networks for groundwater predictions. Journal of Global Optimization (2020). doi:10.1007/s10898-020-00912-0
34. Forrester, A.I.J., Keane, A.J.: Recent advances in surrogate-based optimization. Progress in Aerospace Sciences **45**(1), 50-79 (2009). doi:https://doi.org/10.1016/j.paerosci.2008.11.001
35. Barton, R.R., Meckesheimer, M.: Chapter 18 Metamodel-Based Simulation Optimization. In: Henderson, S.G., Nelson, B.L. (eds.) Handbooks in Operations Research and Management Science, vol. 13. pp. 535-574. Elsevier, (2006)
36. Eason, J., Cremaschi, S.: Adaptive sequential sampling for surrogate model generation with artificial neural networks. Computers & Chemical Engineering **68**, 220-232 (2014). doi:https://doi.org/10.1016/j.compchemeng.2014.05.021
37. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Mathematical Programming **103**(2), 225-249 (2005). doi:10.1007/s10107-005-0581-8
38. McKay, M.D., Beckman, R.J., Conover, W.J.: Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. Technometrics **21**(2), 239-245 (1979). doi:10.1080/00401706.1979.10489755
39. Hüllen, G., Zhai, J., Kim, S.H., Sinha, A., Realff, M.J., Boukouvala, F.: Managing Uncertainty in Data-Driven Simulation-Based Optimization. Computers & Chemical Engineering, 106519 (2019). doi:https://doi.org/10.1016/j.compchemeng.2019.106519
40. Misener, R., Floudas, C.A.: ANTIGONE: Algorithms for coNTinuous / Integer Global Optimization of Nonlinear Equations. Journal of Global Optimization **59**(2), 503-526 (2014). doi:10.1007/s10898-014-0166-2

41. Androulakis, I.P., Maranas, C.D., Floudas, C.A.: αBB: A global optimization method for general constrained nonconvex problems. Journal of Global Optimization **7**(4), 337-363 (1995). doi:10.1007/bf01099647
42. Tawarmalani, M., Sahinidis, N.V.: Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. Mathematical Programming **99**(3), 563-591 (2004). doi:10.1007/s10107-003-0467-6
43. Azar, M.G., Dyer, E.L., K, K.P., #246, rding: Convex relaxation regression: black-box optimization of smooth functions by learning their convex envelopes. Paper presented at the Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, Jersey City, New Jersey, USA,
44. Xu, W.L., Nelson, B.L.: Empirical stochastic branch-and-bound for optimization via simulation. IIE Transactions **45**(7), 685-698 (2013). doi:10.1080/0740817X.2013.768783
45. Sergeyev, Y.D., Kvasov, D.E.: A deterministic global optimization using smooth diagonal auxiliary functions. Communications in Nonlinear Science and Numerical Simulation **21**(1), 99-111 (2015). doi:https://doi.org/10.1016/j.cnsns.2014.08.026
46. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems. Mathematical Programming **10**(1), 147-175 (1976). doi:10.1007/BF01580665
47. Duran, M.A., Grossmann, I.E.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Mathematical Programming **36**(3), 307-339 (1986). doi:10.1007/BF02592064
48. Vapnik, V.: The Nature of Statistical Learning Theory. Springer New York, (1999)
49. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Statistics and Computing **14**(3), 199-222 (2004). doi:10.1023/b:stco.0000035301.49549.88
50. Oliphant, T.E.: A guid to Numpy. Trelgol Publishing USA, (2006)
51. Walt, S.v.d., Colbert, S.C., Varoquaux, G.: The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering **13**(2), 22-30 (2011). doi:10.1109/MCSE.2011.37
52. Hart, W.E., Laird, C., Watson, J.-P., Woodruff, D.L.: Pyomo - Optimization Modeling in Python. Springer Publishing Company, Incorporated, (2012)
53. Baudin, M., Christopoulou, M., Collette, Y., Martinez, J.-M.: pyDOE: The experimental design package for python. In.
54. Pedregosa, F., Ga, #235, Varoquaux, l., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., #201, Duchesnay, d.: Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. **12**, 2825-2830 (2011).
55. Zhai, J., Boukouvala, F.: Nonlinear Variable Selection Algorithms for Surrogate Modeling. AIChE Journal **0**(ja), e16601. doi:10.1002/aic.16601
56. Guzman, Y.A.: Theoretical Advances in Robust Optimization, Feature Selection, and Biomarker Discovery. Academic dissertations (Ph.D.), Princeton University (2016)
57. Bound-constrained programs. http://www.minlp.com/nlp-and-minlp-test-problems. 2019

