

Hyperparameter Tuning with Gaussian Processes for Optimal Abstraction Control in Simulation-based Optimization of Smart Semiconductor Manufacturing Systems

MOON GI SEOK, Department of Computer Science and Engineering, Dongguk University, Seoul, South Korea
WEN JUN TAN, School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore
BOYI SU, School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore
WENTONG CAI, School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore
JISU KWON, School of Electronic and Electrical Engineering, Kyungpook National University, Daegu, South Korea
SEON HAN CHOI, Graduate Program in Smart Factory, Ewha Womans University, Seoul, South Korea

Smart manufacturing utilizes digital twins that are virtual forms of their production plants for analyzing and optimizing decisions. Digital twins have been mainly developed as discrete-event models (DEMs) to represent the detailed and stochastic dynamics of productions in the plants. The optimum decision is achieved after simulating the DEM-based digital twins under various what-if decision candidates; thus, simulation acceleration is crucial for rapid optimum determination for given problems. For the acceleration of discrete-event simulations, adaptive abstraction-level conversion approaches have been previously proposed to switch active models of each machine group between a set of DEM components and a corresponding lookup table-based mean-delay model during runtime. The switching is decided by detecting the machine group's convergence into (or divergence from) a steady state. However, there is a tradeoff between speedup and accuracy loss in the adaptive abstraction convertible simulation (AACS), and inaccurate simulation can degrade the quality of the optimum (i.e., the distance between the calculated optimum and the actual optimum). In this article,

This research is supported by the A*STAR Cyber-Physical Production System (CPPS) – Towards Contextual and Intelligent Response Research Program, under the RIE2020 IAF-PP Grant A19C1a0018, and Model Factory@SIMTech, and partially supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science and ICT (NRF2022R1F1A1066267). This work was also partially supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) under the Artificial Intelligence Convergence Innovation Human Resources Development (IITP-2024-RS-2023-00254592) grant funded by the Korea government (MSIT).
Authors' addresses: M. G. Seok, Dept. of Computer Science and Engineering, Dongguk University, South Korea; e-mail: mgseok@dgu.ac.kr; W. J. Tan, B. Su, and W. Cai, Sch. of Computer Science and Engineering, Nanyang Technological University, Singapore; e-mails: wjtan@ntu.edu.sg, bsu@ntu.edu.sg, ASWTCAI@ntu.edu.sg; J. Kwon, Sch. of Electronic and Electrical Engineering, Kyungpook National University, South Korea, e-mail: kjisu96@knu.ac.kr; S. H. Choi (Corresponding author), Graduate Program in Smart Factory, Ewha Womans University, South Korea; e-mail: seonhan.choi@ewha.ac.kr.



This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2024 Copyright held by the owner/author(s).
ACM 1558-1195/2024/11-ART7
https://doi.org/10.1145/3646549

we propose a simulation-based optimization that optimizes the problem based on a genetic algorithm while tuning specific hyperparameters (related to the tradeoff control) to maximize the speedup of AACS under a specified accuracy constraint. For each individual, the proposed method distributes the overall computing budget for multiple simulation runs (considering the digital twin's probabilistic property) into the hyperparameter optimization (HPO) and fitness evaluation. We propose an efficient HPO method that manages multiple Gaussian process models (as speedup-estimation models) to acquire promising optimal hyperparameter candidates (that maximize the simulation speedups) with few attempts. The method also reduces each individual's exploration overhead (as the population evolves) by estimating each hyperparameter's expected speedup using previous exploration results of neighboring individuals without actual simulation executions. The proposed method was applied to optimize raw-material releases of a large-scale manufacturing system to prove the concept and evaluate the performance under various situations.

CCS Concepts: • **Computing methodologies** → **Systems theory**; **Discrete-event simulation;**

Additional Key Words and Phrases: Simulation-based optimization, runtime abstraction-level conversion, discrete-event modeling, manufacturing-system simulation

ACM Reference Format:
Moon Gi Seok, Wen Jun Tan, Boyi Su, Wentong Cai, Jisu Kwon, and Seon Han Choi. 2024. Hyperparameter Tuning with Gaussian Processes for Optimal Abstraction Control in Simulation-based Optimization of Smart Semiconductor Manufacturing Systems. *ACM Trans. Model. Comput. Simul.* 35, 1, Article 7 (November 2024), 21 pages. https://doi.org/10.1145/3646549

LIST OF ABBREVIATIONS

AACS	Adaptive Abstraction Convertible Simulation
ALC	Abstraction-Level Converter
DEM	Discrete-Event Model
DOS	DEM-Only Simulation
GA	Genetic Algorithm
GP	Gaussian Process
HPO	Hyperparameter Optimization
KS	Kolmogorov Smirnov
MDM	Mean-Delay Model
SBO	Simulation-Based Optimization

1 INTRODUCTION

Semiconductor manufacturing systems have applied Industry 4.0–based smart manufacturing concepts to their plants to enhance the monitoring and adaptive process control capabilities. The increased monitoring capabilities provide a huge volume of real-time data from machines or Internet-of-Things devices, which helps to develop and validate the digital twin (i.e., the virtual model of the plant). Moreover, we also used the data to initialize the digital twin so that we could estimate the performances of various what-if scenarios.

For agile production control, we should quickly compute optimal control solutions after evaluating the production performances of various what-if scenarios [2, 7, 35]. Depending on the measurement of key performance indicators, some indicators, such as overall cycle time, machine groups' maximum (or average) queue lengths, and so on, are computed using discrete-event simulation to consider the production dynamics for accurate estimation [2, 5, 10, 14]. Engineers aim to represent the target manufacturing plant's physical entities, such as machines and automated guided vehicles, considering their stochastic behaviors in terms of operation periods, reworking, and breakdown. Those engineers virtualize each physical entity into a corresponding **discrete-event model (DEM)** component. The components are integrated to build the digital twin (i.e., virtual plant), and

the digital twin is simulated by exchanging events among the components. The digital twin is initialized and maintains its high fidelity online using observed events to synchronize the digital twin and physical plant [29].

Using the digital twin, we can evaluate the performances of various what-if production decision candidates based on different scheduling and dispatching scenarios before applying a candidate to the actual plant. This article deals with finding one or multiple optimal solutions among different candidates through **simulation-based optimization (SBO)** using the stochastic DEM-based digital twin. For the rapid response to the physical plant with an optimal control solution (derived by SBO), minimizing the overall execution time for SBO is essential. For that, we have previously focused on the methods of accelerated simulation of the digital twin [26–28]. The key concept of the methods is runtime model abstraction, which adaptively switches each machine group between the group's high-fidelity DEMs components and a corresponding lookup table-based **mean-delay model (MDM)**.

A group's active model is switched whenever the group converges into or diverges from a steady state in terms of the stationary of queuing jobs in the group. If the group runs in an unsteady state, then the group's DEM components become active to perform event-based complex processing operations based on external or internal events. Among external events, the lot events (symbolizing raw materials/parts flows) are primary entities processed by machines and move across the machines. In a steady state, an MDM becomes active to handle the lot events based on saved mean delays, skipping the DEMs' detailed operations. The switching condition is computed following the **Kolmogorov–Smirnov (KS)** test. The test examines a statistical similarity between two types of consecutive observation samples, consisting of arrival-interval and waiting-time observations, and aims to see the convergence of queuing jobs based on Little's Law. Depending on the lot-release scenario from the inventory, the overall production stability changes. As the virtual plant's overall production becomes more stable, MDMs are more likely to be active than high-fidelity DEM components, which guarantees higher speed. Even in the same stability, depending on the values of the significance level α used in the KS test, the statistical similarities of the consecutive samples are different; lower α allows more invariances between the samples, leading to more simulation speedup with more accuracy loss.

During the SBO, the optimizer generates many individuals who represent control (or decision) candidates. Each individual's expected performance is measured using multiple simulations due to the stochasticity of digital twin. To shorten the simulation execution during SBO, we can set lower α. However, this reduces the performance accuracy of each individual, which can influence the optimal quality in terms of the closeness between the derived optimal by **adaptive abstraction convertible simulation (AACS)** and the actual optimal. We denote that the actual optimal is a value computed through **DEM-only simulation (DOS)** with a sufficiently large simulation budget (i.e., the number of simulation runs for each individual). Thus, we should carefully select the α value for each individual to gain a speedup while satisfying an accuracy constraint (i.e., a defined upper bound of the allowable accuracy loss). When MDMs sample lot types whose delay was not observed during the previous sampling for the MDM activation, these will trigger the switching conditions from MDMs to DEMs. In high-mix and low-volume manufacturing scenarios, a small sample size can reduce the speedup due to the frequency of the sampling new lot types, where the frequent switching reduces the activation duration of MDMS. However, a large sample size hinders early detection of steady-state convergence because of its more extended sampling period, which might not draw a maximum speedup. Hence, the selection of appropriate sample sizes is also essential to the simulation speedup in SBO.

This article proposes an integrated approach to optimize the plant's production control parameters (affecting the target performance indicators) and then tune two important hyperparameters: α

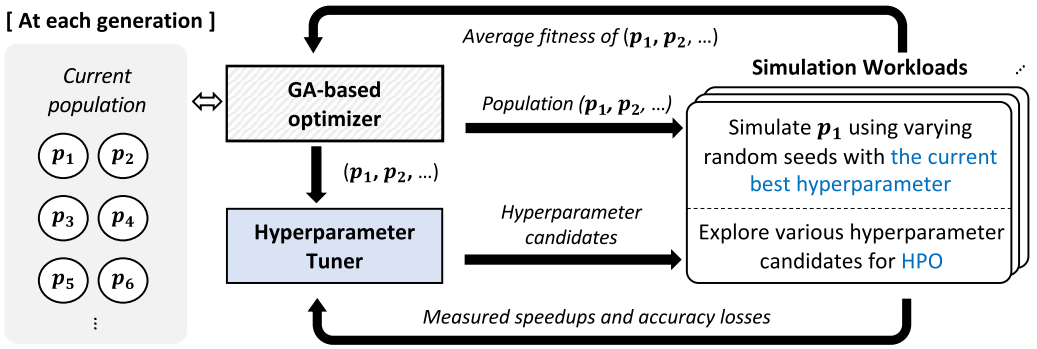


Fig. 1. Unified SBO process incorporating the hyperparameter adjustment for both manufacturing-specific and hyperparameter optimizations.

and the sample size. The approach aims to shorten the overall SBO execution time by maximizing the AACS speedups under the accuracy constraint, which aims to derive the production-control optimum(s) as accurately as those optimized through DOS. We use the **genetic algorithm (GA)** as the default optimization algorithm. Due to the stochastic property of semiconductor manufacturing plants, each individual (provided by the GA-based optimizer) requires a number of simulation runs using different random seeds, as shown in Figure 1. In this article, the number of runs is also known as the *simulation budget*, and we fix each individual's budget for its fitness evaluation. The proposed approach allocates part of each individual's total simulation budget for the **hyperparameter optimization (HPO)**.

