ASME Journal of Manufacturing Science and Engineering
Online journal at:
https://asmedigitalcollection.asme.org/manufacturingscience

Digital Twinning and Optimization of Manufacturing Process Flows

Hankang Lee
Industrial and Manufacturing Engineering,
The Pennsylvania State University,
University Park, PA 16802
e-mail: hxl5495@psu.edu

Hui Yang[1]
Industrial and Manufacturing Engineering,
The Pennsylvania State University,
University Park, PA 16802
e-mail: huiyang@psu.edu

The new wave of Industry 4.0 is transforming manufacturing factories into data-rich environments. This provides an unprecedented opportunity to feed large amounts of sensing data collected from the physical factory into the construction of digital twin (DT) in cyberspace. However, little has been done to fully utilize the DT technology to improve the smartness and autonomous levels of small and medium-sized manufacturing factories. Indeed, only a small fraction of small and medium-sized manufacturers (SMMs) has considered implementing DT technology. There is an urgent need to exploit the full potential of data analytics and simulation-enabled DTs for advanced manufacturing. Hence, this paper presents the design and development of DT models for simulation optimization of manufacturing process flows. First, we develop a multi-agent simulation model that describes nonlinear and stochastic dynamics among a network of interactive manufacturing things, including customers, machines, automated guided vehicles (AGVs), queues, and jobs. Second, we propose a statistical metamodeling approach to design sequential computer experiments to optimize the utilization of AGV under uncertainty. Third, we construct two new graph models—job flow graph and AGV traveling graph—to track and monitor the real-time performance of manufacturing jobshops. The proposed simulation-enabled DT approach is evaluated and validated with experimental studies for the representation of a real-world manufacturing factory. Experimental results show that the proposed methodology effectively transforms a manufacturing jobshop into a new generation of DT-enabled smart factories. The sequential design of experiments effectively reduces the computation overhead of expensive simulations while optimally scheduling the AGV to achieve production throughput cost-effectively. This research is strongly promised to help SMMs fully utilize big data and DT technology for gaining competitive advantages in the global marketplace.
[DOI: 10.1115/1.4063234]

Keywords: design of experiments, digital twin, multi-agent simulation, manufacturing systems, graph model

1 Introduction

Rapid advances in sensing technology usher in a new generation of the Internet of Manufacturing Things, which brings the proliferation of data in the manufacturing jobshop [1]. As such, information visibility is significantly increased in this data-rich environment regarding real-time states and actions of manufacturing things (e.g., sensors, machines, transports, jobs, and queues) in every corner of the factory, thereby providing an unprecedented opportunity to transform a large amount of sensing data from the physical factory into the construction of digital twin (DT) in cyberspace. DT is a digital representation of manufacturing processes that allows for real-time data collection from the physical factory, transformation of the data into intelligence, and simultaneous feedback to the jobshop [2,3]. DT offers a high level of cyber-physical integration to derive data-driven intelligence for smart manufacturing operations.

However, only a small fraction of small and medium-sized manufacturers (SMMs) has initiated or planned to implement the DT technology. The Technology-in-Industry Report of Automation Alley showed that approximately 2% of SMMs in Michigan had implemented artificial intelligence (AI) and/or DT-related technology [4]. Moreover, 48% of respondents had no intention to implement AI technology into their businesses, and almost 30% stated that AI would not increase the competitive advantages of operations. Indeed, many SMMs do not realize the benefits of either AI or DT technology, in part due to constraints in capital investment and technical support hampering their ability to implement DT technology, and partly because SMMs lack advanced sensor technology for data collection and digital representation in cyberspace [5]. Characterized by low-volume production of various types, SMMs necessitate implementing flexible production systems. Additionally, volatility in market demand requires the ability to make decisions quickly. Therefore, it is imperative to address these challenges facing SMMs and realize

[1]Corresponding author.
Manuscript received April 21, 2023; final manuscript received August 16, 2023; published online September 11, 2023. Assoc. Editor: Qing (Cindy) Chang.

the full potential of data analytics and DT technology in advanced manufacturing.

DT technology overcomes practical limitations and offers the flexibility to run experiments in cyberspace, which helps analyze and evaluate various decision-making rules in manufacturing systems [6]. Simulation experiments on DT models are effective and cost-efficient approaches to investigating new decisions, reducing risk, and predicting system behaviors under a particular set of "what-if" scenarios [7]. For example, new policies can be tested without consuming physical resources for acquisition. Also, the model allows to solve optimization problems in manufacturing processes. Time can be compressed or expanded in the model by running simulations. Therefore, simulation modeling with DT is indispensable for manufacturing system analysis. Simulation-enabled optimization of DT models is conducive to improving the performance of manufacturing operations such as facility layout planning, factory design, production scheduling, and quality engineering.

However, computer models of manufacturing systems involve higher levels of complexity, such as functional non-convexity and nonlinear interactions among a network of machines, queues, material handling systems (MHSs), and jobs. It is also worth mentioning that machines and MHSs can degrade or fail, queues can be blocked, and jobs can become backlogged. For example, a multi-agent simulation model of a jobshop with multiple machines and MHSs involves a large number of attribute settings for each machine or MHS, which brings the "curse of dimensionality" problem and expensive computations during the model calibration and optimization. Traditional linear and nonlinear optimization methods (e.g., linear programming, neural networks, genetic algorithms) tend to be limited in handling optimization tasks. Additionally, these conventional methods are black-box approaches that provide limited information about the underlying physics of the manufacturing process. Therefore, there is an urgent need to improve computational efficiency and design effective simulation experiments for manufacturing optimization.

This paper presents the design and development of DT models of manufacturing jobshops for the simulation and optimization of manufacturing process flows. First, we develop a multi-agent simulation model to describe stochastic and nonlinear characteristics among interconnected networks of intelligent manufacturing things, including customers, automated guided vehicles (AGVs), machines, queues, and jobs. Second, we propose a statistical metamodeling approach for the sequential design of computer experiments to optimize the utilization of MHSs under the uncertainty of jobs, machines, and queues. Third, we formulate novel graph models—job flow graph and AGV traveling graph—to monitor and track the real-time status of manufacturing jobshops. These models exploit network-like relationships and provide novel insights into process flows. The proposed simulation-enabled DT approach is evaluated and validated through experimental studies for a representation of a real-world manufacturing jobshop. Experimental results demonstrate that the proposed methodology effectively transforms manufacturing jobshops into a new generation of smart factories. The sequential design of experiments effectively reduces the computation overhead of expensive simulations, while optimally scheduling MHSs to achieve production throughput cost-effectively. Data analytics and DT technology are strongly promising to help SMMs gain a competitive advantage in the global market.

The remainder of this paper is organized as follows: Sec. 2 introduces the research background of digital twins; Sec. 3 details the proposed methodology of the multi-agent simulation model and statistical metamodeling for sequential designs of simulation experiments; Sec. 4 provides experimental design; Sec. 5 evaluates and validates the proposed methodology in simulation experiments; and Sec. 6 concludes this research.

2 Research Background

Manufacturing systems embodying the characteristics of smart factories show advanced levels of intelligence and autonomy in multiple aspects, including process control, machine scheduling, production planning, and maintenance strategies [8]. Digital simulation models have become essential for smart manufacturing, enabling cost-effective analysis and real-time state updates within a factory [9]. Accordingly, research on smart manufacturing tends to focus on addressing specific problems within a conceptual framework that is validated through model-based analysis and simulation experiments. There are a variety of simulation modeling methods, including discrete event simulation (DES) and system dynamics, to address problems in manufacturing processes. For example, Mittal et al. [10] introduced a physical-based nonlinear stochastic differential equation approach to capture downtime dynamics, while Yang et al. [11] developed a system dynamics model to represent multi-stage assembly lines as continuous fluid flows. Simulation experiments also helped synchronize the digital model with the real factory to capture process parameters for building DT [12]. To acquire these input parameters, the manufacturing system was separated into four modules (i.e., fabrication, logistics, storage, and inspection), and DES models were constructed for representing each module. DES can be combined with a metaheuristic, such as genetic algorithms, to obtain an optimal solution for the step size, batch size, and the number of active lines in a production system while reducing the required computational capacity [13].