Appendix I. List of lower dimensional problems

Table 1 List of lower dimensional problems

No.	Name	N	x_{lb}	x_{ub}	f^*
1	AluffiPentini	2	[-1.1513, -1.1]	[-0.942, 1.1]	-0.3524
2	BeckerLago	2	[4.5, 4.5]	[5.5, 5.5]	0
3	Camel3	2	[-1.1, -1.1]	[1.1, 1.1]	0
4	DekkersAarts	2	[-1.1, -16.4396]	[1.1, -13.4506]	-24776.52
5	GoldPrice	2	[-1.1, -1.1]	[1.1, 1.1]	3
6	Hartman3	3	[0.0, 0.0, 0.0]	[1.0, 1.0, 1.0]	-3.8628
7	Hosaki	2	[3.6, 1.8]	[4.4, 2.2]	-2.3458
8	MeyerRoth	3	[3.1667, 9.0, -1.1]	[3.8704, 10.0, 1.1]	0.0019
9	ModRosenbrock	2	[-1.1, -1.1]	[1.1, 1.1]	0
10	MultiGauss	2	[-1.1, -1.1]	[1.1, 1.1]	-1.297
11	box3	3	[-1.1, -1.1, -1.1]	[1.1, 1.1, 1.1]	0
12	brownbs	2	[900000.0, -1.1]	[1100000.0, 1.1]	0
13	camel1	2	[-3.0, -2.0]	[3.0, 2.0]	-1.0316
14	cliff	2	[2.7, 2.8348]	[3.3, 3.4648]	0.1998
15	concha1	2	[-4443.6025, 596.1299]	[-3635.6748, 728.6033]	4.254
16	concha10	2	[-1.1, -1.0]	[1.1, 1.0]	9.7258
17	concha11	2	[9000.0, -10.0]	[10000.0, -9.0]	5.8026
18	concha12	2	[-4143.4454, 2928.5789]	[-3390.0917, 3579.3742]	6.1635
19	concha3	2	[3859.0955, -6994.0971]	[4716.6723, -5722.4431]	7.3971
20	concha5	3	[3262.6655, -1695.5224, 93.1098]	[3987.7022, -1387.2456, 113.8009]	1.4885
21	concha5a	3	[-100.0, -100.0, 29.7399]	[-90.0, -90.0, 36.3487]	2.9675
22	concha8	2	[9000.0, 0.0]	[10000.0, 1.1]	5.7115
23	concha9	2	[4861.7199, -3.1428]	[5942.1022, -2.5714]	7.7031
24	cube	2	[0.9, 0.9]	[1.1, 1.1]	0
25	denschna	2	[-1.1, -1.1]	[1.1, 1.1]	0
26	denschnb	2	[1.8, -1.1]	[2.2, 1.1]	0
27	denschnc	2	[-1.1, 0.9]	[1.1, 1.1]	0
28	denschnd	3	[-1.1, -1.1, -1.1]	[1.1, 1.1, 1.1]	0
29	denschne	3	[-1.1, -1.1, -1.1]	[1.1, 1.1, 1.1]	0
30	draper1	2	[0.0, 0.0]	[1.0, 1.0]	0.005
31	draperg	3	[103.6322, 2.0796, -24.2317]	[126.6616, 2.5417, -19.8259]	7.0133