The main contributions are as follows:

— We propose an HPO approach integrated with conventional GA-based SBO to maximize the speedup of AACS under a user-defined accuracy constraint.
— We designed a hyperparameter tuner that (1) performs HPO with **Gaussian process (GP)** models (as speedup-estimation models) to acquire promising hyperparameter candidates by estimating the candidates' speedups before actual simulations and (2) evaluates the population's fitnesses through multiple AACSs using tuned hyperparameter values.
— We propose an efficient hyperparameter exploration method for a given individual, which reuses the previous exploration results of the individual's neighbors based on the following perspective: an individual's production dynamics will be similar to its neighbors, and the hyperparameter optimums for similar decision candidates would be close.

Distinct from traditional HPO methodologies, which predominantly aim to identify a singular optimal set of hyperparameters for enhancing problem-specific training or optimization processes, our approach focuses on determining multiple hyperparameter optima, each corresponding to an individual entity. This endeavor is aimed at maximizing the speedups of individuals by adeptly managing individuals' own GP-based speedup estimation models. As the population evolves during genetic algorithm-based optimization, our approach augments the reliability of GP models by assimilating results derived from previous simulation explorations, thus empowering the tuning mechanism to deliver more precise hyperparameter optima. The evolution of these GP models amplifies the AACS speedups and reduces the portion of the budget allocated for hyperparameter exploration within HPO. This methodology ultimately leads to a reduction in the stochastic variances of the DEM, enabling more simulation runs for fitness evaluations compared to the hyperparameter explorations within a predetermined simulation budget.

The rest of the article is organized as follows. Section 2 describes the literature overview of the proposed SBO approach. Section 3 specifies the problems of HPO during AACS-based optimization. Section 4 explains the HPO procedures during the SBO. Section 5 shows the detailed procedures of (1) preparing the data used to initialize the GP models by reusing the previous exploration results of neighboring individuals without running actual simulations and (2) the GP-based HPO method. Section 6 shows the experimental results of the proposed approach, and Section 7 concludes the article.

2 RELATED WORKS

When a new demand is triggered, the semiconductor manufacturer often needs to make rapid decisions for production scheduling or material planning to optimize the cycle time, cost, utilization, bottleneck reduction, and so on. To be responsive to the rapid demand changes, optimal solutions need to be found quickly after estimating the performances of various what-if decision candidates. DEM has been typically used to simulate complex production dynamics of semiconductor manufacturing plants [2, 5, 10, 14]. DEM has multiple advantages in accessing the key performance indicators of the plant's various production scenarios. It enables building a complex system model to be built by including detailed subcomponents, which operate in an event-driven manner and exchange events to influence others [24]. We can couple various DEM components to achieve the desired fidelity for detailed processing, routing, and dispatching rules [20]. The process of finding the optimum candidates by simulating the decision candidates is called SBO. One typical way to speed up the SBO is to reduce the total number of simulation runs required by improving the optimization algorithm. Another way is to accelerate the simulation evaluation (the leading overhead of SBO). This article focuses on the runtime improvement of simulation evaluation during the SBO.

Although there are advantages to using DEMs, discrete-event simulations are computationally expensive due to iterative event processing [17]. This shortcoming becomes further accentuated during SBO, where achieving global optimization typically requires tens of thousands of model runs to find the global optimum within a delineated feasible parameter space, thereby obstructing thorough exploration in the optimization phase. A conventional approach to alleviating the computational burden during SBO is employing surrogate models, which are cost-efficient approximated models of compute-intensive simulation models. These surrogate models articulate the relationship between the inputs (i.e., the model's adjustable parameters) and outputs (i.e., the performance measure of the simulation model) using specified mathematical expressions whose parameters are calibrated using sufficient input-output data derived from multiple simulation runs with varied parameter sets.

In efforts to accelerate semiconductor manufacturing simulations, researchers have formulated two main types of surrogates: (1) queuing networks [6, 11, 23] and (2) other analytic closed-form solutions for individual machine groups. Queuing networks comprise queuing nodes assigned to each machine group whose machines execute analogous tasks and share a queue. Each node is modeled using queuing parameters to represent arrival and service processes. Other surrogates incorporate empirical approximations in various closed-forms solutions as constant delays, probability distributions, or exponential/quantile functions [3, 15, 25, 32]. The parameter values for those surrogates are calibrated using simulation outcomes of DEMs. However, surrogate models inherently face significant limitations, notably their inability to assure estimation accuracy in unprecedented or markedly divergent situations (arising from varied demands or orders) compared to those in the training data scenarios. These limitations are emphasized due to the characteristic on-demand requests of SBO aimed at optimizing evolving manufacturing systems. Thus, gathering adequate data for surrogate calibration before surrogate-based optimization is a practical challenge.

Moreover, those surrogate models struggle to accommodate complex scenarios involving high-mix processing, varying setups, machine groups with batching and cascading machines, and dynamic dispatching rules.

To resolve these previous surrogates' problems, we previously proposed an adaptive abstraction-level convertible model (as a new surrogate). We presented an **abstraction-level converter (ALC)** for each machine group to switch the group's active model between a set of dedicated DEMs and a corresponding lookup table-based MDM during runtime. Each detection of a group's steady-state convergence or divergence triggers the model switching. Compared to the previous surrogate models, MDMs are not pre-biased to specific scenarios or predeveloped before simulation runs, because their runtime behaviors rely on observed mean delay information in steady states. ALC detects the steady-state convergence/divergence through the KS test based on two important hyperparameters: significance level α and the sample size. For a production scenario, various hyperparameter values differently improve speedups with some accuracy losses. Moreover, a hyperparameter candidate that guarantees the maximum speedup with the desired accuracy loss for a production scenario might not be effective in another scenario. Thus, we aim to perform the HPO during the GA-based SBO to maximize the simulation speedup under the accuracy constraint when evaluating a lot of individuals' fitnesses.

HPO approaches have been actively studied in the training of machine learning. Those hyperparameters, such as the number of hidden layers, learning rates of stochastic gradient descent, batch size, and so on, directly control the behaviors of training algorithms and significantly affect the performance of machine learning [33]. The HPO approaches require deep knowledge of machine learning algorithms and machine learning-specific appropriate techniques [34]. The HPO can provide several primary advantages: (1) reducing the costly menial work of engineers, (2) improving the accuracy and efficiency of neural network training, and (3) improving the reproducibility and fairness of scientific studies [9].

Some HPO studies have tried to find optimal hyperparameter values for the **evolutionary algorithm (EA)** [8, 12, 19, 31], and Aleti and Moser [1] provided a comprehensive review of HPO. The success of EA applications highly depends on the hyperparameter settings, such as the choices of representation, parent selection, recombination and mutation operators, the probabilities in selection and replacement, population size, and so on. Among the HPO strategies in Reference [1], our approach can be categorized as *adaptive parameter control* that searches problem-specific optimum while monitoring the quality of EA runs to adjust hyperparameters.

This article presents a unified approach, concurrently orchestrating HPO and AALCs during SBO. Unlike conventional HPO studies, which primarily aim to identify a universal optimal set of hyperparameters to enhance training and optimization for specific problems, our approach endeavors to find the individuals' own hyperparameter optima for their maximum speedup while maintaining accuracy constraints. Our proposed approach is advancing our preceding method discussed in Reference [30], seeking to maximize the speedup by incorporating a new critical hyperparameter, the sample size, and the previously used hyperparameter, α. Given the expanded cardinality induced by the new hyperparameter, it becomes crucial to develop an efficient search technique to navigate the enlarged hyperparameter space with minimized simulation runs. For that, we propose a Bayesian optimization-based strategy that proficiently manages individuals' own GP models that are calibrated using observed speedup measurements and used to estimate the speedup of various hyperparameter candidates. The aim is to curtail the overhead of hyperparameter exploration by predicting high-quality initial GP models for new individuals, leveraging neighboring individuals' existing GP models, and underpinning the assumption that optimal hyperparameters for analogous manufacturing scenarios are likely similar. The following section elucidates the challenges in realizing this objective.

3 PROBLEM FORMULATION

To detect a machine group's steady-state convergence (or divergence) of the DEMs, the group's ALC monitors each lot type's inter-arrival times and waiting times in the group's queue. The same lot type guarantees the same priority in dispatching and the same processing recipe (related to processing time). Let $1/\lambda_i$ be an inter-arrival time between an i-type lot and its previous i-type lot, and let W_i be the waiting time of an i-type lot. The ALC manages two types of consecutive paired samples. One type of paired samples consists of λ_i observations for all lot types (i.e., $\forall i = 1, 2, \ldots$), and the other type of paired samples consists of W_i observations. If both paired samples become statistically indistinguishable, then we determine that the group queue converges into a steady state based on queueing theory. In the steady state, we assume that (1) the queue's average number of i-type lots is converged to $\lambda_i \times W_i$ and (2) the average total queue length becomes $\sum_i \lambda_i W_i$. The steady-state convergence condition based on the KS test can be formulated as follows:

$$\sup |F^{-}_{\lambda}(\lambda_i) - F^{+}_{\lambda}(\lambda_i)| \leq b \wedge \sup |F^{-}_{W}(W_i) - F^{+}_{W}(W_i)| \leq b, \tag{1}$$

where F^{-}_{λ} and F^{+}_{λ} are **cumulative distribution functions (CDFs)** of two consecutive arrival-rate samples; F^{-}_{W} and F^{+}_{W} are CDFs of two consecutive waiting-time samples; and $b = C(\alpha, m^{-}, m^{+})$ is an allowable bound that is calculated through function C based on significance-level α and two sample sizes, m^{-} and m^{+}. A general form of statistical invariance bound, $C(\alpha, m^{-}, m^{+})$, is represented as $\sqrt{-\ln(\alpha/2)\frac{1+m^{-}/m^{+}}{2m^{-}}}$ [18]. As hyperparameter α decreases, bound $C(\alpha, m^{-}, m^{+})$ increases inversely. If the group's ALC in a steady state detects a statistical arrival change, then the DEMs are reactivated after generating dummy lot events to make a workload consistency [26–28].

Depending on the production stability of the target scenario candidate, different values of α are required, because there is a tradeoff between speedup and accuracy. Typically, smaller α leads to better speedup at the expense of accuracy. Especially in scenarios whose overall productions are highly unstable, small α can induce a large number of switching between the DEMs and corresponding MDMs, which results in both higher inaccuracy and a slowdown.

Besides detecting a statistical invariance of lot-input arrivals, a steady-state machine group can reactivate the DEMs when observing a new lot type or a machine down. The new lot type means that an input lot's mean delay has not been stored in the MDM's lookup table. ALC sends the new lot to DEMs after reactivating the DEMs. As the sample size increases, we can reduce the chance of observing a new lot type, which reduces the number of DEM reactivations of steady-state groups. However, this leads to spending more time on collecting the consecutive samples (for the convergence testing), which can result in less speedup because of insensitive steady-state convergence detection.

