Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

ADAPTIVE INTELLIGENCE: COMBINING DIGITAL TWIN PRECISION WITH AI FORESIGHT FOR REAL-TIME DECISION MAKING

Chun Hung Chen[1], Jie Xu[1], Enver Yücesan[2]

[1]Dept. of Systems Eng. and Operations Research, George Mason University, Fairfax, VA, USA
[2]Technology and Operations Management Area, INSEAD, Fontainebleau, FRANCE

ABSTRACT

Digital Twins, virtual models that accurately replicate the dynamic behavior of physical systems, are on the cusp of a revolutionary transformation not only for the design but also for real-time control of dynamic systems. In this tutorial, we define the key building blocks of Digital Twins and explore the key challenges they face in enabling real-time decision making. We discuss adaptive intelligence, an innovative reasoning framework that combines AI-driven approaches with simulation optimization techniques to enhance real-time decision making capabilities –hence, the value– of Digital Twins.

1 INTRODUCTION

Computer simulation has become "a tool of *first* resort" mainly due to the confluence of impressive advances in computing hardware and significant developments in stochastic optimization techniques. As noted by Laidler, Nelson, and Pavlidis (2025), "big data, big computing, and big consequences are pushing stochastic simulation beyond its typical role of creating static system designs and good long-run average performance." Such an opportunity is most visible in Digital Twins, where computer simulation is bound to assume an even more central role in real-time decision making. A digital twin (DT) "is a set of virtual information constructs that mimics the structure, context, and behavior of a natural, engineered, or social system (or system of systems), is dynamically updated with data from its physical twin, has a predictive capability, and informs decisions that realize value" (National Academies of Sciences and Medicine 2024). DTs broadly consist of four main components (dos Santos, Montevechi, Campos, Miranda, Queiroz, and Amaral 2024): `physical systems` with humans, materials, and processes; `virtual systems` consisting of models of physical systems; `service systems` enabling simulation, optimization, and communication between the physical and virtual systems; and `data systems` reflecting the information transmitted between systems to enable efficient decision making.

Infrastructure technologies (in both hardware and software) have made it possible to automate data collection from physical systems, enabling the construction of digital models (Lugaresi 2024) and their parametrization for simulation and experimentation (Cen, Herbert, and Haas 2020). Such digital objects, typically referred to as *digital shadows*, receive real-time data from the physical system; however, any operational modification recommended by the simulation executed on the digital object must be incorporated *manually* into its physical counterpart (Kritzinger, Karner, Traar, Henjes, and Sihn 2018). Emerging technologies are rapidly closing the loop for the automatic modification of operational parameters in the physical system, giving rise to *digital twins*.

DTs are developed to ensure information continuity throughout a system's life cycle, enable virtual commissioning, perform real-time monitoring, view, analyze and control process state, predict system behavior, and generate optimal operating policies. As depicted in Figure 1, the physical system provides its digital twin with real-time data, improving the fidelity of the simulation models, while predictions and decisions based on these models are provided to the physical system practically on a real-time basis, modifying its operations (Zhou 2024). There exist nevertheless several challenges for on-line decision-

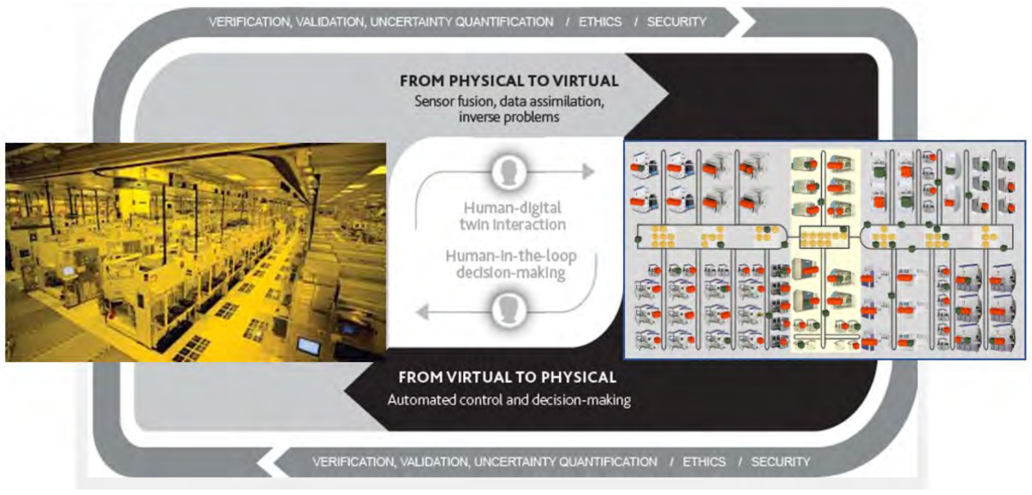
VERIFICATION, VALIDATION, UNCERTAINTY QUANTIFICATION / ETHICS / SECURITY
FROM PHYSICAL TO VIRTUAL
Sensor fusion, data assimilation, inverse problems
Human-digital twin interaction
Human-in-the-loop decision-making
FROM VIRTUAL TO PHYSICAL
Automated control and decision-making
VERIFICATION, VALIDATION, UNCERTAINTY QUANTIFICATION / ETHICS / SECURITY

Figure 1: The relationship between the physical system and its digital twin.

making in DTs, including modeling, analysis, and optimization. The objective of this tutorial is to address the challenges that arise in synchronizing the physical and digital systems in a data-driven fashion to enable real-time decision making. In particular, we will focus on the complementary roles AI/ML and simulation optimization can play in enhancing the efficiency of modeling processes in the `virtual system` and the efficacy of decision support in the `service systems`.

For modeling, in addition to building valid simulation models and verifying their correct execution, it is also necessary to maintain the fidelity of these models over time. In other words, while validation and verification aim to ensure the satisfactory mapping between the physical and virtual systems and the correct functioning of the computational implementation, respectively, accreditation is necessary to evaluate the model's usability in guiding decisions (Sargent, Goldsman, and Yaacoub 2016; Sargent 2020).

For analysis and optimization, a DT must generate optimal operating policies by rapidly evaluating alternatives through its simulation optimization algorithms to support real-time decision making for guiding its physical counterpart (Zhou 2024). However, the duration of a simulation run depends on the complexity of the modeled system, which contributes to the computational load, particularly when a large number of potential solutions must be evaluated. Thus, determining optimal policies for large-scale systems through "naive" simulation becomes inefficient, highlighting the need for reasoning that combines AI/ML techniques with simulation-based optimization methods to expedite the decision making process.

Modeling and optimization challenges are particularly acute in applications with high complexity, high uncertainty, and limited data where dynamics underlying the system are not well understood —consequently only crudely approximated by the digital models. As a result, the DT should be calibrated to reflect the physical system as accurately as possible before it can be optimized to identify an effective control policy for the physical system (Zheng, Xie, Ryzhov, and Choy 2025). As a result, there has been growing interest both in the broader research community and within the Winter Simulation Conference (WSC) in the role artificial intelligence and machine learning (AI/ML) can play in enhancing the real-time decision making capabilities of DT.

This tutorial is organized as follows: Section 2 provides a formal characterization of real-time decision making in DTs. Approaches for stochastic optimization via simulation are also discussed. Section 2.2 proposes the *adaptive intelligence* framework by discussing the synergies between AI/ML and simulation for reasoning to support real-time decision making in DT. Section 4 closes the tutorial by highlighting promising venues for future research and development.