32	draperj	3	[3.2122, 11.5163, 0.0]	[3.926, 14.0755, 1.1]	0.0079
33	drapero	2	[0.0, 1.6902]	[1.1, 2.0658]	2.864
34	engval2	3	[-1.1, -1.1, -1.1]	[1.1, 1.1, 1.1]	0
35	ex005	2	[-1.0, 1.8]	[1.1, 2.0]	-4
36	ex4_1_5	2	[-1.1, -1.1]	[1.1, 1.1]	0
37	ex8_1_3	2	[-1.1, -1.1]	[1.1, 1.1]	3
38	ex8_1_4	2	[-1.1, -1.1]	[1.1, 1.1]	0
39	ex8_1_5	2	[-1.1, -1.1]	[1.1, 1.1]	-1.0316
40	ex8_1_6	2	[3.6, 3.6]	[4.3999, 4.3999]	-10.086
41	fermat2_eps	2	[1.8, -1.1]	[2.2, 1.1]	4.4722
42	fermat2_vareps	3	[1.8, -1.1, 0.0]	[2.2, 1.1, 1.1]	4.4721
43	fermat_eps	2	[1.8, 1.0392]	[2.2, 1.2702]	7.4641
44	fermat_vareps	3	[1.8, 1.0392, 0.0]	[2.2, 1.2702, 1.1]	7.4641
45	gold	2	[-1.1, -1.1]	[1.1, 1.1]	3
46	hatfldd	3	[2.8795, -1.1196, -1.1]	[3.5194, -0.916, 1.1]	0
47	hatflde	3	[2.8776, -1.1103, -1.1]	[3.5171, -0.9084, 1.1]	0
48	himmelbb	2	[-1.1, -1.1]	[1.1, 1.1]	0
49	himmelbg	2	[-1.1, -1.1]	[1.1, 1.1]	0
50	himmelbh	2	[-1.1, -1.1]	[1.1, 1.1]	-1
51	himmelp1	2	[73.0725, 62.2421]	[89.3108, 75.0]	-62.0539
52	hs001	2	[0.9, 0.9]	[1.1, 1.1]	0
53	hs002	2	[1.1019, 1.5]	[1.3468, 1.65]	0.0504
54	hs003	2	[-1.1, 0.0]	[1.1, 1.1]	0
55	hs004	2	[1.0, 0.0]	[1.1, 1.1]	2.6667
56	hs3mod	2	[-1.1, 0.0]	[1.1, 1.1]	0
57	jensmp	2	[-1.1, -1.1]	[1.1, 1.1]	1.2436
58	logros	2	[0.0, 0.0]	[1.1, 1.1]	0
59	maratosb	2	[-1.1, -1.1]	[-0.9, 1.1]	-1
60	median_vareps	2	[0.0, -1.1]	[1.1, 1.1]	4.9424
61	mexhat	2	[1.0276, 1.1733]	[1.256, 1.434]	-0.0401
62	model19	2	[215.0479, 0.0]	[262.8363, 1.1]	0.1246
63	model2	2	[192.4285, -1.0]	[235.1903, 1.1]	1.168
64	model23	3	[2.388, 0.0, -0.1]	[2.9187, 1.1, 0.0]	5.9731
65	model24	3	[65.216, 2.3563, 0.0]	[79.7085, 2.8799, 1.1]	8.0565
66	model3	3	[0.001, 0.0, 0.0]	[1.0, 0.01, 0.1]	2.3845
67	model31	3	[2.8184, 13.6434, 0.0]	[3.4447, 16.6753, 1.1]	0
68	model32	3	[11.9168, 1.3507, 18.09]	[14.565, 1.6508, 22.1099]	0.0001
69	model33	3	[58.4694, 1.3569, 17.9283]	[71.4626, 1.6584, 21.9124]	1.2519

