JOURNAL OF SIMULATION
https://doi.org/10.1080/17477778.2022.2046520

ORIGINAL PAPER

Real-time digital twin-based optimization with predictive simulation learning

Travis Goodwin[a], Jie Xu[a], Nurcin Celik [b] and Chun-Hung Chen[a]

[a]Department of Systems Engineering & Operations Research, George Mason University, Fairfax, VA, USA; [b]Department of Industrial Engineering, The University of Miami, Coral Gables, FL, USA

ABSTRACT

Digital twinning presents an exciting opportunity enabling real-time optimization of the control and operations of cyber-physical systems (CPS) with data-driven simulations, while facing prohibitive computational burdens. This paper introduces a method, Sequential Allocation using Machine-learning Predictions as Light-weight Estimates (SAMPLE) to address this computational challenge by leveraging machine learning models trained off-line in a predictive simulation learning setting prior to a real-time decision. SAMPLE integrates machine learning predictions with data generated by real-time execution of a digital twin in a rigorous yet flexible way, and optimally guides the digital twin simulation to achieve the computational efficiency required for real-time decision-making in a CPS. Numerical experiments demonstrate the viability of SAMPLE to select optimal decisions in real-time for CPS control and operations, compared to those of using only machine learning or simulations.

ARTICLE HISTORY
Received 13 September 2021
Accepted 13 February 2022

KEYWORDS
Digital twin; real-time simulation optimisation; predictive simulation learning; machine learning; cyber-physical systems

1. INTRODUCTION

Recently, the digital twin concept has attracted a tremendous amount of attention from researchers and practitioners across a wide spectrum of scientific and engineering disciplines. One particularly exciting future application of the digital twin concept is the potential to conduct real-time optimisation of the control and operations of a cyber-physical system (CPS) with data-driven simulations.

One such application example is the real-time resource allocation to mitigate disruptions on a system's components as illustrated in Figure 1. When protecting k components from fast approaching disruptions, the decision-maker only has a limited amount of time to determine how M mitigation measures will be allocated to protect components once the disruptions are observed. An example of such a problem is illustrated when managing power grid against disturbances such as an incoming storm system, electricity demand surges, or intermittent generations (Bastani, Damgacioglu et al., 2018; Bastani, Thanos et al., 2018; Damgacioglu & Celik, 2022; Darville & Celik, 2020a, 2020b; Shi & Celik, 2016; Xu et al., 2020; Yavuz et al., 2020). Here, one mitigation measure is the power utility company crew teams, which will be allocated to protect k transmission lines. Given a digital twin that is designed to determine the outcome of the disruptions for a given allocation of mitigation measures, the goal of real-time digital twin-based decision-making would be to select the optimal allocation of mitigation measures from amongst all possible allocations, e.g., to minimise load shedding caused by the storm system.

In this paper, we emphasise the importance of *context* in digital twin-based optimisation, which refers to a collection of both external and internal events, covariates, and states under which a decision is being evaluated. For example, for the problem of determining the optimal mitigation measure allocation of a power grid to improve its weather resilience, weather forecast, electricity demand, and power generation capabilities, etc., are all part of the context under which the mitigation decision is made. Obviously, different contexts will lead to different optimal decisions, making the optimal decision *context dependent*. In fact, the hallmark characteristic of a digital twin is to dynamically assimilate various sensor data into a computational simulation model to reflect the current state of the system and predict the outcomes of decision alternatives.

In principle, once a context is observed, an optimisation problem can then be formulated under the context and solved via simulation optimisation, which is an emerging optimisation methodology that works directly with a stochastic black-box function, e.g., a digital twin. Simulation optimisation has already been used in a wide spectrum of engineering, scientific, and societal applications. Examples include production planning in manufacturing (Calverley et al., 2021; Cao et al., 2021; Hsieh et al., 2007; Pickardt et al., 2010; Song et al., 2019; Zhang et al., 2020), operational planning of power systems (Thanos et al., 2015; Xu

CONTACT Jie Xu jxu13@gmu.edu Department of Systems Engineering & Operations Research, Fairfax, VA 22030, USA
© Operational Research Society 2022.

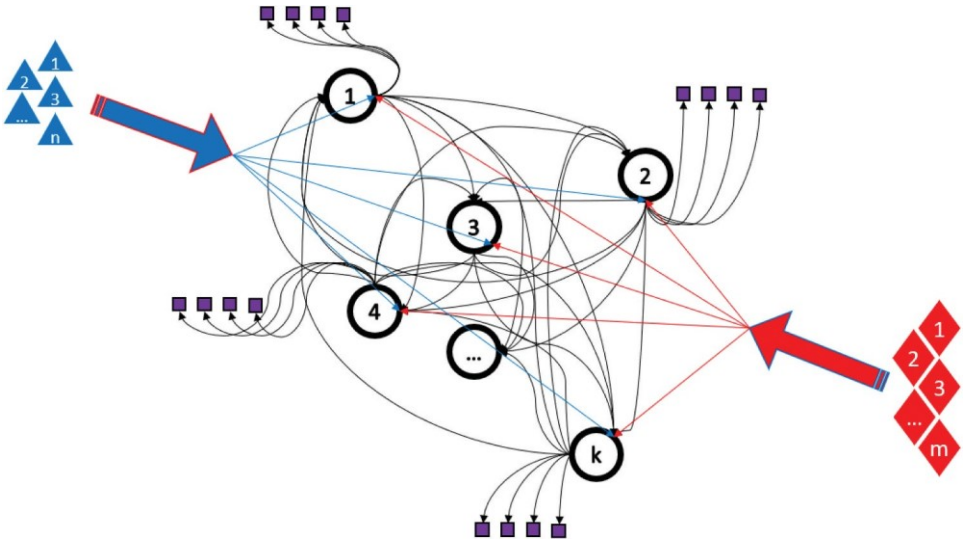

Figure 1. A system disruption mitigation problem with n mitigation measures (blue triangles) to be allocated to protect k interconnected components (black circles) upon observing a disruption vector with m disruptions (red diamonds).

et al., 2020; Yavuz et al., 2020), resource allocation in healthcare and other service systems (T. T. Chen & Wang, 2016; Kasaie & Kelton, 2013; Qiu & song, 2016), and transportation (Zhou et al., 2021).

Before the advent of digital twin, simulation optimisation methodologies were intended to optimise the *planning and design* of a system. Data sets were collected over a period of time and compiled to provide views on the uncertainty in systems' operations and environments, and provide input models, often together with some other expert inputs, for the simulation. The simulation optimisation algorithm then selects a decision that achieves the best performance as measured against the input models. The concept of *context* is typically not explicit and instead encapsulated into the simulation model, representing the belief that the simulation model will internally reflect the future state forecast. Considering the uncertainty in the future, some perturbations to the input models may be added to help understand how the system's performance may change when the future is different from what the curated input models specify. One representative example is the simulation study of the development of drug resistance among parasites by Xu et al. (2014), where many genetic, epidemiological, and climatic parameters were determined based on historical data and past research results, but were perturbed to account for uncertainty in these parameters as well as difficult to predict future changes.

In contrast, an anticipated main usage of simulation optimisation in future digital twin applications is to respond in *real-time* to changes or disturbances in system's operational states or external events, or in other words, to find the optimal decision under a *new context* that has just been observed using dynamic data feed retrieved from sensors and assimilated into the digital twin. While it is possible to argue that a simulation optimisation may be performed for this newly observed context in the same way it has been used in the past, the tight simulation budget constraint due to the short decision time window, and the high computation cost of running digital twin simulations make this approach no longer viable. Without a sufficient amount of simulation budget, existing simulation optimisation methodologies would fail to either identify an (near) optimal decision or estimate its performance accurately.

An intuitive approach is to simulate *all* possible contexts and identify the best decision for each context and store these results in a contingency table, or "playbook". Then, a quick table look-up would suffice to help a decision-maker to select the best decision in real-time. The problem with implementing this solution is the poor scalability; as the cardinality of the context space increases, e.g., with a high-dimensional context space, this leads to exorbitantly high computational cost to solve a simulation optimisation problem for each context.

Machine learning models are powerful predictive tools and can be used to support real-time decision-making. The emerging area of simulation learning (Brantley et al., 2014; Lin et al., 2019; Pedrielli et al., 2019) studies how to design stochastic simulation experiments to generate data that can be used to train machine learning models to predict system performance. Given a predictive machine learning model trained using simulation data, when a new context is observed, the decision-maker may then use the predictive model to quickly select a decision (Damgacioglu et al. (2018, 2019)). One such approach is the covariate Ranking & Selection (R&S) method considered in Shen et al. (2021), which used a linear regression model to model the relationship between simulation output and *covariates* (what we call *context*

in this paper). In Jiang et al. (2020), a logistic regression model was created using simulation output, and was shown to accurately classify risk levels associated with financial portfolios. The classification of these risk levels allows for real-time updating of risk assessments. Pearce and Branke (2017) used a Gaussian random field trained using simulation data sequentially added using the criterion of expected improvement, to relate the system's performance with contexts (referred to as instances in their paper) and decisions. Such a scheme is also referred to as offline simulation online decision (OSOD) by L. J. Hong and Jiang (2019) because machine learning models are trained *offline* before the context is observed, and then an *online* decision upon observing a context is made using the machine learning model predictions.

In this paper, we refer to the aforementioned methods as predictive simulation learning because it relies on simulation learning that is performed before the current decision context is observed, and then uses the machine learning predictions without any further simulation optimisation to select a decision in response to the observed context.

While predictive simulation learning has been demonstrated as a potentially very useful method to support real-time decision-making, major obstacles exist that make it difficult to achieve high-quality decision outcome. For linear models used in Shen et al. (2021) and Jiang et al. (2020), selection of a proper set of basis functions to predict the expected utility of a context has been shown to be difficult in high-dimensional problems. While the Gaussian random field model used by Pearce and Branke (2017) is a very powerful predictive model, its use is still limited to a low-dimensional context space because of the curse of dimensionality. The problem of predictive accuracy is further compounded by the use of stochastic simulation data, often with significantly heterogeneous noise, in model training. More importantly, as explained in Pedrielli et al. (2019), the vast context and decision space and the complex, dynamic and uncertain environment, and limited computing budget for learning often lead to predictive accuracy that ranges from poor to mediocre.