2 FORMAL CHARACTERIZATION

Traditionally, the digital object on the right-hand side of Figure 1, typically a simulation model, uses two sets of inputs, including a control policy governing the behavior of the system, X, and the context in the form of data from the physical system, Y. The execution of the simulation model, in turn, yields, as the simulation output, an estimate of the performance of the system, $J(X,Y)$ given the specific context. The objective of the simulation experiment is to determine the optimal decision (say, the best policy) given the *context*, namely

$$X^*(Y) = \arg\min_{X\in\Theta} J(X,Y),$$

where Θ is the set of potential policies. Since the performance of interest does not typically have a closed-form functional representation, it is estimated as the sample average of a number (say, N) of independent Monte Carlo samples, W_ℓ:

$$J(X,Y) = E_W[f(X,W,Y)] \approx \frac{1}{N}\sum_{\ell=1}^{N} f(X,W_\ell,Y),$$

where f is the performance function that does not admit a closed form.

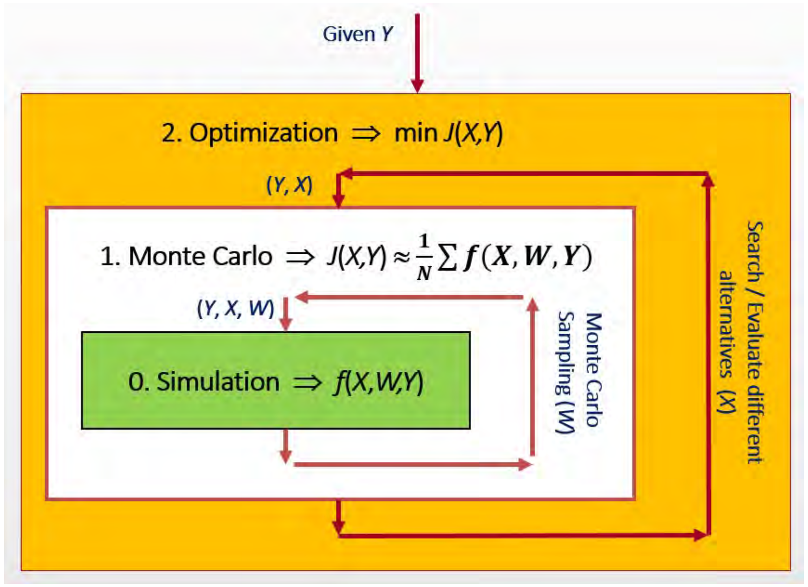
Given Y
2. Optimization ⇒ min J(X,Y)
(Y, X)
1. Monte Carlo ⇒ J(X,Y) ≈ 1/N Σ f(X,W,Y)
(Y, X, W)
0. Simulation ⇒ f(X,W,Y)
Monte Carlo Sampling (W)
Search / Evaluate different alternatives (X)

Figure 2: Simulation optimization within a context.

Within the context of simulation optimization, ranking and selection (R&S) aims at identifying the best policy among a possibly large, but finite, set of competing policies through noisy experiments. As DTs must provide the physical system with an optimal operating policy practically in real time, we formulate R&S algorithms under a fixed-budget setting aiming at efficiently allocating the available sampling budget to alternative policies. To maximize the quality of the final selection, the exploration/exploitation trade-off must be managed in an effective fashion. As illustrated in Figure 2, a larger sampling budget for Monte Carlo simulation (inner loop (0 and 1): exploitation) enhances estimation accuracy while a larger search budget (outer loop (2): exploration) enables the evaluation of a larger number of alternative policies, moving the decision maker closer to optimality.

2.1 A Brief Recap of OCBA

To maximize decision quality, typically expressed in terms of the probability of correct selection (PCS), Optimal Computing Budget Allocation (OCBA) *adaptively* manages the exploration/exploitation trade-off. To this end, Chen, Lin, Yücesan, and Chick (2000) solve the following optimization problem

$$\max_{N_1,N_2,\ldots,N_k} \quad PCS$$
$$\text{subject to} \quad \sum_{i=1}^{K} N_i = N,$$

where N_i represents the simulation budget allocated to policy i and N is the total simulation budget. The asymptotically optimal budget allocation scheme follows the following ratios:

$$\begin{aligned} \frac{N_i}{N_j} &= \frac{(\frac{\sigma_i}{\delta_{b,i}})^2}{(\frac{\sigma_j}{\delta_{b,j}})^2}, \quad \text{for} \quad i \neq j \neq b \\ N_b &= \sigma_b \sqrt{\sum_{i \neq b} \frac{N_i^2}{\sigma_i^2}}, \end{aligned} \tag{1}$$

where σ_i is the standard deviation of the performance associated with policy i and $\delta_{b,i}$ is the "distance" between the expected performance of the "best" policy b and that of policy i. In other words, the computational budget allocated to a policy is directly proportional to its variance and inversely proportional to the square of its distance from the current best policy. Ryzhov (2016) asserts that OCBA yields optimal performance. The speedup achieved by OCBA is demonstrated next through with a small illustration.

Example. Consider a two-stage tandem queueing system where jobs arrive to the first queue according to a Poisson process with unit intensity. The processing times are random variables distributed uniformly U(1, 39) at the first stage, and U(5, 45) at the second stage. The objective is to allocate 31 servers across the two stages so as to minimize the average waiting time of the jobs. In addition, a constraint is imposed whereby each stage should be allocated at least 11 servers. For a 99% probability of correct selection (PCS), Table 1 shows the speed up factor achieved by OCBA with respect to the traditional allocation procedures that are proportional to estimator variance (Rinott 1978).

Table 1: Speedup factor of OCBA over traditional procedures for PCS = 99%.

No. of alternatives	4	10	20	50	75	100
Speedup	1.7	3.5	6.5	12.8	16.3	19.6

2.2 Contextual R&S

Note that while OCBA is 3 to 20 times faster than traditional R&S procedures, such speed up may still not be sufficient for real-time decision making. Moreover, this speed up is obtained within a *given* context, Y. Contextual R&S, also known as R&S with covariates, extends ranking and selection to *personalized* decision-making, in which the best policy is not universal, but varies as a function of some observable covariates that characterize the specific context. The goal of contextual R&S is to devise a selection process that identifies the best alternative once the context is revealed. The context therefore introduces a third layer of complexity, namely generating a sufficient number of contexts to ensure coverage, in addition to the exploration/exploitation trade-off.

Contextual R&S therefore consists of an offline and an online stage (Hong and Jiang 2019). Given the tight time frame for online decision-making, a large portion of the literature focuses on exploration during the offline stage. As a result, we can label the exploration phase as the *learning stage*. Within the learning stage, the literature can be further categorized into two streams based on their representation of the context space. The first stream assumes a discrete context space with moderate cardinality, M, allowing exhaustive sampling of all $K \times M$ design-context combinations. For instance, Gao, Du, and Chen (2019) derive the optimal budget allocation ratios analogous to Equation (1) for constructing an adaptive procedure by maximizing an aggregated metric, namely

$$\begin{aligned} \max_{N_{i_m} \geq 0} \quad & \sum_{i=1}^{M} p_m PCS_m \\ \text{subject to} \quad & \sum_{i=1}^{K} \sum_{m=1}^{M} N_{i_m} = N, \end{aligned}$$

where p_m denotes the likelihood of observing a particular context, N_{i_m} represents the simulation budget allocated to a design-context combination, and N is the total simulation budget. Gao, Du, and Chen (2024) illustrate this method through an application to personalized medicine. Similarly, the dynamic procedure of Li, Lam, and Peng (2024) leverages the Gaussian mixture model to exploit cluster information; Shi, Peng, and Zhang (2023) adapt Top-Two Thompson Sampling and show asymptotic optimality under certain conditions. Further examples under this setting include Jin, Li, and Lee (2019), Alban, Chick, and Zoumpoulis (2021), Çakmak, Wang, Gao, and Zhou (2024), and Zhang, Chen, Huang, and Peng (2024).