However, the manufacturing process involves a diverse range of interactive manufacturing objects and systems, which leads to increased complexity in digital models. As the quantity of manufacturing objects in a factory increases, the number of events that occur also increases exponentially, making simulations more challenging as more computational overhead is necessitated to manage the growing complexity. Traditional DES models are passive and operate based on system definition, which limits their ability to represent real-world manufacturing systems. To overcome these limitations, multi-agent system (MAS) approaches have been introduced, which represent various behaviors and interactions of manufacturing objects, such as workpieces, AGVs, machines, and queues, by modeling agents. These agents can actively behave and interact with each other through their unique characteristics and actions [14], making MAS a more effective approach for modeling complex manufacturing systems. Furthermore, agent-based models (ABM) represent and simulate complex systems consisting of interactions between autonomous agents with unique properties [15,16]. Overall, MAS and ABM provide more accurate and comprehensive representations of manufacturing systems, allowing better decision-making and process optimization.

With rapid advances in smart manufacturing, the production paradigm has experienced the transformative shift from mass production to mass customization and personalization [17]. In conventional mass production systems, decision-making procedures are characterized as repetitive and periodic processes, allowing researchers to pursue optimal solutions through a centralized decision-making approach [18,19]. However, personalized production systems involve the creation of diverse products with distinct operation sequences, rendering centralized decision-making inefficient in responding to this variability [20]. To address this issue, previous studies have sought to address this challenge by defining specific functions as agents and constructing multi-agent models for modeling and simulating manufacturing systems.

For example, a manufacturing execution system was modeled by ABM, incorporating decision-making at the enterprise level by reflecting relationships between order agents and resource agents [21]. Huang and Liao [22] developed a negotiation approach to decision-making models for distributed machine scheduling in parallel machine production systems. Their model consisted of job agents, machine agents, and management agents, representing characteristics of jobs, machines, and supervisors, respectively. The multi-agent factory model developed by Giordani et al. [23] enabled factories to model the decentralized decision-making of machines and mobile robots by decomposing manufacturing systems into production planning and production scheduling levels. In their model, ABM could make decisions at each level.

Adediran et al. [24] proposed an ABM approach integrated with heuristic algorithms to solve disruption problems in flow shop production caused by customers' unexpected alterations in demand quantity, delivery time, and process sequence. In their approach, large-scale manufacturing systems are decomposed into different types of agents, such as customer orders, machines, and operators. Each agent is assigned unique actions and message sequences, facilitating communication and coordination among the different agent types. In another study, Kim et al. [18] developed reinforcement learning models to furnish the MAS with intelligence. Their distributed decision-making approach was applied in low-volume and high-diversity serial production systems, where the system was partitioned into three layers, each layer consisting of different types of interconnected agents.

However, although agents are employed to represent distributed decision-making in factories, their structure characterizes a centralized decision-making procedure in the context of a manufacturing execution system. This characteristic poses limitations in modeling the autonomy and flexibility of systems and providing real-time synchronization of data and decision-making in a dynamically changing environment. Consequently, there is an urgent need to develop a digital factory model that empowers manufacturing objects to make decisions autonomously, interact with other objects, and acquire the intelligence needed for decision-making through learning from the manufacturing system.

This paper introduces a multi-agent simulation model that addresses these issues. Each agent incorporated in the model reflects the characteristics of corresponding manufacturing devices on the shop floor, enabling distributed and automated decision-making. Interactions between these devices are modeled as message transactions. In this study, it is assumed that radio-frequency identification (RFID) sensors are installed in the jobshop to facilitate data collection from these message-passing processes. RFID is a matured technology that identifies and locates assets by capturing wireless data [25]. These sensors can collect real-time process flow status through state transitions and transactions performed by interactions during production.

Real-time and flexible DT models are essential for achieving autonomy in smart manufacturing, especially in personalized production, where processes vary depending on the type of jobs. However, there is a lack of research on real-time manufacturing modeling for sensor-based intelligence embedded in manufacturing objects, which is crucial to realize the full potential of smart manufacturing systems. To address this gap, we propose a multi-agent simulation-enabled DT model that represents the characteristics of the manufacturing things (e.g., jobs/workpieces, MHSs, machines, and queues) as multi-agents and captures their interconnectivity. By leveraging data collected from smart sensors, the agents reflect the behaviors and decisions of each object that actively interacts with other objects through the cloud environment and intelligence from simulation. These decisions are then delivered to manufacturing things in physical factories in the form of production plans, process control, or operation schedules. Variations in states and decisions of manufacturing things can be reflected by the structure and topology of the network model in cyberspace [26]. Network models can also capture the interaction patterns between things in manufacturing systems and reduce the system to a summarized topology [27]. As such, this network model provides a sparse representation of complex system states, interactive dynamics, and interconnected and distributed activities [28].

Constructing a multi-agent simulation model of a jobshop with multiple machines and MHSs is a challenging task due to the need to consider numerous attribute settings for each machine and MHS. This challenge, known as the "curse of dimensionality," makes model calibration and optimization complex [29]. Moreover, manufacturing systems typically involve a large number of manufacturing things, resulting in multi-optimization problems at the process level. As a result, model calibration can be computationally expensive as it requires the simulation of complex factory models in a high-dimensional design space. This poses significant challenges to traditional optimization methods, such as genetic algorithms, linear programming, and metaheuristics, in computer experiments and DT model optimization [30]. Simulations of manufacturing systems are computationally expensive due to the complex, nonlinear, and non-convex nature of the models. This complexity makes it difficult to optimize the design space effectively, leading to challenges in both computer experiments and DT model optimization. To address these issues, the sequential design of experiments offers a more cost-efficient approach that focuses on optimal design and model calibration while minimizing the simulation cost and improving the running speed. This approach has the benefit of representing latent effects between factors and responses through statistical metamodeling for optimization. The metamodel helps search for the optimal solution as well as identify the relationship between control factors and responses. The proposed design approach allows for the exploration of the intelligence of system attributes with the analysis of this relationship.

3 Research Methodology

In this paper, we design and develop a DT model of manufacturing jobshops for modeling and optimization of job process flows. As illustrated in Fig. 1, the proposed simulation-enabled DT approach is composed of three key components:

(1) A multi-agent simulation model is developed to comprehensively describe the complexity and dynamics inherent in digital-integrated manufacturing systems. This model captures the distinctive characteristics of manufacturing devices and represents their actions and states within the overall system. The proposed model reflects the active interactions between various manufacturing things through message transactions and action triggers.
(2) Statistical metamodeling is developed to address the challenges arising from the high levels of complexity associated with manufacturing systems. The sequential design approach incorporating statistical metamodeling significantly improves the computational efficiency of simulation experiments.
(3) Graph models are designed to characterize interactions between manufacturing things to provide insights into the process flows in interactive manufacturing networks.

The combination of these three methods enables a comprehensive and efficient representation of manufacturing jobshops and facilitates the analysis of job process flows.