In this work, we tune two hyperparameters: α and the sample size m, which are used for the AACS of each individual during SBO. There are several previous hyperparameter tuning techniques to improve neural network training. Those hyperparameter tuning techniques attempt to optimize functions whose closed-form expressions and derivatives are unknown and challenging to identify. Typical hyperparameter tuning methods are grid search, random search, and Bayesian optimization [9, 34]. Grid and random searches roam the entire space of available parameter values in an isolated way without paying attention to past results. These tuning techniques can become a time-consuming challenge, especially with large parameter spaces. The search space grows exponentially with the number of parameters tuned. Thus, we employ the Bayesian optimization approach, which guarantees that fewer attempts are required to find the optimal hyperparameter set compared to the grid and random searches. This reduction of attempts is significant when the computational cost in evaluating the objective function is high. In our case, the objective-function evaluation requires the speedup measurement through simulation; thus, the cost is relatively high compared to the analytic calculation of an equation.

Our approach requires users to specify an accuracy constraint as the bound of an allowable accuracy loss. Considering the DEM's stochastic property, each individual's fitness evaluation requires multiple simulation executions to average the results. We prefix each individual's total number of simulation runs (i.e., computing budget) as a user-defined number. For the Bayesian optimization of individuals' speedup-related hyperparameters, we manage multiple GP models to acquire promising hyperparameter candidates after estimating the speedup of each candidate prior to actual simulation-based measurements. The GP models also help quantify uncertainties of the predicted speedups.

The following sections detail these three points:

— The overall framework to tune hyperparameters in SBO.
— Each individual's simulation budget management procedures of the tuner to balance exploration and exploitation.
— The reuse of previous hyperparameter exploration information among neighboring individuals.

4 OVERVIEW OF HYPERPARAMETER TUNING IN SIMULATION-BASED OPTIMIZATION

As shown in Figure 1, the proposed approach requires two essential entities for simultaneous HPO and SBO, which are (1) the GA-based optimizer and (2) the hyperparameter tuner. The GA-based optimizer is for the problem-specific optimization of decision candidates. It generates multiple individuals (i.e., offsprings for the next generation) based on the fitness of every individual in the current population. Each individual symbolizes a manufacturing-decision candidate; a user-defined objective function is used to measure the individual's mean fitness based on multiple simulation results using different random seeds. For each individual $\mathbf{p}_\ell$ in the current new offsprings P_{new}, the tuner cooperates with the GA-based optimizer for the following two important roles.

— Exploration: allocating a partial budget to traverse various hyperparameter points over the hyperparameter space until finding the optimal point.
— Exploitation: evaluating $\mathbf{p}_\ell$'s fitness through AACS using optimal hyperparameter point.

The tuner manages a GP model for each individual $\mathbf{p}_\ell$ to find the optimal hyperparameter point that maximizes an unknown speedup function f:

$$\mathbf{x}_{\text{opt}} = \underset{\mathbf{x}\in\mathcal{A}}{\arg\max}\, f_\ell(\mathbf{x}), \tag{2}$$

where $\mathcal{A}$ denotes the search space of hyperparameter $\mathbf{x} = \langle x^{(1)}, x^{(2)}\rangle \in [0,1]^2$ and $x^{(1)}$ and $x^{(2)}$ are two encoded hyperparameters from original hyperparameters, α and m, respectively. The original hyperparameters α and m are under user-defined search spaces $A \subset \{\alpha \mid 0 < \alpha \leq 2\} \in \mathbb{R}$ and $M \in \mathbb{R}$, i.e., $(\alpha, m) \in A \times M$. The maximum α is 2 (i.e., $\sup A = 2$), which signifies that the candidate does not need AACS. If $\sup A = 2$, then statistical bound $C(\alpha, m, m) = \sqrt{-\ln(\alpha/2)\frac{1+m/m}{2m}}$ (in Equation (3)) becomes zero, and the abstraction conversion will not occur. If bound $C(\alpha, m, m)$ is 2, then the simulation speedup always becomes less than 1 because of unnecessary computational overheads for steady-state detection. We roughly define an α range, $A_{\text{dos}} = \{\alpha \mid 1 \leq \alpha \leq 2\} \subset A$, where each $\alpha \in A_{\text{dos}}$ deactivates ALCs to monitor their groups' steady-state convergences.

Each encoding from α to $x^{(1)}$ (or from m to $x^{(2)}$) is performed considering the scales of $[0,1]$ and A (or M). Decoding from $x^{(1)}$ to α (or from $x^{(2)}$ to m) is also performed when we apply a candidate (derived by the GA) to AACS. These scaling-based encoding and decoding procedures allow (1) the calculation of a kernel between two hyperparameters with equal importance at the GP-based speedup estimation (discussed in Section 5) and (2) the mapping between continuous GP variable $x^{(2)}$ and discrete sample size m.

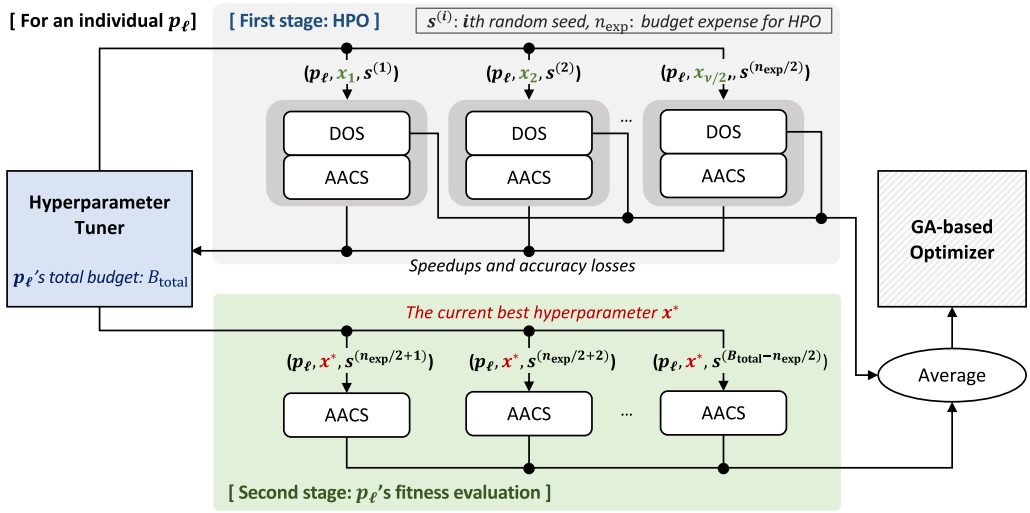


Fig. 2. Detailed comprehensive simulation executions using diverse random seeds for both manufacturing-specific and hyperparameter optimizations.

For each individual, users should specify two types of maximum simulation budgets: the maximum number of total simulations and the maximum number of simulations for the exploration in HPO. We denote two types of budget as B_{total} and $B_{\text{exp}}(< B_{\text{total}})$, respectively. The maximum budget B_{exp} is an even number, but the number of actual simulations for HPO, denoted by $n_{\text{exp}}(\leq B_{\text{exp}})$, can vary depending on the confidence of estimated hyperparameters. The remaining simulation budget, $B_{\text{total}} - n_{\text{exp}}$, is used to run multiple simulations for an individual's fitness evaluation using the best hyperparameter point x_{best} resulting from the current exploration in HPO. The individual's fitness is the average of the simulation results, as many as $B_{\text{total}} - n_{\text{exp}} + n_{\text{exp}}/2$, as shown in Figure 2. The number of simulations for the fitness evaluation is the same as the required number of random seeds.

During the exploration in HPO, multiple simulation pairs are executed for an individual, as many as n_{exp}. In each pair, AACS and DOS for a given individual $\mathbf{p}_\ell$ are conducted by sharing a same random seed to measure the *accuracy loss* and *speedup*. The *accuracy loss*, computed using random seed s and hyperparameters $\mathbf{x}$, is the ratio of difference between AACS- and DOS-driven evaluated fitnesses, such that $loss_\ell(\mathbf{x}, s) = |\left(fit_{\text{aacs}}(\mathbf{p}_\ell, \mathbf{x}, s) - fit_{\text{dos}}(\mathbf{p}_\ell, s)\right)/fit_{\text{dos}}(\mathbf{p}_\ell, s)|$, where $fit_{\text{aacs}}(\mathbf{p}_\ell, \mathbf{x}, s)$ and $fit_{\text{dos}}(\mathbf{p}_\ell, s)$ are the evaluated fitnesses by AACS and DOS, respectively, using $\mathbf{p}_\ell$, $\mathbf{x}$, and s. The *speedup* is conventionally gauged using actual execution times; however, we have devised an alternative approach. Our driving rationale behind this shift is to allocate a greater portion of the total budget (= B_{total}) toward application-specific optimization, thereby curtailing the number of simulation runs for HPO. When we base the speedup evaluation on actual execution times, it necessitates multiple simulation runs for each hyperparameter candidate within HPO due to the unavoidable stochastic variances present, even in measuring execution times of simulation on a standalone computing machine. To circumvent this limitation, we define an alternative metric (as a deterministic gauge for assessing the speedup of AACS versus DOS) as follows.

Definition 1. An individual $\mathbf{p}_\ell$'s AACS speedup for a given pair of hyperparameter $\mathbf{x}$ and random seed s is measured by the ratio of the number of events processed during AACS compared to that of DOS, i.e., the speedup is represented by $ev_{\text{dos}}(\mathbf{p}_\ell, s)/ev_{\text{aacs}}(\mathbf{p}_\ell, \mathbf{x}, s)$,

where $ev_{\text{dos}}(\mathbf{p}_\ell, s)$ and $ev_{\text{aacs}}(\mathbf{p}_\ell, \mathbf{x}, s)$ are the number of events processed during DOS and AACS, respectively, for the given $\mathbf{p}_\ell$, $\mathbf{x}$, and s.

Given a distinct random seed, the complete set of event-driven operations for the target scenario simulation is predetermined. The AACS strategy accelerates traditional discrete-event simulations in semiconductor manufacturing by omitting repetitive event-driven activities within steady-state machine groups. As a result, the enhancement in speedup, attributable to a specific hyperparameter candidate during AACS, is deeply tied to the number of events it can evaporate. For each hyperparameter candidate, there is a deterministic reduction in the total events in the standard DOS specific to the same individual and random seed. Our previous research empirically showed the high correlation between the reduced event count and the consequent simulation acceleration [26].

Using random seed s, the unknown speedup function $f(\mathbf{x})$ is measured by

$$f_\ell(\mathbf{x}) \approx f_\ell(\mathbf{x}, s) = \begin{cases} ev_{\text{dos}}(\mathbf{p}_\ell, s)/ev_{\text{aacs}}(\mathbf{p}_\ell, \mathbf{x}, s) & \text{if } loss_\ell(\mathbf{x}, s) < \theta_{\text{acc}}, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where θ_{acc} is a user-defined accuracy constraint. Though we can reduce any noise in measuring the speedup by computing the reduced events (as Definition 1), $f(\mathbf{x})$ still has a variance, because speedups could change as seed s varies; thus, an amount of statistical variance in the $f_\ell(\mathbf{x})$ measurement is inevitable.

