Asia-Pacific Journal of Operational Research
Vol. 36, No. 6 (2019) 1940011 (35 pages)
© World Scientific Publishing Co. & Operational Research Society of Singapore
DOI: 10.1142/S0217595919400116

Generalized Ordinal Learning Framework (GOLF) for Decision Making with Future Simulated Data

Giulia Pedrielli*
School of Computing Informatics
and Decision Systems Engineering Arizona State University
699 S Mill Ave Tempe, Arizona 85251, USA
gpedriel@asu.edu

K. Selcuk Candan
School of Computing Informatics
and Decision Systems Engineering Arizona State University
699 S Mill Ave Tempe, Arizona 85251, USA
candan@asu.edu

Xilun Chen
School of Computing Informatics
and Decision Systems Engineering Arizona State University
699 S Mill Ave Tempe, Arizona 85251, USA
cupidcxl@gmail.com

Logan Mathesen
School of Computing Informatics
and Decision Systems Engineering Arizona State University
699 S Mill Ave Tempe, Arizona 85251, USA
lmathese@asu.edu

Alireza Inanalouganji
School of Computing Informatics
and Decision Systems Engineering Arizona State University
699 S Mill Ave Tempe, Arizona 85251, USA
ainanlou@asu.edu

Jie Xu
Systems Engineering and Operations Research Department
George Mason University 4400 University Drive Fairfax
Virginia 22030, USA
jxu13@gmu.edu

*Corresponding author.

Chun-Hung Chen
Systems Engineering and Operations Research Department
George Mason University 4400 University Drive Fairfax
Virginia 22030, USA
cchen9@gmu.edu

Loo Hay Lee
Department of Industrial Systems Engineering & Management
National University of Singapore 1 Engineering
Drive 2 Singapore 117576, Singapore
iseleelh@nus.edu.sg

Received 16 August 2018
Revised 8 April 2019
Accepted 5 July 2019
Published 15 November 2019

Real-time decision making has acquired increasing interest as a means to efficiently operating complex systems. The main challenge in achieving real-time decision making is to understand how to develop next generation optimization procedures that can work efficiently using: (i) real data coming from a large complex dynamical system, (ii) simulation models available that reproduce the system dynamics. While this paper focuses on a different problem with respect to the literature in RL, the methods proposed in this paper can be used as a support in a sequential setting as well. The result of this work is the new Generalized Ordinal Learning Framework (GOLF) that utilizes simulated data interpreting them as low accuracy information to be intelligently collected offline and utilized online once the scenario is revealed to the user. GOLF supports real-time decision making on complex dynamical systems once a specific scenario is realized. We show preliminary results of the proposed techniques that motivate the authors in further pursuing the presented ideas.

Keywords: Simulation optimization; multi-fidelity; ordinal learning; stochastic optimization; real-time decision making.

1. Introduction and Motivation

Simulation has been used to support design and optimization of industrial, transportation, water distribution systems among many others. More recently, we have witnessed an increased interest in the use of simulation for the real time operation of large complex systems. Under the perspective of Cyber-Physical Systems (CPS), it has become increasingly apparent that data from the real physical system should be *coupled* with data from the simulated system, to improve decision making. The problem of using real system data to inform the decision process is one of the foci of the machine learning (ML) literature that has undergone an unprecedented development. Simulation models of the environment represent a very important component for most CPSs, where *real* historical data are not enough to support control. In fact, decision making in many critical applications is very challenging due to the *inherent sparsity of available data*, even when accounting for the increasing pervasiveness of sensing technology. There are several causes for this sparsity: sensing large portions of a complex system is often not cost effective (or impossible) for many applications, and making decisions requires information about not only what has happened in the past and what is happening in the present, but also about the future events (possible scenarios), which are inherently unknown and cannot be sensed. In this setting, simulations, which are also based on past data and observations, offer the opportunity to see what sensors cannot capture, including what may happen in the future. We refer to this specific class of simulation generated "data" as look-back data (LBD, including past and present simulated data) and look-ahead data (LAD, future simulated data) and differentiate these from the observed data (OD), which are obtained from sensors and, potentially, have high accuracy, but are inherently sparse. It is important to note that LBD and LAD are potentially expensive to obtain, voluminous, inherently imprecise and dependent upon analysts' goals (Fig. 1).

ML and, in particular, Reinforcement Learning (RL) represents the de-facto standard for sequential decision making where a model of the environment is known (but a closed-form solution to the model is not), a simulator of the environment is available, and data can be gathered from the environment upon actions (Sutton and Barto, 2018). In the context of simulation based optimization, RL literature mainly focuses on long horizon problems, where the *agent*, in contact with the simulated *environment*, acts in order to *maximize a long term expected reward*. While the response of the environment to the action can be evaluated by means of a simulation model, typically the long term reward is approximated using Monte Carlo experiments (Bertsekas, 2008; Powell, 2007; Si *et al.*, 2004). RL can be seen as a form of simulation-based dynamic programming, used to solve Markov and semi-Markov decision problems (Sutton and Barto, 2018). When solving a control-optimization problem, RL can help avoid enumeration of complete transition probability matrices and can help obtain compact representations of the underlying dynamic programming matrices. It is important to note that, in this context, (i) there is a real

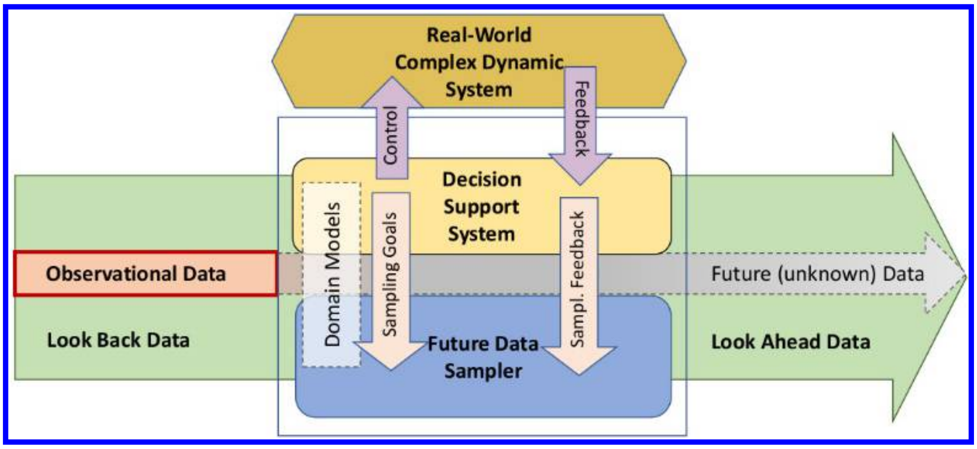


Fig. 1. GOLF framework.

environment in which the agent acts or plans to act; (ii) each action in the real world provides new data; (iii) there is a simulation model to support simulated trials to associate reward values to potential actions in the real world, and (iv) the goal is to decide which actions to take in the real environment to maximize the reward function.

In contrast, in this work, we aim to address a different problem: (i) once again, there is a real environment in which the agent acts or plans to act; (ii) but in this case, there is an *underspecified* simulation model to support simulated trials (e.g., not a Markov/semi-Markov process); (iii) the data from the real-environment arrives potentially independently from the actions of the agent; and (iv) the goal is to decide which simulations to run to help improve the underspecified model. In fact, we aim to address the fundamental challenge of learning about complex dynamical systems by adopting large volume, *potentially low accuracy* simulated data derived from (time-varying) simulation models along with real-world data (past observations). We achieve the objective by proposing the Generalized Ordinal Learning Framework (GOLF) for decision making with simulated data.

Note that the problem we investigate in this paper involves taking the best decision in a *specific* scenario by intelligently using both real (past observations) and simulated data. This is equivalent to sequentially performing one-time optimizations, where the objective is to get as close as possible to the maximum reward for each single scenario. The need to take a decision each time a new scenario is presented makes our optimization structure dynamic, allowing models to change over time. An RL algorithm would instead keep applying a specific stationary policy, upon completion of the learning phase. Nonetheless, while we deal with a different problem, the techniques proposed in this manuscript well fit the policy search research challenge for RL as well as *transfer learning* (Yamada *et al.*, 2018).

Summarizing, Fig. 1 represents the proposed framework where LBDs, ODs, and LADs are all interpreted as low accuracy (offline) information that can be adopted in order to make educated decisions when the future (online) information becomes available. GOLF not only provides a rigorous way to handle these numerous sources of information, but also provides novel methods to guide the process for generating data relevant to the decision process. Sections 1.1 and 1.2 revisit the literature relevant to GOLF highlighting the main critical aspects and provide a detailed description of the contributions of the present work to the literature, respectively.

1.1. *Background*

The problem of taking decisions based upon heterogeneous sources of data with no closed form formulation of the system dynamics, can be brought back to the large research field of black box optimization as well as the issue of exploring policies in RL (Sutton and Barto, 2018). To optimize a black-box function, direct search as well as metamodel-based approaches have been proposed in the literature (Fu, 2015). Direct search methods select and evaluate candidate locations within the solution space according to a sampling algorithm, and update/adapt the sampling procedure based upon the information from the search process. The presence of adaptation and the structure of the sampling rule have led to the development of a plethora of approaches. Examples include hit-and-run (Zabinsky and Smith, 1992; Zabinsky *et al.*, 1993; Solis and Wets, 1981; Brooks, 1958), GRASP (Feo and Resende, 1995) a greedy adaptive random search sampling technique, and the stochastic ruler method (Yan and Mukai, 1992; Alrefaei and Andradóttir, 2001).

Metamodel-based approaches differ from direct search methods in that they consider all sampled points to make a sampling decision by constructing a response surface model that emulates the function to optimize in locations where evaluation has not yet been performed. In this rich research area, efforts have been dedicated to proposing novel statistical models to improve the accuracy of the prediction, as well as to the study of effective sampling criteria that are able to handle the exploration/exploitation dilemma (Li *et al.*, 2010). Concerning the model choice, applications of ANN, kriging models, polynomial regression models and Radial Basis Functions (RBFs) have been proven successful in different application settings (Chen *et al.*, 2015).

In this area, Gaussian Processes (GP), which will be used in this work, have received an important attention (Quan *et al.*, 2013; Ankenman *et al.*, 2010; Yin *et al.*, 2011). Nevertheless, most of these models are designed to handle a single source of data, whereas, as we mentioned, GOLF relies on at least three sources of data: LBD, LAD, and OD. We can regard this problem as Multi-Fidelity simulation optimization a rising field in simulation optimization, which focuses on the use of lower precision models to support expensive simulation optimization algorithms. In this area, Wang *et al.* (2013), propose Multi-Fidelity Optimization with Ordinal Transformation and Optimal Sampling (MO^{2}TOS) that relies on the concept of Ordinal Transformation (OT). OT is a mapping $\mathcal{X} \rightarrow \mathcal{H}$, where $\mathcal{X}$ is d-dimensional discrete space and $\mathcal{H}$ is a one-dimensional rank space constructed by associating to each point of $\mathcal{X}$, the rank computed according to the solution value returned by the low-precision (fidelity) model (this could be the LAD, LBD related to a specific solution with no associated OD). This mapping, as defined by the authors, can be applied to any finite, countable space $\mathcal{X}$. Once the mapped space is computed, the solutions are grouped in sub-sets with respect to $\mathcal{H}$ and sampled according to the Optimal Sampling (OS) scheme. The authors theoretically prove how the use of low-fidelity models (offline data) can lead to improved performance with respect to procedures not using any low-fidelity information. Hsieh *et al.* (2017) further extends the previous contribution by using OT not only to transform the solution space but also to decide on high-fidelity computational budget allocation. Hsieh *et al.* (2016) and Zhang *et al.* (2016a), extend the application of this approach, and Xu *et al.* (2014), proposes an innovative OS methodology that maximizes the estimated probability of selecting the best solution. All the aforementioned approaches handle one low-fidelity model. In the direction of introducing more fidelities, Xu *et al.* (2016a) proposes a novel scheme to weight predictions generated by multiple low fidelities creating another equivalent low-fidelity model. More recently, Min (2017) uses Gaussian Process Regression along with OT to optimize noisy observations of a black-box function. While the authors still focus on discrete optimization, an approach to handle an arbitrary number of low-fidelity models is presented. In the area of continuous optimization, Santner *et al.* (2013) represents the main reference for the modeling of information from several fidelities and discusses co-kriging as a possible framework to capture relationships between several models when we can "rank" models in terms of fidelity. Forrester *et al.* (2007), develops a co-kriging based method where an algorithm is proposed that chooses different sampling points at the different fidelity levels. The authors consider a wing optimization problem as a case study to validate and show the applicability of the proposed method. More recently, Ulaganathan *et al.* (2015), extend the co-kriging formulation with multi-fidelity gradient information. The authors numerically show how the extra information from the multi-fidelity gradient can help the kriging model achieve better prediction accuracy. Liu *et al.* (2016), propose to use a multi-fidelity Gaussian Process with Memetic Differential Evolution, while Chen *et al.* (2015) decompose the high-fidelity response into trend and residual components and use a non-parametric locally weighted regression with smoothing. Osorio and Selvam (2017) propose a method that combines information from evaluating models with different fidelities to optimally control traffic networks. At each iteration, the algorithm decides on the model to evaluate based on an estimate of accuracy loss due to running the low-fidelity model.