To address the aforementioned limitations of simulation optimisation and predictive simulation learning, this paper proposes a new algorithmic approach that integrates predictive simulation learning with simulation after a context is observed. The new algorithm, Sequential Allocation using Machine-learning Predictions as Light-weight Estimates (SAMPLE), makes an important step towards the goal of real-time simulation optimisation for digital twin applications. SAMPLE integrates predictions from *lightweight machine learning models*, which means those models are easy to train and fast to compute, but are of poor prediction accuracy, with simulation output conducted under the observed context. This is the *gist* of SAMPLE that distinguishes it from existing works such as Jiang et al. (2020) and Shen et al. (2021):

- By integrating simulation output for the current context, SAMPLE is able to effectively overcome the problem of predictive errors in machine learning models, and offers a robust method that works well with lightweight machine learning models that are easy to train and deploy in many important real-time decision making situations.
- By integrating machine learning predictions, SAMPLE is able to tremendously reduce the impact of stochastic variability in digital twin simulation estimates due to the severely limited simulation budget in a real-time decision making situation, and translates predictive simulation learning power collected from a large amount of past simulation data into improved decision outcomes in the future.

In the rest of the paper, Section 2 presents the mathematical formulation of the real-time simulation optimisation problem. Section 3 presents the technical details of the proposed SAMPLE algorithm. Section 4 presents numerical experiment results on a benchmark function and a case study. Section 5 states the conclusions and potential future work.

2. Real-time simulation optimisation problem and challenges

Formally, we aim to solve the following optimisation problem. Let Z denote the *context*, e.g., the observed system disruption such as the forecast of wind speeds in an area affected by a storm system, or a new rush order sent to the factory floor of a semiconductor manufacturing plant. Under context Z, the decision-maker needs to select from a set of feasible decisions $X_j = 1; 2; \ldots; m$, e.g., the allocation of utility crew to power substations or transmission lines to perform equipment/line hardening. Let $f(X_j; Z)$ denote the utility of decision X_j given context Z. Our goal is to select $j^*(Z)$, the optimal decision under context Z:

$$j^*(Z) = \arg \max_{j \in \{1;\ldots;m\}} f(X_j; Z):$$

Given the uncertainty in the system, e.g., in weather forecast and weather impact on the power system, $f(X_j; Z)$ is typically a summary statistic of a random variable $F(X_j; Z; \omega)$, where ω represents the uncertainty in the system. In this paper, we consider $f(X_j; Z) = \mathbb{E}_{\mu_\omega}[F(X_j; Z; \omega)]$, where the expectation is taken with respect to the probability measure μ_ω, which may depend on the context Z as well as decision X_j. Because of the complexity of the digital twin model, $f(X_j; Z)$ can only be observed by running simulations that produce independent and identically distributed (IID) observations of $F(X_j; Z; \omega)$. We

denote the output from the l th IID replication by $F_l(X_j; Z; \phi)$. The digital twin simulation estimate of $f(X_j; Z)$ is the sample average

$$\hat{f}(X_j; Z) = \frac{1}{l_j} \sum_{l=1}^{l_j} F_l(X_j; Z; \phi) \quad (1)$$

where l_j is the number of simulation replications ran for X_j under context Z. Then, the best decision selected would be

$$\hat{j}(Z) = \arg \max_{j \in \{1,\ldots,m\}} \hat{f}(X_j; Z). \quad (2)$$

When there is considerable noise in simulation output, a large number of simulation samples need to be collected to achieve an acceptable probability of correction selection (PCS) by simulation. We define this to be the probability that, for an observed context $Z = Z_i$, we observe the correct $\hat{j}$ from our simulation output and denote this as $\mathrm{PCS}_S(Z_i)$

$$\mathrm{PCS}_S(Z_i) = P\left(\hat{j}(Z_i) = j(Z_i)\right).$$

For general Z, Z, Shen et al. (2021) introduces two different ways to understand the probability of correct selection across the entire space of covariates; the probability can be taken with respect to its expected value across covariates or its minimal value across covariates. In this paper, we consider the PCS across Z under its expected value, such that

$$\mathrm{PCS}_S = \mathbb{E}_Z\left[P\left(\hat{j}(Z) = j(Z)\right)\right] \quad (3)$$

where the expectation is taken with respect to the probability distribution of Z. For our numerical experiments, we evaluate this probability by conducting experiments on randomly generated contexts and take the sample average to estimate PCS_S.

Given a particular context Z_i, simulation optimisation literature provides many efficient methods that either achieve a pre-specified certain level of $\mathrm{PCS}_S(Z_i)$ or maximise $\mathrm{PCS}_S(Z_i)$ for a given simulation budget (Fu, 2015; Xu et al., 2016). A recent review of existing procedures that includes descriptions of emerging techniques in simulation optimisation is given by J. L. Hong et al. (2021). Specifically, fixed-confidence Ranking & Selection (R&S) methods determine the number of simulation replications to be allocated to each decision so as to provide some form of probabilistic guarantees, e.g., the PCS or probability of good selection (PGS; Eckman & Henderson, 2021; Kim & Nelson, 2006). Fixed-budget R&S methods aim to optimise the allocation of a given simulation budget to maximise some performance metrics, e.g., PCS or expected opportunity cost (EOC; Branke et al., 2007; Gao et al., 2017; He et al., 2007). Well-known fixed-budget R&S methods adopt either a large-deviation rate optimal approach started by the seminal paper on optimal computing budget allocation (OCBA; C.-H. C.-H. Chen & Lee, 2010; C.-H. Chen et al., 2000; LaPorte et al., 2012, 2015) or Bayesian approach that sequentially allocates simulation budget (Chick et al., 2010; Frazier et al., 2008; Ryzhov et al., 2012).

When there is a sufficient amount of simulation budget, problem (2) can be solved by the aforementioned existing simulation optimisation methods to achieve a reasonably high $\mathrm{PCS}_S(Z_i)$, although the role of context Z_i is not explicitly considered and instead encapsulated into the simulation model itself. However, with a highly constrained simulation budget as in real-time digital twin applications, $\mathrm{PCS}_S(Z_i)$ may be very poor when existing simulation optimisation methods are directly applied. One may consider executing an R&S procedure in a high-performance computing (HPC) environment, such as in Luo et al. (2015), Ni et al. (2017), Zhong and Hong (2021), and Zhong et al. (2021). However, access to computing clusters which provide the magnitude of compute nodes required to successfully implement traditional R&S procedures in the real-time context we consider are often not available. Furthermore, in some instances, the scale of computational complexity to identify good solutions from a given simulation may be such that the required number of compute nodes is intractable to obtain even for those willing to invest in HPC infrastructure.

Recently, OCBA has also been extended to consider the allocation of simulation budget to decisions and a finite set of contexts with an objective to maximise the PCS measured across all contexts for a particular decision (Gao et al., , 2019b). They did not consider the possibility of executing a limited number of simulations once a context is observed. Consequently, assuming that the true optimal alternative is different depending on the observed context (otherwise, the problem degrades to traditional simulation optimisation), such a policy is sub-optimal given the observed information contained in the context. While the decision selected by their procedure is expected to perform reasonably well, it would most likely be sub-optimal for the specific realised context without taking advantage of new information generated by simulating decisions under the observed context.

In the following, we present the proposed SAMPLE framework that effectively handles the aforementioned challenges for potential digit twin applications.

3. SAMPLE framework

In this paper, we assume that the decision-maker has used predictive simulation learning to train a machine learning model using simulation data generated on different contexts and decision alternatives. Then, upon observing Z, the utility of a decision X_j is approximated by the machine learning model prediction $g(X_j; Z)$, which can be evaluated in a negligible

amount of time compared to running digital twin simulations. For notational simplicity, we just consider one machine learning model here. The proposed SAMPLE algorithm is able to work with multiple machine learning models. It is then reasonable to select the best predicted decision by

$$j^{\blacklozenge}(Z) = \arg\max_{j=1,\ldots,m} g(X_j, Z). \quad (4)$$

The probability of selection by predictive simulation learning (PCS $_{\mathrm{P}}$) is then defined by

$$\mathrm{PCS}_{\mathrm{P}} = \mathbb{E}_Z[P(j^{\blacklozenge}(Z) = j^{\blacklozenge}(Z))] \quad (5)$$

The problem of solely relying on $j^{\blacklozenge}(Z)$ is that PCS $_{\mathrm{P}}$ is likely to be low unless $g(\cdot,\cdot)$ has high predictive accuracy. However, as explained in Pedrielli et al. (2019), data sparsity makes it likely that $g(\cdot,\cdot)$ only provides a crude approximation to $f(\cdot,\cdot)$ for the current context Z. The proposed SAMPLE framework integrates information from simulation output $F_i(X_j, Z)$ in equation (1) with the machine learning predictions in equation (4) to produce a new estimate $f(X_j, Z)$, which then gives the following real-time simulation optimisation selection rule

$$j^{\blacklozenge}(Z) = \arg\max_{j=1,\ldots,m} f(X_j, Z);$$

and the PCS with real-time simulation optimisation using the proposed SAMPLE algorithm under a realised context Z_i, is then given by

$$\mathrm{PCS}_{\mathrm{R}}(Z_i) = P(j^{\blacklozenge}(Z_i) = j^{\blacklozenge}(Z_i)).$$

To better measure the performance of SAMPLE in the context space, following the definition of PCS_S as in equation (3), we define $\mathrm{PCS}_{\mathrm{R}}$:

$$\mathrm{PCS}_{\mathrm{R}} = \mathbb{E}_Z[\mathrm{PCS}_{\mathrm{R}}(Z_i)]. \quad (6)$$

Unlike the previous work on context-dependent R&S by Shen et al. (2021), the proposed SAMPLE algorithm is not designed with any prior context distribution in mind. Rather, the premise of SAMPLE is that for any realised context Z_i, given the same online simulation budget, SAMPLE would be able to improve $\mathrm{PCS}_{\mathrm{R}}(Z_i)$ because of the usage of both machine learning predictions and online simulation data. In the following sections, we first show how to construct $f(X_j, Z)$. We then conduct numerical experiments to compare $\mathrm{PCS}_{\mathrm{R}}$ with $\mathrm{PCS}_{\mathrm{S}}$ and $\mathrm{PCS}_{\mathrm{P}}$.

3.1. Overview of sample

The proposed SAMPLE algorithm operates at two stages: predictive simulation learning and real-time simulation optimisation. The predictive simulation learning stage occurs before the decision context Z is observed. The purpose of the predictive simulation learning stage is to train machine learning models using simulation data on various contexts and decision alternatives. SAMPLE can use any of the predictive simulation learning methods in the literature such as Shen et al. (2021) and Pedrielli et al. (2019) in this stage.