70	model36	2	[-1.1, -1.1]	[1.1, 1.1]	1.2436
71	model39	2	[-1.1, -1.1]	[1.1, 1.1]	0.0089
72	model4	3	[14.1058, 0.0, 0.0]	[17.2404, 1.1, 1.0]	0.006
73	model42	3	[0.0, 4.5, 1.1196]	[1.1, 5.0, 1.3684]	2.0397
74	model45	3	[13.9498, 1.0802, 0.0]	[17.0498, 1.3202, 1.1]	0
75	model5	3	[9.0, 990.0, 47.8124]	[10.0, 1100.0, 58.4374]	5.8496
76	nasty	2	[-1.1, -1.1]	[1.1, 1.1]	0
77	price	2	[-1.1, -1.1]	[1.1, 1.1]	0
78	ratkasymptotic	2	[0.0, 0.0]	[1.0, 1.0]	6.3502
79	ratkbates	3	[0.0, 90000.0, 0.0]	[1.1, 100000.0, 1.1]	3.3199
80	rosenbr	2	[0.9, 0.9]	[1.1, 1.1]	0
81	s201	2	[4.5, 5.4]	[5.5, 6.6]	0
82	s202	2	[4.5, 3.6]	[5.5, 4.4]	0
83	s204	2	[-1.1, -1.1]	[1.1, 1.1]	0.0485
84	s205	2	[2.7, -1.1]	[3.3, 1.1]	0
85	s206	2	[0.9, 0.9]	[1.1, 1.1]	0
86	s207	2	[-1.1, -1.1]	[1.1, 1.1]	0
87	s208	2	[0.9, 0.9]	[1.1, 1.1]	0
88	s209	2	[-1.1, -1.1]	[1.1, 1.1]	0
89	s210	2	[-1.1, -1.1]	[1.1, 1.1]	0
90	s211	2	[0.9, 0.9]	[1.1, 1.1]	0
91	s212	2	[-1.1, -1.1]	[1.1, 1.1]	0
92	s214	2	[-1.1, -1.1]	[1.1, 1.1]	0
93	s229	2	[0.9, 0.9]	[1.1, 1.1]	0
94	s240	3	[-1.1, -1.1, -1.1]	[1.1, 1.1, 1.1]	0
95	s242	3	[0.0, 0.0, 0.0]	[1.1, 1.1, 1.1]	0
96	s244	3	[0.0, 9.0, 4.5]	[1.1, 10.0, 5.5]	0
97	s245	3	[-1.1, -1.1, -1.1]	[1.1, 1.1, 1.1]	0
98	s246	3	[-1.1, -1.1, -1.1]	[1.1, 1.1, 1.1]	0
99	s274	2	[-1.1, -1.1]	[1.1, 1.1]	0
100	s290	2	[-1.1, -1.1]	[1.1, 1.1]	0
101	s307	2	[0.0, 0.0]	[1.1, 1.1]	1.2436
102	s309	2	[3.1344, 3.51]	[3.831, 4.29]	-3.9872
103	s311	2	[-4.1572, -3.6115]	[-3.4014, -2.9549]	0
104	s312	2	[18.4114, -38.2704]	[22.5029, -31.3122]	0
105	s328	2	[1.5691, 1.8267]	[1.9178, 2.2327]	1.7442
106	s332	2	[0.0, 0.0]	[1.1, 1.1]	2.7191
107	s333	3	[80.9118, -1.1, -1.1]	[98.8922, 1.1, 1.1]	0.0433
108	s386	2	[4.5, 5.4]	[5.5, 6.6]	0
109	sim2bqp	2	[-1.1, 0.0]	[1.1, 0.5]	0