For the exploration in HPO, the tuner manages each individual's GP model as a surrogate model to predict a hyperparameter point's expected speedup based on the prior measurements before measuring the actual speedup. For the given individual $\mathbf{p}_\ell$ from the GA-based optimizer, the tuner's overall procedures are described in Algorithm 1. If individual $\mathbf{p}_\ell$'s GP model has not been initialized for HPO, then the tuner prepares warmup hyperparameter points ($X_{\text{warm},\ell}$) and their corresponding speedups to initialize the GP model, as Lines 6–13. This data preparation is called a warmup stage for the GP model. The tuner attempts to estimate the speedups of each point in $X_{\text{warm},\ell}$ without real simulations using previous exploration information of individual $\mathbf{p}_\ell$'s neighbors. If point $\mathbf{x}_i \in X_{\text{warm},\ell}$ is predictable without simulation, then the tuner saves $\mathbf{x}_i$ in additional set X_{pred}. Detailed procedures are described in Section 5.1. Otherwise, $f_\ell(\mathbf{x}_i)$ is measured through a pair of DOS and AACS executions, as Line 12. Each point $\mathbf{x}_i$ and its predicted or measured speedup, $f_\ell(\mathbf{x}_i)$, are saved in two types of data sequences: $X_{\text{exp},\ell}$ and $Y_{\text{exp},\ell}$, respectively.

Before the exploration for HPO, the tuner derives the optimal hyperparameter candidate $\mathbf{x}_{\text{best}}$ based on the previous exploration information: $X_{\text{exp},\ell}$ and $Y_{\text{exp},\ell}$, as Line 14. Suppose that $f_\ell(\mathbf{x}_{\text{best}})$ was not measured by simulation (i.e., $\mathbf{x}_{\text{best}} \in X_{\text{pred}}$). In that case, the tuner re-evaluates $f_\ell(\mathbf{x}_{\text{best}})$ through a pair of simulations to compute $f_\ell(\mathbf{x}_{\text{best}})$ accurately, because the GP-based HPO approach intensively explores the vicinity of $\mathbf{x}_{\text{best}}$. Thus, the reliability of $\mathbf{x}_{\text{best}}$'s speedup measurement is critical for successful GP-based HPO.

At the HPO stage, the tuner explores the hyperparameter space $\mathcal{A}$ until exhausting all of budget B_{exp}. Based on previously collected hyperparameters and their speedups, which are saved in $X_{\text{exp},\ell}$ and $Y_{\text{exp},\ell}$, the tuner updates the GP prior, which helps predict the speedup incorporating prior knowledge (kernels) and provide uncertainty measures over prediction. Then, the tuner acquires the optimal hyperparameter candidate, denoted by $\mathbf{x}_{\text{new}}$. Detailed procedures to update the GP prior and acquire $\mathbf{x}_{\text{best}}$ are discussed in Sections 5.1 and 5.2, respectively.

If new hyperparameter point $\mathbf{x}_{\text{new}}$ is close to the current optimum $\mathbf{x}_{\text{best}}$, then we ensure that hyperparameter $\mathbf{x}$ is converged. This hyperparameter convergence is an additional termination condition to prevent further exploration and is defined as follows:

Definition 2. $\mathbf{x}_{\text{new}} = \langle x^{(1)}_{\text{new}}, x^{(2)}_{\text{new}} \rangle$ is converged to $\mathbf{x}_{\text{best}} = \langle x^{(1)}_{\text{best}}, x^{(2)}_{\text{best}} \rangle$ if satisfying $|\mathbf{x}^{(i)}_{\text{new}} - \mathbf{x}^{(i)}_{\text{best}}| < \theta^{(i)}_{\text{conv}}$ for all $i \in \{1, 2\}$,

ALGORITHM 1: Overall procedures of the hyperparameter tuner for individual $\mathbf{p}_i$

1 let FIT_ℓ be the sequence of evaluated fitnesses of individual $\mathbf{p}_\ell$.
2 let $X_{\text{exp},\ell}$ and $Y_{\text{exp},\ell}$ be the sequence of explored hyperparameter points and those points' corresponding speedups for $\mathbf{p}_\ell$, respectively.
3 let X_{pred} be the set of hyperparameters whose speedups are predicted without simulation.
4 $n_{\text{exp}} \leftarrow 0$
5 **if** *GP model was placed* **then** `// Warmup stage`
    `// Section 5.1 details the behaviours in Line (6)-(9).`
6 set initial hyperparameter points $X_{\text{warm},\ell}$.
7 **foreach** $\mathbf{x}_i \in X_{warm,\ell}$ **do**
8 **if** $f_\ell(\mathbf{x}_i)$ *is predictable without simulation* **then**
9 approximate $f_\ell(\mathbf{x}_i)$ using neighbors' exploration results.
10 $X_{\text{pred}} \leftarrow X_{\text{pred}} \cup \{\mathbf{x}_i\}$
11 **else**
12 evaluate $f_\ell(\mathbf{x}_i)$ and $fit_{\text{dos}}(\mathbf{p}_\ell, s)$ through a pair of DOS and AACS using new seed s; append the fitness value to FIT_ℓ; $n_{\text{exp}} \leftarrow n_{\text{exp}} + 2$
13 append $\mathbf{x}_i$ and $f_\ell(\mathbf{x}_i)$ to $X_{\text{exp},\ell}$ and $Y_{\text{exp},\ell}$, respectively.
14 $\mathbf{x}_{\text{best}} \leftarrow \arg\max_{\mathbf{x}_i \in X_{\text{exp},\ell}} f_\ell(\mathbf{x}_i)$
15 **while** $\mathbf{x}_{best} \in X_{pred}$ **do** `//` $f_\ell(\mathbf{x}_{\text{best}})$ `was estimated without real simulations`
16 evaluate $f_\ell(\mathbf{x}_{\text{best}})$ and $fit_{\text{dos}}(\mathbf{p}_\ell, s)$ through a pair of DOS and AACS using new seed s; append $fit_{\text{dos}}(\mathbf{p}_\ell, s)$ to FIT_ℓ; $n_{\text{exp}} \leftarrow n_{\text{exp}} + 2$
17 update $f_\ell(\mathbf{x}_{\text{best}})$ in $Y_{\text{exp},\ell}$; $X_{\text{pred}} \leftarrow X_{\text{pred}} \setminus \{\mathbf{x}_{\text{best}}\}$
18 $\mathbf{x}_{\text{best}} \leftarrow \arg\max_{\mathbf{x}_i \in X_{\text{exp},\ell}} f_\ell(\mathbf{x}_i)$
19 **while** $n_{exp} < B_{exp}$ **do** `// Hyperparameter exploration for HPO`
20 refresh the GP prior by updating kernel matrix based on $X_{\text{exp},\ell}$ and $Y_{\text{exp},\ell}$. `// Section 5.1`
21 acquire new hyperparameter candidate $\mathbf{x}_{\text{new}}$. `// Section 5.2`
22 **if** $\mathbf{x}_{new}$ *and* $\mathbf{x}_{best}$ *are close* **then** `// Termination condition: Definition 2`
23 **break**
24 evaluate $f_\ell(\mathbf{x}_{\text{new}})$ and $fit_{\text{dos}}(\mathbf{p}_\ell, s)$ through a pair of DOS and AACS using new seed s; append $fit_{\text{dos}}(\mathbf{p}_\ell, s)$ to FIT_ℓ; $n_{\text{exp}} \leftarrow n_{\text{exp}} + 2$
25 append $\mathbf{x}_{\text{new}}$ and $f_\ell(\mathbf{x}_{\text{new}})$ to $X_{\text{exp},\ell}$ and $Y_{\text{exp},\ell}$, respectively.
26 **if** $f_\ell(\mathbf{x}_{new}) > f_\ell(\mathbf{x}_{best})$ **then** $\mathbf{x}_{\text{best}} \leftarrow \mathbf{x}_{\text{new}}$
27 **foreach** $i \in \{n_{exp}+1, \ldots, B_{total}\}$ **do** `// Perform fitness evaluations though multiple AACS runs`
28 calculate $fit_{\text{aacs}}(\mathbf{p}_\ell, \mathbf{x}_{\text{best}}, s)$ through AACS using new random seed s; then, append $fit_{\text{aacs}}(\mathbf{p}_\ell, \mathbf{x}_{\text{best}}, s)$ to FIT_ℓ.
29 **return** the mean of all evaluated values in FIT_ℓ to the GA-based optimizer.

where $\Theta_{\text{conv}} = \langle \theta^{(1)}_{\text{conv}}, \theta^{(2)}_{\text{conv}} \rangle$ is a pair of user-defined thresholds for the two-dimensional hyperparameter. If $\mathbf{x}_{\text{new}}$ is not converged, then the tuner evaluates $f_\ell(\mathbf{x}_{\text{new}})$ to compare the new speedup with $f_\ell(\mathbf{x}_{\text{best}})$ for further optimization, as Lines 24–26. Both explored point $\mathbf{x}_{\text{new}}$ and its speedup are saved to $X_{\text{exp},\ell}$ and $Y_{\text{exp},\ell}$ at each iteration.

After the hyperparameter exploration terminates, the tuner evaluates the fitness values through AACS using the remaining simulation budget, $B_{\text{total}} - n_{\text{exp}}$ (see Lines 27 and 28). Then, the tuner returns the mean fitness value to the GA-based optimizer.

Individual $\mathbf{p}_\ell$ in the new offsprings P_{new} could be evaluated at the previous generation but was not selected as an elite. In this case, the tuner reuses the previous exploration and evaluation results

of $X_{\text{exp},\ell}$, $Y_{\text{exp},\ell}$, and FIT_ℓ. Then, the simulation budget for HPO and fitness evaluation is used to (1) enhance the AACS speedup by finding a more optimized hyperparameter point and (2) increase accuracy in the fitness evaluation owing to a larger number of simulation attempts considering DEMs' stochastic properties. If $\mathbf{p}_\ell$'s optimal hyperparameter point was previously converged, then the tuner skips the HPO stage.

5 GAUSSIAN PROCESS-BASED HYPERPARAMETER TUNING

5.1 Warmup and Update of Gaussian Process Model

The overall stability of each individual's production dynamics is the main element in deciding a hyperparameter optimum that maximizes the speedup of the individual's AACS. If production dynamics among neighbors are similar, then a hyperparameter value yields similar speedups during the neighbors' AACSs; thus, we assume that a hyperparameter point's expected speedup would be similar among neighbors.

If the GP model was not initialized for the given individual $\mathbf{p}_\ell$ (from the GA-based optimizer), then the tuner starts a warmup stage. In the warmup stage, the tuner initializes individual $\mathbf{p}_\ell$'s GP model using synthetic data (which is used to initialize the GP) by estimating the speedups of initial hyperparameter points $X_{\text{warm},\ell}$ based on the previous exploration results of adjacent neighbors. $X_{\text{warm},\ell}$ is divided into two subsets: X_{fix} and $X_{\text{rnd},\ell}$ (i.e., $X_{\text{warm}} = X_{\text{fix}} \cup X_{\text{rnd},\ell}$), where X_{fix} is the set of user-specified and prefixed points that are shared by all of the population; $X_{\text{rnd},\ell}$ is the set of randomly sampled points for each individual. The fixed points in X_{fix} should be specified sparsely but identically distributed over the hyperparameter space to mitigate a local optimum trap. The management of the fixed points (X_{fix}) aims to find each point's speedup without actual simulation executions through approximated prediction using neighboring individuals' speedup estimates (or simulation-based measurements) based on the assumption: a point's speedups would be similar among neighboring individuals. More speedup prediction saves the simulation budget used to constitute plenty of the warmup data.