In the real-time simulation optimisation stage, SAMPLE then integrates predictions from the machine learning models with simulations run in real-time on the observed context Z_i to evaluate decision alternatives and recommend a (near) optimal decision in real time. The real-time simulation optimisation stage sets SAMPLE apart from existing methods. Figure 2 presents the information flow in the SAMPLE framework.

3.2. Predictive simulation learning

In the predictive simulation learning stage, SAMPLE generates a large number of contexts and conducts extensive simulation experiments to estimate outcomes of decisions under each context. This process provides a (potentially) vast amount of offline simulation data to train machine learning models that may predict outcomes of different decisions without the need to run simulations. A very well-known example of this offline simulation-based learning approach is Google/Deepmind's AlphaGo (Silver et al., 2016) and AlphaGo Zero (Silver et al., 2017). Trained over 29 million self-played games (Silver et al., 2017), AlphaGo Zero was able to build highly accurate deep neural networks that allow Monte Carlo Tree Search to effectively search the decision space without going down very deep in the game tree. However, most applications would not have access to the computing power required to perform offline simulation experiments at such a scale. Therefore, despite the success of AlphaGo, as pointed out by Pedrielli et al. (2019), data sparsity is a real issue for applications with high-dimensional context and decision spaces. To further exacerbate the problem of predictive accuracy, unlike a perfectly deterministic, rule-based board game like Go, real-world problems lack clear structures and rules, and uncertainty is prevalent both internally and externally, leading to even more complex behaviours that are extremely difficult to predict using machine learning models trained on observed or simulated data.

Another important consideration is the interpretability of machine learning models. While there has been very active research regarding interpretable machine learning in the past several years (Du et al., 2019; Molnar, 2020), simpler models such as linear models and tree-based models are much easier to understand and provide insights on relationships among responses and predictors.

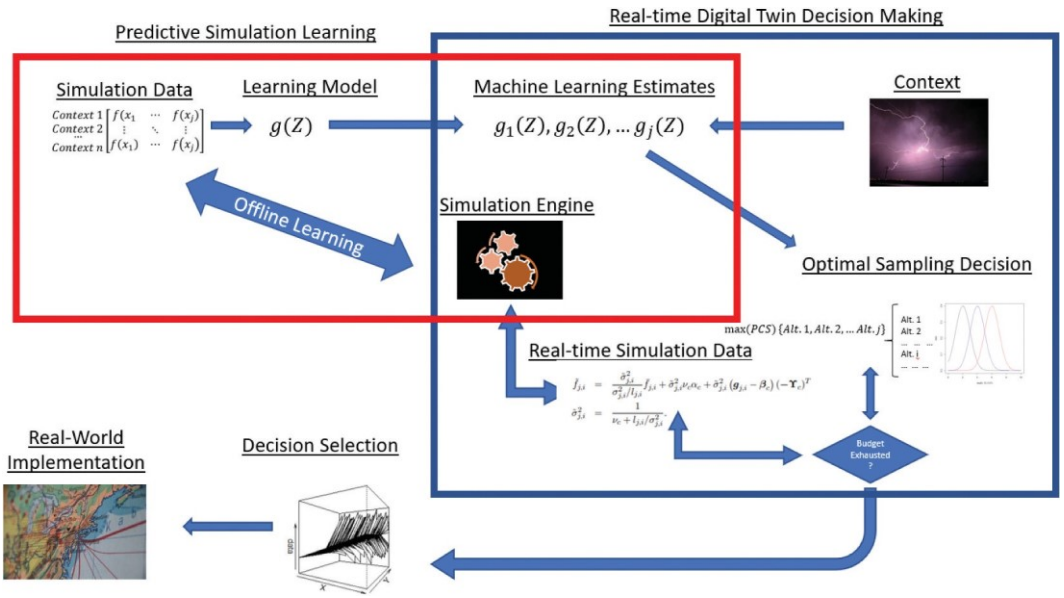

Figure 2. Information flow in the proposed SAMPLE framework.

However, these simpler and much easier to interpret machine learning models may not have high prediction accuracy.

Furthermore, even when an organisation is capable of employing complex and powerful machine learning techniques to create models using adequate data, there may still be a need to implement simulations in real time. For example, such a case may arise when the system for which our machine learning model generates predictions is an adaptive system. When generating predictions for such a system, the value of a particular decision may change over time as the system we are concerned with adapts to any observable pattern in our own decision-making and optimises its own system dynamics against these patterns. This change in value over time of a decision renders our predictive model ineffective. It may be possible, given the time and computing power, to generate a predictive model which ranks decisions that perform well against all possible system configurations as optimal, but in so doing, may result in undesirable opportunity cost resulting from sub-optimal decisions given the current system configuration.

Considering the data sparsity, prediction accuracy, and adaptive system limitations, SAMPLE is designed to work with *lightweight* machine learning models, such that it can effectively boost online decision efficiency and quality by working with machine learning models that are easy to train and interpret, but may not be the most accurate, e.g., compared to "heavy weight" counterparts such as deep neural networks. SAMPLE achieves this important design objective by optimally using real-time simulation sampling budget as explained in the next section.

3.3. Real-time simulation optimisation

Upon observing a context in nature for which a decision is required, relying on machine learning predictions alone would not lead to an optimal decision due to the inaccuracies in predicted decision performance. Online simulation sampling can fill this gap by providing accurate predictions of decision performance upon observing a new context. However, simulations are time-consuming, and simulation output is noisy. SAMPLE alleviates the computational efficiency challenge by utilising machine learning to guide online simulation sampling. Following exhaustion of a limited simulation budget, machine learning predictions and simulation data are assimilated to identify the best decision quickly for real-time response. This is the key premise of SAMPLE that sets it apart from existing works.

To implement this framework, the primary difficulty is evaluating the level of uncertainty surrounding the performance of a particular decision given both the machine learning predictions and the simulation output. Traditional R&S methodologies seek to maximise the PCS (or minimise the EOC) given noisy simulation output data. SAMPLE seeks to maximise the PCS for an observable context given both the observed simulation output and simulation learning predictions. The critical question becomes how one optimally balances the need to explore the solution space using simulation output to safeguard against machine learning prediction errors and the need to exploit machine learning predictions and focus simulation on the top performing alternatives. The work presented in this paper shows that the optimal solution to the exploration/exploitation problem in stochastic simulation under a framework such as

SAMPLE produces better decisions with less effort than when we only consider observable simulation data, making it a good tool to use when selecting decision alternatives in a resource constrained environment. In the following, we first discuss how to integrate machine learning and online simulation data, and then study how to allocate online sampling budget within the SAMPLE framework. The algorithms are based on the work of Peng et al. (2019), who studied how to integrate low-fidelity model output (Celik & Son, 2012) with high-fidelity simulation data.

3.3.1. Integration of machine learning and simulation data

Let $Z_i; i = 1; 2; \ldots; n$ index n contexts, and $X_j; j = 1; 2; \ldots; m$ be the decision alternatives. Let $f(X_j; Z_i)$ denote the utility of decision X_i given context Z_i. Our goal is to select j^*, the optimal decision index under context Z_i:

$$j_i^* = \arg \max_{j \in \{1;\ldots;m\}} f(X_j; Z_i):$$

Here, $f(X_j; Z_i)$ can only be evaluated using stochastic simulations that fully capture the complexity and uncertainty in the system. We assume the simulation sample mean $f(X_j; Z_i)$ is an unbiased estimator of $f(X_j; Z_i)$, and the distribution of the simulation output is normal with a mean of $f(X_j; Z_i)$ and a variance of σ^2_{ji}. When the distribution of the simulation output is not normal, we can instead sample $f(X_j; Z_i)$ as individual observations using the average of batched samples, which then behaves according to a central limit, and yields an approximately normal distribution.

In addition to the simulation model, there are also K machine learning models. Denote by $g_k(X_j; Z_i)$ the prediction from the k th model, and $g(X_i; Z_i) = \{g_1(X_j; Z_i); \ldots; g_K(X_i; Z_i)\}$. Also, denote by $h(X_i; Z_i) = \{f(X_i; Z_i); g(X_i; Z_i)\}$. Following the framework of Peng et al. (2019), $h(X_j; Z_i)$ is assumed to follow a Gaussian mixture model

$$h(X_j; Z_i) \sim \sum_{c=1}^{C} \tau_c \phi(\cdot|\mu_c; \Sigma_c): \quad (7)$$

Here, $\phi(\cdot|\cdot)$ is the probability density function of a $K + 1$ dimensional multivariate normal distribution and C is the number of mixture model components. According to (7), if the performance value of $(X_j; Z_i)$ belongs to component c, then machine learning predictions $g_k(X_j; Z_i); k = 1; \ldots; K$ and the true utility $f(X_j; Z_i)$ follow a multivariate normal distribution with a mean vector μ_c and a covariance matrix Σ_c. The parameter τ_c gives the weight of component c.

The motivation and justification for this choice of representation for decision alternatives' performance is centred on a clustering phenomenon that has been documented in Xu et al. (2016), Peng et al. (2019), Song et al. (2019), and Zhang et al. (2020). The numerical experiments presented in this paper, e.g., as shown in Figure 5, also further demonstrated the existence of this clustering structure. Intuitively, such a clustering structure can be viewed as a characterisation of the general quality of decisions, e.g., excellent, good, poor. In real-world applications, instead of finding the optimal decision, decision makers typically would be satisfied with finding an excellent or even good decision when decision time window is very limited. This clustering structure also matches with this desire and has proved to be a flexible, robust, and effective method to use low-fidelity predictions to help determine the quality of a decision.