3.1 Multi-Agent Simulation Model. The multi-agent simulation model of physical factories consists of a variety of manufacturing agents, each representing various manufacturing things, such as AGVs, machines, queues, and jobs. This model represents digital

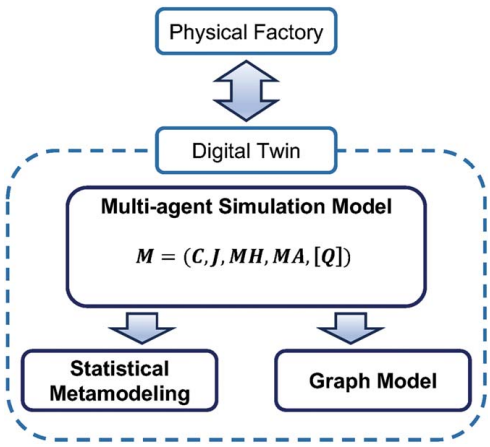

Fig. 1 Flowchart of research methodology

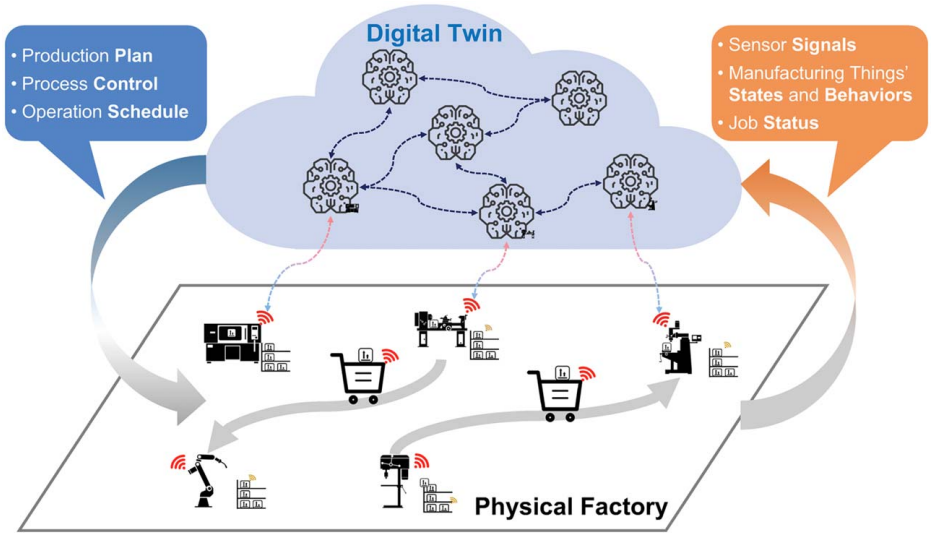

Fig. 2 Illustration of Digital Twin and interconnection between manufacturing things

integration within manufacturing systems and process flows. The proposed model is structured into two layers: the cloud layer and the jobshop layer. The cloud layer serves as the decision-making level, providing an interconnected environment of manufacturing agents in cyberspace. It facilitates communication and interaction among diverse manufacturing agents. On the other hand, the jobshop layer acts as a virtual representation of physical shop floors. The manufacturing things on the jobshop layer actively and autonomously process jobs. As depicted in Fig. 2, sensor signals, including RFID transactions and real-time status information, are collected from manufacturing objects in the jobshop. The states and actions of the manufacturing things are recorded in their digital representations, as outlined in Table 1. Manufacturing objects or things (e.g., jobs/workpieces, MHSs, machines, and queues) in the physical world are reflected as a variety of agents (e.g., Job agents, material handling agents (MHAs), Machine agents, and queue agents) in the digital world. By leveraging the collected data, AI techniques, and simulation results, manufacturing intelligence is constructed in DT. This process enables DT to inform decision-making and provide real-time feedback to its physical counterpart. The proposed approach effectively models and captures these relationships, facilitating the modeling of DT with the message transactions.

The proposed multi-agent simulation model is defined with five tuples, as shown in Eq. (1)

$$M = \left(C, \boldsymbol{J}, \boldsymbol{MH}, \boldsymbol{MA}, I^{\mathrm{Sys}}\right) \tag{1}$$

where C represents a cloud layer, and $\boldsymbol{J}$, $\boldsymbol{MH}$, and $\boldsymbol{MA}$ correspond sets of job agents, MHAs, and machine agents, respectively. I^{Sys} is a tuple of system inputs that include the simulation length, replication number, job arrival time, and a set of parameters for characterizing manufacturing agents.

The proposed model characterizes manufacturing process flows and interactions as shown in Fig. 3. The process begins when a customer order is delivered to the enterprise (a). Then, the cloud system receives the work order, creates a production plan, and transmits it to the manufacturing shop. Subsequently, a job is generated (b), initiating the production processes, and the job is called a work-in-process (WIP) (c). Throughout the production processes, WIP actively interacts with various manufacturing things. For example, it is transported within the factory premises by MHSs to reach its designated destination (d-1). Upon arrival at a machine, the WIP enters the corresponding queue (d-2), subsequently being processed by the machine (d-3). After completing the task, the WIP summons the MHSs to proceed to the next task (d-1). Because a job may involve multiple production tasks, these repetitive actions of the WIP are illustrated as a circular flow in Fig. 3. When all necessary tasks are completed, the WIP reaches a warehouse and becomes a final product (e). This product exits the factory system and is ready to ship to customers. These interactions between the manufacturing things are effectively represented through message transactions occurring during state transitions and actions of the respective agents.

Table 1 Definition of agents in DT

Physical factory	Digital Twin
Job/Workpiece	Job agent
Material handling system (AGVs, gantry system)	Material handling agent
Machine (drilling, lathe, milling, welding)	Machine agent
Queue	Queue agent

3.1.1 Job Agent. A Job agent (J) is a model of workpieces in a production system. The definition of Job agent is presented in Eq. (2)

$$J = \left(S^{\mathrm{Job}}, A^{\mathrm{Job}}, I^{\mathrm{Job}}\right) \tag{2}$$

where S^{Job} is a set of Job states, A^{Job} is a Job action set, and I^{Job} is a tuple of Job inputs, including Job task sequence and basic information about the Job, such as Job ID, type, and created time.

The states of a Job agent (S^{Job}) are displayed in a state-action diagram (Fig. 4(*a*)) along with its associated actions. When a work order arrives at the factory, a Job agent is generated and plans its task sequences (S_1^{Job}). Then, its state transitions to the "Initial Planning" phase, which consists of three states representing the journey from the entrance of the factory to the queue of the first machine. Once the planning phase is complete, the Job agent enters the process task phases. The process task phase is recursive and consists of six states as shown by a blue-dashed circle in Fig. 4(*a*). After the job enters the Queue-In, it transitions to the "in queue" state (S_5^{Job}) and waits for the process to begin. When the process starts, the state moves to "being processed" (S_6^{Job}), and after the process completes, the job waits on the machine until space is available in the Queue-Out (S_7^{Job}). In the queue-out, the job searches for the next machine (S_2^{Job}) and available MHSs (S_3^{Job}) to call. The job is carried to its next destination by the assigned MHS (S_4^{Job}), and the next process task phase is initiated unless its destination is the factory exit.

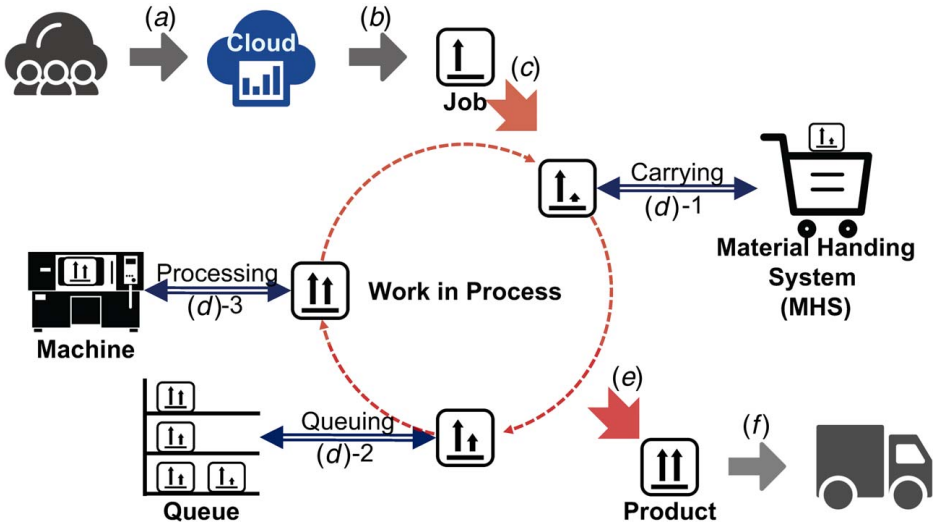

Fig. 3 Illustration of manufacturing process flows

The state transition of Job agents is facilitated by Job actions (A^{Job}), which are outlined in Table 2. These actions allow for modeling communication and interactions between jobs and other manufacturing objects, such as MHS, machines, and queues. As the manufacturing process flow is workpiece-centric, the behaviors of jobs are closely linked to these other objects. The Job actions can trigger the actions of other objects and vice versa.