The second stream deals with a continuous context space where exhaustive sampling is no longer possible. To overcome this challenge, most authors first construct metamodels to characterize the underlying mean performance function of each design over the context space before developing decision-making models. For example, Hu and Ludkovski (2017), Pearce and Branke (2017), and Ding, Hong, Shen, and Zhang (2022) utilize a Gaussian process to model the mean performance function. To guide the sampling process, popular strategies from Bayesian optimization such as the Upper Confidence Bound (UCB), Expected Improvement (EI), and Knowledge Gradient (KG) are widely employed alongside various approximation techniques to explore the continuous context space. Subsequently, on-line decision making is simply reduced to plugging the observed covariates into the metamodel to evaluate performance. In contrast, Keslin, Nelson, Plumlee, Pagnoncelli, and Rahimian (2022) first select a finite set of contexts to conduct traditional R&S at the selected contexts and use the k-nearest neighbors method to directly construct the decision-making model. In the next section, we propose to enrich the on-line decision making stage through adaptive intelligence that harnesses the synergies between AI/ML and simulation optimization —in particular, to handle the settings where the observed covariates are different from what had been evaluated during the off-line stage necessitating on-line reasoning, as illustrated in Figure 3.

3 REASONING WITH DTS: EXPLOITING THE SYNERGIES BETWEEN SIMULATION AND AI/ML

The key feature that distinguishes a DT from conventional simulations is the bi-directional interaction that exists between the two systems: the physical system continually provides its digital twin with real-time data, improving the fidelity of the simulation models, while predictions and decisions based on these models are provided to the physical system, modifying its operations (Zhou 2024). DTs therefore present significant challenges for simulation modeling, analysis, and optimization. To create the digital system, in addition to building valid simulation models and verifying their correct execution, it is also necessary to maintain the fidelity of these models over time; thus, model calibration is of paramount importance. In return, to support real-time decision making, a DT must rapidly complete its evaluation by executing its simulation optimization algorithms for guiding its physical counterpart.

To address these challenges, AI/ML techniques offer significant support even though they might, at first glance, appear incompatible with simulation modeling and analysis: simulation explicitly models the

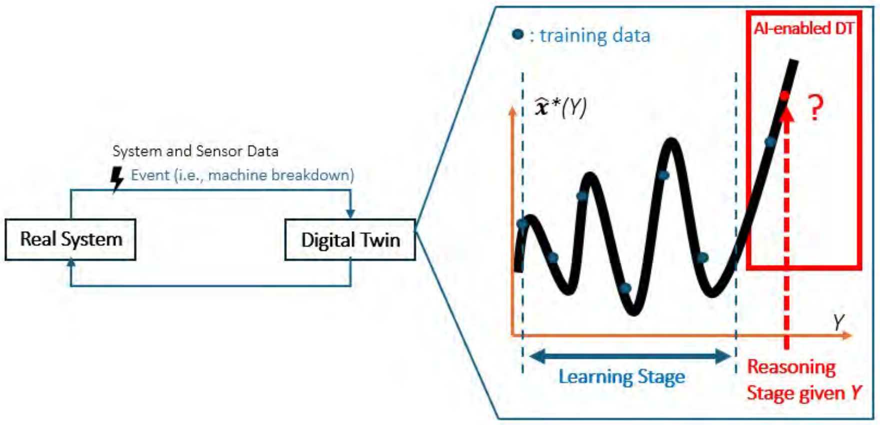

Figure 3: Ranking and Selection with On-Line Reasoning.

logic of a system that maps input parameters into the output performance of interest while ML models are mostly black boxes that fit a parameterized statistical model for input-output mapping (Haas 2024). There exist, however, significant synergies between the two techniques, as depicted in Figure 4, including the generation of scenarios to be explored, the evaluation of scenario robustness, the construction of robust metamodels, and the determination of model fidelity.

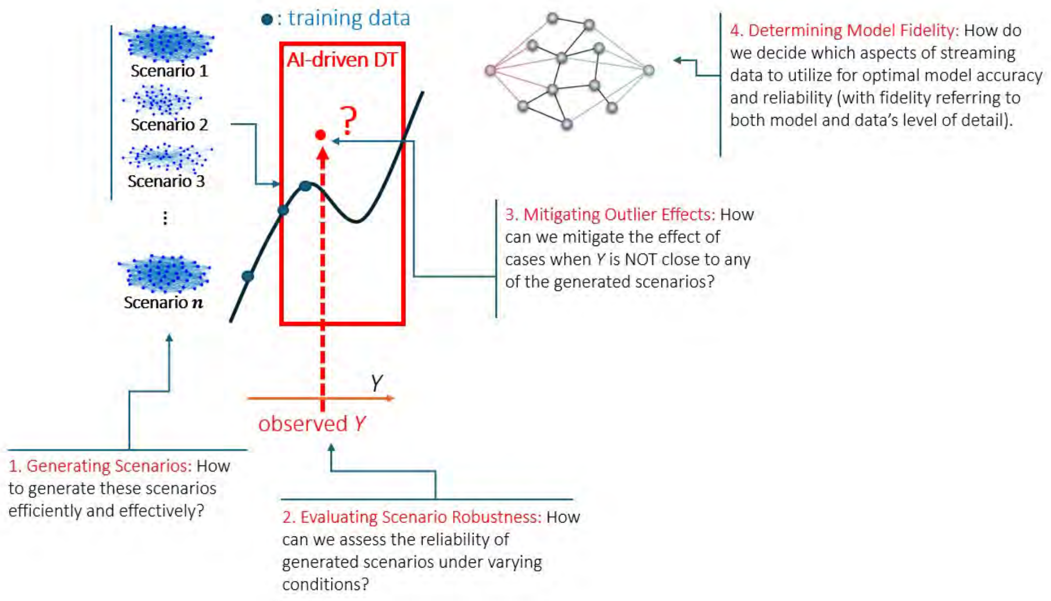
● : training data
Scenario 1
Scenario 2
Scenario 3
Scenario n
AI-driven DT
?
Y
observed Y
1. Generating Scenarios: How to generate these scenarios efficiently and effectively?
2. Evaluating Scenario Robustness: How can we assess the reliability of generated scenarios under varying conditions?
3. Mitigating Outlier Effects: How can we mitigate the effect of cases when Y is NOT close to any of the generated scenarios?
4. Determining Model Fidelity: How do we decide which aspects of streaming data to utilize for optimal model accuracy and reliability (with fidelity referring to both model and data's level of detail).

Figure 4: AI/ML's impact on metamodeling.

Following a brief overview of the potential contributions of AI/ML to the generation of well-calibrated digital models, we will focus on the potential synergies between AI/ML and simulation optimization, particularly on enhancing the ability of the digital system to enable real-time decision making in its physical counterpart. To that end, we first discuss, in Section 3.1, how AI/ML is teamed up with simulation optimization techniques such as Monte Carlo Tree Search (MCTS) to create a world-class *Go* player. In

Section 3.2, we then review in greater detail an *adaptive intelligence* concept for leveraging multi-fidelity information via a flexible algorithmic framework to implement efficient real-time decision making in DTs.

Neural networks (NN) have been developed to improve parameter estimation for input modeling to drive simulations (Cen, Herbert, and Haas 2020). Miao, Huang, and Hu (2025) empirically investigate the efficacy of generative models such as Generative Adversarial Networks (GAN), Variational Autoencoders (VAE), Normalizing Flows, and Diffusion Models for capturing complex dependencies in input models that drive simulations. NNs have also been deployed to enhance metamodeling of the performance function of interest (Cen and Haas 2022). do Amaral, Montevechi, de Carvalho Miranda, and de Sousa Junior (2022) provide a comprehensive overview of various metamodel-based simulation optimization methods and their applications. For instance, polynomial regression is the most common approach in practice (Li, Ng, Xie, and Goh 2010). However, polynomials offer poor fidelity for highly nonlinear, multidimensional problems. In such settings, nonlinear functions such as Kriging and radial basis functions (RBFs) are recommended (Parnianifard and Azfanizam 2020). Advances in learning-based algorithms such as Support Vector Machines (SVM), Decision Trees (DST), Random Forests (RF) have made ML a viable approach for metamodeling (Mohammad Nezhad and Mahlooji 2014). Early experiments with large language models (LLM) provide encouraging results for generative AI-assisted simulation model generation (Carreira-Munich, Paz-Marcolla, and Castro 2024). In the same spirit, Matta and Lugaresi (2023) further extend process mining for automated model generation.