In Peng et al. (2019), a stochastic model-based clustering procedure was derived and can be used to cluster decisions into C components for a context Z_i using both machine learning and stochastic simulation data. In an online decision setting, the number of simulation samples would be very limited, and when noise is significant, including those simulation samples in clustering analysis may not be helpful. Based on this consideration, we propose to use the completely observable machine learning predictions $g(X_j; Z_i)$ to perform clustering analysis, which provides an approximation to the clustering structure for the partially observable data $h(\cdot; \cdot)$.

$$\mu_c = (\alpha_c; \beta_c); \Sigma_c = \begin{bmatrix} \varsigma_c^2 & \Gamma_c \\ \Gamma^T & \Lambda_c \end{bmatrix}; \Sigma_c^{-1} = \begin{bmatrix} \nu_c & \Upsilon_c \\ \Upsilon_c & \Omega_c \end{bmatrix}$$

In the above, α_C, β_c represent the average performance of alternatives contained in the c^{th} cluster according to the simulation and K predictive models, respectively. The variance of the alternatives' performance in the c^{th} cluster according to the simulation is ς^2, and Γ_c the correlation between the simulation output and each of the predictive models. The covariance matrix Λ_c is for the predictions for all K predictive models for alternatives in the c^{th} cluster. Parameters of Σ_c^{-1} are computed directly from these values. Here β_c and Λ_c only depend on $g(X_j; Z_i)$ data. Denote by $\Theta_C = \{\beta_c; \Lambda_c; \tau_c : c = 1; \ldots; C\}$ the set of model parameters to be estimated. It can be shown that the likelihood of these parameters is given as

$$L_g(\Theta|\Theta_C) = \prod_{i=1} \left(\sum_{c=1} \tau_c \phi(g|\beta_c; \Lambda_c) \right)!$$

This likelihood can be maximised using the expectation-maximisation (EM) algorithm, and we denote the estimated parameters as Θ_C^*. Since C is unknown, it can be treated as a tuning parameter. The Bayesian

information criteria (BIC) can be utilised to discount the given likelihood in order to determine the optimal number of components $C^{\blacklozenge}$ as

$$C^{\blacklozenge} = \arg\max_{C=1,\ldots,\blacklozenge} \log\left(L_{g(\blacklozenge|\blacklozenge,C)}\right) - \frac{d+1}{2}\frac{d+2}{\blacklozenge}C - 1\,\blacklozenge\log\frac{m}{2} \quad (8)$$

where d is the number of parameters to be estimated and $\blacklozenge$ is a pre-specified upper bound for C. Once these parameters have been determined, initial online simulation sample of $f(\blacklozenge; Z_i)$ would be generated upon observing the context Z_i. We have the following theorem on the posterior distribution of $f(X_j; Z_i)$. For notation simplicity, we replace $(X_j; Z_i)$ with subscripts in the following, e.g., instead of writing $f(X_j; Z_i)$, we use $f_{j;i}$.

Theorem 1. Conditional on decision X_j belonging to cluster c, the posterior distribution of $f(X_j; Z_i)$ is normal with mean $f(X_j; Z_i)$ and variance $\sim^2(X_j; Z_i)$ as given below:

$$f_{j;i} = \sigma^2_{j;i}\left(\frac{\tilde{\sigma}^2_{j;i}}{l_{j;i}}\right)\left(f_{j;i} + \nu_c\left(\nu_c + \tilde{\sigma}^2_{j;i}\right)^{\blacklozenge}\left(g_{j;i} - \beta_c\right)(\gamma - \gamma_c)^T\right) \quad (9)$$

$$\sim^2_{j;i} = \nu_c + \frac{1}{l_{j;i}}\sigma^2 \quad (10)$$

For the proof of Theorem 1, please refer to Peng et al. (2019). The model parameters $\alpha_c; \beta_c; \gamma_c; \nu_c$ in (9) and (10) are estimated via the EM algorithm using the number of clusters $C^{\blacklozenge}$ determined using the BIC as described previously in (8). Given simulation output, one can easily compute $f_{j;i}$ and σ^2 , and subsequently compute the posterior mean and variance. For a more detailed description of the process, please see, Peng et al. (2019). As can be seen from (10), as online simulation sample size $l_{j;i}$ increases, the posterior variance goes to zero, providing accurate predictions. However, $l_{j;i}$ would typically be small and thus simulation estimates would have considerable noise. The term ν_c is the precision from the Gaussian mixture model and helps reduce the posterior variance with the information from machine learning predictions $g_{j;i}$.

Equation (9) reveals how $g_{j;i}$ are used with online simulation sample mean $f_{j;i}$ to predict $f_{j;i}$. The weight for $f_{j;i}$ is given by the ratio of the posterior variance and the variance of $f_{j;i}$. As more simulation samples are collected, this weight increases. The second term in the right-hand side of (9) comes from the estimated mixture component mean α_c:

$$\wedge_c = \frac{\sum_{j=1}^{m}\eta_{j;c}\,\blacklozenge_{j;i}}{\sum_{j=1}^{m}\eta_{j;c}}; \quad c = 1;\ldots;C \quad (11)$$

In (11), the binary variable $\eta_{i;c}$ indicates if decision X_i is grouped into component c. Peng et al. (2019) showed how to compute clustering statistics using $g(\blacklozenge)$ to determine $\eta_{j;c}$. Therefore, $\wedge_c$ pools the online simulation estimate for all decisions grouped into the same component c such that limited online simulation information may be shared among multiple decisions. The last term in (11) uses the component-wise linear relationship captured by the multivariate normal model between $f(\blacklozenge)$ and $g(\blacklozenge)$ to directly predict $f_{j;i}$ using $g_{j;i}$.

3.3.2. Optimal online sampling policy

After $l_{\blacklozenge}$ online simulations have been run for each decision $j = 1;\ldots;m$, we would select the decision with the largest posterior mean as the best decision, i.e.,

$$j_i^{\blacklozenge} = \arg\max_{j\in\{1;\ldots;m\}} f_{j;i}$$

Due to simulation noise, the selection of $j^{\blacklozenge}$ may be incorrect, i.e., $j_i^{\blacklozenge} \neq j_i^{\blacklozenge}$. An optimal online sampling policy aims to maximise the probability of correctly selecting the best decision with a sampling budget L:

$$\max_{l_{j;i};j\in\{1;\ldots;m\}} P\left(j_i^{\blacklozenge} = j_i^{\blacklozenge}\right)$$

$$s.t. \sum_{j\in\{1;\ldots;m\}} l_{j;i} \leq L$$

Under the Gaussian mixture modelling framework given in (7), the posterior distribution of $f(\blacklozenge; Z_i)$ is normal, which facilitates the derivation of an optimal sampling allocation rule as given in the theorem below.

Theorem 2. Let $\delta_{j;i} = f_{j^{\blacklozenge};i} - f_{j;i}, \forall j \neq j^{\blacklozenge}$, and $\nu_{j;i} = \sigma^2_{j;i} + \nu_{j;i}$. An asymptotically optimal solution $l_{j;i}^{\blacklozenge}; j \in \{1;\ldots;m\}$ to problem (12) satisfies the following equations

$$\frac{l_{j_1;i}}{l_{j_2;i}} = \frac{\nu_{j_1;i}\delta_{j_2;i}}{\nu_{j_2;i}\delta_{j_1;i}}; \quad j_1 \neq j_2 \neq j^{\blacklozenge} \quad (12)$$

$$l_{i}^{\blacklozenge} = \left(\frac{\sigma}{\delta_{j;i}}\right)^2 \sum_{j\neq j^{\blacklozenge}} \nu_{j;i} \quad (13)$$

The proof of Theorem 2 follows the derivation of the original OCBA policy in C.-H. Chen et al. (2000) and is omitted here. The set of equations given in Theorem 2 can be easily solved to obtain the optimal allocation of online simulation budget to all decisions.

When the sampling policy is implemented, l_0 initial simulation samples are collected for each alternative and the sample variances $\hat{\sigma}^2_{j,i}$ are plugged into equations (9) and (10), and then the quantity $v_{j,i}$ is calculated and plugged into equations (12) and (13) to determine the allocations.

3.4. Implementation of sample

Algorithm 1 outlines the main steps of the SAMPLE algorithm. The decision-maker specifies the number of simulation replications l_0 that each alternative receives at the beginning of the online sampling stage. The total number of online simulation samples per SAMPLE iterations ΔL is another parameter specified by the decision-maker, which can be determined based on the decision time window and simulation run time. The choice of ΔL is related to the online computing platform to make use of any parallel computing power available. For notational simplicity, we drop the subscript i for the context under which a decision is to be selected.

Algorithm 1 (SAMPLE algorithm)

INPUT: context Z_i, a set of decision alternatives $\{X_1, X_2, \ldots, X_m\}$, machine learning predictions $\{g_j : j = 1, \ldots, m\}$, total simulation budget L, the number of initial simulation replications allocated to a decision l_0, the number of simulation samples spent at each online iteration ΔL.

Initialise

Estimate the Gaussian mixture model using $\{g_j : j = 1, \ldots, m\}$; set $l_j = l_0$; $j = 1, \ldots, m$; set iteration counter $k = 1$; set expended simulation budget $L_k = ml_0$.

LOOP: WHILE $L_k < L$ **DO**

Online simulation:

1: Simulate X_j; $j = 1, \ldots, m$ l_j times, compute sample variance $\hat{\sigma}_j$ and sample mean $\hat{f}_j$.

2: Compute posterior mean f_j and variance $\tilde{\sigma}^2_j$ by plugging $\hat{\sigma}^2_j$ and $\hat{f}_j$ into (9) and (10).

3: Compute l_j, the allocation of ΔL simulation samples to X_j; $j = 1, \ldots, m$, using (12) and (13).

4: Set $k = k + 1$ and $L_k = L_k + \Delta L$.

END OF LOOP

Decision: Return the decision with the highest f_j.

5. Numerical experiments

The efficiency of the SAMPLE framework was first tested with the widely used Griewank function in Section 4.1 and then on a system disruption mitigation experiment in Section 4.2. We compare the achieved PCS as a function of the total number of algorithm (SAMPLE and its competitors) iterations executed to compare the efficiency of each of the selection procedures tested. The specific PCS metrics, PCS_S; PCS_P; PCS_R, are defined in equations (3), (5) and (6). In these equations, the expectations are taken with respect to a uniform random distribution on a context space Z. Therefore, in the following experiments, we randomly sampled contexts and took the average PCS achieved under these sampled contexts as estimates of PCS_S; PCS_P, and PCS_R, respectively.

5.1. Experiments with a benchmark function

The efficiency of SAMPLE was first tested on the widely used Griewank function from the global optimisation literature (Griewank, 1981; Taghiyeh & Xu, 2016). The d-dimensional Griewank function is given by the formula

$$f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

The Griewank function is typically defined on a hypercube, where all $x_i \in [-600, 600]$. For testing purposes, we only consider values on the interval $[-10, 10]$. In this experiment, we set $d = 5$. The five-dimensional Griewank function has a global minimum at $x = [0, 0, 0, 0, 0]$.