The Job agent also models uncertainties of workpieces due to other manufacturing objects. The effects of MHS and machine breakdowns are expressed as actions A_8^{Job} and A_9^{Job}, respectively. Additionally, the impact on jobs due to lack of queue-out space is reflected as S_7^{Job}. The representation of uncertainty by states and actions allows the proposed model to effectively represent complex relationships between manufacturing objects.

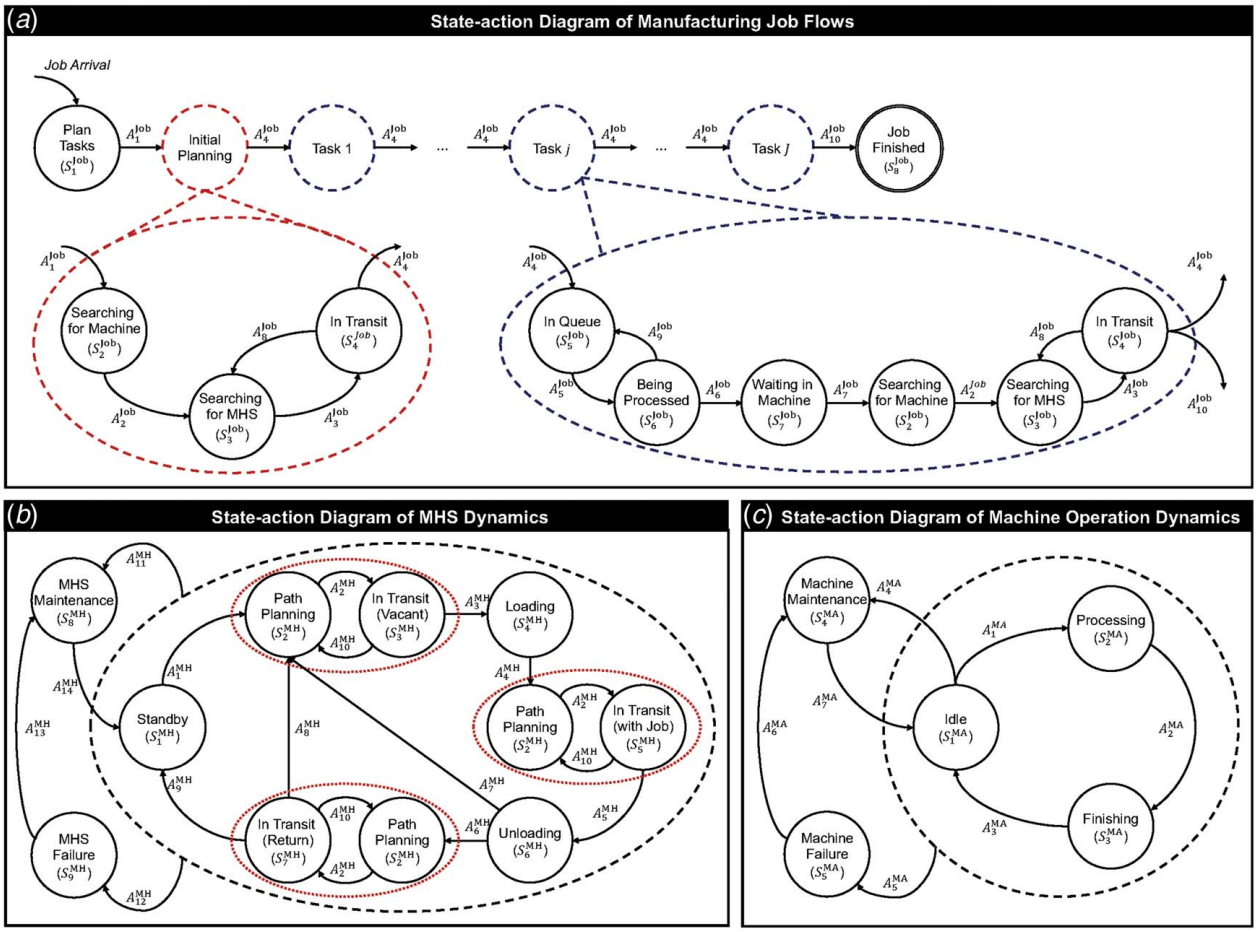

Fig. 4 State-action diagrams of (*a*) manufacturing job flows, (*b*) MHS dynamics, and (*c*) machine operating dynamics

Table 2 List of actions for the Job agent

Action	Description	Interaction
A_1^{Job}	Searching and selecting an available machine	A_3^{Q}
A_2^{Job}	Searching and selecting an available MHS	$A_1^{\text{MH}}, A_7^{\text{MH}}, A_8^{\text{MH}}$
A_3^{Job}	Being loaded to MHS	$A_4^{\text{MH}}, A_4^{\text{Q}}$
A_4^{Job}	Arriving at the next machine, being unloaded from MHS and entering Queue-In	$A_6^{\text{MH}}, A_7^{\text{MH}}, A_1^{\text{Q}}$
A_5^{Job}	A process of the job beginning	$A_1^{\text{MA}}, A_2^{\text{Q}}$
A_6^{Job}	A process of the job ending	A_2^{MA}
A_7^{Job}	Entering Queue-Out, and searching and selecting an available machine	$A_3^{\text{MA}}, A_3^{\text{Q}}$
A_8^{Job}	Being unloaded due to the failure of MHS	$A_{11}^{\text{MH}}, A_{12}^{\text{MH}}$
A_9^{Job}	Being reallocated to another machine due to the failure of the currently assigned machine	$A_5^{\text{MA}}, A_1^{\text{Q}}$
A_{10}^{Job}	Arriving at the warehouse, being unloaded from MHS, and exiting the jobshop	$A_6^{\text{MH}}, A_7^{\text{MH}}$

3.1.2 Material Handling Agent. An MHA (i.e., denoted by the symbol *MH*) is a digital representation of an MHS, such as an AGV, robot, and gantry system, that handles and delivers jobs. All MHAs are generated at the initial stage of the factory model. The MHA can be formally defined in Eq. (3)

$$MH = \left(S^{\text{MH}}, A^{\text{MH}}, I^{\text{MH}}\right) \tag{3}$$

where S^{MH} is a set of states for the MHA, A^{MH} is an MHA action set, and I^{MH} is a tuple of MHA inputs, including operating rules of the MHS and basic information about the MHS, such as its ID, type, failure rate, and information of the root node which is the station where the MHS stays for charging and maintenance. The operating rules encompass path-searching rules, maintenance policies, and distributions of maintenance and repair times for the MHS.

The states of the MHA (S^{MH}) are illustrated in Fig. 4(*b*). The initial state of the MHA is "standby" (S_1^{MH}), which represents the MHS staying on the root node. The MHS searches and plans a route before moving (S_2^{MH}). When the MHS is traveling, it can be in one of the three "in-transit" states. The MHS is either moving to pick up a job (S_3^{MH}), carrying a job (S_5^{MH}), or returning to the root node for charging and maintenance (S_7^{MH}). Loading and unloading operations are expressed by S_4^{MH} and S_6^{MH}, respectively, both with time delay functions. Maintenance and failure of the MHS are represented by states S_8^{MH} and S_9^{MH}, respectively. The MHS can break down under any condition except when it is under maintenance as displayed by an arrow of A_{12}^{MH} in Fig. 4(*b*). The impact on MHS's travel due to system disruptions, such as obstacles in the jobshop, is modeled as A_{12}^{MH}.