In summary, OT and co-kriging-based methods are the two families of approaches more related to GOLF that tackle multi-fidelity simulation-optimization problems. OT is powerful in that it allows to reduce the dimensionality of the solution space and it reduces sampling to selecting the solution with the best associated rank. While the concept has been proven very successful in the literature, it needs to be further extended to cope with specific challenges such as the ability to handle continuous input variables. This is particularly important not only when we have continuous problems, but also when the number of solutions is particularly large and the computational cost of the low-fidelity model, while low, is not negligible. While co-kriging-based approaches can solve this issue, most of the algorithms require to know the rank of the models in terms of fidelity and several authors highlight the computational burden associated with the estimation of the co-kriging model hyper-parameters (Santner *et al.*, 2013).

1.2. *Contribution*

In this paper, we propose a data driven two-stage (offline and online) decision support system, addressing a fundamental gap in the ability to leverage real and simulated data for effective real-time decision making. We use a new source of information (*look-back data*, *LBD*, *and look-ahead data*, *LAD*) and we propose innovative methods to generate, learn, and integrate those to provide *reliable* recommendations. We highlight three major contributions resulting from this work:

- *Contribution* 1: *Measuring relevance of simulation instances in a given decision making context.* Instead of fully relying on traditional fit and error measures, which fail to capture the application context, we propose novel criteria that dynamically and adaptively maximize the "explicability", capture "complexity", and/or encourage "diversity" for simulation samples.
- *Contribution* 2: *Budgeted and incremental LAD and LBD learning and sampling.* Grounded in these novel metrics, we develop an innovative family of budgeted simulation sampling algorithms that help construct models iteratively learned from past observations, while providing insights into possible future scenarios. These new learning and sampling methodologies face fundamental challenges given by non-smooth functions and high-dimensional spaces.
- *Contribution* 3: *LAD and LBD driven real-time decision making.* We formulate the decision problem accounting for inaccuracy of simulation ensembles in a way that enhances the algorithmic speed of decision making with voluminous LAD and LBD, appropriate to the required fidelity that needs to be provided to the decision maker. Ordinal Learning will be the key method to allow the effective use of offline generated data and models and integrate them iteratively with newly generated information once the scenario of interest becomes available to the decision maker.

2. Data and Simulation Based Decision Making and Control

In this work, we look at the specific problem of optimizing/controlling a large complex dynamical system when multiple sources of data, including LBD and LAD in addition to real OD, are used as estimates of the system performance. Formally, the problem is to identify the decision $\boldsymbol{x}$ satisfying:

$$\boldsymbol{x} = \underset{\boldsymbol{x}\in\mathbb{X}}{\operatorname{argmax}}\, \mathcal{J}(\boldsymbol{x}, \boldsymbol{Y} \mid \boldsymbol{Y} = \boldsymbol{y}). \tag{1}$$

Since $\mathcal{J}$ is a performance resulting from the operations of a large complex system, it is not known in closed form and, instead, data (either real or simulated for past or future scenarios) need to be adopted to as evaluations. $\boldsymbol{x}$ represents the decision, while $\boldsymbol{Y}$ refers to the system parameters that are *not* controllable by the user (e.g., demand level for a product, location of the demand). The problem in Eq. (1) is not solvable in closed form, nevertheless, data are available and can be generated that provide *useful* information towards the goal of solving the problem in Eq. (1), such as

- The optimal solution observed/computed under a different scenario, $\boldsymbol{x}^*(\boldsymbol{y}')$.
- The function value observed/simulated for a suboptimal solution for the same scenario $\boldsymbol{Y} = \boldsymbol{y}$, i.e., $\mathcal{J}(\boldsymbol{x}, \boldsymbol{Y} \mid \boldsymbol{Y} = \boldsymbol{y})$.

- A function value observed/simulated for a suboptimal solution in a different scenario $\boldsymbol{Y} = \boldsymbol{y}'$, i.e., $\mathfrak{J}(\boldsymbol{x}, \boldsymbol{Y} \mid \boldsymbol{Y} = \boldsymbol{y}')$.

This generalizes the learning typically implemented in RL, where a decision is made, hopefully, based on solutions/decisions in the past (i.e., $\boldsymbol{x}^*(\boldsymbol{y}')$). We argue that different information can be used and generated to solve the problem in Eq. (1). In fact, once $\boldsymbol{Y}$ becomes known, we may not have enough time to run a large number of expensive simulations to take a decision. While data generated through simulation experiments provide a rich body of information, the simulation ensembles generated in the past almost certainly deviate from actual observations in the real world. Hence, solely relying on decisions learned from simulated scenarios would inevitably lead to sub-optimal outcomes. On the other hand, fully exploring the decision space is not a viable approach because of the short time-window decision makers face and the time-consuming nature of high-fidelity simulations that account for the current observations.

The fundamental question, therefore, is how to make use of simulated data along with the current observational data (i.e., the knowledge of $\boldsymbol{Y} = \boldsymbol{y}$) to efficiently search the decision space while generating *few* additional simulations. Existing techniques are of limited value because of the complexity, high dimensionality, and the black-box nature of most real-world problems. In this paper, we propose the novel GOLF that tackles the problem formalized in Eq. (1) by intelligently generating and learning from simulations obtained by fixing the scenarios $\boldsymbol{Y} = \boldsymbol{y}$ and decision $\boldsymbol{x}$, and *transfers* the learned information to the current scenario when it becomes known. The new methodologies proposed look into four main challenges:

(a) Generate intelligently and evaluate offline scenarios $\boldsymbol{Y} = \{\boldsymbol{y}\}$.
(b) Estimate the function value $\mathfrak{J}(\boldsymbol{x}, \boldsymbol{Y} \mid \boldsymbol{Y} = \boldsymbol{y})$ by intelligently sampling "interesting" actions $\boldsymbol{x} \in \mathbb{X}$.
(c) Learn the distribution $g(\mathbf{Y}) := (\boldsymbol{x}^* \mid \boldsymbol{Y}) \in \ \text{argmax}_{\boldsymbol{x} \in \mathbb{X}} \mathfrak{J}(\boldsymbol{x}, \boldsymbol{Y} \mid \mathbf{Y} = \boldsymbol{y})$ of optimal decisions under scenarios $\boldsymbol{y}$.
(d) Once the scenario is known, use offline information to guide online sampling through ordinal learning and efficiently identify the best guess for $\boldsymbol{x}^*$.

In the following, we present GOLF in its general architecture and detail the different components.

3. Methodology: GOLF for Simulation Based Decision and Control

GOLF separates the learning and optimization into two phases: (1) *offline*: GOLF tries to construct appropriate models for the function of interest using available historical data together with simulated data. This is done before the scenario of interest is revealed to the user; (2) *online*: once the scenario of interest becomes known, GOLF selects data, generated offline, with a few carefully selected new simulations.

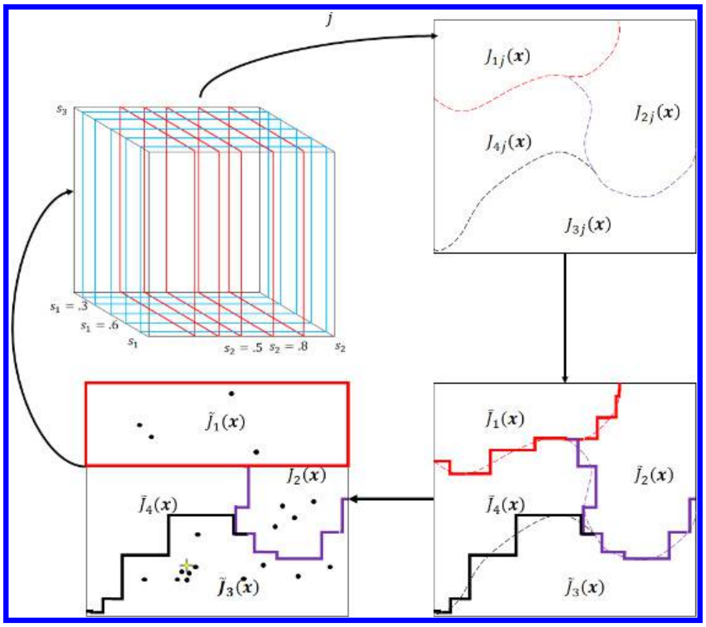


Fig. 2. GOLF partitions learning.

The *offline phase* tackles the problems of (1) non-smooth objective functions and (2) high-dimensional solution spaces. In view of the first problem, we propose a proof of concept for a novel framework to iteratively learn partitions of the solution space (or a transformation of it) allowing for the estimation of better models. Figure 2 shows the main idea behind the offline learning scheme. Since we deal with complex systems, it is natural to argue that *multiple* models may better represent the system behavior in different regions. As an example, Fig. 2 (top right) shows four different functions ($\mathcal{J}_{1,j}, \ldots, \mathcal{J}_{4,j}$) where j is the index of the "slice" considered by the model. Slicing is a type of projection that GOLF uses to tackle the problem of high dimensionality. Subsequently, our sampling algorithms will give a different focus to different sub-regions driven by two main goals: (1) provide a good approximation of the function across the solution/scenario space (bottom right representation in Fig. 2) and (2) concentrate simulation effort in regions that are most relevant to the optimization problem (bottom left representation in Fig. 2). Therefore, GOLF needs to adaptively learn how the solution space (or a transformed version of it) is partitioned and which model best represents the system behavior in each of these learned partitions. Sections 3.1 and 3.2 deal with the problem of defining effective criteria for the partitioning and estimation of the sub-models, respectively. Section 3.3 deals with the efficient partitioning of the solution space to increase the computational feasibility of the offline phase.

When all the offline data have been created and the information about the scenario of interest becomes available, the online procedure needs to effectively use the offline information and generate few expensive online data to solve a complex optimization problem (Adaptive Ensemble Ordinal Learning (AEOL), Sec. 3.4).

3.1. *Learning from simulation: Metrics to define experiments relevance in GOLF*

We address the problem of how to define "interesting" configurations that sampling should give higher priority to. We are interested in scenarios $\boldsymbol{Y} = \{\boldsymbol{y}\}$ and configurations $\boldsymbol{X} = \{\boldsymbol{x}\}$ that can "easily" explain past observations, while preserving fit quality. A common way to assess how well a set of simulations/data explain a phenomenon of interest is to define a measure of fit, such as root-mean square error (RMSE) or perplexity (in probabilistic settings), that help compare a set of simulations to the corresponding observations. A challenge in time-variant domains is that the degree of pairwise-matching between corresponding simulation and observation instances may be difficult to define, especially if the observations themselves are incomplete and or noisy and if the processes driving different variates is asynchronous (Yu-Ru *et al.*, 2011).

We argue that focusing on the maximization (or minimization) of a single measure of fit may drive the simulation effort towards a small set of apparently promising parameter configurations, that collectively provide a likely explanation of the observations. Another shortcoming of the basic fit criteria is that they do not necessarily promote parsimony of the resulting explanations, which can be critical in decision making (Viana *et al.*, 2014; Myers and Anderson-Cook, 2009; Muller and Piche, 2011; Goel *et al.*, 2007; Santner *et al.*, 2013; Kleijnen, 2015; Le Gratiet and Cannamela, 2015). Traditionally, model complexity is measured probabilistically, examples are Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) (Ozdogan, 1987). However, evaluating the interpretative-complexity of an ensemble is all but trivial and several difficult questions need to be addressed: (a) interpretative-complexity is related to the effort required to estimate a function, but how can we measure/predict, such an effort with highly non-linear models? (b) interpretative-complexity is context-dependent: while, in some contexts, harmonic signals may have an easy interpretation, elsewhere polynomials may be natural for the explanation of a phenomenon.

In this paper, we introduce a novel approach to embed complexity considerations in our family of ensembles. Specifically, let $\mathcal{C}(f(x))$ be the interpretative-complexity associated to function $f(x)$. Table 1 presents a set of axioms that characterize the complexity functional $\mathcal{C}(f(x))$, used to associate appropriate complexity invariants, $\mathcal{C}(\sin), \mathcal{C}(\exp)$, and $\mathcal{C}(\log)$ specific to an application domain (i.e., defined by the user). It is important to highlight how this new formulation of complexity not only considers the *shape of the surface* defined by the function, but also the *specific form the description of the function* takes. In fact, the form to describe the function plays an important role in our complexity measure since the raw function may take several steps before being converted into its final simpler form, but this requires

Table 1. Axioms of interpretative-complexity.

$f(x) = b \rightarrow \mathcal{C}(f(x)) = 0$
$f(x) = x \rightarrow \mathcal{C}(f(x)) = 1$
$f(x) = x^m \rightarrow \mathcal{C}(f(x)) = \begin{cases} m, & \text{if } m > 0 \\ \frac{1}{m}, & \text{otherwise} \end{cases}$
$f(x) = \sin x \rightarrow \mathcal{C}(f(x)) = \mathcal{C}(\sin)$
$f(x) = e^x \rightarrow \mathcal{C}(f(x)) = \mathcal{C}(\exp)$
$f(x) = \log x \rightarrow \mathcal{C}(f(x)) = \mathcal{C}(\log)$
$\mathcal{C}(f(g(x))) = \mathcal{C}(f(x)) \times \mathcal{C}(g(x))$
$\mathcal{C}(f(x) + g(x)) = \max\{\mathcal{C}(f(x)), \mathcal{C}(g(x))\}$
$\mathcal{C}(f(x) \times g(x)) = \mathcal{C}(f(x)) + \mathcal{C}(g(x))$

one to spend more effort and time, hence justifying its consideration as part of the complexity measure. Several approaches may be used to define the complexity of a function, while this paper does not want to be exhaustive, we highlight how complexity should be an integral part of any model evaluation criteria. In this work, we will show preliminary results that use the complexity indicator to enhance more traditional model performance metrics such as R^2 (Alrefaei and Andradóttir, 2001; West *et al.*, 2012), AIC (Ozdogan, 1987), and Coverage (Burnham and Anderson, 2004; Wong *et al.*, 2011). In particular, we discuss how complexity metrics can be derived as extension of the Akaike Based Metrics. A similar approach can be adopted for R^2 and Coverage (Fig. 3).