Using this information, we define the cost for a decision $x = X_j$ under a particular context $z = Z_i$ to be:

$$f(x, z) = \sum_{i=1}^{d} \frac{(x_i - z_i)}{4000} - \prod_{i=1}^{d} \cos\left(\frac{(x_i - z_i)}{\sqrt{i}}\right) + 1$$

The l th simulation output is then obtained by adding to $f(x, z)$ a noise term to ε_l that is an IID realisation from a normal distribution $\varepsilon_l \sim N(0, \sigma^2)$:

$$F_l(x, z) = f(x, z) + \varepsilon_l$$

To test the effect of simulation learning prediction error and variability, we use the following equation to generate "simulation learning predictions" $g(x, z)$:

$$g(x, z) = f(x, z) + \gamma$$

Here γ is an IID realisation of a normal random variable $N(\mu_g, \sigma^2)$, with μ_g being the average prediction error and σ^2 the variance in prediction errors.

We create a static list of 1,000 randomly generated decisions, which serve as the set of feasible alternatives from which a decision-maker would select her decision. For each randomly generated decision denoted as x^j, $i = 1, \ldots, 1000$, its i th element x^j, $i = 1, \ldots, 5$, is generated uniformly randomly on the $[0, 10]$ interval. Any duplicate is removed, and a new one is added to the set of decisions. For testing purposes, we ensure that for a given context, a single decision from the previously generated 1000 decisions will be the unique

optimal decision. Without loss of generality, we utilise the first 200 decision alternatives to serve as the 200 contexts that we test. Therefore, for each context tested, there is a decision that is identical to the context and is thus the globally optimal decision.

In our experiments, we varied several procedure variables for each selection procedure, which affect the efficiency of the algorithms. Table 1 lists the design parameters that are tested.

The experiment is executed as follows. For each context, initial performance estimates are gathered by simulating each alternative l_0 replications. For the SAMPLE algorithm, initialisation also includes generating the simulation learning predictions and computing the Gaussian mixture model parameters. Following initialisation, each selection procedure executes 100 iterations. For each iteration, ΔL replications are allocated in accordance with the selection procedure. At the end of each iteration (to include initialisation), the selected optimal alternative is recorded. We conducted 20 macro-replications for each experimental parameter configuration given in Table 1 and selection procedure. According to Table 1, there are 18 unique combinations of $l_0; \Delta L; \sigma^2$ to conduct experiments for OCBA and equal allocation (EA) selection procedures. EA refers to the sampling policy where replications are distributed to alternatives with equal probability. Implementation of EA and OCBA for these experiments was equivalent to their implementation in a scenario where the context was assumed and integrated into the design of the simulation. The behaviour of an alternative in the simulation given the observation of the covariates is fixed, so OCBA and EA can be applied to determine the optimal alternative in this fixed setting. Unlike SAMPLE, this is performed without any prior knowledge given by a set of predictive models. For each of those 18 combinations, there are 6 unique SAMPLE experiment parameter combinations related to the simulation learning predictive model parameters (μ_g and σ^2). This results in 108 total experiments. Given the large number of experiments, only a subset of 24 representative experiments conducted are presented here in Figure 3, while the remainder of the results provides similar observations as derived from these 24 experiments.

In Figure 3, each plot shows PCS curves for each selection procedure, one for each of the 20 macro replications conducted for the given experiment. We can see that in all cases, the SAMPLE algorithm has a clear and substantial increase in PCS during initial sampling when compared against OCBA and EA. Given the mean and variance of performance for each selection procedure in each experiment, we can conduct a two-way t-test to infer the statistical significance of the difference. The p-values associated with these tests for a variety of different PCS measures are given in the following table.

Table 1. Experimental design for comparing SAMPLE framework efficiency.

Parameter	Values
Number of Alternatives m	1000
Predictive Estimate Error μ_n	0, 1
Predictive Estimate Variance σ^2	0, 0.1, 1
Initial Replications per Alternative l_0	3, 10
Allocation Budget per Iteration ΔL	100, 500, 1000
Simulation output variance σ^2	1, 4, 9

For each of these experiments, we also calculate the probability that SAMPLE reduces the required number of simulation replications to achieve various levels of PCS. In Table 2 and Table 3, we present the p-values associated with the two-way t-tests conducted against our hypothesis. The null hypothesis used for our statistical testing is as follows:

$$H_0 : N_{PCS=p}(\text{OCBA or EA}) = N_{PCS=p}(\text{SAMPLE}) \tag{14}$$

where $N_{PCS=p}(\blacklozenge)$ is the number of replications needed to achieve a PCS level of x with the given selection procedure. The PCS values that we test our hypothesis against are .1, .25, .5, .7, and .85. Values which are annotated as N/A indicate that neither selection procedure was able to reach that given level of PCS.

We can observe from these results that, in a general sense, SAMPLE outperforms OCBA and EA with statistical significance. The only exceptions are seen in experiments 13–17, when comparing SAMPLE with OCBA at a PCS level of 0.7. For these experiments, the variances of the simulation are very low, and the accuracy of our predictive models are relatively poor. As a result, OCBA was able to achieve similar performance as SAMPLE.

5.2. System disruption mitigation experiments

Figure 4 illustrates a system where a system operator is given five assets to protect against disruptions, each with a different utility. There are 10 disruption events that may happen simultaneously to these assets. The system operator is able to allocate 10 different mitigation measures to protect the five assets. The mitigation measures all have the same probability, denoted as p, of successfully stopping a disruption event. If the mitigation measure is not successful, a disruption event is able to disrupt the system's operation with the same probability q.

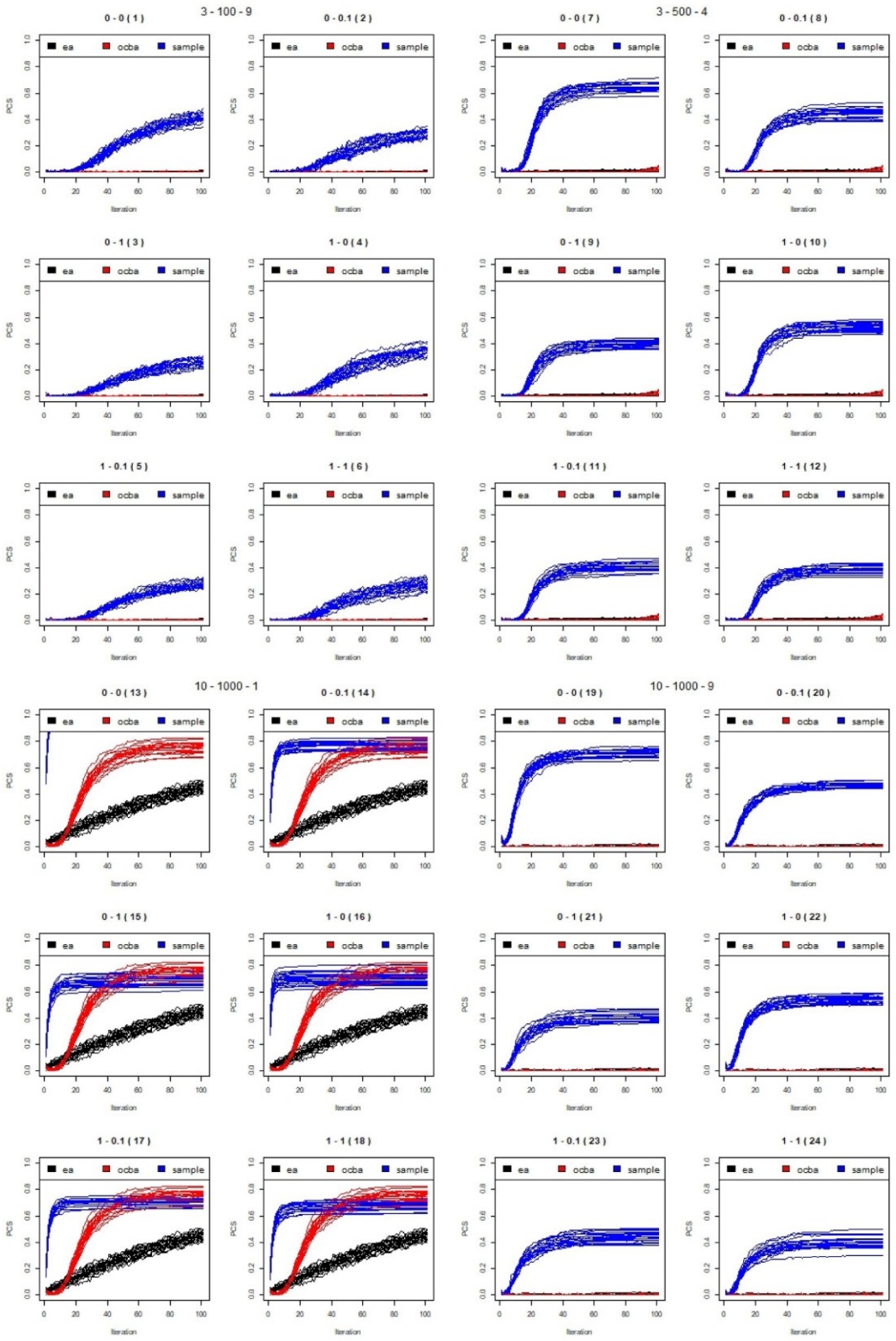

Figure 3. Results from 4 out of 24 experiments on the benchmark test problem. Each experiment was presented in a sub-figures with 6 plots laid in a 3×2 grid. The title of each sub-figure contains three numbers depicting l_0, ΔL, and σ^2, respectively. Each plot is titled using two additional numbers depicting μ_q and σ^2 for the SAMPLE procedure, respectively. The horizontal axis shows the algorithm iteration count. The vertical axis shows the PCS achieved by a procedure after a certain number of iterations.

Table 2. *p*-values from two-way *t*-tests comparing SAMPLE and EA.