Table 3 List of actions for the MHA

Action	Description	Interaction
A_1^{MH}	Receiving call from a job	A_2^{Job}
A_2^{MH}	Departing travel	—
A_3^{MH}	Arriving and beginning loading a job	—
A_4^{MH}	Finishing loading a job	A_3^{Job}
A_5^{MH}	Arriving and beginning unloading a job	—
A_6^{MH}	Finishing unloading a job (no jobs waiting)	A_4^{Job}
A_7^{MH}	Finishing unloading a job (other jobs waiting)	$A_2^{\text{Job}}, A_4^{\text{Job}}$
A_8^{MH}	Receiving call from a job during travel	A_2^{Job}
A_9^{MH}	Arriving at the root node	—
A_{10}^{MH}	Travel disrupted by system	—
A_{11}^{MH}	Beginning the preventive maintenance of the MHS	A_8^{Job}
A_{12}^{MH}	The MHS breaking down	A_8^{Job}
A_{13}^{MH}	Beginning the reactive maintenance of the MHS	—
A_{14}^{MH}	The MHS being repaired	—

Table 4 List of actions for the Machine agent

Action	Description	Interaction
A_1^{MA}	Beginning a process	$A_5^{\text{Job}}, A_1^{\text{Q}}$
A_2^{MA}	Ending a process	A_6^{Job}
A_3^{MA}	Releasing the finished job	$A_7^{\text{Job}}, A_4^{\text{Q}}$
A_4^{MA}	Beginning preventive maintenance	—
A_5^{MA}	The machine breaking down	A_9^{Job}
A_6^{MA}	Beginning reactive maintenance	—
A_7^{MA}	The machine being repaired	—

Table 3 outlines actions of the MHA (A^{MH}) provoking the MHA's state transitions. Interactions and communications between MHSs and jobs are represented as actions. These actions are triggered by messages from Job agents and vice versa. For example, A_1^{MH}, A_7^{MH}, and A_8^{MH} are triggered by requests to pick up jobs (A_2^{Job}). A_3^{MH}, A_4^{MH}, A_5^{MH}, A_6^{MH}, and A_7^{MH} induce actions of jobs by loading and unloading them. In the path planning phase during A_2^{MH}, the MHS selects a traveling route based on the next destination information provided by the job. Overall, the MHA operates based on a communication protocol between MHSs and jobs, with actions triggered by these agents to execute the necessary tasks.

3.1.3 Machine Agent. A Machine agent (*MA*) represents machines in factories that process jobs. When initiating the model, all Machine agents are generated and placed. The Machine agent is defined as Eq. (4)

$$MA = \left(S^{\text{MA}}, A^{\text{MA}}, I^{\text{MA}}, Q\right) \tag{4}$$

where S^{MA} is a set of Machine states, A^{MA} is a Machine action set, and I^{MA} is a tuple of Machine inputs that include the operating rules of the machine and basic information about the machine, such as Machine ID, work type, failure rate, and location. The operating rules consist of maintenance policies and distributions of maintenance and repair times for the machine. Q is a Queue agent representing a queue associated with the machine, as described in the Queue Agent section.

The states of the Machine agent (S^{MA}) are depicted in Fig. 4(*c*). Each Machine agent starts in the "Idle" state (S_1^{MA}) when it is generated in the DT model. As a job arrives at the queue, the machine's status transitions to the "Processing" state (S_2^{MA}). After the process is complete, the machine is in the "Finishing" state (S_3^{MA}). The length of time stuck in this state is influenced by the capacity of the associated queue. For example, if the Queue-Out is filled, the finished job becomes blocked and stays in the machine. States S_4^{MA} and S_5^{MA} correspond to situations where the machine is under maintenance or repair (due to a failure), respectively.

The Machine agent's states change in response to actions (A^{MA}), as outlined in Table 4. Machine actions can trigger actions of jobs and queues, and vice versa. For example, actions A_1^{MA}, A_2^{MA}, and A_3^{MA} directly initiate job actions, and these actions also impact queues. The state of the queue can influence actions A_3^{MA}. The maintenance work is captured by action A_4^{MA}. The machine can break down under any state (A_5^{MA}). In particular, if the machine fails while processing a job in the S_2^{MA} state, the current process is immediately paused, and the job is sent back to the Queue-In to wait for the machine to recover. This action triggers A_9^{Job}. In summary, the Machine agent operates based on interactions among machines, jobs, and queues, with various actions triggered.

3.1.4 Queue Agent. A Queue agent (Q) represents queues associated with machines, which are categorized into two types: Queue-In and Queue-Out. Jobs waiting to be processed by the machine are placed in the Queue-In (Q^{In}). Upon arrival, a job enters the corresponding Queue-In and waits until the machine is available. The top-priority job determined by the queuing rule of the Queue-In is then processed by the machine. After the process

completes, the job enters the Queue-Out (Q^{Out}) and waits for an MHS to ship it to the next destination.

The Queue agent can be defined as Eq. (5)

$$Q = \left(S^{\mathrm{Q}}, E^{\mathrm{Q}}, I^{\mathrm{Q}}\right) \tag{5}$$

where S^{Q} is a set of Queue states, A^{Q} is a Queue action set, and I^{Q} is a tuple of Queue inputs, including capacities of Q^{In} and Q^{Out} ($N^{\mathrm{Q^{In}}}$ and $N^{\mathrm{Q^{Out}}}$), and queuing rules for both types of queues.

The state of the Queue agent (S^{Q}) is a tuple of $n^{\mathrm{Q^{In}}}$ and $n^{\mathrm{Q^{Out}}}$, indicating the number of jobs in Queue-In and Queue-Out, respectively, where $n^{\mathrm{Q^{In}}} \in \left[0, N^{\mathrm{Q^{In}}}\right]$, and $n^{\mathrm{Q^{Out}}} \in \left[0, N^{\mathrm{Q^{Out}}}\right]$. The initial state of the Queue agent is (0, 0), and state transitions occur when actions are taken. The Queue agent interacts with its associated Machine agent and Job agents, with these interactions modeled as actions of a Queue agent (A^{Q}), as defined in Table 5. The actions of the Queue agent account for uncertainty in the queues. If $n^{\mathrm{Q^{In}}} = N^{\mathrm{Q^{In}}}$, the corresponding Machine agent will not accept a newly arrived job. Similarly, if $n^{\mathrm{Q^{Out}}} = N^{\mathrm{Q^{Out}}}$, the Machine agent will not process the new job. The Queue agent effectively represents the operations of queues and performs based on communication and interactions among queues, associated machines, and jobs, with actions to execute the necessary tasks.

3.2 Statistical Metamodeling for Sequential Design of Computer Experiments in Digital Twin. The DT model represents a complex network of interactive manufacturing processes and involves a large number of nonlinear interactions and functions that operate in a high-dimensional space of experimental factors. This complexity leads to non-convex functions and nonlinear interactions that pose significant challenges for conventional optimization tasks. Further, solving optimization problems with this complex model requires heavy computational resources due to a large number of simulation experiments necessary [31]. Simulation experiments with the DT model generate outcome responses with respect to input factors. However, the high computational overhead is required to collect large amounts of data to construct a response surface corresponding to input factors. Hence, there is an urgent need to develop computationally efficient design methods for computer experiments to reduce the number of required simulation runs required for response surface estimation.