Once adequately trained, NNs are extremely efficient in supporting real-time decision making (Wang, Kallus, and Sun 2024). For instance, by implementing a recurrent neural network inspired simulation approach, Wang and Hong (2022) extend the analysis and optimization of a capacitated multi-echelon production-inventory system under a base stock policy, which was originally analyzed by Glasserman and Tayur (1994) and Glasserman and Tayur (1995), to an industrial scale. Similarly, graph neural networks have emerged as a highly effective deep learning architecture for combinatorial optimization problems (Norman, Dawadi, and Yedidsion 2024; Soykan and Rabadi 2024). Kumar, Peng, Wu, and Zeevi (2025) evaluate the ability of large-language models (LLM) to solve stochastic modeling problems by evaluating the solutions generated by LLM models to graduate-level homework and doctoral qualifying exam questions.

Training NNs, however, requires significant quantities of data, which may not be readily available from the physical system. Computer simulation may play a key role here. For instance, to efficiently manage the bi-directional interaction between the physical and digital systems, Zheng, Xie, Ryzhov, and Choy (2025) use model-based reinforcement learning (MBRL) specifically to construct digital models of systems whose dynamics are not fully understood. More specifically, they consider a setting where a DT is specified by a finite number of calibration parameters, which are set based on a limited amount of expensive physical experimental data.Using an actor-simulator framework, the control policy obtained by simulating the digital twin is used to guide the acquisition of new data from the physical system with the overall goal of determining an optimal control policy for the physical system. In other words, the approach alternates between calibrating the digital twin, identifying information to be acquired in the next experiment, and learning an RL policy to optimize the physical system. Their framework is anchored on an uncertainty function quantifying the discrepancy between the two systems with respect to the RL objective of maximizing the cumulative discounted reward. During calibration, the next experiment is selected to maximize the uncertainty function; in contrast, during policy optimization, the reward function is augmented by penalizing high-uncertainty actions. This approach results in a targeted exploration of the state-action space, identifying those regions that yield the greatest benefit for policy optimization.

The potential for collaboration between simulation optimization and NNs is illustrated in Figure 5, where the y-axis, *Learning Coverage*, reflects the intensity of training the NN has received while the x-axis shows the remaining *Time to the Decision Point* corresponding to the computational budget available for the simulation optimization algorithms. The way ML and simulation optimization are combined to enable real-time decision making is illustrated in further detail over the next two subsections.

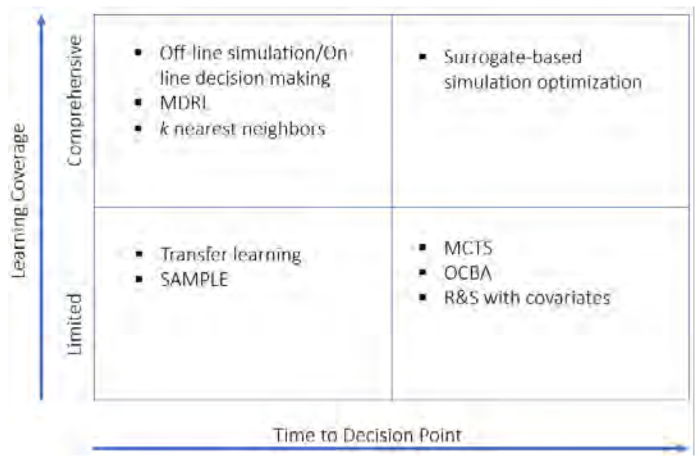

Figure 5: Adaptive intelligence for real-time decision-making.

3.1 How "Deepminded" is AI for Decision Making: from Monte Carlo Tree Search to AlphaGo

Players of the board game *Go* face a tremendously large decision space at every turn. To put the depth of the decision space into context, a player can choose, on average, between 150 to 250 possible moves per turn. In comparison, chess players have an average number of 37 moves per turn. Furthermore, the number of moves in *Go* typically ranges from 180 to 250 compared to 40 to 60 moves in a chess game. The number of possible board configurations in *Go* is estimated to be around 10^{170}! In comparison, the number of atoms in the observable universe is estimated to be between 10^{78} and 10^{82}. A *Go* player or a *Go* computer program also needs to make a decision in a fairly short amount of time. In major international tournaments such as the Samsung Cup, each player has 3 hours of main time. After using the main time, the player enters five periods of 60-second *byo-yomi* within which a move must be made. If the player takes more than 60 seconds to make the move, the current period expires and a new period starts. After the full five minutes are exhausted, the player must ultimately make a move within the next 60 seconds; otherwise, the player loses the game. Because of the complexity and the depth of the decision space, a human-expert level *Go* computer program had long been regarded as out of reach in the foreseeable future before the development of deep neural networks (DNN).

A board game like *Go* represents an ideal DT environment in many important ways. The state of the system, i.e., the board configuration, is known perfectly to the virtual system along with the rules of the game. Conceptually, the virtual system can evaluate the outcome of the game for all possible moves and select the winning move. The information flow from the virtual system to its physical twin is also trivial. A human assistant simply places a piece on the game board following the "optimal" decision made by the DT. Therefore, the critical task of the DT is to efficiently evaluate the strength of each possible move in an extremely large game space.

Monte Carlo Tree Search (MCTS) is a depth-first tree search algorithm using Monte Carlo samples, designed to effectively address the very large number of moves per turn (Fu, Qiu, and Xu 2024). Before the widely publicized success of Google DeepMind's AlphaGo (Silver, Huang, Maddison, Guez, Sifre, Van Den Driessche, Schrittwieser, Antonoglou, Panneershelvam, Lanctot, et al. 2016) that beat a top human *Go* player, Lee Sedol, in 2016 (referred to as AlphaGo Lee), the most powerful commercial *Go* programs such as Crazy Stone (Coulom 2007), Fuego (Enzenberger, Müller, Arneson, and Segal 2010), and Pachi (Baudiš and Gailly 2011) were based on MCTS. These MCTS-driven *Go* programs represent the application of simulation optimization for DT-based decision making, as captured in the bottom right quadrant of Figure 5. However, these MCTS-driven *Go* programs require a considerable amount of computational time to simulate the game tree before a strong move can be identified. Without the assistance of AI, they only achieve strong amateur (human) player level.

Before the arrival of DNNs, earlier efforts to use neural networks to predict moves only led to computer *Go* programs with very limited play strength. The deep convolutional neural network-based (DCNN) *Go* program of Clark and Storkey (2015) advanced this "pure" AI approach, using a DCNN trained in a supervised learning (SL) pipeline to predict a move in any state of the game board. While the DCNN achieves fairly strong predictive accuracy, the actual playing strength is limited and turns out to be slightly worse than MCTS-based *Go* programs like Fuego. Nontheless, DCNN *Go* is much faster in making a move than MCTS *Go* programs. Therefore, despite the sophisticated AI used, due to the limited training relative to the extremely large game space, the DCNN *Go* program only represents the bottom left quadrant of Figure 5.

The phenomenal success of AlphaGo serves as a compelling demonstration of *adaptive intelligence*, a sophisticated form of AI characterized by its capacity to attain *"superhuman performance"* by adapting the AI models' predictions to the current state of the game, using real-time insights generated through MCTS simulations. AlphaGo thus represents the top right quadrant of Figure 5. In AlphaGo, the `service system` of the DT includes a policy network $p_\sigma(a|s)$ trained from SL pipeline and a value network $v_\theta(s)$, which predicts the winning rate from a state s, and the MCTS algorithm. The value network $v_\theta(s)$ is derived using self-play data generated with an improved policy network $p_\rho(a|s)$. The policy network $p_\rho(a|s)$ is trained using reinforcement learning starting from $p_\sigma(a|s)$. The improved $p_\rho(a|s)$ has a winning rate of 80% against $p_\sigma(a|s)$ and is thus used to generate better training data for the value network $v_\theta(s)$ using self-play.