PCS	.1	.25	.5	.7	.85
1	9.353e-134	1.274e-125	N/A	N/A	N/A
2	3.649e-127	5.003e-106	N/A	N/A	N/A
3	1.571e-127	3.580e-08	N/A	N/A	N/A
4	4.317e-130	2.844e-110	N/A	N/A	N/A
5	3.747e-133	6.070e-16	N/A	N/A	N/A
6	4.082e-125	2.246e-09	N/A	N/A	N/A
7	1.779e-150	7.261e-151	2.369e-129	1.619e-01	N/A
8	7.524e-151	5.539e-142	1.619e-01	N/A	N/A
9	5.969e-143	7.292e-131	N/A	N/A	N/A
10	4.157e-154	8.445e-148	3.579e-08	N/A	N/A
11	1.111e-149	1.711e-137	N/A	N/A	N/A
12	7.320e-147	2.042e-134	N/A	N/A	N/A
13	2.267e-17	7.653e-33	4.323e-16	1.587e-181	7.296e-159
14	2.412e-17	1.820e-32	4.343e-16	6.352e-137	N/A
15	2.492e-17	1.867e-32	4.360e-16	3.482e-03	N/A
16	2.590e-17	7.619e-33	4.335e-16	4.805e-05	N/A
17	2.172e-17	1.877e-32	4.351e-16	2.247e-09	N/A
18	2.239e-17	1.931e-32	4.361e-16	8.100e-03	N/A
18	1.877e-154	6.351e-156	9.023e-139	3.440e-07	N/A
20	1.575e-154	4.706e-145	7.726e-02	N/A	N/A
21	6.265e-154	7.264e-129	N/A	N/A	N/A
22	8.460e-152	1.088e-152	2.672e-21	N/A	N/A
23	1.218e-154	7.534e-137	1.619e-01	N/A	N/A
24	8.791e-152	9.976e-129	N/A	N/A	N/A

The parameters utilised for these experiments are given in Table 4. For a detailed description of one such example in the context of power grid hardening for improved weather resilience, please refer to Xu et al. (2020).

The disruption vector $Z \frac{1}{4} \mathbf{f}z_1; \ldots; z_5\mathbf{g}$ provides the context under which the system operator determines the mitigation decision $X \frac{1}{4} \mathbf{f}x_1; \ldots; x_5\mathbf{g}$. Assuming all 10 mitigation measures will be used there are $1; 001$ context-decision pairs. To develop benchmark performance measures for each context vector, the theoretically optimal mitigation measure allocation $X_{\mathbf{f}}\eth Z_i\mathbf{p}$ was computed for each Z_i.

5.2.1. Predictive simulation learning

Two machine learning models were trained in the offline stage. The first was a linear regression model. Because linear models are often preferred by decision makers thanks to their easy interpretability, efforts were made to compare and select basis functions to improve the predictive accuracy of a specific linear model trained for use in this experiment. The linear regression equation found to be effective for this experiment is given below:

$$f\eth X; Z\mathbf{p} \frac{1}{4} \sum_{k\frac{1}{4}1}^{5} \beta_{1;k}x_k \mathbf{p} \beta_{2;k}z_k \mathbf{p} \beta_{3;k}x_k z_k \qquad (15)$$

Table 3. *p*-values from two-way *t*-tests comparing SAMPLE and OCBA across 24 experiment scenarios.

PCS	.1	.25	.5	.7	.85
1	9.353e-134	1.274e-125	N/A	N/A	N/A
2	3.649e-127	5.003e-106	N/A	N/A	N/A
3	1.571e-127	3.580e-08	N/A	N/A	N/A
4	4.317e-130	2.844e-110	N/A	N/A	N/A
5	3.747e-133	6.070e-16	N/A	N/A	N/A
6	4.082e-125	2.246e-09	N/A	N/A	N/A
7	1.779e-150	7.261e-151	2.369e-129	1.619e-01	N/A
8	7.524e-151	5.539e-142	1.619e-01	N/A	N/A
9	5.969e-143	7.292e-131	N/A	N/A	N/A
10	4.157e-154	8.445e-148	3.579e-08	N/A	N/A
11	1.111e-149	1.711e-137	N/A	N/A	N/A
12	7.320e-147	2.042e-134	N/A	N/A	N/A
13	1.401e-43	8.669e-44	3.106e-37	6.759e-02	7.296e-159
14	1.202e-43	1.335e-41	3.809e-36	6.843e-02	N/A
15	1.463e-43	6.984e-43	4.545e-35	9.999e-01	N/A
16	1.568e-43	1.055e-43	7.097e-37	9.974e-01	N/A
17	1.179e-43	1.863e-42	4.711e-36	8.853e-01	N/A
18	1.643e-43	7.478e-43	3.287e-35	N/A	N/A
18	1.877e-154	6.351e-156	9.023e-139	3.440e-07	N/A
20	1.575e-154	4.706e-145	7.726e-02	N/A	N/A
21	6.265e-154	7.264e-129	N/A	N/A	N/A
22	8.460e-152	1.088e-152	2.672e-21	N/A	N/A
23	1.218e-154	7.534e-137	1.619e-01	N/A	N/A
24	8.791e-152	9.976e-129	N/A	N/A	N/A

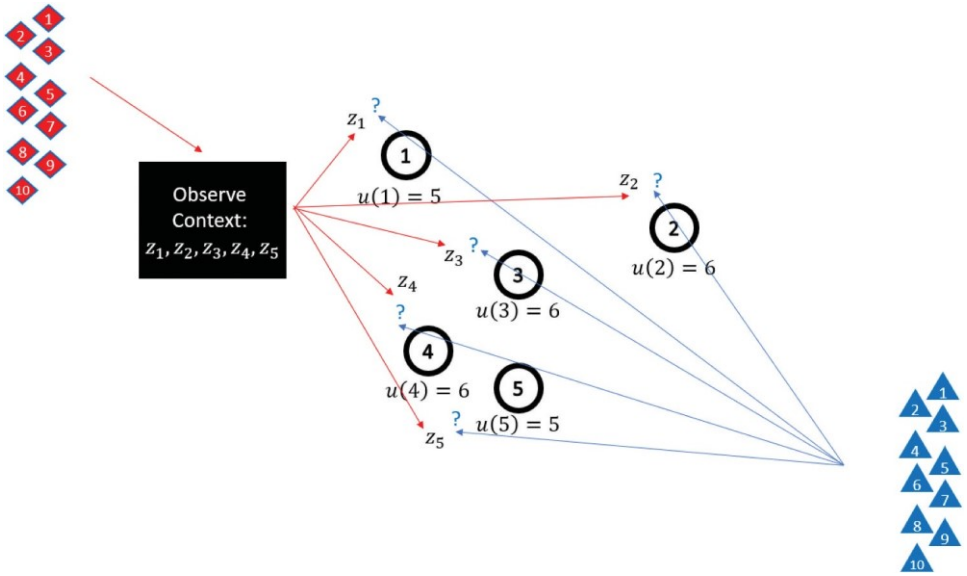

Figure 4. Allocation of 10 mitigation measures to protect 5 assets against 10 incoming disruptions.

The second was a nonlinear regression model known as multi-adaptive regression spline (MARS (Friedman (1991)).

Latin hypercube designs (LHD; Santner et al. (2018)) were used to sample independently 200 disruption vectors and mitigation vectors. Then, pairing each disruption vector with all 200 mitigation vectors created 40,000 different design points. Thirty simulation replications were then run for each design point to generate training data. For the linear regression model, elastic net (Zou and Hastie (2005)) was used to estimate regression coefficients. Model tuning was done using a repeated 10-fold cross-validation process over multiple L_1 and L_2 regularisation parameters, for which the optimal settings were found to be ð0; 10Þ for the L_1 and L_2 regularisation parameter, respectively. For the MARS model, the data was also transformed using the same basis functions from equation (15). The tuning parameters were the degree of branching for splines and the number of splines to utilise. The optimal parameters were found to be ð2; 30Þ, respectively.

We plot predictions made by these two machine learning models for six different disruption scenarios in Figure 5. In the figure, all 1; 001 mitigation vectors are sorted in the ascending order of true utility values. We observe large prediction errors, but the relative ordering of mitigation vectors using machine learning predictions appear to be useful, as evidenced by the overall increasing trend in prediction values. In addition, except for the middle and right plots in the first

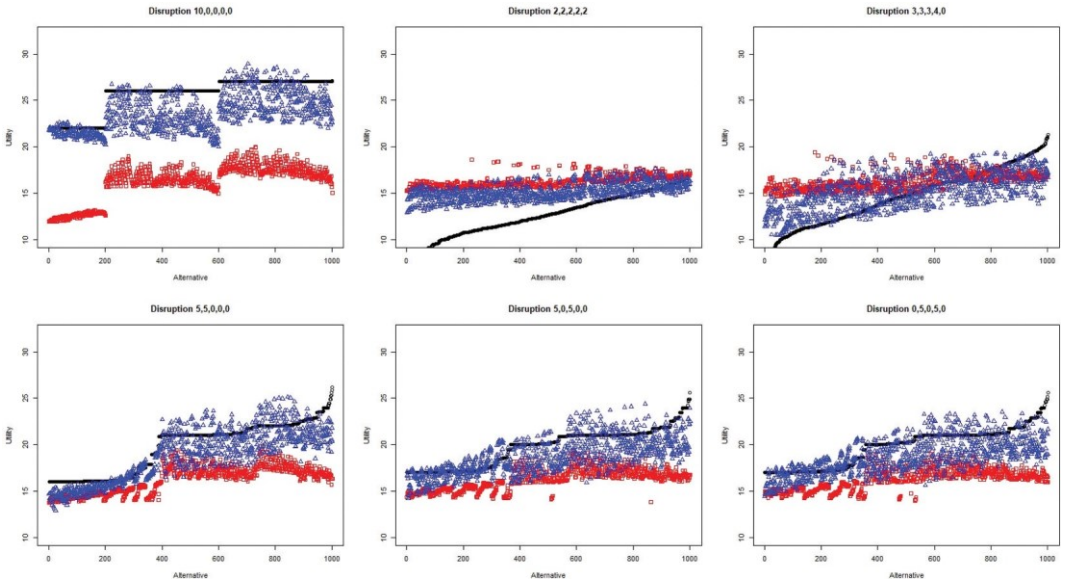

Figure 5. Utility values achieved by different mitigation vectors sorted in ascending order of utility values for six disruption scenarios. Black points are true expected utility values, blue points are predictions from Mars model, red points are predictions from the linear regression model. Mitigation vectors are sorted in the ascending order of true utility values.

Table 4. Experiment parameters with 10 disruptions impacting 5 assets protected by 10 mitigation measures.