Algorithm 1 Sequential design of computer experiments

Input: Dataset D with factors $\mathbf{X}$ and response $\mathbf{y}$

1: Construct the initial statistical metamodel ($\tilde{f}(\mathbf{X})$)
2: **while** $\mathbf{x}_N \notin \mathbf{X}$
3: **for** each $i = 1, \ldots, |\mathbf{y}|$ do
4: Extract $\mathbf{x}_i$ and y_i as a validation dataset
5: Form a training set D_i by excluding the validation set from D (D_i consisting of $\mathbf{X} \setminus \mathbf{x}_i$ and $\mathbf{y} \setminus y_i$)
6: Compute the discrepancy $z_i = |y_i - \tilde{f}(\mathbf{x}_i)|$
7: **end for**
8: Construct a discrepancy prediction model ($\tilde{g}(\mathbf{X})$) to estimate the discrepancy surface
9: Compute PI based on updated $\mathbf{z}^*$
10: Evaluate discrepancies with the PI and extract the next design point $\mathbf{x}_N = \arg\max_x \mathrm{PI}(\mathbf{X})$
11: Collect response value of the next design (y_N) by the simulation experiment with respect to $\mathbf{x}_N$
12: Update the dataset: $(\mathbf{X}, \mathbf{y}) = \left(\begin{bmatrix}\mathbf{X}\\ \mathbf{x}_N\end{bmatrix}, \begin{bmatrix}\mathbf{y}\\ y_N\end{bmatrix}\right)$
13: Update the statistical metamodel ($\tilde{f}(\mathbf{X})$) with the updated dataset
14: **end while**

The proposed approach, shown in Fig. 5, consists of three core modules to develop statistical metamodels for addressing the challenges of the exhaustive search in the design space. The first module develops statistical metamodels that construct response surfaces according to input factors, based on a sample of experimental outcomes. This statistical metamodel is a surrogate of multiple simulations to predict the response surface. The second module constructs the discrepancy surface, which is based on the discrepancy of responses from the DT model and the statistical metamodel. The discrepancy surface provides information about the accuracy of the statistical metamodel that the proposed approach aims to maximize. In the third module, the probability of improvement (PI) is computed, which helps identify an additional design point required to improve the prediction power of the statistical metamodel [32]. Once the additional design point is identified, the corresponding simulation outcome is added to update the statistical metamodel. This process continues by selecting the next best design point and updating the statistical metamodel until the additional design point converges into the existing data set. By integrating these three modules into the proposed sequential design method of computer experiments, complex DT models can be calibrated more efficiently.

Table 5 List of actions for the Queue agent

Action	Description	Interaction
A_1^{Q}	Job entering Queue-In	$A_4^{\mathrm{Job}}, A_9^{\mathrm{Job}}, A_1^{\mathrm{MA}}$
A_2^{Q}	Job exiting Queue-In	A_5^{Job}
A_3^{Q}	Job entering Queue-Out	$A_1^{\mathrm{Job}}, A_7^{\mathrm{Job}}$
A_4^{Q}	Job exiting Queue-Out	A_3^{Job}

In the first module, a Gaussian process (GP) modeling approach is adopted to develop the statistical metamodels for predicting response surfaces. In statistical metamodeling for the design of computer experiments, the response is defined as simulation outcomes of the DT model ($\mathbf{y}$). The GP is particularly useful due to the provision of both mean and variance estimations of the response, enabling statistical analysis of the predicted response surface. Furthermore, the GP model is robust to handle the nonlinear and non-convex characteristics of the DT model, allowing for the uncertainty analysis of responses. Specifically, the statistical metamodel ($\tilde{f}(\ldots)$) acts as a surrogate of multiple simulations of the DT model ($\mathbf{y} = f(\mathbf{X})$), where $\mathbf{X}$ is a matrix of control factors and $\mathbf{y}$ is a vector of responses [30].

The second module involves constructing discrepancy surfaces based on the collected simulation outcomes. This module requires a discrepancy prediction model ($\mathbf{z} = \tilde{g}(\mathbf{X})$), where each element of the response is defined as $z_i = \left|y_i - \tilde{f}(\mathbf{x}_i)\right|$ for $i = 1, \ldots, |\mathbf{y}|$. The proposed design approach forms the discrepancy prediction model using a leave-one-out method to avoid overfitting and overcome the constraints of a small dataset. This technique trains the model with all data except for one observation and uses the excluded observation to evaluate the model's prediction accuracy. This process repeats for each observation, resulting in a discrepancy model that generalizes well to new data.

The measure of PI helps identify an additional design point to enhance the prediction power of the statistical metamodel. The objective is to minimize the discrepancy between the simulation results and the predicted responses by the statistical metamodel. The PI module calculates the PI value for each point on the discrepancy surface using the formula defined as follows:

$$\mathrm{PI} = \max\left\{\Phi\left(\frac{z_i - T_{\max}}{s(\mathbf{z})}\right)\right\} \tag{6}$$

where $\Phi(\cdot)$ is the cumulative distribution function of a normal distribution. $T_{\max}$ is a target value defined as $T_{\max} = z_{\max} + 0.25|z_{\max}|$, where $z_{\max}$ is the maximum discrepancy of the training dataset. The point with the highest PI indicates the largest difference between the predicted response by the statistical metamodel ($\tilde{\mathbf{y}} = \tilde{f}(\mathbf{X})$) and the

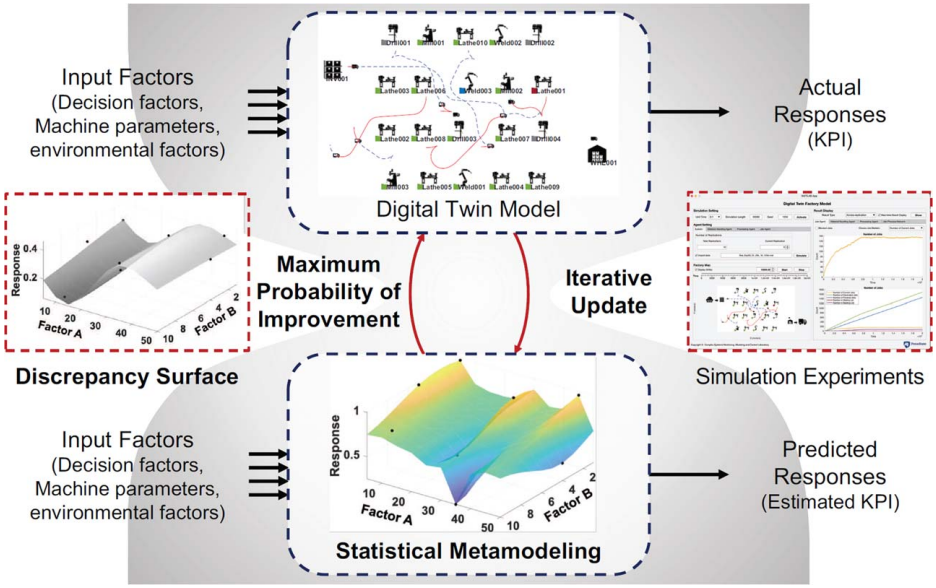

Fig. 5 Flowchart of the sequential design of experiments

actual response ($\mathbf{y}$), so it has the highest likelihood to collect experimental results at this point for subsequent iterations.

Algorithm 1 outlines the detailed procedures of the proposed sequential design method. The algorithm starts with constructing the initial statistical metamodel ($\hat{f}(\mathbf{X})$) based on the initial dataset D. Then, the dataset is split to calculate the discrepancies at each point. Next, the computed discrepancies with respect to the input space are used to form the discrepancy surface through a discrepancy prediction model ($\tilde{g}(\mathbf{X})$). After computing the PI on the discrepancy surface, the additional design point $\mathbf{x}_N$ with the highest PI value is determined. Then, the simulation experiment on $\mathbf{x}_N$ is conducted, and the outcomes are updated to the dataset $\mathbf{X}$. The statistical metamodel ($\hat{f}(\mathbf{X})$) is updated based on $\mathbf{X}$, and these procedures are iterative until the additional design point converges.