Akaike Based Metrics. More specifically, let $\mathcal{M}$ be a model learned from data $\mathcal{X}$. Let $\pi(\mathcal{M})$ be the number of parameters to be inferred in the model $\mathcal{M}$ and $\mathcal{L}$ be the

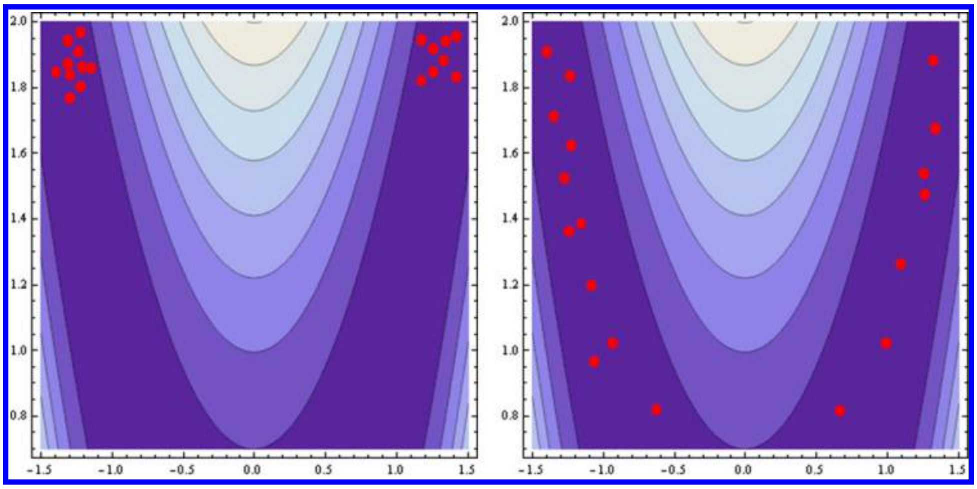


Fig. 3. Optimal fit (on the left) against O-Diversity maximizing sampling (on the right).

maximum value of the likelihood function for the model; i.e., $\mathcal{L}(\mathcal{M}, \mathcal{X}) = P(\mathcal{X} \mid \boldsymbol{\theta}, \mathcal{M})$, where $\boldsymbol{\theta}$ are the optimal parameter values for the likelihood function. Considering these definitions, the AIC penalty for a model results in the following:

$$\mathrm{AIC}(\mathcal{M}, \mathcal{X}) = 2\pi(\mathcal{M}) - 2\ln \mathcal{L}(\mathcal{M}, \mathcal{X}).$$

From the definition, we can see how the AIC aims to find a balance between the number of parameters to be inferred (which is a measure of the model complexity) and the model's ability to explain the data. The AIC penalizes models with a large numbers of parameters. The model complexity (i.e., the number of model parameters) that the AIC as previously defined relies on is rather limited. To overcome this challenge, we introduce the *Complexity Guided Akaike information* (C-AIC) that makes use of the complexity rules in Table 1 to associate a complexity penalty for each model in the model dictionary, namely

$$C\text{-}\mathrm{AIC}(\mathcal{M}, \mathcal{X}) = 2\mathcal{C}(\mathcal{M}) - 2\ln \mathcal{L}(\mathcal{M}, \mathcal{X}).$$

Intuitively, C-AIC replaces the term $\pi(\mathcal{M})$ (the number of free parameters) with the complexity measure, $\mathcal{C}(\mathcal{M})$, that considers both the number of free parameters, but also the interpretability of the model).

3.2. *Sampling in the action space to reconstruct $\mathcal{J}(x, Y \mid Y = y)$*

Constructing ensembles is very important to make effective use of available historical data for scenarios $t\{\boldsymbol{Y} = \boldsymbol{y_\ell}\}_{\boldsymbol{\ell=1}}^{L}$ occurred in the past. The main aspect for this response estimation is to efficiently sample in the space of the candidate solutions, $\mathcal{X}$ for several scenarios $\{\boldsymbol{Y} = \boldsymbol{y_\ell}\}_{\boldsymbol{\ell=1}}^{L}$. The criteria proposed in Sec. 3.1 are at the basis for the reconstruction of such ensembles. In this regard, weighted and additive models start from fitting several surrogates over the entire parameter space, perform an evaluation of the fitting quality, and then compute *static weights* over the parameter space to maximize a measure of the fit or prediction error (Muller and Piche, 2011; Goel *et al.*, 2007; Meng and Ng, 2015). In either case, there is no mechanism to explicitly control the interpretative complexity. In addition, most of the approaches are not sequential in nature thus preventing incremental learning.

While several space partitioning approaches have been proposed in ML, our focus is fundamentally different: rather than training a classifier with given observations, we are interested in understanding parameter configurations that are worth sampling and which scenarios are worth exploring. We argue that dynamic and adaptive, instead of static, weighting structures should be used for generating an appropriate ensemble, where weighting is performed not only based upon fitting, but also the complexity of the resulting model. In fact, a complex system may show completely different patterns in different regions of the parameter space. To ensure that the ensemble is created in a way that properly covers and describes the underlying phenomenon, GOLF starts by sampling from the complete parameter space as a whole and, *as needed*, the same space is partitioned into finer regions. The basic idea is that, if all current partitions match the target quality requirement or the offline simulation budget has been consumed in its entirety, the process ends. Specifically, we propose a novel *rank stability-based strategy*, which selects the partition with the worst rank-stability (under the appropriate penalty function, see Sec. 3.1) for further investigation.

Notation and Definitions. Let us be given a complex system, $\mathcal{S}$, with $\mathcal{N}$ input parameters, such that the ith input parameter can take $\mathcal{J}_i$ distinct values. Also, let $\mathcal{X}$ be the set of offline simulation that have been selected to be executed so far. Also, let $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_p\}$ be a set of non-overlapping partitions of the input parameter space, such that $\bigcup_{\mathcal{P}_i \in \mathcal{P}} \mathcal{P}_i = \mathcal{J}_1 \times \mathcal{J}_2 \times \cdots \times \mathcal{J}_N$ (thus guaranteeing the desired *coverage*). Now let $\mathcal{X}_i$ denote the (non-empty) subset of $\mathcal{X}$ that fall in partition $\mathcal{P}_i$. Now, consider a specific model dictionary (we saw model dictionaries for the definition of complexity in Sec. 3.1) $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k\}$ such that each model $\mathcal{M}_j$ is associated with a penalty p_{ij}, where i is the index of the specific penalty criterion (for example we could use the C-AIC from Sec. 3.1).

Rank Stability-Based Partitioning. Let us assume that the partition $\mathcal{P}_i$ is selected for further investigation and we extend the simulation set, $\mathcal{X}_i$, with new simulation instances, which we refer to as $\Delta\mathcal{X}_i$. Let us denote the extended simulation set as $\mathcal{X}'_i = X_i \cup \Delta\mathcal{X}_i$. This new simulation set has associate a revised penalty, p'_{ij} for each model $\mathcal{M}_j \in \mathcal{M}$. With such information, it is possible to define the rank stability as $\mathrm{RS}(\mathcal{P}_i, X_i, \Delta\mathcal{X}_i)$, for each partition $\mathcal{P}_i \in \mathcal{P}$ related to the simulation set, $\mathcal{X}_i$, and new instances, $\Delta\mathcal{X}_i$, as

$$\mathrm{RS}(\mathcal{P}_i, \mathcal{X}_i, \Delta\mathcal{X}_i) = \sum_{\substack{M_j \in M \\ \text{in decreasing order of penalty } p_{ij}}} \frac{\left(\min_j p'_{ij} - p'_{ij}\right)/2}{\log_2 j + 1},$$

where $\min_j p'_{ij}$ is the minimum penalty computed for all models in $\mathcal{M}$. Intuitively, $\mathrm{RS}(\cdot)$ is analogous to the normalized discounted cumulative gain (NDCG) of the old model ranking as a function of the new model ranking and quantifies if the additional simulation samples for the given partition result in a major shift in the ranking of the models in the model dictionary. The numerator re-normalizes the model penalties, such that the models with lower penalty are given higher weights in measuring rank stability. Intuitively, if the rank stability is low, it means that the current partition is hard to describe with $\mathcal{X}_i$ or $\mathcal{X}'_i$. In contrast, if the rank stability is high, it means that additional simulations do not impact the ranks of the models in the dictionary. In such a circumstance, additional simulation samples for the given partition may not be necessary. Once a partition, $\mathcal{P}_i$ is picked, GOLF assigns a set, $\Delta\mathcal{X}_i$, of new simulations to the partition. Once these simulations have been executed, models within $\mathcal{M}$, are re-ranked based on the revised set, $\mathcal{X}'_i$ of simulations for the given partition and a new rank stability measure, $rs_i = \mathrm{RS}(\mathcal{P}_i, \mathcal{X}_i, \Delta\mathcal{X}_i)$ is computed for the partition. If rs_i is less than a predefined threshold $\bar{rs}$, the partition is not rank-stable and needs to be further partitioned. However, if $rs_i \geq \bar{rs}$, $\mathcal{P}_i$ may or may not need further partitioning, a look-ahead perspective can be used to take this decision.

Look-ahead partitioning. In particular, (a) $\mathcal{P}_i$ is first *virtually* split into sub-partitions and (b) a model ranking is obtained for each sub-partition based on the resulting simulation instances allocation; (c) next, the rank stability of each sub-partition $\mathcal{P}_i$ is computed and compared with the reference threshold $\bar{rs}$: if none of the partitions $\mathcal{P}_i \in \mathcal{P}$ fail the rank stability test, it means the order of the models obtained when considering the sub-partitions is aligned with the order of the models for the original partition $\mathcal{P}_i$. In this case, the original partition $\mathcal{P}_i$ does not need to be further split; if a partition $\mathcal{P}_i \in \mathcal{P}$ fail the rank stability test, it is composed of heterogeneous regions and, thus, needs to be further split into smaller partitions. When the process ends (either due to budget completion or rank-stable partitions), we have a simulation ensemble and a ranked list of instantiated models for each partition. One advantage of this approach is that the user can be provided with, not one, but several top alternative models as candidates for each partition.

3.3. *Learning the distribution* $g(\boldsymbol{Y}) := (\boldsymbol{x}^* \mid \boldsymbol{Y}) \in \mathbf{argmax}_{\boldsymbol{x}\in\mathbb{X}} \mathcal{J}(\boldsymbol{x}, \boldsymbol{Y})$: *A Large Scale Bayesian Optimization approach* (*LSBO*)

While the methods in Sec. 3.2 target the reconstruction of response surfaces to represent our cost function, $\mathcal{J}(\boldsymbol{x}, \boldsymbol{y})$ for past historical scenarios, this part of our work focuses on scenarios that *have never occurred*. In this case, we need to consider $\boldsymbol{x}$, i.e., the configuration parameters, as well as the "scenario" parameters $\boldsymbol{y}$. Specifically, we develop a method to reconstruct the optimal condition for any parameter setting that we wish to test as possible future (i.e., the action/strategy that the decision maker would plausibly apply). If we refer to $\boldsymbol{Y}$ as the parameter setting describing the future condition of the complex system, simulating the function $\mathcal{J}(\boldsymbol{x}, \boldsymbol{Y} \mid \boldsymbol{Y} = \boldsymbol{y})$ requires us to "reconstruct" the unknown interventions and strategies in the future, for which we need to construct meaningful sample paths. We propose to tackle this problem in the form of a large scale stochastic optimization and we propose the Large Scale Bayesian Optimization (LSBO) algorithm. Let us refer to the function $\boldsymbol{x}^*(Y) \in \mathrm{argmax}_{x\in\mathbb{X}} \mathcal{J}(\boldsymbol{x}, Y)$. Sampling in the space of parameter settings to learn $\boldsymbol{x}^*(Y)$ is difficult due to the size of the set of parameters $\boldsymbol{Y}$, which makes any approach based on enumeration impractical. At the same time, the sampling algorithms such as those presented in Sec. 3.2 would not help with this dimensionality issue. This requires *a novel and transformative sampling paradigm.* In this work, we propose a projection method that helps dealing with the problem of sampling in high dimensions. The proposed approach has two main components: (1) random adaptive projection schemes that allow to iteratively sample in lower-dimensional spaces; (2) optimal samplers that can quickly return an estimate of $\boldsymbol{x}^*(Y)$.

Random Projections. Let us refer to the parameter space as $\mathbb{Y} \subseteq \mathbb{R}^d$ where d is a very large value. At this point, we want to partition the space to obtain a satisfactory model for $\boldsymbol{x}^*(Y)$. While the criteria defined in Sec. 3.1 are still useful in this setting, we cannot afford to sample from the original space due to the size of $\boldsymbol{Y}$ and the fact that optimization needs to be solved for each parameter configuration. We propose to sequentially decompose the space $\mathbb{Y}$ into a number of M subspaces, all sharing one dimension s (to be chosen) $\mathbb{Y}^s \subseteq \mathbb{Y}, s = 1, 2, \ldots,$ where M can be dynamically optimized and $\bigcup_{s=1}^{M} \mathbb{Y}^s \supseteq \mathbb{Y}$. Figure 4 graphically shows the main idea behind parameter space slicing. Once projections have been established, we can perform "local" optimization. As a result, each local algorithm returns a *projected proposal of the distribution of the optimal selection as a function of the subset of considered parameter settings.* We can refer to this information as $\hat{g}(\mathbb{Y}^s)$ and we need to find an appropriate way to "pass" this estimate to the other projections (Fig. 5). In this regard, we propose the *integrated complexity* along with the *integrated diversity* indicators to communicate each projection to which scenario value they should focus for improving the estimation of $\hat{g}(\mathbb{Y})$. The reason to extend the criteria presented in Sec. 3.1 is that we need to integrate with respect to the shared dimension. In other words, we need to provide the most interesting parameter setting, which is also the most robust with respect to the shared dimension.