110	simbqp	2	[-1.1, 0.0]	[1.1, 0.5]	0
111	sisser	2	[-1.1, -1.1]	[1.1, 1.1]	0
112	st_cqpjk2	3	[0.0, 0.0, 0.0]	[1.0, 1.0, 1.0]	-12.5
113	st_e39	2	[3.6, 3.6]	[4.3999, 4.3999]	-10.086
114	stattools	2	[0.9909, 0.0]	[1.2111, 1.1]	0.0418
115	tranter2	3	[0.0, -1.0, 0.0]	[0.1, 0.0, 0.1]	0.0003
116	tre	2	[-1.1, -1.1]	[1.1, 1.1]	0
117	zangwil2	2	[3.6, 8.1]	[4.4, 9.9]	-18.2
118	zhenglog	3	[65.216, 2.3563, 0.0]	[79.7085, 2.8799, 1.0]	8.0565

Appendix II. List of high dimensional problems

Table 2 List of higher dimenional problems

No.	Name	N			
1	Expo	10	[-1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0]	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	-1
2	Neumaier2	4	[0.0, 1.7999, 1.8001, 2.7]	[1.0, 2.0, 2.2002, 3.3]	0
3	Neumaier3	10	[9.0, 16.2, 21.6, 25.2, 27.0, 27.0, 25.2, 21.6, 16.2, 9.0]	[11.0, 19.8, 26.4, 30.8, 33.0, 33.0, 30.8, 26.4, 19.8, 11.0]	-210
4	Paviani	10	[8.4152, 8.4152, 8.4152, 8.4152, 8.4152, 8.4152, 8.4152, 8.4152, 8.4152, 8.4152]	[10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0]	-45.7785
5	PowellQ	4	[-1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1]	0
6	PriceTransistor	9	[0.0, 0.0, 0.9, 1.8001, 7.2, 7.1997, 4.5, 0.0, 1.8]	[1.1, 1.1, 1.1, 2.2001, 8.8, 8.7997, 5.5, 1.1, 2.2001]	0
7	Shekel10	4	[3.6007, 3.6005, 3.5997, 3.5996]	[4.4008, 4.4007, 4.3996, 4.3995]	-10.5364
8	Shekel7	4	[3.6005, 3.6006, 3.5995, 3.5996]	[4.4006, 4.4008, 4.3994, 4.3996]	-10.4029
9	Shekelfox10	10	[7.2225, 8.2367, 4.6026, 6.8589, 4.1076, 4.2399, 2.6964, 5.5134, 0.0, 4.4838]	[8.8275, 10.0, 5.6254, 8.3831, 5.0204, 5.1821, 3.2956, 6.7386, 1.1, 5.4802]	-10.2088
10	Shekelfox5	5	[7.2224, 8.2366, 4.6025, 6.8588, 4.1077]	[8.8274, 10.0, 5.6253, 8.3829, 5.0205]	-10.4056
11	Wood	4	[-1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1]	0
12	biggs6	6	[1575174.0197, 15.9149, -110000000.0, 4.6679, 1.5403, 2.0937]	[1925212.6907, 19.4515, -90000000.0, 5.7052, 1.8826, 2.5589]	0.0057
13	brownal	10	[-1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, 0.9]	[1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1]	0