AlphaGo trained and deployed several DNNs, which played different structural roles from the perspective of the DT framework. A key component of the virtual system is the fast policy network $p_\pi(a|s)$ used by AlphaGo for quickly sampling moves in its simulation of the game tree from a leaf node expanded until the end of the game. This is an instance of using a lightly trained NN to improve both the computational efficiency and the prediction accuracy for the virtual system. Later with a trained value network $v_\theta(s)$ to predict the winning rate from a state s, AlphaGo may terminate the simulation early when the outcome of the game is quite certain to further improve the efficiency of the virtual system of a DT. With the fast policy network, it is expected that MCTS would have better performance, but would still fall short of expert human player level. An earlier version of AlphaGo with a less accurate policy network and value network that defeated European Go champion Hui Fan in 2015 was referred to as AlphaGo Fan. AlphaGo Fan achieved a play strength much higher than using just state-of-the-art AI (DCNN *Go*) and MCTS only. With improved AI training, AlphaGo Lee further improved over AlphaGo Fan.

DeepMind further improved the DNN structure and came up with a reinforcement learning training pipeline that did not use any human knowledge, e.g., without the supervised learning policy network $p_\sigma(s,a)$, and named the new *Go* program AlphaGo Zero (Silver, Schrittwieser, Simonyan, Antonoglou, Huang, Guez, Hubert, Baker, Lai, Bolton, et al. 2017). The new DNN combines the policy network and the value network into a single network, which results in a slightly reduced accuracy for predicting moves, but improved value predictions and better performance when combined with MCTS. The improved AI makes AlphaGo Zero the strongest computer *Go* program reported.

It is worth noting that when the DNN of AlphaGo Zero is used without MCTS, which is mapped to the top left quadrant of Figure 5, its play strength is even slightly worse than that of AlphaGo Fan, although it is considerably stronger than MCTS *Go* programs. From the evolution of computer Go programs, one insight we derive is no matter how "deepminded" AI is, a search process using the virtual system allows the computer program to adapt to the current state of the game board and achieve significant improvement in play strength. Therefore, the efficiency of MCTS and the way MCTS uses information from AI to deliver *adaptive intelligence* have considerable impact on the quality of the decision. Figure 6 depicts the play strength (measured by Elo rating) of these different *Go* programs as summarized in Silver, Schrittwieser, Simonyan, Antonoglou, Huang, Guez, Hubert, Baker, Lai, Bolton, et al. (2017).

To demystify the setting, let us briefly expand on how the MCTS in AlphaGo Lee works. The supervised policy network $p_\sigma(a|s)$ is used in AlphaGo to provide the prior probabilities of moves when a leaf node is

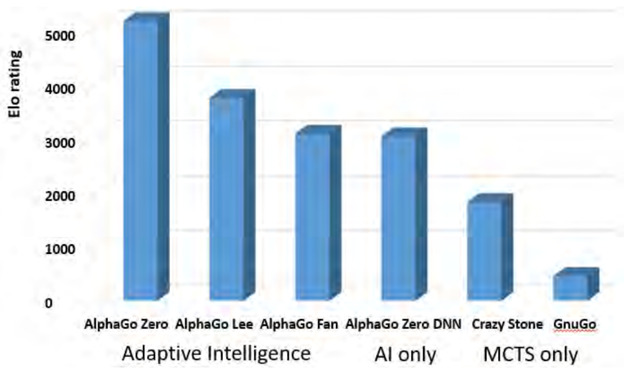

Figure 6: Improving computer Go program by combining MCTS and AI to achieve adaptive intelligence.

expanded. It is worth noting AlphaGo performs better with $p_\sigma(a|s)$ instead of using the improved policy network $p_\rho(a|s)$. For a leaf node s_L, its value $V(s_L)$ is evaluated using both the value network $v_\theta(s)$ and the simulated outcome z_L using the roll-out policy network $p_\pi(a|s)$, in the form of a weighted average:

$$V(s_L) = (1-\lambda)v_\theta(s_L) + \lambda z_L. \tag{2}$$

The MCTS algorithm then back-propagates $V(s_L)$ to update the visit count $N(s,a)$ and the action value $Q(s,a)$ for edges that the algorithm traverses in the simulation:

$$N(s,a) = \sum_{i=1}^{n} I(s,a,i), \quad Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} I(s,a,i)V(s_L^i),$$

where $I(s,a,i)$ is an indicator function that takes the value of one if the ith simulation traverses edge (s,a) and zero otherwise. If at time t, a decision has to be made after n simulations, MCTS uses an upper confidence bound (UCB) type of selection rule to choose an action under the current state s_t:

$$a_t = \arg\max_a \; Q(s_t,a) + \frac{Cp_\sigma(s_t,a)}{1+N(s_t,a)}. \tag{3}$$

We conclude this section with two remarks related to the two key MCTS steps described above. First, equation (3), the MCTS selection policy, is based on the well-known UCB policy for multi-armed bandit problem with an objective to minimize cumulative regret. For the *Go* program as well as for many other applications, the objective is to select the best decision for the current state instead of minimizing cumulative regret. Fu, Qiu, and Xu (2024) showed how an OCBA policy designed for MCTS (Li, Fu, and Xu 2021) can improve the performance of MCTS for applications such as a computer Othello program. Second, equation (2) is a crude way to combine information from AI, $v_\theta(s_L)$, and from the simulation, z_L. It is rigid and involves a hyperparameter λ that requires tuning with no clear guidance. Next, we show how to combine estimates from multiple sources in a flexible and rigorous way to achieve better performance.

3.2 Enhanced Reasoning with DTs

While academic and industrial communities explore ML and deep learning (DL) methods, real-world implementations impose severe constraints on model selection. First, complex models like DNNs are suitable only in data-rich environments, which are not always available. Second, simpler models provide better interpretability, but may sacrifice accuracy, highlighting the trade-off between interpretability and accuracy. Finally, real-time applications in adaptive systems emphasize the importance of computational efficiency and speed. The increasing complexity of physical systems and the inherent stochasticity in operating environments make it challenging to obtain high-quality predictions using AI/ML – particularly in high-dimensional problems. The difficulty is compounded by the expansive decision space, making it even tougher to ensure accurate decision-making with a constrained computing budget.

To address these challenges, Goodwin, Xu, Celik, and Chen (2024) propose the Sequential Allocation using Machine learning Predictions as Light-weight Estimates (SAMPLE) method as another example of *adaptive intelligence*. By integrating simulation optimization with ML predictions, SAMPLE offers a robust framework that combines lightweight ML models with high-accuracy digital twin simulation output to enhance decision-making capabilities for discrete decision spaces. By lightweight ML models, we refer to ML models that are trained offline and deployed online to make predictions in real time, but possibly with substantial prediction errors. In other words, SAMPLE prioritizes lightweight ML models, focusing on ease of training, interpretability, and efficiency in online decision-making. In particular, SAMPLE improves the accuracy of decision-making during real-time simulation optimization by integrating ML predictions with simulation data by efficiently allocating the online sampling budget.