Optimal Sampling. In relation to the second item, it is important to highlight that obtaining an evaluation of $\boldsymbol{x}^*(Y)$ implies a full optimization, therefore we need to carefully allocate the budget in an appropriate way. Even if this phase of the approach is offline, solving a complex non-linear optimization for each point in the scenario space would be computationally unattainable. In this direction, we

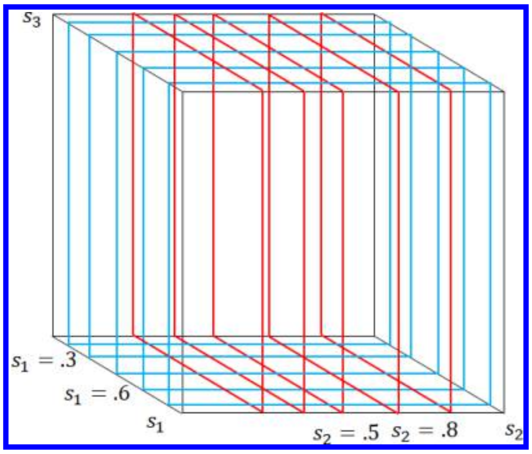


Fig. 4. Scenario space "slicing". The original space has 3 dimensions. The main idea is to consider $M = 2$ subspaces of 2 dimension each. Similar to Zhang *et al.* (2017) we decompose the space in a way such that 1 dimension is shared between the different projections, this allows us to increase the sampling density.

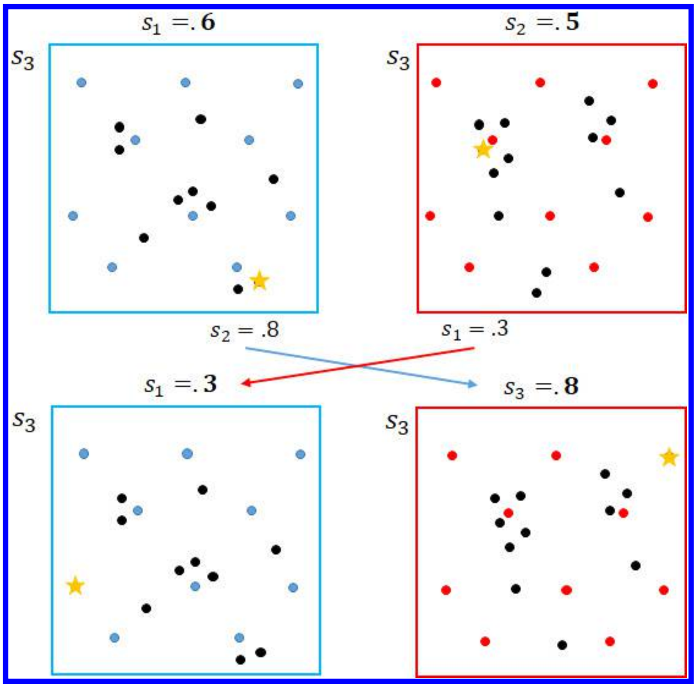


Fig. 5. Information exchange between parameter slices.

propose to extend the multi-fidelity algorithm presented by the authors in Inanlouganji *et al.* (2018), and further detailed in Sec. 3.4, in order to consider the presence of a low fidelity solution, which implies a completely different algorithmic framework. Nonetheless, the promising results in Sec. 4.2 give us confidence of the viability of the approach. Even in the "decomposed space" (Zhang *et al.*, 2017), a significant challenge in budget-constrained simulation ensemble generation remains: different parameter configurations and different models in the model-dictionary can have different associated cost/quality trade-offs. While the ensemble generation can be interpreted as a generalization of the Optimal Computing Budget Allocation problem (initially developed in Chen *et al.* (2000)) for discrete optimization, several key challenges remain unsolved in this context (Kandasamy *et al.*, 2016; Ryzhov *et al.*, 2010; Benamara *et al.*, 2016): (a) how much budget to dedicate to each iterative ensemble generation process and (b) how to allocate budget to the different parameters configurations, when different conditions lead to substantially different simulation costs in terms of required computational time. In our preliminary work (Pedrielli and Ng, 2016), we have shown some results in this direction, but two main challenges still remain: (1) how to choose the number of iterations (i.e., how much to sample "now" and how much to reserve for "later"), (2) how to allocate the budget among different candidate points and parameter configurations for a given total budget. We propose a novel formulation of the budget allocation in the form of stochastic control, with the objective function considering the cost of sampling, derived from the choice of the sample set, and the cumulative gain in terms of improved accuracy conditional on the current sampling decision and cumulated over all iterations.

3.4. *Online decision making through ordinal learning*

Given the size of the configuration space and the scenarios, the probability that the LAD ensemble contains the current configuration (i.e., the present was already observed in the past) is zero. However, the information contained in the simulation ensembles can be valuable in helping to learn the outcomes of decisions in an online manner.

A water distribution network example. To give a first conceptual idea of the proposed approach, we refer to an example in water infrastructure management. Let us imagine that we want to optimally configure a water network regulation and pump schedules to satisfy dynamically varying demands for multiple types of consumers (private, industrial). The system has eight pumps of three types transmitting water to 11 types of customers (11 demand profiles). There are a total of 10,500 configurations obtained by selecting different possible transmissions, pumps allocation, and alternative schedules. The offline data (LAD) consist of the simulation results for all 10,500 configurations under a single value of demand from the different customers. Demand is known for the past, but clearly not for the future. Therefore, once the demand for supply becomes known, we are interested in learning the performance of different configurations such that we can adopt the optimal configuration. The left plot in Fig. 6 shows the performance levels of all 10,500 configurations, which is obtained through extensive simulations that cannot be done online. We observe the large variances in performance and the consequent challenge in search such a configuration space (x-*axis* in Fig. 6).

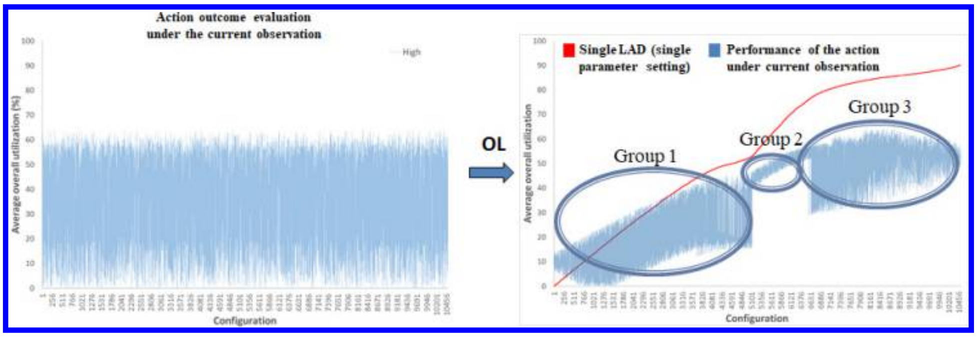


Fig. 6. (Color online) Online ordinal learning of 10,500 configurations using look-ahead data versus actual performance under observed parameters setting. The red smooth curve shows the ordered performance of all configurations given by look-ahead data (1 single parameter configuration is considered). The blue jagged band shows the actual performance under the observed parameters setting.

We propose a highly effective *ordinal learning* (*OL*) method to learn from LAD. OL orders all 10,500 configurations based on the LAD performance, already available from offline phase, selecting the configuration with best performance (the leftmost one in the right plot of Fig. 6) as the best decision for the current scenario. While the selected decision is not necessarily optimal, it has near optimal performance. OL is extremely simple and fast to execute for online applications and it is also independent of the dimensionality of the configuration space, and thus is generally amenable. *This preliminary result demonstrates that OL may be a highly efficient and effective online learning method to synthesize LAD with current data.*

In the illustrative example above, offline LAD for a configuration $\boldsymbol{x}$ is only available for a specific parameter setting $\boldsymbol{Y} = \boldsymbol{y_\ell}$. However, LAD driven sampling generates many more LAD for a specific selection $\boldsymbol{x}$. AEOL is what we propose to make use of LAD in online decision making extending our preliminary approach depicted in Fig. 6. The approach enabling to embed LAD ensembles instead of a single parameter configuration is to determine a weight assigned to different LAD ensembles based on their "relevance" with respect to the current observational data.

$$\begin{aligned} J(\boldsymbol{x};\boldsymbol{\theta} \,|\, \boldsymbol{Y}=\boldsymbol{y}) = \sum_{\ell=1}^{L} & w_\ell(\boldsymbol{x})[\alpha_\ell(\boldsymbol{x};\boldsymbol{\theta_\alpha}) J_{\boldsymbol{Y}=\boldsymbol{y_\ell}}(\mathcal{M}_{\boldsymbol{\ell}}(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{M}});\boldsymbol{\theta}_{\boldsymbol{y}_\ell} \,|\, \boldsymbol{Y}=\boldsymbol{y}_\ell) \\ & + B_\ell(\mathcal{M}_{\boldsymbol{\ell}}(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{M}});\boldsymbol{\theta}_{B,\boldsymbol{y}_\ell} \,|\, \boldsymbol{Y}=\boldsymbol{y}_\ell)], \quad \boldsymbol{x} \in \mathbb{X}. \end{aligned} \tag{2}$$

In our preliminary work (Inanlouganji *et al.*, 2018), we assumed that $J_{\boldsymbol{Y}=\boldsymbol{y}_\ell}$ and J, i.e., the LAD ensembles and observational data, as well as the bias process B_ℓ representing the distance between $J_{\boldsymbol{JY}=\boldsymbol{y}_\ell}$ and the observed response J can be modeled as GP. In model (2), $\boldsymbol{\theta}_{LF}$ and $\boldsymbol{\theta}_B$ refer to the vector of the hyper-parameters needed to construct a predictor for the GP $J_{\boldsymbol{Y}=\boldsymbol{y}_\ell}$ and B, respectively. In Eq. (1), we have L LAD ensembles. The multiplier $\alpha_\ell(\boldsymbol{x};\boldsymbol{\theta}_\alpha)$ is generally a function of the location $\boldsymbol{x}$ and is parametrized through $\boldsymbol{\theta}_\alpha$. Also, $\mathcal{M}(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{M}})$ represents a general mapping function from the look-ahead scenario space to the current observational space. This mapping function is parametrized through $\boldsymbol{\theta}_{\mathcal{M}}$, which requires the development of learning mechanisms to be efficiently estimated. Finally, $w_\ell(\boldsymbol{x})$ represents the adaptive ensemble weight that associates a different importance score to a look-ahead scenario used in the prediction of the unknown function $J(\boldsymbol{x};\boldsymbol{\theta} \,|\, \boldsymbol{Y})$. Section 3.4.1 illustrates the methods we have explored so far to estimate the model in (2), while Sec. 3.4.2 focuses on the iterative algorithm for the online selection of few expensive simulations together with the conditional selection of offline generated data in order to quickly solve the online optimization problem.

3.4.1. *Estimation of the multi-fidelity model*

To derive the model in Eq. (2) we will have two separate sets of sampled points, $\{\mathbb{X}_{\mathbf{Y}=\boldsymbol{y}_\ell}\}_{\ell=1}^{L}$ representing the collection of points sampled in scenarios $\boldsymbol{Y} = \mathbf{y}_\ell$ that differ from the target scenario of interest. These scenarios can be sampled offline according to the methods introduced in Secs. 3.2 and 3.3. Therefore, concerning the online phase, we can see this set of points and the related function evaluations as available at "zero cost". The online phase is therefore concerned about: (1) understanding which points in the offline set represent valuable information with respect to the current scenario, (2) sample a restricted set of points, which we will refer to as $\mathbb{X}_{\boldsymbol{B}}$, for the current scenario to add to the offline evaluations to improve the decision making process. From an online perspective, $\{\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}\}_{\ell=1}^{L}$ is the "cheapest" set since all the evaluations are available at the moment when the scenario of interest is revealed. Therefore, we will generally have $|\mathbb{X}_B| \ll |\{\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}\}_{\ell=1}^{L}|$. Given a desired prediction location $\boldsymbol{x}_0$, to estimate $\hat{B}(\boldsymbol{x}_0) \,|\, \mathbb{X}_B; \boldsymbol{\theta}_B$, which, for simplicity, will be referred to as $\hat{B}(\boldsymbol{x}_0)$ from now on, we first obtain the elements of observed bias vector, $\boldsymbol{b}$ using Eq. (3) below

$$b(\boldsymbol{x}) = J_{Y=y}(\boldsymbol{x}) - G(\{J_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\mathcal{M}_{\boldsymbol{\ell}}(\boldsymbol{x};\boldsymbol{\theta}_M);\boldsymbol{\theta}_{\boldsymbol{y}_\ell} \,|\, \boldsymbol{Y}=\boldsymbol{y}_\ell)\}_{\ell=1}^{L}), \quad \boldsymbol{x} \in \mathbb{X}_B, \tag{3}$$

where $G(\cdot)$ is an appropriate function of the several offline data available at a specific location. Such a function has the role to associate larger weight to using observed bias, $\boldsymbol{b}$ vector as the response, a Gaussian process can be trained to estimate the bias at any given solution $\boldsymbol{x}_0$ as follows:

$$\hat{B}(\boldsymbol{x}_0) \sim \mathcal{N}(\mu_B(\boldsymbol{x}_0), \sigma_B^2(\boldsymbol{x}_0)), \tag{4}$$

$$\mu_B(\boldsymbol{x}_0) = \boldsymbol{c}^{\boldsymbol{T}}(\boldsymbol{x}_0, \mathbb{X}_B)\boldsymbol{K}^{-1}\boldsymbol{b}, \tag{5}$$

$$\sigma_B(\boldsymbol{x}_0) = \boldsymbol{K}(\boldsymbol{x}_0,\boldsymbol{x}_0) - \boldsymbol{c}^{\boldsymbol{T}}(\boldsymbol{x}^s, \mathbb{X}_B)\boldsymbol{K}^{-1}\boldsymbol{c}(\mathbb{X}_B, \boldsymbol{x}_0). \tag{6}$$