Asset	Utility	Prob. of mitigation success (p)	Prob. of disruption success (q)
1	5	0.6	0.8
2	6	0.6	0.8
3	10	0.6	0.8
4	6	0.6	0.8
5	5	0.6	0.8

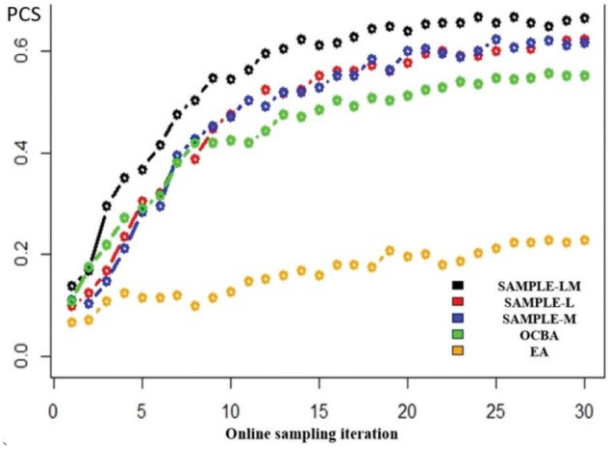

Figure 6. Comparison of PCS achieved by five different online sampling methods.

row of Figure 5, we notice the clear clustering structure of mitigation vectors based on their true utility values. Importantly, we observe that the clustering of mitigation vectors closely matches the clustering structure. These observations provide the empirical evidence on the viability of the Gaussian mixture model-based framework to integrate machine learning and online simulation data.

5.2.2. Real-time simulation sampling and decision

Five different real-time simulation sampling methods were used to select an optimal mitigation measure allocation upon observing a disruption vector, and their performances were compared to determine their accuracy. First, an EA scheme was used to equally allocate a limited simulation budget to all feasible mitigation measures. Second, a sequential implementation of the asymptotically optimal OCBA algorithm was used. Then, SAMPLE was used with the linear regression model (SAMPLE-L), with the MARS model (SAMPLE-M), and finally with both linear and MARS models (SAMPLE-LM) to allocate a simulation budget.

For an observed disruption vector, the sampling procedure carried out 30 online sampling iterations. Considering the prevalence of parallel computing in most applications, in each iteration, the online sampling algorithm determines the allocation of ΔN replications, where ΔN is set based on the parallel computing power. For example, if 100 openMP threads are available to run simulations simultaneously, then we may set $\Delta N = 100$. In the experiment, we set $\Delta N = 1{,}000$. At the end of an iteration, the design with the best estimated utility would be selected as the best mitigation vector. The sampling procedure then assimilates simulation results to update the allocation for the next iteration and proceeds. We set the number of iterations to $K_I = 30$.

The accuracy of different simulation sampling procedures were compared in two different ways. For a disruption vector Z_i, the mitigation vector selected by a procedure at the end of iteration k, $\hat{b}^k(Z_i)$, was compared against the true optimal mitigation vector benchmark allocation policy, $X_i(Z_i)$. This was done for $i = 1, \ldots, I_Z$ randomly generated disruption vectors, and the PCS achieved by each procedure is calculated by the percentage of times that $\hat{b}^k(Z_i) \in X_i(Z_i)$ for all I_Z disruption scenarios across all iterations $k = 1, \ldots, K_I$. In the experiment, we set $I_D = 250$. Figure 6 plots the PCS achieved by different simulation sampling procedures as a function of simulation sampling iterations. SAMPLE generally outperforms EA and OCBA.

Figure 6 plots the PCS achieved by different real-time simulation sampling procedures as a function of online sampling iterations. It is immediately clear that optimally allocating a limited sampling budget has a major impact on the quality of the decision.

When simulation budget is equally allocated, there was very little improvement in PCS, hovering around 20% even after 30 iterations. When SAMPLE only used one machine learning model, SAMPLE's performance was slightly lower than that of OCBA because machine learning predictors have considerable errors. However, as iteration increases, both SAMPLE-L and SAMPLE-M were able to outperform OCBA, showing SAMPLE's ability to integrate both simulation and machine learning predictions to improve performance. It is even more interesting to see how SAMPLE-LM was able to outperform OCBA early on and was considerably better than SAMPLE-L and SAMPLE-M as well. This is a very promising result showing that SAMPLE was able to integrate multiple machine learning models' predictions to achieve further performance improvement. To highlight the computational efficiency improvement achieved by SAMPLE-LM, we notice that OCBA after 30 iterations achieved a PCS around 0.5. It only took seven iterations for SAMPLE-LM to achieve that level of PCS. Therefore, SAMPLE-LM was more than 4 times faster than OCBA in this experiment.

6. Conclusions and future research

In this paper, a new framework is proposed to support online digital-twin-based decision with offline-learned machine learning models. A case study on disruption mitigation demonstrated the potential of the proposed SAMPLE framework. Notably, SAMPLE was able to benefit from two machine learning models that individually have large prediction errors, but when integrated with simulation data, achieved a substantial improvement in computational efficiency. In the benchmark function experiment across 108 experiment scenarios, SAMPLE outperforms OCBA with a high level of statistical significance in the vast majority of scenarios, except for a few with highly favourable setups for OCBA and unfavourable setups for SAMPLE. In the system disruption mitigation experiment, SAMPLE was able to achieve a four times speed-up using machine learning models with low predictive accuracy.

There are several important directions to further enhance the performance of the proposed SAMPLE framework. First, while the Gaussian mixture model has proved to be effective and flexible, for some applications, other models may better capture the relationship between machine learning predictions and simulation output. Exploring alternative models to integrate offline trained machine learning predictions with online simulation output may lead to significant performance improvement of the proposed SAMPLE framework. Second, machine learning models used in the experiments have been built using the well-known least-squares method. However, for this particular problem, the ability of a machine learning model to correctly rank alternatives (specifically, to determine the best alternative) is more important than its ability to minimise the sum of squared errors based on how the online sampling procedure uses the machine learning model predictions. Therefore, other estimation methods that may lead to better ordinal ranking performance may further enhance the performance of the proposed SAMPLE framework.

Finally, incorporation of data from real-time simulation iterations could prove beneficial. For the case where we are presented with a context, which we have already investigated in depth in the offline learning stage, the incorporation of this information may not help improve machine learning models' prediction accuracy. However, it is far more likely that we are running simulations to make a decision in the event of a context we have not investigated. In such a case, there is the potential of incorporating information from previously ran simulations to adjust estimates from the offline-learned machine learning models.

Acknowledgments

T. Goodwin, J. Xu, C.-H. Chen, and N. Celik were supported in part by the Air Force Office of Scientific Research under grant FA9550-19-1-0383. J. Xu was also supported in part by the National Science Foundation under grant DMS-1923145 and UChicago Argonne LLC under grant 1F-60250. C.-H. Chen was also supported in part by the National Science Foundation under grant FAIN-2123683. Numerical experiments were conducted using resources provided by the Office of Research Computing at George Mason University (URL: https://orc.gmu.edu) and funded in part by grants from the National Science Foundation (Award Numbers 1625039 and 2018631).

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the Air Force Office of Scientific Research [FA9550-19-1-0383]; National Science Foundation[DMS-1923145, FAIN-2123683]; UChicago Argonne LLC [1F-60250].

ORCID

Nurcin Celik http://orcid.org/0000-0002-5479-7911

References

Bastani, M., Damgacioglu, H., & Celik, N. (2018). A δ-constraint multi-objective optimization framework for operation planning of smart grids. *Sustainable Cities and Society*, *38*, 21–30. https://doi.org/10.1016/j.scs.2017.12.006

Bastani, M., Thanos, A. E., Damgacioglu, H., Celik, N., & Chen, C.-H. (2018). An evolutionary simulation optimization framework for interruptible load management in the smart grid. *Sustainable Cities and Society*, *41*, 802–809. https://doi.org/10.1016/j.scs.2018.06.007

Branke, J., Chick, S. E., & Schmidt, C. (2007). Selecting a selection procedure. *Management Science*, *53*(12), 1916–1932. https://doi.org/10.1287/mnsc.1070.0721

Brantley, M. W., Lee, L. H., Chen, C.-H., & Xu, J. (2014). An efficient simulation budget allocation method incorporating regression for partitioned domains. *Automatica*, *50* (5), 1391–1400. https://doi.org/10.1016/j.automatica.2014.03.011

Calverley, J., Currie, C., Onggo, B. S., Monks, T., & Higgins, M. (2021). Simulation optimisation for improving the efficiency of a production line. In Proceedings of the Operational Research Society Simulation Workshop 2021 (SW21) M. Fakhimi, D. Robertson, and T. Boness, eds. pp. 137–144. https://doi.org/10.36819/SW21.014 .

Cao, Yiyun, Currie, Christine, Onggo, Bhakti Stephan, and Higgins, Michael. (2021). Simulation optimization for a digital twin using a multi-fidelity framework. Proceedings of the 2021 Winter Simulation Conference, Phoenix, AZ. December 12-15, 2021, IEEE. DOI: 10.1109/WSC52266.2021.9715498

Celik, N., & Son, Y.-J. (2012). Sequential monte carlo-based fidelity selection in dynamic- data-driven adaptive multi-scale simulations. *International Journal of Production Research*, *50*(3), 843–865. https://doi.org/10.1080/00207543.2010.545445

Chen, C.-H., & Lee, L. H. (2010). *Stochastic simulation optimization: An optimal computing budget allocation.* World scientific.

Chen, C.-H., Lin, J., Yücesan, E., & Chick, S. E. (2000). Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems*, *10*(3), 251–270. https://doi.org/10.1023/A:1008349927281

Chen, T., & Wang, C. (2016). Multi-objective simulation optimization for medical capacity allocation in emergency department. *Journal of Simulation*, *10*(1), 50–68. https://doi.org/10.1057/jos.2014.39

Chick, S. E., Branke, J., & Schmidt, C. (2010). Sequential sampling to myopically maximize the expected value of information. *INFORMS Journal on Computing*, *22*(1), 71–80. https://doi.org/10.1287/ijoc.1090.0327

Damgacioglu, H., & Celik, N. (2022). A two-stage decomposition method for integrated optimization of islanded ac grid operation scheduling and network reconfiguration. *International Journal of Electrical Power &. Energy Systems*, *136*, 107647. https://doi.org/10.1016/j.ijepes.2021.107647

Damgacioglu, H., Celik, E., & Celik, N. (2018). Intra-cluster distance minimization in DNA methylation analysis using an advanced tabu-based iterative *k* k-medoids clustering algorithm (t-clust). *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *17*(4), 1241–1252. https://doi.org/10.1109/TCBB.2018.2886006

Damgacioglu, H., Celik, E., & Celik, N. (2019). Estimating gene expression from high- dimensional DNA methylation levels in cancer data: A bimodal unsupervised dimension reduction algorithm. *Computers & Industrial Engineering*, *130*, 348–357. https://doi.org/10.1016/j.cie.2019.02.038

Darville, J., & Celik, N. (2020a). Microgrid operational planning using deviation clustering within a dddas framework. In *International conference on dynamic data driven application systems*, Boston, MA, USA, ACM. (pp. 77–84). https://doi.org/10.36819/SW21.014 .