SAMPLE operates in two distinct stages: off-line (simulation) learning and simulation optimization to enable on-line reasoning for real-time decision making, separated by whether the context Y has been observed. The goal in the off-line (simulation) learning stage is to train an appropriate ML model using simulation data that had been collected across various contexts, Y_i, and policy alternatives, X_j. This trained model therefore enables SAMPLE to predict outcomes for different policies without running additional simulations during the real-time optimization stage. Consider Figure 7 where we let $Y_i, i = 1,2,\cdots,n$, represent the contexts and $X_j, j = 1,2,\cdots,m$, denote the policy alternatives. The utility of a policy X_j in a given context Y, denoted as $f(X_j,Y)$, can only be estimated through stochastic simulations. The simulation sample mean $\bar{f}(X_j,Y)$ is assumed to be an unbiased estimator of $f(X_j,Y)$, with the simulation output following a normal distribution with a mean $\bar{f}(X_j,Y)$ and variance σ^2_{ji}. In addition, let $g_k(X_j,Y)$ represent the prediction from the k^{th} ML model fitted during the off-line (simulation) learning step, where $g(X_j,Y) = \{g_1(X_j,Y),\cdots,g_K(X_j,Y)\}$.

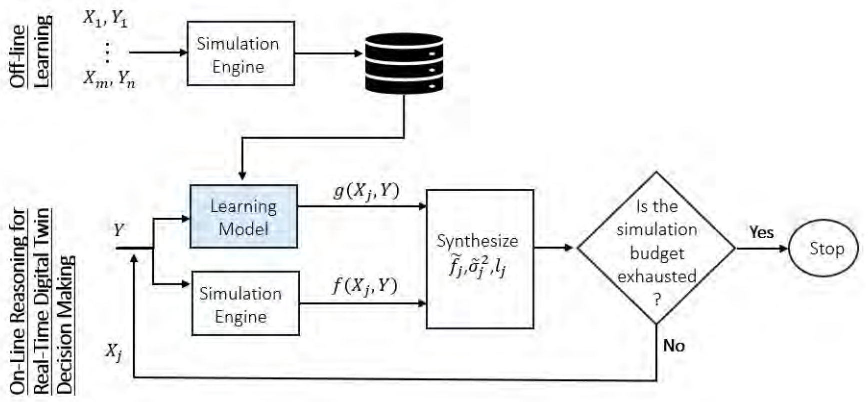
Off-line Learning
X_1, Y_1
X_m, Y_n
Simulation Engine
On-Line Reasoning for Real-Time Digital Twin Decision Making
Y
Learning Model
$g(X_j,Y)$
Simulation Engine
$f(X_j,Y)$
Synthesize $\tilde{f}_j, \tilde{\sigma}_j^2, l_j$
Is the simulation budget exhausted ?
Yes
Stop
No
X_j

Figure 7: The framework of SAMPLE.

The computational efficiency of SAMPLE is grounded in the *ordinal transformation* (OT) concept introduced in Xu, Zhang, Huang, Chen, Lee, and Çelik (2014) and Xu, Zhang, Huang, Chen, Lee, and Çelik (2017). As shown in the right side of Figure 8, alternative decisions X in a discrete decision space are evaluated using an ML model and ranked based on the ML predictions. Compared to the indexing scheme of decisions based on their covariates shown on the left side of Figure 8, the ordinal space reveals the clustering of decisions based on their performance. It is worth noting that the clustering of decisions based on the ML predictions are well aligned with the clustering based on the high-fidelity simulation results despite possibly significant errors in ML predictions. OT thus provides a flexible and robust way to exploit ML to improve the efficiency of simulation optimization techniques that use expensive DT simulations.

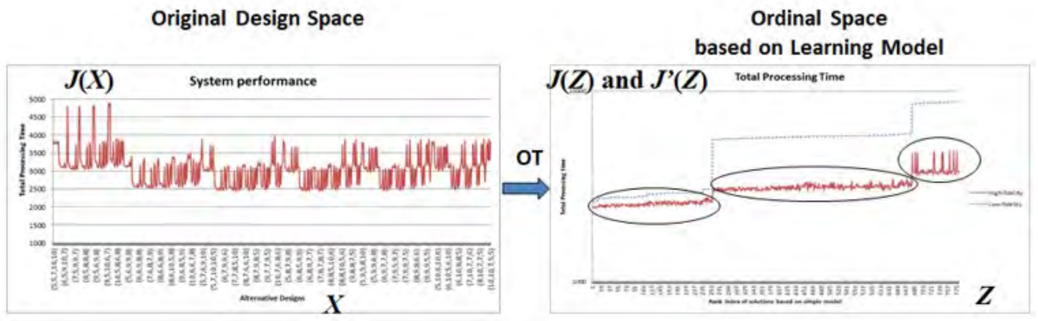

Figure 8: Overview of ordinal transformation.

The details of the mathematical development are provided in (Goodwin, Xu, Celik, and Chen 2024). Here, we focus on the implementation of SAMPLE in large-scale DTs. The decision-maker specifies three key parameters for the online sampling stage: the total simulation budget N, the initial number of simulation replications allocated to each policy n_0, and the number of simulation samples allocated per online iteration ΔN. Once the context Y is observed and the set of policy alternatives $\{X_1,\cdots,X_m\}$ are known, ML predictions $\{g_j : j=1,\cdots,m\}$ can be quickly obtained using the fitted model from the previous stage. With this information, a Gaussian mixture model is then estimated to represent the distribution of outcomes for the policy alternatives. As illustrated in Figure 7, SAMPLE proceeds as follows:

Step 0: Initialize the parameters by setting $n_j = n_0$ for $j=1,\cdots,m$, the iteration counter $t=1$, and the expended simulation budget $N_t = mn_0$.

Step 1: Simulate $X_j, j=1,\cdots,m$, for n_j independent replications. Compute the sample mean $\bar{f}_j$ and the sample variance $\hat{\sigma}_j^2$ for each decision alternative X_j.

Step 2: Compute the posterior mean $\tilde{f}_j$ and variance $\tilde{\sigma}_j^2$ for each decision alternative X_j by substituting $\bar{f}_j$ and $\hat{\sigma}_j^2$ as follows (Peng, Xu, Lee, Hu, and Chen 2019):

$$\tilde{f}_j = \frac{\tilde{\sigma}_j^2}{\sigma_j^2/n_j}\bar{f}_j + \tilde{\sigma}_j^2 \nu_c \alpha_c + \tilde{\sigma}_j^2 (g_j - \beta_c)(-\gamma_c)^T \tag{4}$$

$$\tilde{\sigma}_j^2 = \frac{1}{\nu_c + n_j/\sigma_j^2} \tag{5}$$

Step 3: Compute n_j for each decision alternative X_j using the following equations (Chen, Lin, Yücesan, and Chick 2000, Chen and Lee 2010, Perry, Xu, Huang, and Chen 2022):

$$\frac{n_{j_1}}{n_{j_2}} = \frac{\nu_{j_1}\delta_{j_2}}{\nu_{j_2}\delta_{j_1}}, j_1 \neq j_2 \neq j^*, \quad n_j^* = \sqrt{\nu_{j^*}}\sqrt{\sum_{j\neq j^*}\frac{n_j^2}{\nu_j}} \tag{6}$$

Step 4: Stop the process if N_t reaches N, and select the decision with the highest posterior mean $\tilde{f}_j$ as the optimal decision. Otherwise, proceed to Step 1 after updating the iteration counter $t=t+1$, and the expended simulation budget $N_t = N_t + \Delta N$.

SAMPLE enhances the accuracy of decision-making during real-time simulation optimization by combining the ML predictions with simulation data while efficiently allocating the online sampling budget. When the objective is to identify a decision that maximizes a single objective, adaptive intelligence can efficiently determine the best choice among a set of known decision alternatives for any given context.

4 CONCLUDING COMMENTS

Digital Twins, virtual models that accurately replicate the dynamic behavior of physical systems, are on the cusp of a revolutionary transformation not only for the design but also for real-time control of dynamic systems. In this tutorial, we define the key building blocks of Digital Twins and explore the key challenges they face in enabling real-time decision making.