For each offline scenario that we have tested, we estimate $\hat{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x}_0) \,|\, \mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}$; $\boldsymbol{\theta}_{\mathbf{Y}=\boldsymbol{y}_\ell}$, considering all the configurations that have been sampled according to the methods in Sec. 3.2, i.e., $\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}$. We will refer to these models, for simplicity of notation, as $\hat{J}_\ell(\boldsymbol{x}_0)$. Specifically, for each scenario $\boldsymbol{Y}=\boldsymbol{y}_\ell$, we use $\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}$ as input matrix and $\boldsymbol{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x} : \boldsymbol{x} \in \mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell})$, the vector of the offline response values for a specific scenario and the locations in the set $\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}$, as the output. A Gaussian process model can be estimated to obtain the following predictors:

$$\hat{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x}_0) \sim \mathcal{N}(\mu_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x}_0), \sigma_{\boldsymbol{Y}=\boldsymbol{y}_\ell{}^2}(\boldsymbol{x}_0)), \tag{7}$$

$$\mu_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x}_0) = \boldsymbol{c}^{\boldsymbol{T}}(\boldsymbol{x}_0, \mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell})\boldsymbol{K}^{-1}\boldsymbol{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x} : \boldsymbol{x} \in \mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}), \tag{8}$$

$$\sigma_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x}_0) = \boldsymbol{K}(\boldsymbol{x}_0,\boldsymbol{x}_0) - \boldsymbol{c}^{\boldsymbol{T}}(\boldsymbol{x}_0, \mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell})\boldsymbol{K}^{-1}\boldsymbol{c}(\boldsymbol{x}_0, \mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}). \tag{9}$$

Finally, we can estimate $\hat{J}_\ell(\boldsymbol{x}_0)$ using a linear combination of the two predictions in (4) and (7), obtaining

$$\hat{J}_{Y=y}(\boldsymbol{x}_0) \sim \mathcal{N}(\mu_{Y=y}(\boldsymbol{x}_0), {\sigma_{Y=y}}^2(\boldsymbol{x}_0)), \tag{10}$$

$$\mu_{Y=y}(\boldsymbol{x}_0) = G_\mu(\{\mu_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x}_0)\}_{\ell=1}^{L}) + \mu_B(\boldsymbol{x}_0), \tag{11}$$

$${\sigma_{Y=y}}^2(\boldsymbol{x}_0) = G_\sigma(\{{\sigma_{\boldsymbol{Y}=\boldsymbol{y}_\ell}}^2(\boldsymbol{x}_0)\}_{\ell=1}^{L}) + {\sigma_B}^2(\boldsymbol{x}_0). \tag{12}$$

It is important to highlight again how, to guarantee adequacy of the additivity of the processes, we should have $\{\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}\}_{\ell=1}^{L} \cap \mathbb{X}_{\boldsymbol{B}} = \boldsymbol{\emptyset}$. While this is not the case, we have $|X_B| \ll |\{\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}\}_{\ell=1}^{L}|$, thus decreasing the impact of the dependency effect generated by the sampling policy. As a result, the sampling process for the offline observations and the one for the learning of the offline/online dependency do not overlap. We propose a third model to guide the decision whether to perform an online sampling decision or not. Specifically, we derive the predicted offline response using the same relationship as in Eq. (3), but with the focus of estimating the offline response, i.e., we look at the relationship in (2) as $J_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x};\boldsymbol{\theta}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}) = G^{-1}(J_{Y=y}(\boldsymbol{x};\boldsymbol{\theta}_{Y=y}) - B(\boldsymbol{x};\boldsymbol{\theta}_B)), \boldsymbol{x} \in X$. While the $\boldsymbol{B}$ component is the same as in (4)–(6), $\hat{J}_{Y=y}(\boldsymbol{x};\boldsymbol{\theta}_{\boldsymbol{Y}=\boldsymbol{y}_\ell})$ is estimated only using the online evaluations $\boldsymbol{J}_{\boldsymbol{Y}=y|\mathbb{X}_B}$. In order to distinguish this predictor from the one in Eq. (10), we refer to it as $\hat{J}_{\boldsymbol{Y}=y|\mathbb{X}_{\boldsymbol{B}}}(\boldsymbol{x};\boldsymbol{\theta}_{\boldsymbol{Y}=\boldsymbol{y}_\ell})$, and we refer to this alternative estimation of the offline response model as $\hat{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_{\boldsymbol{B}}}$ and we derive it as it follows:

$$\hat{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_B}(\boldsymbol{x}_0) \sim \mathcal{N}(\mu_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_{\boldsymbol{B}}}(\boldsymbol{x}_0), {\sigma_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_{\boldsymbol{B}}}}^2(\boldsymbol{x}_0)), \tag{13}$$

$$\mu_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_B}(\boldsymbol{x}_0) = \mu_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_B}(\boldsymbol{x}_0) - \mu_B(\boldsymbol{x}_0), \tag{14}$$

$${\sigma_{\boldsymbol{Y}=\boldsymbol{y}_\ell\,|\,\mathbb{X}_{\boldsymbol{B}}}}^2(\boldsymbol{x}_0) = \sigma^2_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_{\boldsymbol{B}}}(\boldsymbol{x}_0) + \sigma^2_B(\boldsymbol{x}_0). \tag{15}$$

We highlight that, different from the prediction in (10)–(12), the derivation for the conditional density is not an exact result, because we do not account for the inherent dependency between the processes $\hat{J}_{\boldsymbol{Y}=y}$ and $\hat{B}$ generated by the fact that the set of online sampled points $\mathbb{X}_{\boldsymbol{Y}=y}$ and the set of sampled points in the bias set X_B are the same.

3.4.2. *Online sampling algorithm*

Let us consider a single offline scenario $\boldsymbol{Y} = \boldsymbol{y}_\ell$. We investigate *two versions of the algorithm* for online sampling: (1) the offline data driven selection and (2) expensive online selection. The first version uses the offline evaluation at the best location up to iteration k, i.e., $\boldsymbol{x}^*_{k,\boldsymbol{Y}=\boldsymbol{y}_\ell} \in \operatorname{argmin}_{\boldsymbol{x}\in\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}} J_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x})$, where the achieved low-fidelity function value is referred to as $J^b_{k,\boldsymbol{Y}=\boldsymbol{y}_\ell}$ and $\mathbb{X}_{k,\boldsymbol{Y}=\boldsymbol{y}_\ell}$ is the set of locations that have been selected from the offline data set, up to iteration k. The second approach uses the expensive online evaluation at the best location up to iteration k, i.e., $\boldsymbol{x}^*_{k,Y=y} \in \operatorname{argmin}_{\boldsymbol{x}\in\mathbb{X}_{Y=y}} J_{Y=y}(\boldsymbol{x})$, where the achieved function value is referred to as $J^b_{k,Y=y}$ and $\mathbb{X}_{k,Y=y}$ is the set of locations that have been sampled online. In this work, we adopt the Expected Improvement (EI) as sampling criteria (Locatelli, 1997; Jones *et al.*, 1998). The EI is a common sampling criterion in surrogate-based optimization and its derivation is discussed in Jones *et al.* (1998).

Offline data driven selection. The improvement at any location, $\boldsymbol{x}$, as $I(\boldsymbol{x}) = \max(J^b_{k,\boldsymbol{Y}=\boldsymbol{y}_\ell} - \hat{J}_{k,\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x}), 0)$, where $\hat{J}_{k,\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x})$ is the prediction obtained from model (7) at iteration k, when $\mathbb{X}_{k,\boldsymbol{Y}=\boldsymbol{y}_\ell}$ points have been selected from the offline data set. The EI results

$$\boldsymbol{x}_k^* = \underset{\boldsymbol{x}\in\mathbb{X}\setminus\mathbb{X}_{k,\boldsymbol{Y}=\boldsymbol{y}_\ell}}{\text{argmax}} \; E[I(\boldsymbol{x})] = \int_{-\infty}^{J^b_{\boldsymbol{Y}=\boldsymbol{y}_\ell}} (J^b_{\boldsymbol{Y}=\boldsymbol{y}_\ell} - \hat{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x}))^+ f(\hat{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x}))dy, \tag{16}$$

where $\mathbb{X}_{k,\boldsymbol{Y}=\boldsymbol{y}_\ell}$ is the set of selected points from the offline data set, $f(\hat{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x}))$ is the density function of the predicted offline response (in this case normal with mean and variance calculated as functions of candidate solution, $\boldsymbol{x}$, using Eqs. (8) and (9)) (Criterion 1 in Algorithm 1, Step 3.1). Note that *for the second version of the algorithm*, a similar equation can be derived to calculate the EI, by simply replacing $J^b_{\boldsymbol{Y}=\boldsymbol{y}_\ell}$ with $J^b_{Y=y}$, $\hat{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x})$ with $\hat{J}_{Y=y}(\boldsymbol{x})$ and $\mathbb{X}_{k,\boldsymbol{Y}=\boldsymbol{y}_\ell}$ with $\mathbb{X}_{k,\boldsymbol{Y}=y}$ (Criterion 2 in Algorithm 1, Step 3.1). In addition to choosing the location where sampling should be performed (i.e., a candidate solution), the GOLF framework tries to determine whether expensive online sampling is required. We devise a statistical certificate of the relationship between the low and high-fidelity models, which we present in this section. For the candidate point at iteration k, $\boldsymbol{x}_k^*$, we want to certify the validity of our model in Eq. (2) representing the relationship we constructed between offline and online models. We achieve this objective with the following statistical test:

$$\begin{cases} H_0 : \hat{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x}_k^*) \sim N(\mu_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_B}(\boldsymbol{x}_k^*),\ \sigma_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_B}(\boldsymbol{x}_k^*)) \\ H_1 : \text{Otherwise} \end{cases}$$

$$\to Q = \frac{J_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x}_k^*) - \mu_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_B}(\boldsymbol{x}_k^*)}{\sigma_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_B}(\boldsymbol{x}_k^*)} \geq -Z_c. \tag{17}$$

The idea behind this test is to check whether the relationship we have assumed between offline and online models is valid at this location. If the null hypothesis in Eq. (17) cannot be rejected, there is no need for online sampling. Otherwise, online sampling needs to be performed to update the offline observations predictor using Eq. (7), and the online predictor in Eq. (10). The pseudocode for the explained procedure is presented in Algorithm 1.

4. Preliminary Results

In this section, we show some preliminary results for the learning and optimization methods to demonstrate the potential of our new framework. In particular, we first focus on the offline methods for the derivation of models from past scenarios (Secs. 3.1 and 3.2) that are the input to the online phase. Subsequently, Sec. 4.2 focuses on a proof of concept for the online optimization phase showing numerical performance of the methods in Sec. 3.4, but using a single offline scenario.