14	brownden	4	[-12.7539, 11.8833, -1.1, -1.1]	[-10.435, 14.524, 1.1, 1.1]	8.5822
15	dixon3dq	10	[-1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1]	0
16	extrosnb	10	[-1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1]	0
17	hart6	6	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	-3.3229
18	hatflda	4	[0.0, 0.0, 0.0, 0.0]	[1.1, 1.1, 1.1, 1.1]	0
19	hatfldb	4	[0.0, 0.0, 0.0, 0.0]	[1.1, 0.8, 1.1, 1.1]	0.0056
20	hatfldc	4	[0.9, 0.0, 0.0, -1.1]	[1.1, 1.1, 1.1, 1.1]	0
21	heart8ls	8	[-1.1, -1.1, -1.1, -1.1, 2.2449, -1.4104, -1.5231, 1.3994]	[1.1, 1.1, 1.1, 1.1, 2.7437, -1.154, -1.2462, 1.7104]	0
22	hilberta	10	[-1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1]	0
23	hs038	4	[-1.1, -1.1, 0.9, 0.9]	[1.1, 1.1, 1.1, 1.1]	0
24	hs045	5	[0.0, 1.8, 2.7, 3.6, 4.5]	[1.0, 2.0, 3.0, 4.0, 5.0]	1
25	hs110	10	[8.4152, 8.4152, 8.4152, 8.4152, 8.4152, 8.4152, 8.4152, 8.4152, 8.4152, 8.4152]	[9.999, 9.999, 9.999, 9.999, 9.999, 9.999, 9.999, 9.999, 9.999, 9.999]	-45.7785
26	kowalik	4	[0.0, 0.0, 0.0, 0.0]	[0.42, 0.42, 0.42, 0.42]	0.0003
27	model13	6	[0.0, 0.0, 0.0, 2.7, 1.4018, 4.5]	[1.0, 1.1, 1.0, 3.3, 1.7134, 5.5]	0
28	model14	6	[0.0, 2.7071, 1.3976, 4.5026, 0.0, 0.9052]	[1.1, 3.3086, 1.7082, 5.5032, 1.1, 1.1063]	0
29	model15	6	[0.0, 0.0, 0.0, 2.6564, 1.4243, 4.4877]	[1.0, 1.0, 1.0, 3.2467, 1.7408, 5.485]	0
30	model16	4	[0.0, 0.0, 0.0, 0.0]	[1.1, 1.1, 1.1, 1.1]	0.0003
31	model18	5	[0.0, 1.7423, -1.6112, 0.0, 0.0]	[1.0, 2.1294, -1.3182, 0.05, 0.05]	0.0001
32	oslbqp	8	[2.5, 0.0, 0.0, 0.0, 0.5, 0.0, 0.0, 0.0]	[2.75, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1]	6.25
33	palmer1c	8	[79.4675, -180.8224, 131.4538, -113.7649, 50.3336, -19.6229, 2.738, -1.1]	[97.1269, -147.9456, 160.6658, -93.0804, 61.5188, -16.0551, 3.3465, 1.1]	0.0976
34	palmer1d	7	[79.3008, -175.177, 110.7352, -71.2179, 22.361, -5.4716, -1.1]	[96.9232, -143.3266, 135.343, -58.2692, 27.3301, -4.4768, 1.1]	0.6527
35	palmer2c	8	[17.4504, -61.5965, 59.6872, -61.8856, 36.9104, -19.8414, 3.8019, -1.1]	[21.3282, -50.3971, 72.951, -50.6336, 45.1127, -16.2339, 4.6468, 1.1]	0.0144