Darville, J., & Celik, N. (2020b). Simulation and optimization for unit commitment using a regionbased sampling (rbs) algorithm. In *Iie annual conference. proceedings* (pp. 1424–1430). New Orleans, LA, IISE.

Du, M., Liu, N., & Hu, X. (2019). Techniques for interpretable machine learning. *Communications of the ACM*, *63* (1), 68–77. https://doi.org/10.1145/3359786

Eckman, D. J., & Henderson, S. G. (2021). Fixed-confidence, fixed-tolerance guarantees for ranking-and-selection procedures. *ACM Transactions on Modeling and Computer Simulation (TOM A CS)*, *31*(2), 1–33. https://doi.org/10.1145/3432754

Frazier, P. I., Powell, W. B., & Dayanik, S. (2008). A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, *47*(5), 2410–2439. https://doi.org/10.1137/070693424

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1), 1–67. https://doi.org/10.1214/aos/1176347963

Fu, M. C. (2015). *Handbook of simulation optimization.* Springer.

Gao, S., Chen, W., & Shi, L. (2017). A new budget allocation framework for the expected opportunity cost. *Ope Rations Research*, *65*(3), 787–803. https://doi.org/10.1287/opre.2016.1581

Gao, S., Du, J., & Chen, C.-H. (2019b). Selecting the optimal system design under covariates. In *2019 ieee 15th international conference on automation science and engineering (case)* (pp. 547–552).Vancouver, Canada, IEEE.

Griewank, A. O. (1981). Generalized descent for global optimization. *Journal of Optimization Theory and Applications*, *34*(1), 11–39. https://doi.org/10.1007/BF00933356

He, D., Chick, S. E., & Chen, C.-H. (2007). Opportunity cost and ocba selection procedures in ordinal optimization for a fixed number of alternative systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and R Eviews)*, *37*(5), 951–961. https://doi.org/10.1109/TSMCC.2007.900656

Hong, J. L., Fan, W., & Luo, J. (2021). Review on ranking and selection: A new perspective. *Frontiers of Engineering Management*, *8*(3), 321–343. https://doi.org/10.1007/s42524-021-0152-6

Hong, L. J., & Jiang, G. (2019). Offline simulation online application: A new framework of simulation-based decision making. *Asia-Pacific Journal of Ope Rational Research*, *36*(6), 1940015. https://doi.org/10.1142/S0217595919400153

Hsieh, B.-W., Chen, C.-H., & Chang, S.-C. (2007). Efficient simulation-based composition of scheduling policies by integrating ordinal optimization with design of experiment. *IEEE Transactions on Automation Science and Engineering*, *4*(4), 553–568. https://doi.org/10.1109/TASE.2007.906342

Jiang, G., Hong, L. J., & Nelson, B. L. (2020). Online risk monitoring using offline simulation. *INFORMS Journal on Computing*, *32*(2), 356–375. https://doi.org/10.1287/ijoc.2019.0892

Kasaie, P., & Kelton, W. D. (2013). Simulation optimization for allocation of epidemic-control resources. *IIE Transactions on Healthcare Systems Engineering*, *3*(2), 78–93. https://doi.org/10.1080/19488300.2013.788102

Kim, S.-H., & Nelson, B. 2006. Selecting the best system. In S. H. SG, and B. Nelson (Eds.), *Elsevier hand books in operations research and management science: Simulation.* Elsevier. Chapter 17 p.

LaPorte, G. J., Branke, J., & Chen, C.-H. (2012). Optimal computing budget allocation for small computing budgets. In *Proceedings of the 2012 winter simulation conference (wsc)* (pp. 1–13).Berlin, Germany, IEEE.

LaPorte, G. J., Branke, J., & Chen, C.-H. (2015). Adaptive parent population sizing in evolution strategies. *Evolutionary Computation*, *23*(3), 397–420. https://doi.org/10.1162/EVCO_a_00136

Lin, Y., Nelson, B. L., & Pei, L. (2019). Virtual statistics in simulation via k nearest neighbors. *INFORMS Journal on Computing*, *31*(3), 576–592. https://doi.org/10.1287/ijoc.2018.0839

Luo, J., Hong, L. J., Nelson, B. L., & Wu, Y. (2015). Fully sequential procedures for large-scale ranking-and-selection problems in parallel computing environments. *Operations Research*, *63*(5), 1177–1194. https://doi.org/10.1287/opre.2015.1413

Molnar, C. (2020). *Interpretable machine learning*. Lulu. com.

Ni, E. C., Ciocan, D. F., Henderson, S. G., & Hunter, S. R. (2017). Efficient ranking and selection in parallel computing environments. *Operations Research*, *65*(3), 821–836. https://doi.org/10.1287/opre.2016.1577

Pearce, M., & Branke, J. (2017). Efficient expected improvement estimation for continuous multiple ranking and selection. In *2017 winter simulation conference (wsc)* (pp. 2161–2172).Las Vegas, NV, USA, IEEE.

Pedrielli, G., Selcuk Candan, K., Chen, X., Mathesen, L., Inanalouganji, A., Xu, J., … Lee, L. H. (2019). Generalized ordinal learning framework (golf) for decision making with future simulated data. *Asia-Pacific Journal of Operational Research*, *36*(6), 1940011. https://doi.org/10.1142/S0217595919400116

Peng, Y., Xu, J., Lee, L. H., Hu, J., & Chen, C.-H. (2019). Efficient simulation sampling allocation using multifidelity models. *IEEE Transactions on A Utomatic Control*, *64*(8), 3156–3169. https://doi.org/10.1109/TAC.2018.2886165

Pickardt, C., Branke, J., Hildebrandt, T., Heger, J., & Scholz-Reiter, B. (2010). Generating dispatching rules for semiconductor manufacturing to minimize weighted tardiness. In *Proceedings of the 2010 winter simulation conference* (pp. 2504–2515). Baltimore, MD, USA, IEEE.

Qiu, Y., & Song, J. (2016). A multiobjective simulation optimization of the macrolevel patient flow distribution. *Healthcare Analytics*, 303.Wiley. https://doi.org/10.1002/9781118919408.ch10

Ryzhov, I. O., Powell, W. B., & Frazier, P. I. (2012). The knowledge gradient algorithm for a general class of online learning problems. *Operations Research*, *60*(1), 180–195. https://doi.org/10.1287/opre.1110.0999

Santner, T. J., Williams, B. J., Notz, W. I., & Williams, B. J. (2018). *The design and analysis of computer experiments* (Vol. 2). Springer.

Shen, H., Hong, L. J., & Zhang, X. (2021). Ranking and selection with covariates for personalized decision making. *INFORMS Journal on Computing*, 33(4). https://doi.org/10.1287/ijoc.2020.1009

Shi, X., & Celik, N. (2016). Relative entropy-based density selection in particle filtering for load demand forecast. *IEEE Transactions on Automation Science and Engineering*, *14*(2), 946–954. https://doi.org/10.1109/TASE.2016.2552221

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., & Van Den Driessche, G. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489. others https://doi.org/10.1038/nature16961

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., & Guez, A. (2017). Mastering the game of go without human knowledge. *Nature*, *550*(7676), 354–359. others https://doi.org/10.1038/nature24270

Song, J., Qiu, Y., Xu, J., & Yang, F. (2019). Multi-fidelity sampling for efficient simulation- based decision making in manufacturing management. *IISE Transactions*, *51*(7), 792–805. https://doi.org/10.1080/24725854.2019.1576951

Taghiyeh, S., & Xu, J. (2016). A new particle swarm optimization algorithm for noisy optimization problems. *Swarm Intelligence*, *10*(3), 161–192. https://doi.org/10.1007/s11721-016-0125-2

Thanos, A. E., Bastani, M., Celik, N., & Chen, C.-H. (2015). Dynamic data driven adaptive simulation framework for automated control in microgrids. *IEEE Transactions on Smart Grid*, *8*(1), 209–218. https://doi.org/10.1109/TSG.2015.2464709

Xu, J., Huang, E., Hsieh, L., Lee, L. H., Jia, Q.-S., & Chen, C.-H. (2016). Simulation optimization in the era of industrial 4.0 and the industrial internet. *Journal of Simulation*, *10*(4), 310–320. https://doi.org/10.1057/s41273-016-0037-6

Xu, J., Vidyashankar, A., & Nielsen, M. (2014). Drug resistance or re-emergence? simulating equine parasites. *ACM Transactions on Modeling and Computer Simulation*, *24*(4), 201–2023. https://doi.org/10.1145/2627736

Xu, J., Yao, R., & Qiu, F. (2020). Mitigating cascading outages in severe weather using simulation-based optimization. *IEEE Transactions on Rer Systems*, *36*(1), 204–213. https://doi.org/10.1109/TPWRS.2020.3008428

Yavuz, A., Darville, J., Celik, N., Xu, J., Chen, C.-H., Langhals, B., & Engle, R. (2020). Advancing self-healing capabilities in interconnected microgrids via dynamic

data driven applications system with relational database management. In *2020 winter simulation conference (wsc)* (pp. 2030–2041). IEEE.

Zhang, F., Song, J., Dai, Y., & Xu, J. (2020). Semiconductor wafer fabrication production planning using multi-fidelity simulation optimisation. *International Journal of Production Research*, *58*(21), 6585–6600. https://doi.org/10.1080/00207543.2019.1683252

Zhong, Y., & Hong, J. (2021). Knockout-tournament procedures for large-scale ranking and selection in parallel computing environments. *Operations Research*, 70(1), 432–453. https://doi.org/10.1287/opre.2020.2065

Zhong, Y., Liu, S., Luo, J., & Hong, L. J. (2021). Speeding up paulson's procedure for large-scale problems using parallel computing. *INFORMS Journal on R Computing*. https://doi.org/10.1287/ijoc.2020.1054

Zhou, C., Xu, J., Miller-Hooks, E., Zhou, W., Chen, C.-H., Lee, L. H., & Li, H. (2021). Analytics with digital-twinning: A decision support system for maintaining a resilient port. *Decision Support Systems*, *143*, 113496. https://doi.org/10.1016/j.dss.2021.113496

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, *67* (2), 301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x