4.1. *Model learning through complexity*

In this part of the experimentation, we focus on the proposed methods for partition learning together with model estimation providing a validation of the components in

Algorithm 1

```
                Initialization
Step 0          $k = 0; \mathbb{X}_{B,k} = \boldsymbol{\emptyset}$, $\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell,k} = \boldsymbol{\emptyset}$, $J_{Y=y|\mathbb{X}_B} = \boldsymbol{\emptyset}$,
                $J_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}} = \boldsymbol{\emptyset}$, $J_{Y=y|\mathbb{X}_B} = \boldsymbol{\emptyset}$.
                Set the initial number of samples $n_{0B}$, and $n_{0\boldsymbol{Y}=\boldsymbol{y}_\ell}$, using
                a Latin hypercube design.
                Set the maximum number of online simulations $CB_{\boldsymbol{Y}=y}$;
                Set $\mathbb{X}_{B,k} \leftarrow \{\boldsymbol{x_i}\}_{i=1}^{n_{0,B}}$, update $\mathbb{X}_{LF,k} \leftarrow \{\boldsymbol{x_i}\}_{i=1}^{n_{0,LF}}$;
        While   $(CB_{Y=y} > 0)$
                Expensive update
Step 1          $\mathbb{X}_B \leftarrow \mathbb{X}_{B,k} \cup \mathbb{X}_B$;
                $\boldsymbol{J}_{Y=y|\mathbb{X}_B} \leftarrow \boldsymbol{J}_{Y=y|\mathbb{X}_B} \cup \{\boldsymbol{J}_{Y=y|\mathbb{X}_B}(\boldsymbol{x}); \boldsymbol{x} \in \mathbb{X}_{B,k}\}$,
                $\boldsymbol{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_B} \leftarrow \boldsymbol{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_B} \cup \{\boldsymbol{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_B}(\boldsymbol{x}); \boldsymbol{x} \in \mathbb{X}_{B,k}\}$;
                Evaluate $b(\boldsymbol{x}) \leftarrow (J_{\boldsymbol{Y}=y|\mathbb{X}_B}(\boldsymbol{x}) - J_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_B}(\boldsymbol{x}))$, $\boldsymbol{x} \in \mathbb{X}_{B,k}$;
                With $\boldsymbol{J}_{Y=y|\mathbb{X}_B}$, $\boldsymbol{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_B}$, $\mathbb{X}_B$ construct:
                $\hat{B} \sim \mathrm{GP}(\mu_B, \sigma_B^2)$ using (4)–(6);
                Cheap Update
Step 2          $\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell} \leftarrow \mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell,k} \cup \mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}$;
                $\boldsymbol{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_{LF}} \leftarrow \boldsymbol{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_{LF}} \cup \{J_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_{LF}}(\boldsymbol{x}); \boldsymbol{x} \in \mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell,k}\}$;
                With $\boldsymbol{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}}$, $\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}$: $\hat{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell} \sim \mathrm{GP}(\mu_{\boldsymbol{Y}=\boldsymbol{y}_\ell}, \sigma_{\boldsymbol{Y}=\boldsymbol{y}_\ell^2})$
                using (7)–(9);
        Criterion 1
Step 3.1  Compute EI($\boldsymbol{x}$) using Eq. (16), set:
          $\boldsymbol{x_k^*} \leftarrow \operatorname{argmax}_{x\in\mathbb{X}} \mathrm{EI}(\boldsymbol{x})$
          $= \int_{-\infty}^{J^b_{\boldsymbol{Y}=\boldsymbol{y}_\ell}} (J^b_{\boldsymbol{Y}=\boldsymbol{y}_\ell} - \hat{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x}))^+$
          $f(\hat{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x}))dJ$
        Criterion 2
          Compute EI$(\boldsymbol{x})$ as:
          $\boldsymbol{x_k^*} \leftarrow \operatorname{argmax}_{x\in\mathbb{X}} \mathrm{EI}(\boldsymbol{x})$
          $= \int_{-\infty}^{J^b_{\boldsymbol{Y}=y}} (J^b_{\boldsymbol{Y}=y} - \hat{J}_{\boldsymbol{Y}=y}(\boldsymbol{x}))^+$
          $f(\hat{J}_{\boldsymbol{Y}=y}(\boldsymbol{x}))dy$
Step 3.2    Evaluate $J_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(\boldsymbol{x_k^*})$;
            Update $J_{\boldsymbol{Y}=\boldsymbol{y}_\ell{}^b \leftarrow \min_{x\in\mathbb{X}_{Y=y_\ell}} \cup \mathbb{X}_B} \boldsymbol{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell}\cup\mathbb{X}_B}$, $\mathbb{X}_{\boldsymbol{Y}=\boldsymbol{y}_\ell,k+1} \leftarrow \boldsymbol{x_k^*}$,
            Certificate:
            Estimate $\hat{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell|\mathbb{X}_B}(\boldsymbol{x_k^*})$ using Eqs. (13)–(15);
            Use $J_{\boldsymbol{Y}=\boldsymbol{y}_\ell}(x_k^*)$ and $\hat{J}_{\boldsymbol{Y}=\boldsymbol{y}_\ell|X_B}(x_k^*)$ in Condition (17) holds;
            If (Condition (17) holds TRUE)
            $\mathbb{X}_{B,k+1} \leftarrow \{\}$;
            Go to Step 2
            Else
            $X_{B,k+1} \leftarrow \boldsymbol{x_k^*}$;
              , $CB_{Y=y} \leftarrow CB_{Y=y} - 1$;
              Go to Step 1
              End
            END
```

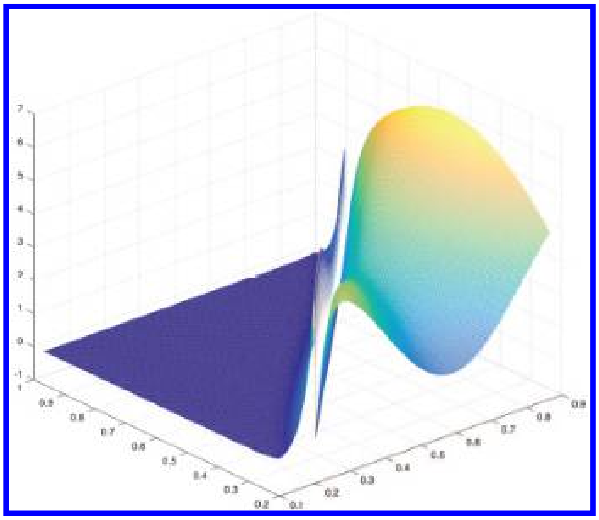


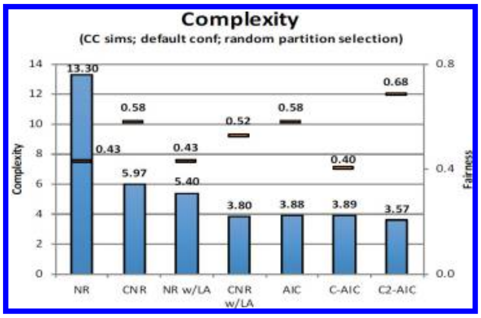
Fig. 7. Example of complex function to identify.

Secs. 3.1 and 3.2. Figure 7 represents the output from a continuous complex system. The true function, supposed to be unknown for the sake of the experiment, is:

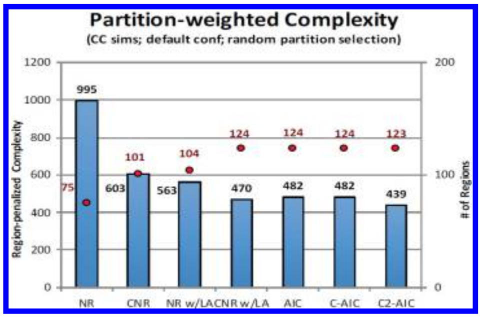
$$f(x_1, x_2) = \begin{cases} \Lambda(x_1, x_2)(e \cdot \sin(8x_1) + 5 \cdot \sin(3x_2)) & \text{If } x_1 < x_2, \\ (1 - \Lambda(x_1, x_2))(e \cdot \sin(8x_1) + 5 \cdot \sin(3x_2)) & \text{If } x_1 \geq x_2, \end{cases}$$

where $\Lambda(x_1, x_2)$ is an exponential smoothing term, which guarantees continuity at $x_1 = x_2$. Different simulation ensemble creation strategies sample the parameter space differently, thus potentially leading to different degrees of fit and model complexity. This needs to be evaluated to compare different strategies. In this section, we use the following measures of fit and complexity:

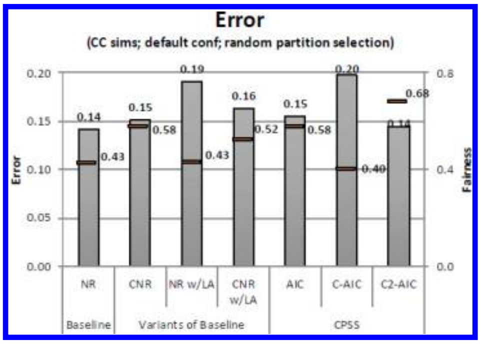
- *Error*: For each partition, we define the average error as absolute difference between the ground truth and the predicted output of the system. This difference is multiplied by the volume of the partition. Finally, all the volume-weighted errors are summed up, resulting in the error measure over the parameter space.
- *Complexity*: We also assess the complexity of the resulting models. For this purpose, we first compute the complexity for each partition using the rules in Table 1. Then the obtained value is multiplied by the volume of the partition. The volume-weighted values are summed across partitions resulting in final complexity measure. To ensure that the sampling strategies do not return a very large number of low-complexity models, we also use a partition weighted complexity measure, $p_{\text{complexity}}$, obtained by multiplying the overall complexity by the number of partitions resulting from the sampling strategy: low $p_{\text{complexity}}$ values indicate a small number of partitions with low complexities.



(a)



(b)



(c)

Fig. 8. Preliminary results on sampling based on different criteria. (a) Obtained surface complexity and fairness with partitioning based on R^2, complexity-driven R^2, complexity driven R^2 with look-ahead partitioning, AIC, complexity driven AIC and look-ahead complexity driven AIC. (b) Obtained region complexity and number of regions with partitioning based on R^2, complexity-driven R^2, complexity driven R^2 with look-ahead partitioning, AIC, complexity driven AIC and look-ahead complexity driven AIC. (c) Error obtained for the different algorithms.

- *Fairness*: This metric is designed to make sure that the fitness or complexity are uniform across the parameter space, formally: fairness $= \frac{\mathcal{P}}{\sum_{\mathcal{P}_i \in \mathcal{P}} \exp(\rho_i)}$, where ρ_i is the error (or complexity) value (as defined above) for the partition, $\mathcal{P}_i \in \mathcal{P}$. This formula is analogous to the well-known *f-score* measure (Selçuk Candan and Maria, 2010), and it returns a high score when *all* the partitions have similarly high fits (or low complexities).

Figure 8 shows preliminary results for the alternative sampling strategies, including purely fit-based, AIC-based, and axiomatic (complexity driven). In particular, $\mathcal{C}$-AIC replaces the number of coefficients in the AIC equation with the complexity function $\mathcal{C}$ (Table 1), and $\mathcal{C}$2-AIC replaces them with $(\mathcal{C} \cdot r)$ where r is the size of the region corresponding to each sample.

Figures 8(a)–8(c) show the *complexity*, $p_{\text{complexity}}$, and *error* results for different penalty measures and split strategies: (a) normalized R^2 (NR) as the baseline; and three variants of the baseline: (b) complexity-guided NR (CNR); (c) NR with look-ahead; (d) CNR with look-ahead; the proposed GOLF complexity-guided parameter space sampling algorithm (CPSS in Fig. 8(c)) using: (e) AIC; (f) complexity-guided AIC (C-AIC); and (g) complexity-guided and coverage-weighted AIC (C2-AIC). As we see in Fig. 8(a), the baseline (pure normalized R^2 (NR) based approach) leads to a very high model complexity. Combining NR with complexity-guidance and look-ahead based split (CNR w/LA) significantly reduces model complexity and improves fairness. We observe that the last three approaches (AIC, C-AIC, and C2-AIC), where we apply rank-stability and look-ahead by default, match the complexity performance of the CNR w/LA. As expected, we obtain the best complexity and fairness outcomes using C2-AIC, which leverages complexity-guided and coverage-weights, along with rank-stability and look-ahead for split decisions. Figure 8(b) shows how, also in terms of the partition-weighted complexity ($p_{\text{complexity}}$), the complexity-guided and coverage-weighted strategies with rank-stability and look-ahead, lead to better models, despite the fact that the number of resulting partitions is generally higher than the pure NR strategy.

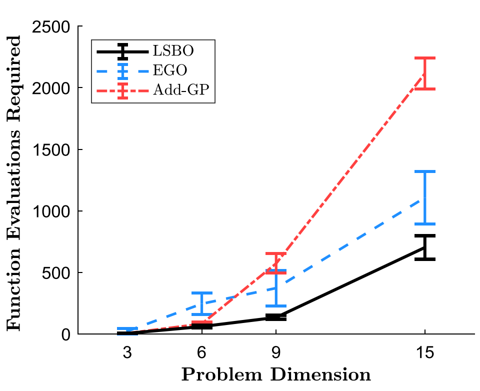
Figure 8(c) shows the impact of the different strategies on the degrees of fitness of the resulting models. In particular, the pure NR strategy leads to a low error rate (0.14); however, it also leads to low (0.43) error fairness. It is interesting to observe how the low error rate, 0.14, is also matched by the C2-AIC strategy and that C2-AIC also provides a significantly higher degree of fairness (0.68), indicating that, the proposed strategies are also effective in terms of the goodness of fit of the resulting models.

Overall, Fig. 8 reports promising results showing that, when coupled with appropriate quality measures, sequential sampling can lead to improved fit, while allowing for increased explanatory power.

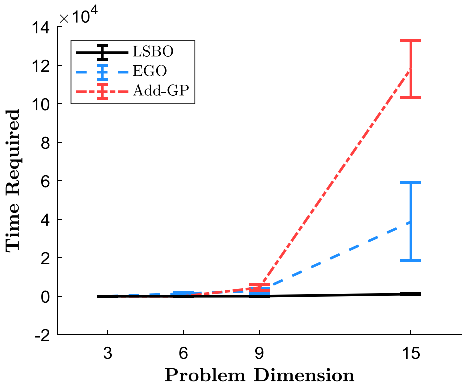
4.2. *Slicing and projection mechanism for offline LSBO*

In this section, we show the validation of the method for slicing and projection in high-dimensional scenarios space presented in Sec. 3.3. The problem of black box optimization in large dimensions has recently attracted the attention from several authors and, herein, we compare our slicing and projection through decomposition with state of the art algorithms: REMBO (Wang *et al.*, 2013), and the Bayesian Optimization (BO) with Additive GP recently proposed in Wang *et al.* (2017). Existing high-dimensional BO paradigms aim to address two issues: (a) identifying a low-dimensional structure (LDS) and (b) efficiently optimizing over an acquisition function (AF). REMBO, for instance, assumes there is an effective low dimensionality that can be specified and randomly embedded into, to accomplish both of these. Similarly, additive Gaussian process BO (add-GP) aims to learn the LDS through statistical modeling techniques such that a decomposition of the objective function can be exploited to efficiently optimize the AF. The proposed LSBO method, goes beyond these by providing additional degrees of freedom towards exploiting hidden problem structures, thus, offering a shift from high-dimensional objective function projection and AF likelihood decomposition, to low-dimensional BO/AF optimization with exact sampling. This leads to computationally tractable AF optimization even when rich problem structure is present, an inherent trade-off in the current paradigm. While not explored in the current submission, this further enables, decomposition methods (such as dependency/factor graph learning or Bayesian learning via Gibbs sampling/structured kernels) to identify appropriate subspace partitions and communication patterns.