Let $f_k(\mathbf{x}_i) \in Y_{\text{exp},k}$ be the previously measured speedup of hyperparameter $\mathbf{x}_i \in X_{\text{exp},k}$ when evaluating the fitness of an individual $\mathbf{p}_k$. Let $\mathbf{P}_{\text{nbr},\ell}$ be the set of $\mathbf{p}_\ell$'s neighbors in all current and previous population P, i.e., $\mathbf{P}_{\text{nbr},\ell} = \{p_k \mid d_{(\ell,k)} < d_{\max}, p_k \in P\}$, where $d_{k,\ell}$ is the distance between two individuals: $\mathbf{p}_k$ and $\mathbf{p}_\ell$; and $d_{\max}$ is a user-defined threshold that determines the maximum radius of the neighborhood. To estimate the speedup of an individual $\mathbf{p}_\ell$'s hyperparameter point $\mathbf{x}_i \in X_{\text{fix}}$, the tuner checks whether there are enough neighbors who previously explored ($\mathbf{x}_i$), i.e., $|N_\ell(\mathbf{x_i})| > \omega$, where $N_\ell(\mathbf{x_i}) = \{k \mid p_k \in \mathbf{P}_{\text{nbr},\ell}, \mathbf{x}_i \in X_{\text{exp},k}\}$ is the index set of $\mathbf{p}_\ell$'s neighbors who previously explored $\mathbf{x}_i$; and ω is the dimension of problem-specific decision space, i.e., $\mathbf{p}_\ell = [p_{\ell,1}, \dots, p_{\ell,\omega}]$, where $p_{\ell,i}$ is ith production parameter to be optimized. If $|N_\ell(\mathbf{x_i})| \geq \omega$, then the tuner estimates $f_\ell(\mathbf{x}_i)$ using multiple gradients, denoted by $\{\nabla g_i(\mathbf{p}_k)\}$, where $\nabla g_i(\mathbf{p}_k)$ is the gradient of the $\mathbf{x}_i$'s speedup at neighbor $\mathbf{p}_k \in \mathbf{P}_{\text{nbr},\ell}$. With respect to neighbor $\mathbf{p}_k$, we estimate the speedup $f_\ell(\mathbf{x}_i)$ by

$$f_{(\ell,k)}(\mathbf{x}_i) \approx f_k(\mathbf{x}_i) + \nabla g_i(\mathbf{p}_k) \cdot (\mathbf{p}_\ell - \mathbf{p}_k). \tag{4}$$

Though the function $f_k(\mathbf{x}_i)$ is unknown, we can approximate the gradient $\nabla g_i(\mathbf{p}_k)$ using neighbors' previously measured speedups as follows:

$$\begin{bmatrix} \mathbf{p}_{n_1} - \mathbf{p}_k \\ \vdots \\ \mathbf{p}_{n_\tau} - \mathbf{p}_k \end{bmatrix} \nabla g_i(\mathbf{p}_k) = \begin{bmatrix} f_{n_1}(\mathbf{x}_i) - f_k(\mathbf{x}_i) \\ \vdots \\ f_{n_\tau}(\mathbf{x}_i) - f_k(\mathbf{x}_i) \end{bmatrix}, \tag{5}$$

where $\mathbf{p}_{n_1}, \dots, \mathbf{p}_{n_\tau}$ are the common neighbors of $\mathbf{p}_\ell$ and $\mathbf{p}_k$, i.e., $p_{n_j} \in \mathbf{P}_{\text{nbr},\ell} \setminus \{\mathbf{p}_k\}$; and $\tau(\geq \omega)$ is the number of used previous speedups of the common neighbors. Suppose that we simplify (5) as

$\mathcal{P}\nabla g_i(\mathbf{p}_k) = \mathcal{Y}$, where $\mathcal{P} = [\mathbf{p}_{n_1} - \mathbf{p}_k; \ldots, \mathbf{p}_{n_\tau} - \mathbf{p}_k]$ and $\mathcal{Y} = [f_{n_1}(\mathbf{x}_i) - f_k(\mathbf{x}_i); \ldots; f_{n_\tau}(\mathbf{x}_i) - f_k(\mathbf{x}_i)]$. Then, we can solve $\nabla g_i(\mathbf{p}_k)$ through ordinary least square approximation as follows:

$$\nabla g_i(\mathbf{p}_k) = (\mathcal{P}^T\mathcal{P})^{-1}\mathcal{P}^T\mathcal{Y}. \tag{6}$$

To reduce the matrix multiplication overhead in the least square approximation, we can set an upper bound of τ, denoted by θ_{nbr}. If the number of the common neighbors, $|\mathbf{P}_{\text{nbr},\ell} \setminus \{\mathbf{p}_k\}|$, is larger than θ_{nbr}, then the tuner selects partial speedup results of the most adjacent neighboring individuals, as many as θ_{nbr}.

Then, overall expected speedup $f_\ell(\mathbf{x}_i)$ with respect to all gradients $\{\nabla g_i(\mathbf{p}_k) \mid k \in N_\ell(\mathbf{x})\}$ is estimated by

$$f_\ell(\mathbf{x}_i) \approx \sum_{k \in N_\ell(\mathbf{x}_i)} \left(w_{(\ell,k)} f_{(\ell,k)}(\mathbf{x}_i) / \sum_k w_{(\ell,k)} \right), \tag{7}$$

where $w_{(\ell,k)} = 1/d_{(\ell,k)}$ is the inverse of the distance. This weight-based estimation follows the perspective: The gradients of more adjacent neighbors are most reliable. If enough speedups of hyperparameter $\mathbf{x}_i \in X_{\text{fix}}$ (i.e., $N_\ell(\mathbf{x}_i) \leq \omega$) cannot be collected from neighbors, then the tuner evaluates $f_\ell(\mathbf{x}_i)$ through a pair of real executions of DOS and AACS.

After an individual's HPO and fitness evaluation end, the individual notifies his neighbors of his new executions. Among neighbors, some neighbors can predict new speedup of a hyperparameter point if the number of the point's neighbors' estimates reaches the minimum neighbor size (which is the same as the dimension of problem-related decision space, ω). Some neighbors can renew a hyperparameter's existing speedup estimate if their numbers of neighbors' estimates are less than θ_{nbr}.

Unlike X_{fix}, the tuner attempts to initialize $X_{\text{rnd},\ell}$ by selecting the neighbors' optimal hyperparameter points that maximize the speedup. Let $x_{\text{best},k}$ be the current best hyperparameter point x_{best} of individual $\mathbf{p}_k$, i.e., $\mathbf{x}_{\text{best},k} = \arg\max_{\mathbf{x}_i \in X_{\text{exp},k}} f_k(\mathbf{x}_i)$. Then, all possible candidates of $X_{\text{rnd},\ell}$, denoted by $X_{\text{can},\ell}$, are derived by $X_{\text{can},\ell} = \{\mathbf{x}_{\text{best},k} \mid \mathbf{p}_k \in \mathbf{P}_{\text{nbr},\ell}\} \setminus X_{\text{fix}}$. If $|X_{\text{can},\ell}|$ is less than a user-defined threshold, denoted by θ_{rnd}, the tuner generates random points to constitute candidate X_{can} whose size is to be θ_{rnd}. With respect to $X_{\text{can},\ell}$, we initialize $X_{\text{rnd},\ell}$ as follows: $X_{\text{rnd},\ell} = \arg\max_{X \subset X_{\text{can},\ell}, |X| = \theta_{\text{rnd}}} \sum_{\mathbf{x}_{\text{best},k} \in X} f_k(\mathbf{x}_{\text{best},k})$. The tuner measures the speedup of each new hyperparameter point in X_{rnd}, and then builds the GP models based on the initial hyperparameter set $X_{\text{exp},\ell} (= X_{\text{fix}} \cup X_{\text{rnd}}$ and their corresponding speedups $Y_{\text{exp},\ell}$.

At the HPO stage, the GP prior is updated whenever $X_{\text{exp},\ell}$ and $Y_{\text{exp},\ell}$ are changed. Let us denote $X_{\text{exp},\ell}$ and $Y_{\text{exp},\ell}$ by two n-dimensional vectors of hyperparameters and their spedups, respectively, such that $X_{\text{exp},\ell} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ and $Y_{\text{exp},\ell} = f_\ell(X_{\text{exp},\ell}) = [f_\ell(\mathbf{x}_1), \ldots, f_\ell(\mathbf{x}_n)]$. We can generally represent the GP prior, $\Pr(\cdot)$, as a multivariate normal distribution of n-dimensional vector:

$$\Pr(X_{\text{exp},\ell}) \approx \mathcal{N}\left(\mu(X_{\text{exp},\ell}), \Sigma(X_{\text{exp},\ell}, X_{\text{exp},\ell})\right), \tag{8}$$

where $\mu(X_{\text{exp},\ell}) = f_\ell(X_{\text{exp},\ell})$ and $\Sigma(X_{\text{exp},\ell}, X_{\text{exp},\ell}) = [K(x_1, x_1), \ldots, K(x_1, x_n); \ldots; K(x_n, x_1), \ldots, K(x_n, x_n)]$. For the covariance function K (i.e., kernel function), one of the most common kernels is the squared exponential covariance function:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2\beta^2}||\mathbf{x}_i - \mathbf{x}_j||^2), \tag{9}$$

where β is a user-defined characteristic length-scale of the process.

Suppose that we infer the value of $f(\mathbf{x}_{\text{new}})$ at new hyperparameter $\mathbf{x}_{\text{new}}$. We estimate $f(\mathbf{x}_{\text{new}})$ based on the given prior using Bayes's rule as follows:

$$f(\mathbf{x}_{\text{new}}) \mid \Pr(X_{\text{exp},\ell}) \approx \mathcal{N}(\mu_*(\mathbf{x}_{\text{new}}), \sigma_*^2(\mathbf{x}_{\text{new}})), \tag{10}$$

$$\mu_*(\mathbf{x}_{\text{new}}) = \Sigma(\mathbf{x}_{\text{new}}, X_{\text{exp},\ell})\Sigma(X_{\text{exp},\ell}, X_{\text{exp},\ell})^{-1}\left(f(X_{\text{exp},\ell}) - \mu(X_{\text{exp},\ell})\right) + \mu(X_{\text{exp},\ell}), \tag{11}$$

$$\sigma_*^2(\mathbf{x}_{\text{new}}) = \Sigma(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{new}}) - \Sigma(\mathbf{x}_{\text{new}}, X_{\text{exp},\ell})\Sigma(X_{\text{exp},\ell}, X_{\text{exp},\ell})^{-1}\Sigma(X_{\text{exp},\ell}, \mathbf{x}_{\text{new}}). \tag{12}$$

Based on the GP-bassed $f(\mathbf{x}_{\text{new}})$ estimation method, the following section details how to acquire the next hyperparameter point based on the posterior distribution.