DTs, which generalize and extend the power of simulation, have thus emerged as an important and popular tool. To provide decision support for the physical system almost in real-time, DTs inherit the challenges of efficiency in simulation optimization, which have been extensively investigated in the simulation community. For instance, OCBA, which adaptively manages the exploration/exploitation trade-off with the aim of maximizing the efficiency of simulation optimization, is 3 to 20 times faster than traditional R&S procedures. However, such speed up may still not be sufficient for real-time decision making.

AI/ML techniques appear to be well positioned to further enhance the efficiency of traditional simulation optimization approaches since, once well trained, they are very fast. However, training these black-box models requires extensive data. Moreover, when a scenario occurs in an area where the model is not well trained, the output of AI/ML may not be reliable. Additional training or adjustment of the AI/ML model would prove to be expensive and excessively time consuming.

This tutorial therefore focused on exploring potential synergies between AI/ML models and simulation optimization techniques in achieving *adaptive intelligence*. A concrete illustration of such synergies can be found in the way AI/ML is teamed up with simulation optimization techniques such as MCTS to create a world-class *Go* player. In this case, instead of performing additional training (or transfer learning) to improve the metamodel, AI/ML is directly integrated into DT in response to the realized/observed scenario.

In another illustration of adaptive intelligence, SAMPLE provides a framework whereby an AI/ML model, which had been trained offline, serves as an engine for the transformation of the decision space into an ordinal space, strengthening the neighborhood structure. A multi-fidelity OCBA is then developed to take advantage of this new neighborhood structure to maximize efficiency in evaluating alternative decisions. SAMPLE is just one possible concept of adaptive intelligence. Other approaches that explore the synergies between AI/ML and simulation optimization can surely be developed. This is why we believe that AI/ML represents an important domain for simulation research.

REFERENCES

Alban, A., S. E. Chick, and S. I. Zoumpoulis. 2021. "Expected Value of Information Methods for Contextual Ranking and Selection: Clinical Trials and Simulation Optimization". In *2021 Winter Simulation Conference*, 1–12 https://doi.org/10.1109/WSC52266.2021.9715303.

Baudiš, P., and J.-l. Gailly. 2011. "PACHI: State of the Art Open Source Go Program". In *Proceedings of the Advances in Computer Games 2011 International Conference*. November 20th-22nd, Tilburg, Netherlands, 24-38.

Carreira-Munich, T., V. Paz-Marcolla, and R. Castro. 2024. "DEVS Copilot: Towards Generative AI-assisted Formal Simulation Modeling Based on Large Language Models". In *2024 Winter Simulation Conference*, 2785–2796. IEEE https://doi.org/10.1109/WSC63780.2024.10838994.

Çakmak, S., Y. Wang, S. Gao, and E. Zhou. 2024. "Contextual Ranking and Selection with Gaussian Processes and OCBA". *ACM Transactions on Modeling and Computer Simulation* 34(2):1–24.

Cen, W., and P. Haas. 2022. "Enhanced Simulation Metamodeling via Graph and Generative Neural Networks". In *2022 Winter Simulation Conference*, 2785–2796 https://doi.org/10.1109/WSC57314.2022.10015361.

Cen, W., E. Herbert, and P. Haas. 2020. "NIM: Modeling and Generation of Simulation Inputs via Generative Neural Networks". In *2020 Winter Simulation Conference*, 584–595 https://doi.org/10.1109/WSC48552.2020.9383966.

Chen, C.-H., and L.-H. Lee. 2010. *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*. World Scientific Publishing Co.

Chen, C.-H., J. Lin, E. Yücesan, and S. E. Chick. 2000. "Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization". *Discrete Event Dynamic Systems* 10:251–270.

Clark, C., and A. Storkey. 2015. "Training Deep Convolutional Neural Networks to Play Go". In *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France: July 4th-7th, Lille, France, 1766-1774. https://doi.org/10.5555/3045118.

Coulom, R. 2007. “Computing “Elo Ratings” of Move Patterns in the Game of Go”. *ICGA Journal* 30(4):198–208.

Ding, L., L. J. Hong, H. Shen, and X. Zhang. 2022. “Technical Note—Knowledge Gradient for Selection with Covariates: Consistency and Computation”. *Naval Research Logistics* 69(3):496–507.

do Amaral, J. V. S., J. A. B. Montevechi, R. de Carvalho Miranda, and W. T. de Sousa Junior. 2022. “Metamodel-based Simulation Optimization: A Systematic Literature Review”. *Simulation Modelling Practice and Theory* 114:102403.

dos Santos, C., J. Montevechi, A. Campos, R. Miranda, J. Queiroz, and J. Amaral. 2024. “Simulation-Based Digital Twins: An Accreditation Method”. In *2024 Winter Simulation Conference*, 2856–2867 https://doi.org/10.1109/WSC63780.2024.10838895.

Enzenberger, M., M. Müller, B. Arneson, and R. Segal. 2010. “Fuego—an Open-source Framework for Board Games and Go Engine Based on Monte Carlo Tree Search”. *IEEE Transactions on Computational Intelligence and AI in Games* 2(4):259–270.

Fu, M. C., D. Qiu, and J. Xu. 2024. “A Tutorial for Monte Carlo Tree Search in AI”. In *2024 Winter Simulation Conference*, 16–30 https://doi.org/10.5555/3712729.3712731.

Gao, S., J. Du, and C.-H. Chen. 2019. “Selecting the Optimal System Design under Covariates”. In *Proceedings of the 2019 IEEE Conference on Automation Science and Engineering*. August 22$^{\text{nd}}$-26$^{\text{th}}$, Vancouver, Canada, 547-552.

Gao, S., J. Du, and C.-H. Chen. 2024. “A Contextual Ranking and Selection Method for Personalized Medicine”. *Manufacturing and Service Operations Management* 26(1):167–181.

Glasserman, P., and S. Tayur. 1994. “The Stability of a Capacitated Multi-echelon Production-inventory System under a Base Stock Policy”. *Operations Research* 42(5):913–925.

Glasserman, P., and S. Tayur. 1995. “Sensitivity Analysis for Base Stock Levels in Multi-echelon Production Inventory Systems”. *Management Science* 41(2):263–281.

Goodwin, T., J. Xu, N. Celik, and C.-H. Chen. 2024. “Real-time Digital Twin-based Optimization with Predictive Simulation Learning”. *Journal of Simulation* 18(1):47–64.

Haas, P. 2024. “Tutorial: Artificial Neural Networks for Discrete-Event Simulation”. In *2024 Winter Simulation Conference*, 116–130 https://doi.org/10.1109/WSC63780.2024.10838940.

Hong, L. J., and G. Jiang. 2019. “Offline Simulation Online Application: A New Framework of Simulation-based Decision Making”. *Asia-Pacific Journal of Opertional Research* 36(06):1940015.

Hu, R., and M. Ludkovski. 2017. “Sequential Design for Ranking Response Surfaces”. *SIAM/ASA Journal on Uncertainty Quantification* 5(1):212–239.

Jin, X., H. Li, and L. H. Lee. 2019. “Optimal Budget Allocation in Simulation Analytics”. In *Proceedings of the 15th International Conference on Automation Science and Engineering*. August 22$^{\text{nd}}$-26$^{\text{th}}$, Vancouver, Canada, 178–182.

Keslin, G., B. L. Nelson, M. Plumlee, B. K. Pagnoncelli, and H. Rahimian. 2022. “A Classification Method for Ranking and Selection with Covariates”. In *2022 Winter Simulation Conference*, 1–12: IEEE https://doi.org/10.1109/WSC57314.2022.10015235.

Kritzinger, W., M. Karner, G. Traar, J. Henjes, and W. Sihn. 2018. “Digital Twin in Manufacturing: A Categorical Literature Review and Classification”. In *Proceedings of the 16th IFAC Symposium on Information Control Problems in Manufacturing*. June 11$^{\text{th}}$-13$^{\text{rd}}$, Bergamo, Italy, 1016-1022.