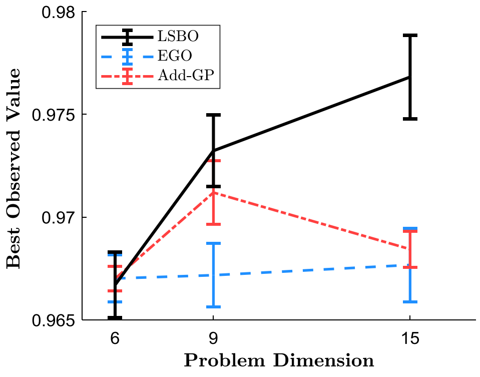
Scalability of LSBO against Competitors. Figures 9(a) and 9(b) above clearly show that LSBO is the best competitor both in terms of the number of function evaluations needed and the computational work to achieve a desired function target



(a)



(b)



(c)

Fig. 9. Performance of LSBO on several benchmark problems. (a) Average function evaluations to observe a function value of 0.01 or less, ±2 standard errors, over 50 replications. (b) Average wall clock time to observe a function value of 0.01 or less, ±2 standard errors, over 50 replications. (c) Average cancer detection performance of best trained neural net, ±2 standard errors, with constrained wall clock.

accuracy (i.e., less than 0.01 when the true global minimum is 0). The results also confirm that LSBO scales significantly better than the competitors: the accuracy and execution time gains provided by LSBO increases as the problem dimensionality increases. LSBO's computational gain primarily comes from space decomposition that, by construction, leads the algorithm to perform only low-dimensional optimizations. Note, when the additive assumption fails, the computational effort required to solve for the likelihood function increases dramatically for Add-GP, as seen in Fig. 9(b).

Application of SCOOP in Neural Network Hyper-Parameter Search. We test the application of LSBO in the construction neural nets to classify cancer data. Pattern neural nets were trained with Matlab's neural network toolbox, and publicly available classification data (https://github.com/zi-w/Structural-Kernel-Learning-for-HDBBO). As competitors, LSBO, EGO, and Add-GP were tested for varying dimensional versions of the neural net search problem, with decision variables being the number of nodes in each layer and the number of decision variables (dimension) corresponding to the number of layers. A wall clock time constraint was implemented for each problem dimensionality encompassing both neural net training and algorithm search times (600, 1,800, and 3,600 s for 6, 9, and 15 dimensions, respectively).

Figure 9 illustrates the average best accuracy achieved by each algorithm across 20 replications. Two standard error confidence intervals are reported. The results show that LSBO significantly outperforms both EGO and Add-GP in terms of solution accuracy, as the number of problem dimensions increases. Note that Add-GP shows non-monotonic performance behavior: an increase in the number of problem dimensions (layers in the neural net) does not necessarily imply that the best observed value will also increase. This is due to the exponential increase in the time Add-GP requires in structured kernel learning (as highlighted by Fig. 2), which consumes significantly more time for deciding hyper-parameters of the neural net and leaves significantly less time for actual training of the classifier with the chosen hyper-parameters.

4.3. *Generalized ordinal learning with single offline scenario at a time*

In this section, both versions of the proposed algorithm for the online learning phase (Sec. 3.4) will be evaluated using the Efficient Global Optimization (EGO) method which will be used as the benchmark to demonstrate the benefit of accounting for offline information. In fact, EGO only considers online generated data. First, the test function and different offline models of interest are introduced; then, the numerical experiments and results are discussed. For the test function, as the optimal solution is known, the algorithm stops whenever at least one of the following conditions is met: (1) $\frac{J^b_{Y=y}-J^*_{Y=y}}{J^*_{Y=y}} \leq 0.01$; (2) $\text{Itr}_{Y=y} >= 500$; (3) $\text{Itr}_{Y=y} >= 50$. Here $\text{Itr}_{Y=y_\ell}$ and $\text{Itr}_{Y=y}$ are the number of offline and online samples, respectively. The function

Table 2. Different offline-data driven models.

Offline model	Model form	Correlation with the true model
$J_{Y=\boldsymbol{y_1}}(\boldsymbol{x})$	$-2\prod_{i=1}^{d}\sin(\pi x_i)$	0.87
$J_{Y=\boldsymbol{y_2}}(\boldsymbol{x})$	$-0.8\prod_{i=1}^{d}\sin(5\pi x_i)$	0.49
$J_{Y=\boldsymbol{y_3}}(\boldsymbol{x})$	$2\prod_{i=1}^{d}\sin(\pi x_i)$	−0.87
$J_{Y=\boldsymbol{y_4}}(\boldsymbol{x})$	$0.8\prod_{i=1}^{d}\sin(5\pi x_i)$	−0.49

$J_{Y=y}(\boldsymbol{x}, \boldsymbol{y})$ below will be considered as the true model where $x_i \in [0.1, 1]$ (Zabinsky *et al.*, 2010)

$$J(\boldsymbol{x}, \boldsymbol{Y} \mid \boldsymbol{Y} = \boldsymbol{y}) = y_1 \prod_{i=1}^{d} \sin(\pi x_i) - y_2 \prod_{i=1}^{d} \sin(5\pi x_i). \tag{18}$$

With $\boldsymbol{y} = \begin{bmatrix} y_1 = -2.5 \\ y_2 = -1.0 \end{bmatrix}$. Table 2 reports four different offline generated models that we used along with their correlation coefficient characterizing their dependency with the true, unknown, model.

The proposed algorithm in its two versions along with the benchmark EGO (Jones *et al.*, 1998) are used to optimize the test function introduced above and their performance is compared. EGO only samples in high-fidelity space and works independently of the low-fidelity model of choice to be used in the proposed multi-fidelity algorithm. All algorithms were run for 50 macro-replications and we considered, as performance metric, the relative distance between the reported optimal solution and the real optimal solution of the true model, i.e., $\frac{\|\hat{x}_i^* - \boldsymbol{x}^*\|}{\|\boldsymbol{x}^*\|}$, where $\boldsymbol{x}^*$ is the global optimal solution of the high-fidelity model and $\hat{\boldsymbol{x}}_i^*$ is the optimal solution reported by the algorithm at the ith macro-replication. We also report the number of online simulations ran before convergence/stop of the algorithms over different macro-replications as "Online Simulations" in both Tables 3 and 4.

Table 3. Statistics on optimality distance for the first version of proposed algorithm.

Low-fidelity model	Dimension	Proposed algorithm				EGO	
		$\frac{\|\hat{x}_i^* - \boldsymbol{x}^*\|}{\|\boldsymbol{x}^*\|}$		#Online simulations		$\frac{\|\hat{x}_i^* - \boldsymbol{x}^*\|}{\|\boldsymbol{x}^*\|}$	
		Average	Std err	Average	Std err	Average	Std err
$J_{Y=\boldsymbol{y_1}}(\boldsymbol{x})$	3	0.07	0.01	2.28	0.17	0.17	0.02
	4	0.04	0.01	2.76	0.23	0.16	0.02
$J_{Y=\boldsymbol{y_2}}(\boldsymbol{x})$	3	0.00	0.00	2.38	0.11	0.17	0.01
	4	0.00	0.00	2.30	0.10	0.16	0.02
$J_{Y=\boldsymbol{y_3}}(\boldsymbol{x})$	3	0.09	0.01	3.18	0.28	0.15	0.01
	4	0.05	0.01	2.86	0.29	0.14	0.01
$J_{Y=\boldsymbol{y_4}}(\boldsymbol{x})$	3	0.00	0.00	2.38	0.11	0.17	0.01
	4	0.00	0.00	2.42	0.16	0.16	0.02

Table 4. Statistics on optimality distance for the second version of proposed algorithm.

Offline model	Dimension	Proposed algorithm				EGO	
		$\frac{\|\hat{x}_i^* - \boldsymbol{x}^*\|}{\|\boldsymbol{x}^*\|}$		#Online simulations		$\frac{\|\hat{x}_i^* - \boldsymbol{x}^*\|}{\|\boldsymbol{x}^*\|}$	
		Average	Std err	Average	Std err	Average	Std err
$J_{Y=\boldsymbol{y_1}}(\boldsymbol{x})$	3	0.00	0.00	2.02	0.11	0.18	0.01
	4	0.00	0.00	2.08	0.07	0.17	0.02
$J_{Y=\boldsymbol{y_2}}(\boldsymbol{x})$	3	0.13	0.02	15.02	0.97	0.10	0.02
	4	0.16	0.01	31.86	1.21	0.03	0.01
$J_{Y=\boldsymbol{y_3}}(\boldsymbol{x})$	3	0.00	0.00	2.02	0.11	0.18	0.01
	4	0.00	0.00	2.60	0.19	0.15	0.02
$J_{Y=\boldsymbol{y_4}}(\boldsymbol{x})$	3	0.12	0.02	14.78	1.02	0.09	0.02
	4	0.17	0.01	31.82	1.43	0.04	0.01

For each macro-replication, we ran the proposed algorithms and EGO with the same random number seed. Upon observing convergence of our algorithm, we also stopped EGO, i.e., EGO could run as many online simulations as those needed by the GOLF counter-part to converge. It is important to highlight how EGO, as any traditional optimization tool, only uses the online generated information to take a decision, while the GOLF algorithms uses a large amount of offline evaluations of the functions $J_{Y=\boldsymbol{y_i}}(\boldsymbol{x}), i = 1, 2, 3, 4$. Table 3 shows the results obtained from the first version of the algorithm that calculates EI over the predicted online (true) response. From the results, it can be observed that our approach outperforms EGO no matter which offline model is used. *This leads us to arguing that it is compelling to implement the version of the algorithm with multiple offline models to maximize the performance.*

Also, the gap between the performance of the first version of the algorithm and EGO is larger when second and fourth offline models are used. While the reason for this result is highly related to the specific sampling criteria, it reveals that model correlation may not be the only important metric to characterize the relevance of offline information. In fact, apparently the non-linear dependency of the high-frequency component and the original function is helping our search procedure.

Table 4 shows the results from the second version of the algorithm, which calculates the EI over the predicted offline model. Analyzing the results, it can be observed that, while the algorithm still outperforms EGO when the offline model is highly correlated with the true model, it fails when correlation is weak. While optimizing according to solely the offline information can have advantages when the offline model has optimization-amenable characteristics, relying too much on offline data will slow down the search especially when the offline model and data are “bad”, i.e., not informative of the real system. This importantly brings to light the need to have good offline data.

Considering both Tables 3 and 4, we can see how, with “good” offline models, the *second version of the algorithm outperforms the first version*, in that it requires a smaller number of online simulations on average. However, with less correlated low-fidelity models, it is the *first version of the algorithm that does a better job of guiding the search* towards optimal solution to the true model. This suggests that a *switching mechanism* between the two criteria based on local correlation between the models within the algorithm may be effective in more complex cases.

5. Conclusions and Future Research

We propose the GOLF framework to support real-time decision making for complex systems in the presence of large data coming from both the sensing system as well as from complex simulation models. GOLF separates the problem into an offline phase where novel methods take the perspective of learning from past data with the aid of simulation in the attempt to construct reliable models to explain past observations. We first propose a novel complexity driven perspective to guide sampling and model generation and preliminary results show the effectiveness of the novel class of algorithms in identifying appropriate models "for the past" in non-smooth high-dimensional set ups. The data and models become input information to online GOLF, where a novel ordinal learning framework is proposed to leverage on past offline generated data and quickly provide a scheme for additional sampling that provides the solution for the scenario of interest.

The preliminary implementation of the several components of GOLF motivates us to further explore GOLF. Specifically, several opportunities are currently being investigated in applications such as smart manufacturing as well as Cyber Physical Systems at large (e.g., self-driven vehicles, water networks). The complexity driven framework still relies on important assumptions on the type of models to be included as well as on the way to assign complexity invariants. Both aspects are under analysis and generalization is required to fully exploit GOLF. Concerning the ordinal learning online GOLF phase, the ability to handle several scenarios simultaneously as well as investigating innovative mapping techniques.

Acknowledgments

The research in this paper was partially supported by Grants NSF#1829238 "Exploring Discrete Event Dynamics to Model and Control Intelligent Manufacturing Systems"; NSF#1827757 "Data-Driven Services for High Performance and Sustainable Buildings", NSF#1610282 "DataStorm: A Data Enabled System for End-to-End Disaster Planning and Response", NSF#1633381 "Discovering Context-Sensitive Impact in Complex Systems", and "FourCmodeling": EUH2020 Marie Sklodowska-Curie Grant Agreement No. 690817. Also, this work has been supported in part by National Science Foundation Under Awards NSF#1462409 and NSF#1462787 and National Natural Science Foundation of China Under Grant 71720107003 and 61603321, and the Ministry of Education, Singapore (Grant MOE2015-T2-2-005).

References

Alrefaei, MH and S Andradóttir (2001). A modification of the stochastic ruler method for discrete stochastic optimization. *European Journal of Operational Research*, 133, 160–182.

Ankenman, B, BL Nelson and J Staum (2010). Stochastic kriging for simulation metamodeling. *Operations Research*, 58, 371–382.