3.3 Graph Models. The proposed methodology includes two types of graph models to extract features of the manufacturing process flow:

(1) the job flow graph: $G^{\text{Job}} = (V, E^{\text{Job}}, W^{\text{Job}})$
(2) the AGV traveling graph: $G^{\text{MH}} = (V, E^{\text{MH}}, W^{\text{MH}})$

In both graph models, the set V denotes a set of nodes representing machines and two special nodes such as inventory and warehouse. These graph models allow for a comprehensive understanding of the layout and interconnectivity of the jobshop.

In the case of the job flow graph G^{Job}, the inventory serves as the source node, and the warehouse functions as the sink node, representing the entry and exit points for jobs, respectively. The set $E^{\text{Job}} \subseteq V \times V$ represents the directed edges that depict the flow of jobs between the machines. Additionally, a weight set W^{Job} denotes the frequency of job flow between nodes. The value of w_{ij}^{Job} is incremented when a job arrives at a particular machine (A_4^{Job} or A_{10}^{Job}). Analysis of the job flow graph provides insight into the sequence of operations and the dependencies between different tasks. This understanding helps identify critical paths, potential delays, and opportunities for improving the overall efficiency of the manufacturing process flow.

On the other hand, the AGV traveling graph G^{MH} does not specify any source or sink nodes. The set $E^{\text{MH}} \subseteq V \times V$ represents the directed edges that illustrate the flow of AGVs between machines. Similar to the job flow graph, a weight set W^{MH} shows the frequency of AGV travel between machines. The value of w_{ij}^{MH} is incremented when an AGV arrives at a specific machine (A_3^{MH}, A_5^{MH}, or A_9^{MH}). By examining the AGV traveling graph, we can gain insights into the complexity of the jobshop environment and the various paths that AGVs can take to transport jobs between machines. This graph model allows an understanding of the physical layout of the jobshop and the interconnections between machines and AGVs.

Overall, the graph models enhance comprehension of the manufacturing jobshop, facilitating the analysis and optimization of production flow, the study of AGV routing, and the acquisition of insights into the dynamics of the manufacturing process. These models provide visual representations that support the interpretation and analysis of the jobshop environment, aiding decision-making and optimization efforts.

4 Experimental Design

The proposed simulation-enabled DT approach is evaluated and validated through a case study of deploying AGVs under uncertain conditions involving MHSs, machines, queues, and jobs. The analysis of the simulation focuses on three key performance metrics to assess the impact of the number of AGVs on the manufacturing process flow: the expected utilization of AGVs (y_1), which is pertinent to production throughput; the average number of jobs waiting for AGVs (y_2), indicating the level of process delay due to AGVs; and the mean number of jobs in Queue-In (y_3), which shows the level of process delay caused by the machine's processing capacity.

With these three metrics as a basis, the objective function for this case study aims to minimize the expected cost ($C(x)$) associated with the number of AGVs (x). The objective function is defined as follows:

$$C(x) = C_{\text{AGV}}x + C_{\text{Idle}}\big(1 - y_1(x)\big) + C_{\text{Backlog}}y_2(x) + C_{\text{Queuing}}y_3(x) \tag{7}$$

where C_{AGV}, C_{Idle}, C_{Backlog}, and C_{Queuing} denote the setup cost of each AGV, the cost incurred due to AGVs being idle, the cost resulting from job backlog caused by AGV delays, and the cost associated with queuing at machines, respectively. The functions $y_1(x)$, $y_2(x)$, and $y_3(x)$ represent the values of each performance metric corresponding to the number of AGVs.

The case study is subject to constraints imposed by the model and configuration of the digital factory. We model the manufacturing process through the proposed DT model and conduct simulation experiments within the jobshop environment reference of the FAME laboratory at the Pennsylvania State University [33]. The

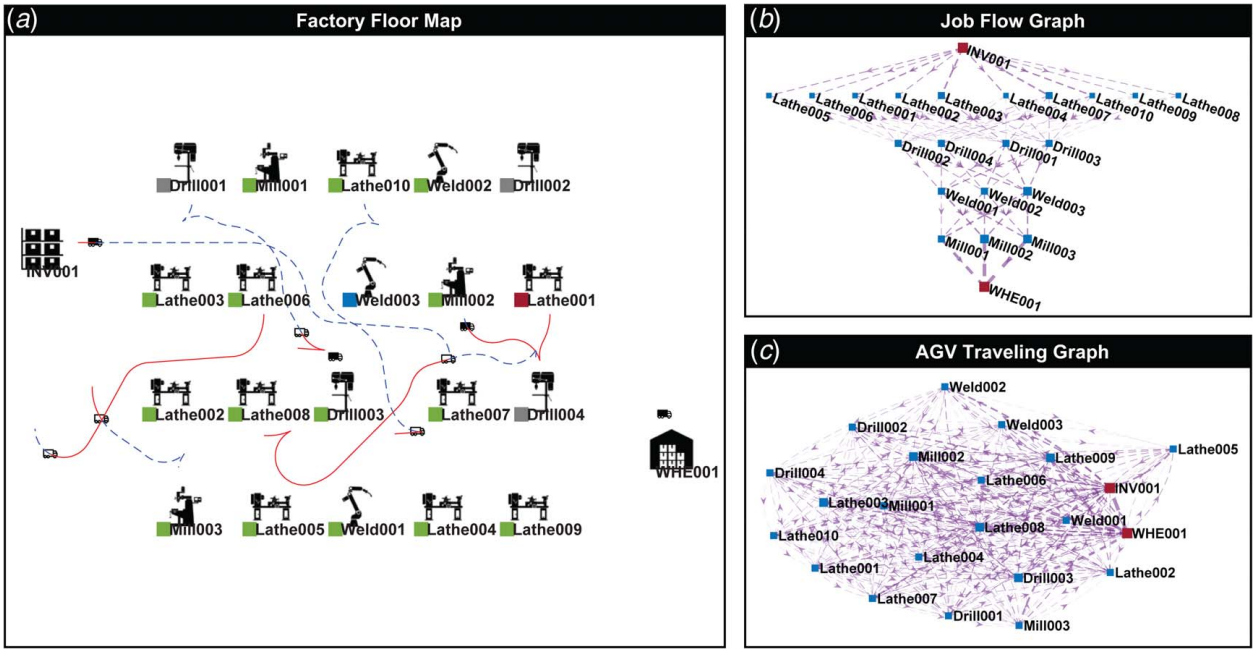

Fig. 6 Illustrations of (*a*) DT factory floor map, (*b*) job flow graph of the "CPQ" type, and (*c*) AGV traveling graph between machines

jobshop consists of 20 machines, including 10 lathe machines, 4 drilling machines, 3 welding machines, and 3 milling machines. To provide a visual representation, Fig. 6(*a*)[2] illustrates an example of the manufacturing jobshop, which incorporates a total of nine AGVs. This simulation setup provides an environment to test the proposed methodology and evaluate its performance under various scenarios.

The manufacturing process in this case study involves AGVs that load raw materials from the inventory upon the arrival of work orders. Once all processes are completed, AGVs transport finished products to the warehouse for shipment. AGVs determine their paths through a Hybrid A* algorithm. The factory model reflects the uncertainty of machine or AGV failure occurrence by employing stochastic distributions. Preventive maintenance policies with regular maintenance periods are implemented for both machines and AGVs. The capacity of the queues is set to ten and five for each Queue-In and Queue-Out, respectively, resulting in uncertainty of blocking the queue. The job types processed in the jobshop are summarized in Table 6. Six types of jobs randomly arrive at the factory in proportion. To reflect the uncertainty of market demand and facilitate optimal decision-making, two key control factors have been identified: the job arrival rate and the number of deployed AGVs. In each case, the number of deployed AGVs is varied from 1 to 10, while the job arrival rate spans from 5 s to 50 s. These control factors provide a comprehensive range of scenarios to assess the system's performance and determine the most effective configuration.