5.2 Hyperparameter Acquisition Using Gaussian Process Model

Typical GP approaches use an acquisition function to choose a new hyperparameter point. There are a few common acquisition functions. In this work, we use **expected improvement (EI)** as a default acquisition function. The EI function of an individual $\mathbf{p}_\ell$ calculates the expected speedup improvement of any hyperparameter point $\mathbf{x}$ when exploring the vicinity of the current optimum $\mathbf{x}_{\text{best},\ell}$. EI for $\mathbf{p}_\ell$ is calculated by

$$\text{EI}_\ell(\mathbf{x}) = \mathbb{E}[I_\ell(\mathbf{x})] = \int_{-\infty}^{\infty} I_\ell(\mathbf{x})\phi(z)dz, \tag{13}$$

where $\phi(z)$ is the probability density function of the normal distribution $\mathcal{N}(0,1)$, i.e., $\phi(z) = \frac{1}{\sqrt{2\pi}}\exp(-z^2/2)$, and $I(\mathbf{x})$ is the function that measures the difference between the expected speedup at $\mathbf{x}$ and the current optimum value $\mathbf{x}_{\text{best}}$, which is defined as follows:

$$I_\ell(x) = \max\left(0, f_\ell^\circ(\mathbf{x}) - f_\ell(\mathbf{x}_{\text{best}})\right), \tag{14}$$

where $f_\ell^\circ(\mathbf{x}) = f_\ell(\mathbf{x}) \mid \Pr(X_{\text{exp},\ell})$. We need to get rid of operator $\max(\cdot)$ in Equation (14). For that, the integral can be divided into two components: one where $f_j^\circ(\mathbf{x}) - f_\ell(\mathbf{x}_{\text{best}})$ is positive and the other where it is negative. In Reference [16], the point is calculated as $\mathbf{z}_o$ such that $\mathbf{z}_0 = \frac{f_\ell(\mathbf{x}_{\text{best}}) - \mu_*(\mathbf{x})}{\sigma_*(\mathbf{x})}$, where $\mu_*(\mathbf{x})$ and $\sigma_*(\mathbf{x})$ are the mean and the standard deviation of the GP posterior predictive at point $\mathbf{x}$, respectively (see Equations (11) and (12)). Then, function EI is calculated in Reference [16], and expressed by

$$\text{EI}_\ell(\mathbf{x}) = (\mu_*(\mathbf{x}) - f_\ell(\mathbf{x}_{\text{best}}))\,\Phi\left(\frac{\mu_*(\mathbf{x}) - f_\ell(\mathbf{x}_{\text{best}})}{\sigma_*(\mathbf{x})}\right) + \sigma_*(\mathbf{x})\phi\left(\frac{\mu_*(\mathbf{x} - f_\ell(\mathbf{x}_{\text{best}})}{\sigma_*(\mathbf{x})}\right), \tag{15}$$

where $\Phi(\cdot)$ is the standard normal distribution function. Formulation (15) works for $\sigma_*(\mathbf{x}) > 0$; otherwise, if $\sigma_*(\mathbf{x}) = 0$ (as it happens at the observed data point), then it holds that $\text{EI}(\mathbf{x}) = 0$.

Using the EI function, the next hyperparameter point is calculated as

$$\mathbf{x}_{\text{new}} = \arg\max_{\mathbf{x}\in\mathcal{A}} \text{EI}(\mathbf{x}). \tag{16}$$

A variety of methods to solve (16) have been studied. Unlike the objective $f_\ell(\mathbf{x})$ in our speedup optimization problem in (2), $\text{EI}_\ell(\mathbf{x})$ is inexpensive to evaluate and allows easy evaluation of first- and second- order derivatives. We can implement the EI algorithm using a continuous first- or second-order optimization method. We utilize one common technique that has worked well to calculate first derivatives and then use the quasi-Newton method L-BFGS-B [21].

If two points $\mathbf{x}_{\text{best}}$ and $\mathbf{x}_{\text{new}}$ meet the termination condition (Definition 2), then the tuner stops exploration in HPO. Otherwise, after computing $f_\ell(\mathbf{x}_{\text{new}})$ and an individual p_ℓ's fitness through a pair of AACS and DOS executions, the tuner continues the exploration if the exploration budget remains.

6 EXPERIMENTATION

We designed a use case to evaluate the performance of the proposed SBO that optimized the hyperparameter values using a given budget for the simulation speedup. Through various experiments, first, we aim to demonstrate the importance of selecting an optimal hyperparameter point for a

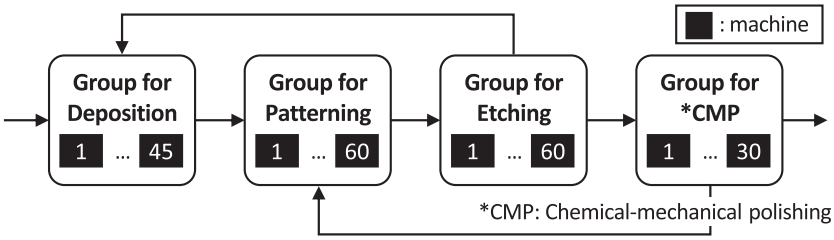


Fig. 3. The structure of the target wafer-fabrication model (four types of machine groups whose number of machines are 45, 60, 60, and 30).

given production scenario by displaying diverse changes in the AACS accuracy and speedup according to the target hyperparameter points. Then, we will show the performance of the proposed HPO-based AACS-based optimization compared to other AACS-based optimizations with fixed hyperparameter points and DOS-based optimization.

As shown in Figure 3, the model has four types of machine groups for deposition, patterning, etching, and chemical-mechanical polishing. The detailed subprocesses in each machine can differ depending on a lot's product recipe. Model parameters are referenced from References [13, 22].

There are five types of products, each requiring different mask layers (such as 4, 8, 12, 16, or 20). As the required number of layers increases, the number of reentrances also increases (e.g., deposition → patterning → etching → deposition $\cdots$). Thus, each wafer lot revisits a machine group multiple times for different processing steps, and the number of reentrances depends on the target lot's product type. Each lot contains 25 wafers (as raw materials). We attempt to produce 100-wafer lots for each product type, i.e., 500-wafer lots for five product types should be produced by different machines, according to their own processing recipes. Depending on the lot-release rate of each product type from inventory, overall performances, such as **mean cycle time (MCT)** or throughput, could vary considerably. Typically, as the intervals of lot releases increase, the MCT becomes longer and the throughput decreases due to the reduction in machine utilization. Rather than multi-objective optimization methods, we conducted a single objective problem by integrating the two important performance indices, MCT and throughput, because a unique control solution is typically required to optimize the target manufacturing plant.

To minimize the MCT and maximize the throughput through SBO, we define a performance index (which is expressed in an integrated form) as follows:

$$\mathrm{PI}(\mathbf{p}) = w_1 k_{\mathrm{mct}}(\mathbf{p}) + w_2 / k_{\mathrm{tp}}(\mathbf{p}), \tag{17}$$

where $\mathbf{p} = (p_1, p_2, \ldots, p_5)$ is an individual that indicates the combination of lot-release intervals for five product types, p_i is the lot-release interval for the ith product type, w_1 and w_2 are user-defined weights, $k_{\mathrm{mct}}(\mathbf{p})$ is the estimated MCT through simulation for population $\mathbf{p}$ for lot-release candidates, and; $k_{\mathrm{tp}}(\mathbf{p})$ is the estimated throughput (average lots produced per hour). SBO is intended to minimize the defined performance index, which minimizes the MCT and maximizes the throughput.

In the first experiment, we measured the speedups and accuracy errors (that is, the estimation differences between AACS and DOS) using simple lot-release scenarios: Twenty-five wafer lots for all product types (i.e., 5 lots per each product type) are released together from inventory at a fixed rate. We present the measurement results in Figure 4, varying the release interval of each product type from 100 to 180 (minutes).

When significance level α becomes lower, the AACS allows more statistical invariances between two consecutive samples (related to the stability decision), which increases the chance of switching

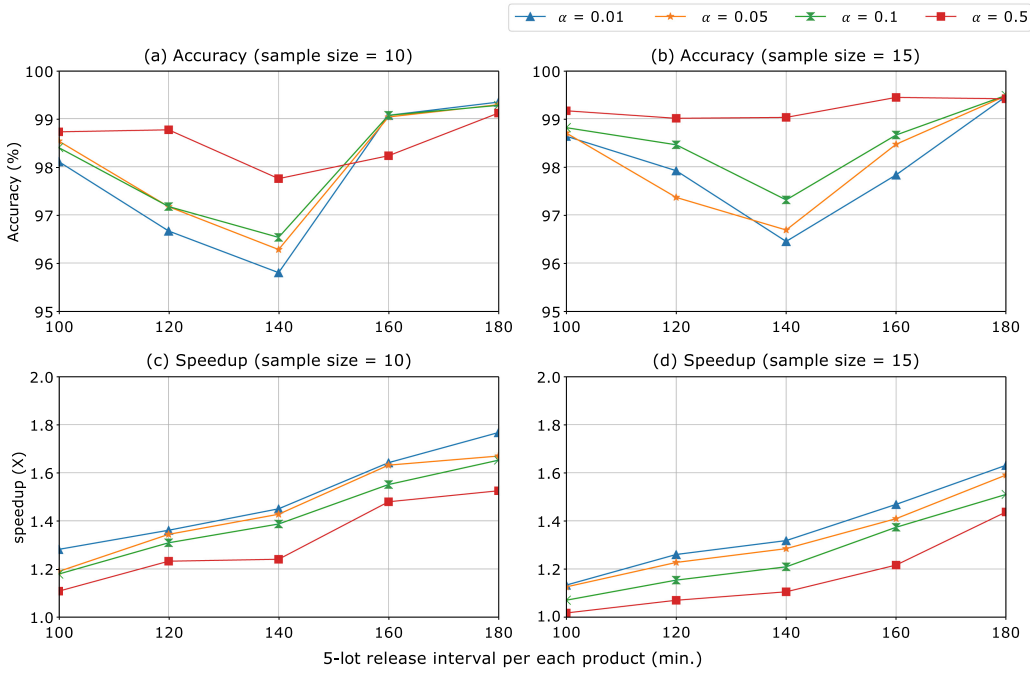


Fig. 4. Speedups and accuracy losses of AACS varying the lot-release intervals of five product types.

from DEMs to MDMs and decreases the chance of switching from MDMs to DEMs through less stringent decisions of steady-state convergences. Having more activated MDMs instead of their corresponding high-fidelity DEMS guarantees more speedup but decreases the accuracy. When the sample size becomes smaller, ALCs detect steady-state convergences (or divergences) more quickly, which helps increase the MDMs' activation periods and guarantees better speedups (see Figure 4(c) and (d)). MDMs process events based on the mean-delay information; the mean information is derived using observed delay data during the sampling period. If the sampling period (that is taken time to collect all observations as many as the defined sample size) is short, then the derived means could be a biased representative of behaviors observed in a specific simulation time region, resulting in lower accuracy.