Benamara, T, P Breitkopf, I Lepot and C Sainvitu (2016). Adaptive infill sampling criterion for multi-fidelity optimization based on Gappy-POD. *Structural and Multidisciplinary Optimization*, 1–13.

Bertsekas, DP (2008). *Approximate Dynamic Programming.*

Brooks, SH (1958). A discussion of random methods for seeking maxima. *Operations Research*, 6, 244–251.

Burnham, KP and DR Anderson (2004). Multimodel inference: Understanding AIC and BIC in model selection. *Sociological Methods & Research*, 33, 261–304.

Chen, CH, J Lin, E Yücesan and SE Chick (2000). Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems*, 10, 251–270.

Chen, R, J Xu, C-H Chen and L-H Lee (2015). An effective learning procedure for multi-fidelity simulation optimization with ordinal transformation. In *Proc. 2015 IEEE Conf. Automation Science and Engineering*, pp. 702–707, Gothenburg, Sweden.

Chen, CH and LH Lee (2010). *Stochastic Simulation Optimization*: *An Optimal Computing Budget Allocation*, World Scientific Publishing Co.

Feo, TA and MG Resende (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6, 109–133.

Forrester, AI, A Sóbester and AJ Keane (2007). December. Multi-fidelity optimization via surrogate modeling. In *Proeedings of the Royal Society of London A*: *Mathematical, Physical and Engineering Sciences*, Vol. 463, No. 2088, pp. 3251–3269. The Royal Society.

Fu, MC (ed.) (2015). *Handbook of Simulation Optimization*, Vol. 216. New York: Springer.

Goel, T, RT Haftka, W Shyy and NV Queipo (2007). Ensemble of surrogates. *Structural and Multidisciplinary Optimization*, 33, 199–216.

Hsieh, L, E Huang, S Zhang, KH Chang and CH Chen (2016). Application of multi-fidelity simulation modeling to integrated circuit packaging. *International Journal of Simulation and Process Modeling*, 28, 195–208.

Hsieh, L, E Huang and CH Chen (2017). Equipment utilization enhancement in photolithography area through a dynamic system control using multi-fidelity simulation optimization with big data technique. *IEEE Transactions on Semiconductor Manufacturing*, 30, 166–175.

Inanlouganji, A, G Pedrielli, G Fainekos and Pokutta (2018). Continuous simulation optimization with model mismatch using gaussian process regression. Submitted to *Winter Simulation Conference.*

Jerri, AJ (1977). The Shannon sampling theorem — Its various extensions and applications: A tutorial review. *Proceedings of the IEEE*, 65, 1565–1596.

Jones, DR, M Schonlau and WJ Welch (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13, 455–492.

Kandasamy, K, G Dasarathy, J Oliva, J Schneider and B Poczos (2016). Gaussian process optimisation with multi-fidelity evaluations. In *NIPS.*

Kang, Y, L Mathesen, G. Pedrielli and F Ju (2017). Multi-fidelity modeling for analysis of serial production lines. In *2017 IEEE CASE Conference.*

Kleijnen, JPC (2015). Regression and Kriging metamodels with their experimental designs in simulation: Review.

Le Gratiet, L and C Cannamela (2015). Cokriging-based sequential design strategies using fast cross-validation techniques for multi-fidelity computer codes. *Technometrics*, 57, 418–427.

Li, Y, SH Ng, M Xie and TN Goh (2010). A systematic comparison of metamodeling techniques for simulation optimization in decision support systems. *Applied Soft Computing*, 10, 1257–1273.

Li, J, W Liu, G Pedrielli, LH Lee and EP Chew (2017). Optimal computing budget allocation to select the non-dominated systems — A large deviations perspective. *IEEE Transactions on Automated Controls*.

Liu, B, S Koziel and Q Zhang (2016). A multi-fidelity surrogate-model-assisted evolutionary algorithm for computationally expensive optimization problems. *Journal of Computational Science*, 12, 28–37.

Locatelli, M (1997). Bayesian algorithms for one-dimensional global optimization. *Journal of Global Optimization*, 10, 57–76.

Mathesen, L, G Pedrielli and SH Ng (2017). Trust region based stochastic optimization with adaptive restart: A family of global optimization algorithms. In *2017 Winter Simulation Conference* (*WSC*). IEEE (*To Appear*).

Meng, Q and SH Ng (2015). An additive global and local Gaussian process model for large data sets. In *Proceedings of 2015 Winter Simulation Conference*, IEEE Press.

Min, C (2017). Multi-fidelity optimization with Gaussian regression on ordinal transformation space. Master dissertation.

Muller, J and R Piche (2011). Mixture surrogate models based on Dempster-Shafer theory for global optimization problems. *Journal of Global Optimization*, 51, 79–104.

Myers, RH and CM Anderson-Cook (2009). *Response Surface Methodology*: *Process and Product Optimization Using Designed Experiments*, Vol. 705. John Wiley & Sons.

Osorio, C and KK Selvam (2017). Simulation-based optimization: Achieving computational efficiency through the use of multiple simulators. *Transportation Science*, 51, 395–411.

Ozdogan, H (1987). Model selection and Akaike's information criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, 52, 345–370.

Pedrielli, G and SH Ng (2015). eTSSO: Adaptive Search Method for Stochastic Global Optimization Under Finite Budget. Working paper.

Pedrielli, G *et al.* Empirical analysis of the performance of variance estimators in sequential single-run ranking selection: The case of time dilation algorithm. WSC'16.

Pedrielli, G and SH Ng (2016, December). G-star: A new kriging-based trust region method for global optimization. In *2016 Winter Simulation Conference* (*WSC*). IEEE.

Pedrielli, G and SH Ng (2015, December). Kriging-based simulation-optimization: A stochastic recursion perspective. In *2015 Winter Simulation Conference* (*WSC*), pp. 3834–3845. IEEE.

Pesaran, MH and RJ Smith (1994). A generalized R^2 criterion for regression models estimated by the instrumental variables method. *Econometrica*, 62, 705–710.

Powell, WB (2007). *Approximate Dynamic Programming*: *Solving the Curses of Dimensionality*, Vol. 703. John Wiley and Sons.

Quan, N, J Yin, SH Ng and LH Lee (2013). Simulation optimization via kriging: A sequential search using expected improvement with computing budget constraints. *Iie Transactions*, 45, 763–780.

Ryzhov, IO, PI Frazier and WB Powell (2010). On the robustness of a one-period look-ahead policy in multi-armed bandit problems. *Procedia Computer Science*, 1, 1635–1644.

Santner, TJ, BJ Williams and WI Notz (2013). *The Design and Analysis of Computer Experiments*. Springer Science & Business Media.

Selçuk Candan, K and LS Maria (2010). *Data Management for Multimedia Retrieval*, Cambridge University Press.

Si, J, AG Barto, WB Powell and D Wunsch (Eds.) (2004). *Handbook of Learning and Approximate Dynamic Programming*, Vol. 2. John Wiley and Sons.

Solis, FJ and RJB Wets (1981). Minimization by random search techniques. *Mathematics of Operations Research*, 6, 19–30.

Sutton, RS and AG Barto (2018). *Reinforcement Learning: An Introduction*. MIT Press.

Ulaganathan, S, I Couckuyt, F Ferranti, E Laermans and T Dhaene (2015). Performance study of multi-fidelity gradient enhanced kriging. *Structural and Multidisciplinary Optimization*, 51, 1017–1033.

Viana, FA, TW Simpson, V Balabanov and V Toropov (2014). Special section on multidisciplinary design optimization: Metamodeling in multidisciplinary design optimization: How far have we really come? *AIAA Journal*, 52, 670–690.

Wang, Z, M Zoghi, F Hutter, D Matheson and N De Freitas (2013). Bayesian optimization in high dimensions via random embeddings. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Wang, Z, C Li, S Jegelka and P Kohli (2017). Batched high-dimensional Bayesian optimization via structural kernel learning. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70, pp. 3656–3664, JMLR.org.

West, SG, BT Aaron and W Wei (2012). Model fit and model selection in structural equation modeling. In *Handbook of Structural Equation Modeling*, pp. 209–231.

Wong, HS, TJ Chin, J Yu and D Suter (2011 November). Dynamic and hierarchical multi-structure geometric model fitting. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pp. 1044–1051. IEEE.

Xinsheng, L, K Selcuk Candan and ML Sapino (2018). M2TD: Multi-task tensor decomposition for sparse ensemble simulations. Submitted to ICDE 2018.

Xu, J, S Zhang, E Huang, CH Chen, LH Lee and N Celik (2016a). Mo2tos: Multi-fidelity optimization with ordinal transformation and optimal sampling. *Asia-Pacific Journal of Operational Research*, 33, 1650017.

Xu, J, S Zhang, CC Huang, CH Chen, LH Lee and N Celik (2014). An ordinal transformation framework for multi-fidelity simulation optimization. In *Proc. 2014 IEEE Conf. Automation Science and Engineering*, pp. 385–390, Taipei, Taiwan, 2014.

Xu, J, E Huang, CH Chen and LH Lee (2015). Simulation optimization: A review and exploration in the new era of cloud computing and big data. *Asia-Pacific Journal of Operational Research*, 32.

Xu, J, E Huang, L Hsieh, LH Lee, QS Jia and CH Chen (2016b). Simulation optimization in the era of Industrial 4.0 and the Industrial internet. *Journal of Simulation*, 10, 310–320.

Yamada, M, J Chen and Y Chang (2018). *Transfer Learning: Algorithms and Applications*. Morgan Kaufmann.

Yan, D and H Mukai (1992). Stochastic discrete optimization. *SIAM Journal on Control and Optimization*, 30, 594–612.

Yin, J, SH Ng and KM Ng (2011). Kriging metamodel with modified nugget-effect: The heteroscedastic variance case. *Computers & Industrial Engineering*, 61, 760–777.

Yu-Ru, L, K Selçuk Candan, H Sundaram and X Lexing (2011). SCENT: Scalable compressed monitoring of evolving multirelational social networks. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 7(Suppl.), 29.

Zabinsky, ZB, RL Smith, JF McDonald, HE Romeijn and DE Kaufman (1993). Improving hit-and-run for global optimization. *Journal of Global Optimization*, 3, 171–192.

Zabinsky, ZB, B David and K Charoenchai (2010). Stopping and restarting strategy for stochastic sequential search in global optimization. *Journal of Global Optimization*, 46.2, 273–286.

Zabinsky, ZB and RL Smith (1992). Pure adaptive search in global optimization. *Mathematical Programming*, 53, 323–338.

Zhang, S, J Xu, LH Lee, EP Chew, WP Wong and CH Chen (2017). Optimal computing budget allocation for particle swarm optimization in stochastic optimization, *IEEE Transactions on Evolutionary Computation*, 21, 206–219.

Zhang, S, J Xu, E Huang and CH Chen (2016a). A new optimal sampling rule for multifidelity optimization via ordinal transformation. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 670–674, IEEE.

Biography

Giulia Pedrielli is currently an Assistant Professor in the School of Computing Informatics and Decision Systems Engineering at Arizona State University. Her research activity is in the field of stochastic simulation of complex systems and simulation optimization. Her focus is on multi-model large scale optimization applied to control of time varying systems. She is in the Editorial Board for the Journal of Simulation and the Journal of Flexible Services and Manufacturing.

K. Selcuk Candan is a Professor of computer science and engineering at Arizona State University and the director of ASU's Center for Assured and Scalable Data Engineering (CASCADE). His primary research interest is in the area of management and analysis of non-traditional, heterogeneous, and imprecise data. He is currently in the editorial boards of the ACM Transactions on Database Systems, IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Cloud Computing, and the Journal of Multimedia. He has served in the organization and program committees of various scientific conferences.

Xilun Chen is a PhD candidate in Computer Science at Arizona State University. His research field is largescale data mining, Personalized Recommender Systems, and data management. His research focuses on large scale multi-layer spatio-temporal ensemble models estimations with applications in epidemiology.

Logan Mathesen is a PhD student in industrial engineering at Arizona State University. His research is in design and analysis of black-box optimization methods, with applications in cyber-physical system testing and falsification.

Alireza Inanalouganji is a PhD student in industrial engineering at Arizona State University. His research is in design and analysis of black-box optimization methods, with applications in power and water networks with a focus on disaster management.

Jie Xu is Associate Professor of Systems Engineering and Operations Research at George Mason University. He received his PhD degree in Industrial Engineering and Management Sciences from Northwestern University. His research interests include Monte Carlo simulation, simulation-based optimization, computational intelligence, and applications in risk management and aviation.

Chun-Hung Chen is a Professor of Systems Engineering and Operations Research at George Mason University. He received his PhD degree from Harvard University in 1994. He served as Co-Editor of the Proceedings of the 2002 Winter Simulation Conference and Program Co-Chair for 2007 INFORMS Simulation Society Workshop. He has served on the editorial boards of IEEE Transactions on Automatic Control, IEEE Transactions on Automation Science and Engineering, IIE Transactions, Journal of Simulation Modeling Practice and Theory, and International Journal of Simulation and Process Modeling. He is a Fellow of IEEE.

Loo Hay Lee is an Associate Professor with the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore. He is also appointed by the Shanghai Municipal Education Commission as an Eastern Scholar Professor for the Shanghai Maritime University. Dr. Lee has served as an Associate Editor for the IEEE Transactions on Automatic Control, IIE Transactions, and the IEEE Transactions on Automation Science and Engineering. He is currently the Co-Editor for the Journal of Simulation and is a Member in the advisory board for OR Spectrum.