Kumar, A., T. Peng, Y. Wu, and A. Zeevi. 2025. “Testing LLM Abstraction Capabilities on Stochastic Modeling Problems”. In *2025 Winter Simulation Conference*.

Laidler, G., B. L. Nelson, and N. G. Pavlidis. 2025. “Simulation Shapelets: Comparing Characteristics of Time-dynamic Trajectories”. *Journal of Simulation*:1–18 https://doi.org/10.1080/17477778.2024.2449033.

Li, H., H. Lam, and Y. Peng. 2024. “Efficient Learning for Clustering and Optimizing Context-dependent Designs”. *Operations Research* 72(2):617–638.

Li, Y., M. C. Fu, and J. Xu. 2021. “An Optimal Computing Budget Allocation Tree Policy for Monte Carlo Tree Search”. *IEEE Transactions on Automatic Control* 67(6):2685–2699.

Li, Y.-F., S. H. Ng, M. Xie, and T. Goh. 2010. “A Systematic Comparison of Metamodeling Techniques for Simulation Optimization in Decision Support Systems”. *Applied Soft Computing* 10(4):1257–1273.

Lugaresi, G. 2024. “Process Mining as Catalyst of Digital Twins for Production Systems: Challenges and Research Opportunities”. In *2024 Winter Simulation Conference*, 3082–3093 https://doi.org/10.1109/WSC63780.2024.10838896.

Matta, A., and G. Lugaresi. 2023. “Digital Twins: Features, Models, and Services”. In *2023 Winter Simulation Conference*, 46–60 https://doi.org/10.1109/WSC60868.2023.10407260.

Miao, Z., Z. Huang, and Z. Hu. 2025. “An Empirical Study of Generative Models as Input Models for Simulation”. In *Proceedings of the Winter Simulation Conference*.

Mohammad Nezhad, A., and H. Mahlooji. 2014. “An Artificial Neural Network Meta-model for Constrained Simulation Optimization”. *Journal of the Operational Research Society* 65:1232–1244.

National Academies of Sciences, E., and Medicine. 2024. “Foundational Research Gaps and Future Directions for Digital Twins”. Technical report, NASEM, Washington, DC.

Norman, D., P. Dawadi, and H. Yedidsion. 2024. “Yield Improvement Using Deep Reinforcement Learning for Dispatch Rule Tuning”. In *2024 Winter Simulation Conference*, 1865–1876 https://doi.org/10.5555/3712729.3712884.

Parnianifard, A., and A. S. Azfanizam. 2020. "Metamodel-based Robust Simulation-optimization Assisted Optimal Design of Multiloop Integer and Fractional-order PID Controller". *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields* 33(1):e2679.

Pearce, M., and J. Branke. 2017. "Efficient Expected Improvement Estimation for Continuous Multiple Ranking and Selection". In *2017 Winter Simulation Conference*, 2161–2172 https://doi.org/10.1109/WSC.2017.8247948.

Peng, Y., J. Xu, L. H. Lee, J. Hu, and C.-H. Chen. 2019. "Efficient Simulation Sampling Allocation Using Multifidelity Models". *IEEE Transactions on Automatic Control* 64(8):3156–3169.

Perry, M., J. Xu, E. Huang, and C.-H. Chen. 2022. "Robust Sampling Budget Allocation under Deep Uncertainty". *IEEE Transactions on Systems, Man and Cybernetics: Systems* 52(10):6339–6347.

Rinott, Y. 1978. "On Two-Stage Selection Procedures and Related Probability-Inequalities". *Communications in Statistics* A7:799–811.

Ryzhov, I. O. 2016. "On the Convergence Rates of Expected Improvement Methods". *Operations Research* 64(6):1515–1528.

Sargent, R. 2020. "Verification and Validation of Simulation Models: An Advanced Tutorial". In *2020 Winter Simulation Conference*, 16–29 https://doi.org/10.1109/WSC48552.2020.9384052.

Sargent, R. G., D. M. Goldsman, and T. Yaacoub. 2016. "A Tutorial on the Operational Validation of Simulation Models". In *2016 Winter Simulation Conference*, 163–177 https://doi.org/10.5555/3042094.3042126.

Shi, X., Y. Peng, and G. Zhang. 2023. "Top-Two Thompson Sampling for Contextual Top-mc Selection Problems". *arXiv preprint arXiv:2306.17704*.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, *et al*. 2016. "Mastering the Game of Go with Deep Neural Networks and Tree Search". *Nature* 529(7587):484–489.

Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, *et al*. 2017. "Mastering the Game of Go without Human Knowledge". *Nature* 550(7676):354–359.

Soykan, B., and G. Rabadi. 2024. "Optimizing Job Shop Scheduling Problem through Deep Reinforcement Learning and Discrete Event Simulation". In *2024 Winter Simulation Conference*, 2607–2618 https://doi.org/10.1109/WSC63780.2024.10838905.

Wang, K., N. Kallus, and W. Sun. 2024. "The Central Role of the Loss Function in Reinforcement Learning". *arXiv preprint arXiv:2409.12799v1*.

Wang, T., and L. J. Hong. 2022. "Large-scale Inventory Optimization: A Recurrent Neural Networks Inspired Simulation Approach". *INFORMS Journal on Computing* 35(1):196–215.

Xu, J., S. Zhang, E. Huang, C.-H. Chen, L. H. Lee, and N. Çelik. 2014. "Efficient Multi-Fidelity Simulation Optimization". In *2014 Winter Simulation Conference*, 3940–3951 https://doi.org/10.1109/WSC.2014.7020119.

Xu, J., S. Zhang, E. Huang, C.-H. Chen, L. H. Lee, and N. Çelik. 2017. "MO^{2}TOS: Multi-fidelity Optimization with Ordinal Transformation and Optimal Sampling". *Asia-Pacific Journal of Operational Research* 33(3):1–27.

Zhang, G., S. Chen, K. Huang, and Y. Peng. 2024. "Efficient Learning for Selecting Top-m Context-Dependent Designs". *IEEE Transactions on Automation Science and Engineering* 22:3210–3225.

Zheng, H., W. Xie, I. O. Ryzhov, and K. Choy. 2025. "Digital Twin Calibration with Model-based Reinforcement Learning". In *arXiv:2501.02205v1*.

Zhou, E. 2024. "Data-Driven Simulation Optimization in the Age of Digital Twins: Challenges and Developments". In *Proceedings of the Winter Simulation Conference*, 31–45 https://doi.org/10.1109/WSC63780.2024.10838871.

AUTHOR BIOGRAPHIES

CHUN HUNG CHEN is a Professor at the Department of Systems Engineering & Operations Research at George Mason University. His research focuses on stochastic simulation and optimization. He served as Co-Editor of the Proceedings of the WSC 2002 and Program Co-Chair for 2007 Informs Simulation Society Workshop. He has been an IEEE fellow since 2015. He received his Ph.D. degree from Harvard University in 1994. His email address is cchen9@gmu.edu and his website is https://mason.gmu.edu/~cchen9/.

JIE XU is Professor at the Department of Systems Engineering & Operations Research at George Mason University. He received the Ph.D. degree in industrial engineering & management sciences from Northwestern University in 2009. His research interests are digital twin, stochastic simulation and optimization, with applications in data centers, energy systems, healthcare, manufacturing, and transportation. His email address is jxu13@gmu.edu and his website is https://mason.gmu.edu/~jxu13/.

ENVER YÜCESAN is a Professor at the Technology and Operations Management Area and the ADCB Chair in International Management at INSEAD, France. Over the past three decades, he has contributed to the WSC in various roles. In 2015, he received the I-Sim Distinguished Service Award. He has been elected an INFORMS Fellow in 2022. His email address is enver.yucesan@insead.edu and his website is https://www.insead.edu/faculty/enver-yucesan.