If the release interval becomes smaller, then the overall production becomes unstable; thus, we cannot get noticeable speedups through AACS because of the lower activation rate of MDMs. If the release interval is extremely low, then each ALC's consecutive samples of queueing parameters differ evidently. In these precarious production scenarios, lower α can accidentally cause ALCs to activate the MDMs. However, these mistakes are corrected afterwards through the subsequent steady-state divergence detection, which increases the active-model switches between DEMs and MDMs and could degrade the simulation execution time to be worse than DOS. As the release interval increases, the MDMs are activated more often, which leads to higher speedups. Since the MDMs guarantee higher accuracy in more stable production scenarios, the accuracy of AACS increases as the release intervals increase. If the release interval is extremely low, then the AACS accuracy also becomes higher, because high-fidelity DEMs are mainly active instead of MDMs. The AACS accuracies are the lowest in an intermediate area, which is neither very stable nor unstable (see the release interval region around 140 in Figure 4).

As such, the speedups and accuracies are different according to the hyperparameter combinations and production scenarios, as shown in Figure 4. When evaluating the fitnesses of various production candidates, we should carefully decide specific hyperparameter points that maximize the speedup under an accuracy constraint, especially in exploring the candidates around an accuracy-deficient region (e.g., the release intervals around 140). As the AACS inaccuracy becomes larger during SBO, the calculated optimal problem-related individual might be far from the actual optimum points derived through DOS-based SBO.

In the second experiment, we aim to find the optimal lot-release intervals of five product types, denoted by $\mathbf{p}_{\text{opt}}$, i.e., $\mathbf{p}_{\text{opt}} = \arg\min_{\mathbf{p}\in\mathbf{P}_{\text{all}}} \text{PI}(\mathbf{p})$, where $\mathbf{P}_{\text{all}} = \{(p_1, \dots, p_5) \mid \forall i \in \{1, \dots 5\}, p_i \in \{100, 105, \dots, 200\}\}$. We assigned two weights considering the scales of two indices, w_1 and w_2, to prioritize throughput over cycle time, which makes the GA-based optimizer prefer to explore the problem-related decision candidates that cause relatively unstable production whose accuracy could problematic in AACS (see Figure 4(a) and (b)). In this case, appropriate hyperparameter selection is essential to prevent a large amount of AACS accuracy.

The GA-based optimizer for the SBO is developed based on the integer-encoding and the Boltzmann selection methods. For the GA, we set the population size, the maximum number of population generations, crossover rate, swap rate, and inversion rate to $15, 30, 0.7, 0.3$, and 0.2, respectively.

For the tuner, considering the stochastic simulation, we set the total simulation budget B_{total} as 12 and the maximum number of exploration B_{exp} as 8. The minimum number of neighbors for the gradient-based speedup estimation is 5 (which equals the dimension of the target decision parameter space). We set the maximum number of neighbors, θ_{nbr}, as 8. The maximum radius of the neighborhood $d_{\max}$ is 7. We set the accuracy constraint for HPO, which is the allowable accuracy loss, as 2.5%. The tuner explores a defined hyperparameter space $\{(\sigma, m)\}$ such that $0.02 \leq \sigma \leq 1.0$ and $m = \{10, 11, \dots, 20\}$. The set of fixed points, X_{fix}, consists of 16 points that are equally distributed over the defined space, i.e., $\{(\sigma, m)\} = \{(0.02, 10), (0.3476, 10), (0.6752, 10), (1.0, 10), (0.02, 16), \dots, (1.0, 20)\}$. If a hyperparameter point's α is one, then we equally assign the point's measured speedup as 1.0 without executing a pair of DOS and AACS (i.e., $1 \in A_{\text{dos}}$) to trigger the DOS-based fitness evaluation when this hyperparameter point is decided as an optimal point. The size of additional random points for the GP initialization, X_{rnd}, is 3. During the GP-based HP optimization using L-BFGS-B, the maximum number of iterations is 20.

Based on the specified parameter values, we traced the execution times of evaluation and exploration of each individual using an experimental machine with an Intel Xeon E5-1650 3.6 GHz CPU and 32 GB of memory. During a single execution of SBO, the averages of each individual's budget expense for the HP exploitation changed over the population generation, as shown in Figure 5. As the number of population generations increases, the tuner can estimate the speedup of fixed points in X_{fix} using neighbors' previous exploration results without simulation. In the beginning, the entire exploration budget B_{exp} (= 8) is consumed. In this case, the fitness is the average of four DOS results obtained during the HP exploration and four AACS results using the current optimum hyperparameter point. As the generation evolves, the expense of the HPO budget decreases, which helps allocate more budget to the fitness evaluation, considering stochastic properties. The longer an individual survives or the more newly generated by the GA-based optimizer, the likelier it is that we can reduce the possibility of wrong fitness prediction caused by stochastic variances. During the SBO execution, the hyperparameter points used to evaluate the whole population's fitnesses through AACS are distributed as shown in Figure 6.

Based on the outlined manufacturing and AACS configurations, Figure 7 offers our comparative assessment of the SBO performances, both in terms of overall execution times and the quality of the computed optimal manufacturing solutions, with a focus on achieving minimized

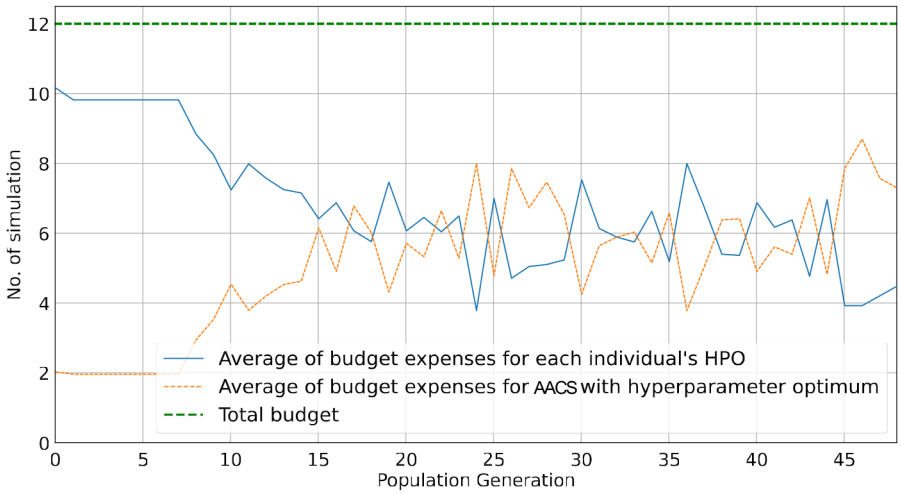


Fig. 5. Traces of average budget expenses for the HP explorations and Execution-time traces of exploration and evaluations of the individual candidates.

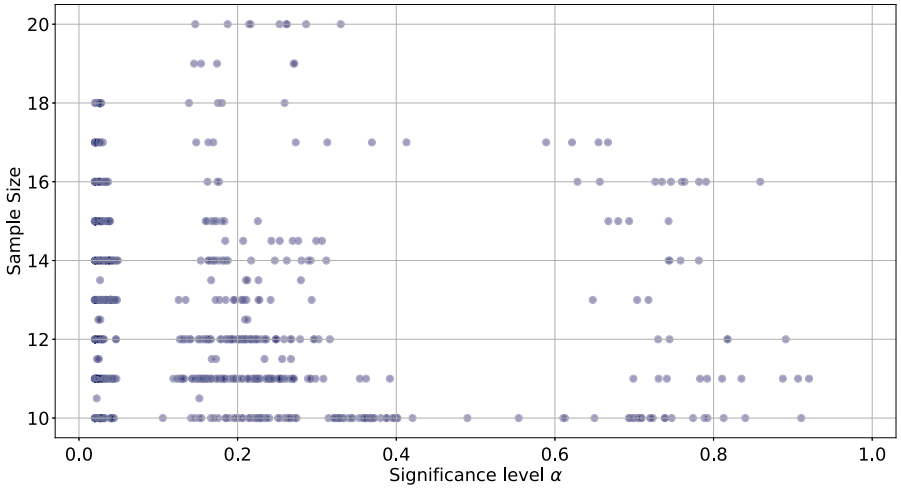


Fig. 6. Distribution of hyperparameter configurations employed in AACS-based fitness assessments during SBO.

average fitness. This comparative assessment encompasses three distinct simulation paradigms: (1) DOSs, (2) AACS that consistently operates with fixed hyperparameters throughout SBO, and (3) AACS wherein hyperparameters undergo adaptive refinement via HPO. For the second simulation method within the SBO, we used the static hyperparameter configurations where alpha is either $0.01, 0.05, 0.1,$ and 0.5, all paired with a sample size of 20. As shown in Figure 7(a), lower α values with the fixed sample size reduce the overall executions of the SBO. We recomputed the fitness of problem-related optimal individuals by running DOS 50 times (see Figure 7(b)).

Among the fixed α values, the minimum α ($= 0.01$) reduces the overall execution time to about 345 seconds (about 22.8%) compared to the optimization using DOS. However, the fitness of the optimum individual derived through AACS-based optimization using the minimum α is as much

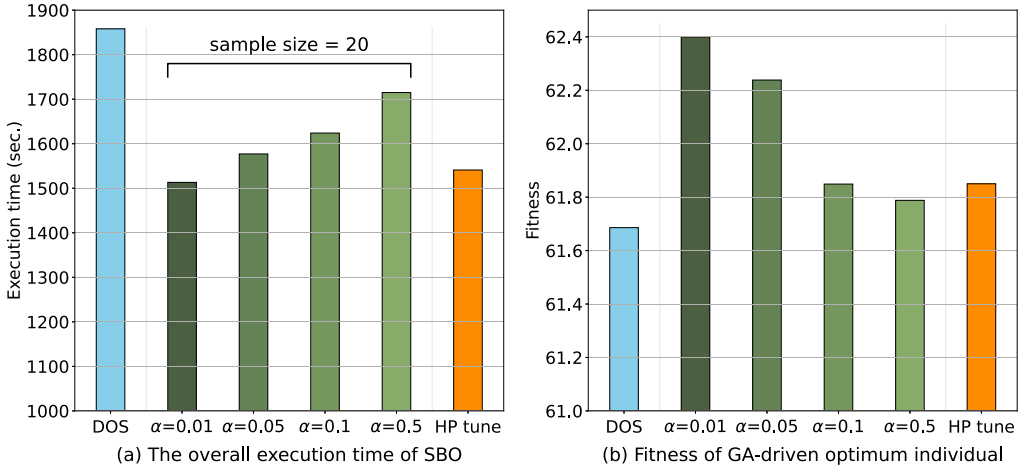


Fig. 7. Comparison of SBO execution times and average fitness of determined optima (lower is favored) using DOS, fixed hyperparameter simulations (green bars), and HPO-driven simulation.

as 2.4% higher than that of the individual obtained by typical DOS-based optimization. The proposed SBO approach (labeled as α tune in Figure 7) adopts different hyperparameter points for each individual to maximize the speedups under the defined accuracy constraint. Our method resulted in a slightly lower speedup (about 20.5%) compared to AACS-driven SBO with the minimum α. However, the fitness of the optimum individual according to HPO-based SBO is closer to the actual optimum (driven by DOS-based SBO). Compared to other hyperparameter points (whose α values are 0.05, 0.1, and 0.5), the proposed method guarantees a similar quality of problem-related optimum individual with more reduced execution times. Moreover, our approach eradicates the efforts or concerns about hyperparameter selection.