36	palmer3c	8	[8.8357, -46.723, 61.6137, -69.2331, 42.0604, -23.1822, 4.5807, -1.1]	[10.7992, -38.2279, 75.3057, -56.6453, 51.4071, -18.9673, 5.5986, 1.1]	0.0195
37	palmer4c	8	[8.716, -52.2234, 80.4934, -105.4979, 67.6167, -38.0683, 7.5732, -1.1]	[10.6529, -42.7283, 98.3808, -86.3165, 82.6427, -31.1468, 9.2562, 1.1]	0.0503
38	palmer5c	6	[33.7833, -1.9033, 36.7144, -1.1, 3.3381, -1.1]	[41.2907, -1.5572, 44.8732, 1.1, 4.0798, 1.1]	2.1281
39	palmer5d	4	[72.2262, -145.3165, 46.4761, -1.1]	[88.2764, -118.8954, 56.8041, 1.1]	8.7339
40	palmer6c	8	[9.5573, -58.0874, 98.4797, -165.0954, 140.3039, -102.3164, 25.7913, -3.8617]	[11.6811, -47.5261, 120.364, -135.0781, 171.4826, -83.7134, 31.5227, -3.1596]	0.0164
41	palmer7c	8	[4.0889, -58.043, 218.1999, -617.6092, 646.5388, -514.1436, 132.1612, -19.4072]	[4.9975, -47.4897, 266.6888, -505.3166, 790.2141, -420.6629, 161.5304, -15.8786]	0.602
42	palmer8c	8	[4.1178, -50.6413, 139.4599, -261.0124, 208.3661, -138.9187, 32.0861, -4.442]	[5.0329, -41.4338, 170.4509, -213.5556, 254.6697, -113.6607, 39.2164, -3.6344]	0.1598
43	powell	4	[-1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1]	0
44	pspdoc	4	[-1.1, -1.1, -1.1, -1.1]	[-1.0, 1.1, 1.1, 1.1]	2.4142
45	s256	4	[-1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1]	0
46	s257	4	[0.0, -1.1, 0.0, -1.1]	[1.1, 1.1, 1.1, 1.1]	0
47	s258	4	[-1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1]	0
48	s259	4	[1.2923, 1.8568, -1.1, -1.1]	[1.5794, 2.2695, 1.1, 1.0]	-8.5446
49	s260	4	[-1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1]	0
50	s266	5	[-1.1, -1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1, 1.1]	1
51	s271	6	[-1.1, -1.1, -1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1, 1.1, 1.1]	0
52	s272	6	[0.0, 9.0, 3.6, 0.0, 4.5, 2.7]	[1.1, 11.0, 4.4, 1.1, 5.5, 3.3]	0
53	s273	6	[-1.1, -1.1, -1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1, 1.1, 1.1]	0
54	s275	4	[-1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1]	0
55	s276	6	[-1.1, -1.1, -1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1, -0.8, 1.1]	0
56	s281	10	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	[1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1]	0

57	s282	10	[-1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1]	0
58	s291	10	[-1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1]	[1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1]	0
59	s294	6	[-1.1, -1.1, -1.1, -1.1, -1.1, -1.1]	[-0.6, 1.1, 1.1, 1.1, 1.1, 1.1]	3.9739
60	s295	10	[-1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1, -1.1]	[-0.6, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1]	3.9866
61	s352	4	[-11.2459, 10.7176, -1.1, -1.1]	[-9.2012, 13.0993, 1.1, 1.1]	9.0323
62	s358	5	[-0.5, 1.7423, -1.6112, 0.001, 0.001]	[0.5, 2.1294, -1.3182, 0.1, 0.1]	0.0001
63	s370	6	[-1.1, 0.9112, -1.1, 1.1344, -1.6651, -1.1]	[1.1, 1.1137, 1.1, 1.3865, -1.3624, 1.1]	0.0023
64	s371	9	[-1.1, -1.1, -1.1, -1.1, 0.9007, -2.8795, 3.694, -3.458, 0.9474]	[1.1, 1.1, 1.1, 1.1, 1.1009, -2.356, 4.5148, -2.8293, 1.1579]	0
65	schwefel	5	[-0.5, -0.5, -0.5, -0.5, -0.5]	[0.4, 0.4, 0.4, 0.4, 0.4]	0
66	shekel	4	[3.6, 3.6001, 3.6, 3.6001]	[4.4, 4.4001, 4.4, 4.4001]	-10.1532
67	st_bsj3	6	[89.1, 89.1, 89.1, 89.1, 89.1, 89.1]	[99.0, 99.0, 99.0, 99.0, 99.0, 99.0]	-86768.55
68	tranter	6	[-1.0, 0.0, -1.0, 0.0, -1.1, -1.1]	[0.0, 1.0, 0.0, 1.1, 0.0, 0.0]	0.0045
69	weibull3	4	[5.9903, 4.9944, 0.0, 1.5865]	[7.3215, 6.1042, 1.0, 1.939]	0.1338

1

1