It is still necessary to reduce the gap between the fitness of optimal individuals obtained by our method and by DOS-based optimization. The reason is assumed to be that we fixed the accuracy constraint among individuals. In future works, we will design a new hyperparameter tuning method for SBO with adaptive control of accuracy constraints. We will also extend our approach using the optimal computing budget allocation concept [4] to adjust the simulation budget, considering both the probability of correct selection and HBO.

7 CONCLUSION

We previously proposed AACS through steady-state convergence or divergence detection based on two consecutive queueing-parameter distributions (sampled from DEMs of machine groups). The speedup and accuracy loss of the AACS vary according to (1) the specific hyperparameter point (a combination of significance level α and the sample size) and (2) the overall stability of the simulation scenarios. When exploring problem-related individuals over decision candidates during GA-based SBO, we must carefully select hyperparameters to maximize the speedup under a given accuracy constraint. To achieve the goal, we proposed an efficient hyperparameter tuning method to find the optimal hyperparameter point used for each individual's AACS during SBO. To find optimal hyperparameter points efficiently with few attempts, we employed the GP model for each individual, which is used to estimate the speedup of the next hyperparameter candidate in advance as a surrogate model. We proposed the discrete-gradient-based speedup estimation method to prepare the data (used for the GP initialization) without actual simulation execution based on neighboring individuals' previous exploration results. As the population generation evolves and

optimization converges, the hyperparameter exploration overheads are reduced via the estimation of more GP models' warmup data (used to initialize the GP odels) using the exploration results of enlarged neighboring individuals. The proposed method was applied to the lot-release-interval optimization of a large-scale wafer-fabrication scenario. It showed that the proposed method generates highly optimized results (compared to other AACSs with fixed hyperparameter points) with reduced execution times.

REFERENCES

[1] Aldeida Aleti and Irene Moser. 2016. A systematic literature review of adaptive parameter control methods for evolutionary algorithms. *ACM Comput. Surv.* 49, 3 (2016), 1–35.

[2] June-Young Bang and Yeong-Dae Kim. 2010. Hierarchical production planning for semiconductor wafer fabrication based on linear programming and discrete-event simulation. *IEEE Transactions on Automation Science and Engineering* 7, 2 (2010), 326–336. https://doi.org/10.1109/TASE.2009.2021462

[3] Jennifer Bekki, J. W. Fowler, Gerald Mackulak, and Barry Nelson. 2010. Indirect cycle time quantile estimation using the Cornish–Fisher expansion. *IIE Trans.* 42 (01 2010). https://doi.org/10.1080/07408170903019135

[4] Chun-Hung Chen and Loo Hay Lee. 2011. *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation.* Vol. 1. World Scientific.

[5] M. A. Chik, A. B. Rahim, A. Z. Md Rejab, K. Ibrahim, and U. Hashim. 2014. Discrete event simulation modeling for semiconductor fabrication operation. In *Proceedings of the IEEE International Conference on Semiconductor Electronics (ICSE'14).* 325–328. https://doi.org/10.1109/SMELEC.2014.6920863

[6] D. P. Connors, G. E. Feigin, and D. D. Yao. 1996. A queueing network model for semiconductor manufacturing. *IEEE Trans. Semiconduct. Manufact.* 9, 3 (1996), 412–427. https://doi.org/10.1109/66.536112

[7] Carlos Alberto Barrera Diaz, Tehseen Aslam, and Amos H. C. Ng. 2021. Optimizing reconfigurable manufacturing systems for fluctuating production volumes: A simulation-based multi-objective approach. *IEEE Access* 9 (2021), 144195–144210. https://doi.org/10.1109/ACCESS.2021.3122239

[8] Ágoston E. Eiben, Robert Hinterding, and Zbigniew Michalewicz. 1999. Parameter control in evolutionary algorithms. *IEEE Trans. Evol. Comput.* 3, 2 (1999), 124–141.

[9] Matthias Feurer and Frank Hutter. 2019. Hyperparameter optimization. In *Automated Machine Learning*. Springer, Cham, 3–33.

[10] D. Fronckowiak, A. Peikert, and K. Nishinohara. 1996. Using discrete event simulation to analyze the impact of job priorities on cycle time in semiconductor manufacturing. In *Proceedings of the IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop on Theme-Innovative Approaches to Growth in the Semiconductor Industry (ASMC'96).* 151–155. https://doi.org/10.1109/ASMC.1996.557987

[11] Dean Grosbard, Adar Kalir, Israel Tirkel, and Gad Rabinowitz. 2013. A queuing network model for wafer fabrication using decomposition without aggregation. In *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE'13).* 717–722. https://doi.org/10.1109/CoASE.2013.6653941

[12] Changwu Huang, Yuanxiang Li, and Xin Yao. 2019. A survey of automatic parameter tuning methods for metaheuristics. *IEEE Transactions on Evolutionary Computation* 24, 2 (2019), 201–216.

[13] James P. Ignizio. 2009. *Optimizing Factory Performance: Cost-Effective Ways to Achieve Significant and Sustainable Improvement* (1st ed.). McGraw–Hill, New York, NY.

[14] J. Jimenez, B. Kim, J. Fowler, G. Mackulak, and You In Choung. 2002. Operational modeling and simulation of an interbay AMHS in semiconductor wafer fabrication. In *Proceedings of the Winter Simulation Conference*, Vol. 2. 1377–1382. https://doi.org/10.1109/WSC.2002.1166405

[15] Rachel T. Johnson, John W. Fowler, and Gerald T. Mackulak. 2005. A discrete event simulation model simplification technique. In *Proceedings of the 37th Conference on Winter Simulation (WSC'05).* Winter Simulation Conference, 2172–2176.

[16] Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient global optimization of expensive black-box functions. *J. Global Optim.* 13, 4 (1998), 455–492.

[17] Yeong-Dae Kim, Sang-Oh Shim, Bum Choi, and Hark Hwang. 2003. Simplification methods for accelerating simulation-based real-time scheduling in a semiconductor wafer fabrication facility. *IEEE Transactions on Semiconductor Manufacturing* 16, 2 (2003), 290–298. https://doi.org/10.1109/TSM.2003.811890

[18] Donald E. Knuth. 1997. *The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms.* Addison Wesley Longman.

[19] Oliver Kramer. 2010. Evolutionary self-adaptation: A survey of operators and strategy parameters. *Evol. Intell.* 3, 2 (2010), 51–65.

[20] James T. Lin and Chien-Ming Chen. 2015. Simulation optimization approach for hybrid flow shop scheduling problem in semiconductor back-end manufacturing. *Simul. Model. Pract. Theory* 51 (2015), 100–114. https://doi.org/10.1016/j.simpat.2014.10.008

[21] Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Program.* 45, 1 (1989), 503–528.

[22] Lars Mönch, John W. Fowler, and Scott J. Mason. 2012. *Production Planning and Control for Semiconductor Wafer Fabrication Facilities: Modeling, Analysis, and Systems* (1st ed.). Springer, New York, NY.

[23] D. Nazzal and L. F. McGinnis. 2005. Queuing models of vehicle-based automated material handling systems in semiconductor fabs. In *Proceedings of the Winter Simulation Conference.* 8 Pages. https://doi.org/10.1109/WSC.2005.1574540

[24] Ashkan Negahban and Jeffrey S. Smith. 2014. Simulation for manufacturing system design and operation: Literature review and analysis. *J. Manufact. Syst.* 33, 2 (2014), 241–261. https://doi.org/10.1016/j.jmsy.2013.12.007

[25] Oliver Rose. 2007. Improved simple simulation models for semiconductor wafer factories. In *Proceedings of the Winter Simulation Conference.* 1708–1712. https://doi.org/10.1109/WSC.2007.4419793

[26] Moon Gi Seok, Wentong Cai, and Daejin Park. 2021. Hierarchical aggregation/disaggregation for adaptive abstraction-level conversion in digital twin-based smart semiconductor manufacturing. *IEEE Access* 9 (2021), 71145–71158. https://doi.org/10.1109/ACCESS.2021.3073618

[27] Moon Gi Seok, Wentong Cai, Hessam S. Sarjoughian, and Daejin Park. 2020. Adaptive abstraction-level conversion framework for accelerated discrete-event simulation in smart semiconductor manufacturing. *IEEE Access* 8 (2020), 165247–165262. https://doi.org/10.1109/ACCESS.2020.3022276

[28] Moon G. I. Seok, Chew Wye Chan, Wentong Cai, Hessam S. Sarjoughian, and Daejin Park. 2020. Runtime abstraction-level conversion of discrete-event wafer-fabrication models for simulation acceleration. In *Proceedings of the ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS'20).* Association for Computing Machinery, New York, NY, 83–92. https://doi.org/10.1145/3384441.3395982

[29] Moon Gi Seok, Wen Jun Tan, Wentong Cai, and Daejin Park. 2023. Digital-twin consistency checking based on observed timed events with unobservable transitions in smart manufacturing. *IEEE Transactions on Industrial Informatics* 19, 4 (2023), 6208–6219. https://doi.org/10.1109/TII.2022.3200598

[30] Moon Gi Seok, Wen Jun Tan, Boyi Su, and Wentong Cai. 2022. Hyperparameter tunning in simulation-based optimization for adaptive digital-twin abstraction control of smart manufacturing system. In *Proceedings of the ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS'22).* Association for Computing Machinery, New York, NY, 61–68. https://doi.org/10.1145/3518997.3531024

[31] E. S. Skakov and V. N. Malysh. 2018. Parameter meta-optimization of metaheuristics of solving specific NP-hard facility location problem. In *Journal of Physics: Conference Series*, Vol. 973. IOP Publishing, 012063.

[32] C. P. L. Veeger, L. F. P. Etman, E. Lefeber, I. J. B. F. Adan, J. van Herk, and J. E. Rooda. 2011. Predicting cycle time distributions for integrated processing workstations: An aggregate modeling approach. *IEEE Trans. Semiconduct. Manufact.* 24, 2 (2011), 223–236. https://doi.org/10.1109/TSM.2010.2094630

[33] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. 2019. Hyperparameter optimization for machine learning models based on Bayesian optimization. *J. Electr. Sci. Technol.* 17, 1 (2019), 26–40.

[34] Li Yang and Abdallah Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (2020), 295–316.

[35] Enrique Ruiz Zúñiga, Matias Urenda Moris, Anna Syberfeldt, Masood Fathi, and Juan Carlos Rubio-Romero. 2020. A simulation-based optimization methodology for facility layout design in manufacturing. *IEEE Access* 8 (2020), 163818–163828. https://doi.org/10.1109/ACCESS.2020.3021753

Received 23 December 2022; revised 9 October 2023; accepted 11 January 2024