To validate the effectiveness of the proposed sequential design method, we evaluate the accuracy of statistical metamodels and the computational efficiency of experimental design methods. The accuracy of statistical metamodels is assessed through a performance metric: the coefficient of determination (R^2), which measures the proportion of variance in the predicted response. R^2 measures the validity of experimental design methods with respect to three responses. The computational efficiency is evaluated based on the number of additional experimental runs required to construct an effective response surface.

[2]See YouTube demo in https://youtu.be/52AKA6hyP7E

Comparative experiments are conducted to compare the effectiveness of the proposed sequential design method against two other design approaches, namely the MaxPro design method and a random selection approach. The MaxPro design method selects additional experimental points based on predetermined criteria from existing candidate points [34], while the random selection approach randomly chooses additional points. The comparative experiments aim to benchmark the proposed design method against these comparison approaches with a focus on evaluating the accuracy of statistical metamodels and the computational efficiency of the design methods. For accuracy evaluation, each case is constrained by the same simulation budget, which corresponds to the same number of additional experimental points as determined in the proposed sequential design method. In terms of efficiency analysis, the comparative design methods are allowed to collect more experimental points until their R^2 value approaches that of the proposed design result. This number of experimental points provides a performance measure of computational efficiency.

5 Experimental Results

5.1 Digital Twin and Network Models. The proposed multi-agent simulation model of DT enables real-time monitoring of the manufacturing process. A factory map, shown in Fig. 6(*a*), illustrates the status of machines and MHSs in the jobshop. The machine icons on the map are accompanied by small boxes indicating the machine's state. The "Idle" state (S_1^{MA}) is shown as a gray

Table 6 List of randomly generated job types, their task sequences, and proportions

Job type	Task sequence	Proportion (%)
"CPK"	Lathe – Weld – Mill – Drill	6.25
"CPQ"	Lathe – Drill – Weld – Mill	6.25
"CPB"	Lathe – Drill – Weld – Drill	12.5
"CPN"	Lathe – Weld – Mill – Drill – Lathe	12.5
"CPP"	Lathe – Mill	50
"CPR"	Lathe – Mill – Drill	12.5

box, while a green-colored box represents the "Processing" (S_2^{MA}) state. During maintenance (S_3^{MA}), the box color changes to blue, and after a breakdown, it turns red until the machine gets repaired (S_4^{MA}). The factory map also shows the paths of AGVs at a particular time with a red line and a blue-dashed line. Once an AGV sets its destination, the planned route is displayed as a blue-dashed line. After the AGV moves, its travel history path changes to a red line. The AGV icon has two different colors: black when carrying a job, and white when moving without any load.

With the proposed methodology, graph models of jobshop networks can be constructed to gain insights into the bottlenecks and strengths of relationships of job flows between machines. Figure 6(*b*) depicts a job flow graph of "CPQ" jobs, where the source and sink of the network coincide with the inventory "INV001" and the warehouse "WHE001," respectively. The edges in the graph denote the direction of job flows between corresponding vertices, and the edge width reflects the number of jobs flowing. Job process flows have a direction between machines, making the network graph directed. In this example, processing the "CPQ" type requires four tasks, as described in Table 6. Therefore, the warehouse is connected to milling machines, while the inventory is linked to lathe machines. Combining the job flow network with the factory map enables the analysis to reveal that the milling machines are situated far from the warehouse, leading to longer travel times for AGVs and higher utilization.

Analyzing the factory network from another perspective of manufacturing objects such as AGVs, the AGV traveling network can track the travel history of AGVs within machine nodes. Unlike the job flow network, this network solely represents characteristics of facilities in the jobshop, reflecting machines as nodes and AGVs as the flow objects. Therefore, it can reveal characteristics of the factory beyond the direct effects of different job types. Figure 6(*c*) shows an AGV traveling network for "AGV003" from the case scenario. This network is directed with weighted edges where the widths represent the number of moves between vertices. The remarkable result is that the AGV travels from the warehouse to the inventory the most frequently, accounting for 2.935% of the number of moves among 345 edges. In the case study, the distance between the inventory and the warehouse is the longest. This result shows that the AGV travels from the warehouse to the inventory vacantly, causing inefficiency in the travel scheduling of AGVs with a high number of vacant travels.

5.2 Comparative Studies of Statistical Designs. The modeling of manufacturing systems presents inherent complexities, including high-dimensional design spaces and nonlinear, nonconvex response surfaces. These challenges make it difficult to accurately capture the system dynamics. Moreover, the computational time required for simulations tends to increase exponentially as the number of agents increases, further exacerbating the modeling process. To address these challenges, statistical metamodeling is employed as a surrogate for intricate simulation models.

The proposed sequential design method, in combination with statistical metamodeling, aims to optimize decision-making in manufacturing systems while mitigating the computational burden. A comparative experiment has been conducted to evaluate the effectiveness of the proposed sequential design approach. Specifically, its accuracy and computational efficiency are assessed in comparison to the MaxPro design approach and random selection. Three key responses, namely the expected utilization of AGVs (y_1), the average number of jobs waiting for AGVs (y_2), and the mean number of jobs in Queue-In (y_3), serve as performance metrics for evaluation.

Within this case study, the proposed sequential design of computer experiments commences with an initial dataset comprising eight simulation results, which serves as the foundation for constructing the statistical metamodel. To form the response surface for y_1, an additional 13 simulation experiments are conducted, while y_2 necessitates 9 additional runs, and y_3 requires 1 more point. The construction of the statistical metamodel incorporating these supplementary experimental results demonstrates the high effectiveness of the proposed method, as evidenced by R^2 values of 0.981 for y_1, 0.840 for y_2, and 0.817 for y_3. These results present that the statistical metamodel implemented in the proposed design method can effectively serve as a substitute for complex simulation models. Also, this study's findings underscore the efficacy of the proposed design method in reducing the number of required simulation runs from 100 to 21, 17, and 9 for the cases of y_1, y_2, and y_3, respectively.

Figure 7 illustrates the outcomes of a comparative analysis conducted to evaluate the effectiveness of different design approaches, namely the sequential design, MaxPro design, and random selection, based on the R^2 metric. Three case studies were conducted to assess the responses y_1, y_2, and y_3, with a fixed simulation budget denoting the number of additional simulation runs required by the proposed sequential design. For y_1, the budget is set at 13 additional simulation runs, while y_2 requires 9 supplementary runs, and y_3 necessitates 1 additional run.

The proposed design approach exhibits superior performance in terms of R^2 compared to both the MaxPro and random design approaches. Specifically, the R^2 value achieved by the proposed method for y_1 is 0.981, whereas the values for the MaxPro and random design approaches are 0.919 and 0.750, respectively. Similarly, for y_2, the R^2 value obtained with the proposed method is 0.840, while the values for the MaxPro and random design approaches are 0.753 and 0.585, respectively. Regarding y_3, the proposed method yields an R^2 value of 0.817, whereas the values for the MaxPro and random design approaches are 0.617 and 0.540, respectively.

These results demonstrate a robust correlation between the expected utilization (y_1) and both the number of deployed AGVs and the job arrival rate. However, it is significant to note that the average numbers of jobs waiting for AGVs (y_2) and waiting in a queue (y_3) cannot be fully explained by these two control factors alone, leading to a lower prediction accuracy compared to y_1. The reason for this disparity lies in the fact that the backlog of jobs is influenced by additional factors beyond the number of AGVs, such as the capacity limitations of the jobshop and individual machines. A more detailed analysis of the relationship between the control factors and the responses will be discussed in the forthcoming section.

The computational efficiency of experimental design approaches is evaluated based on the number of additional simulation runs required to achieve a comparable level of accuracy to the proposed sequential design. For y_1, additional simulation experiments are conducted for comparative methods until the R^2 value reaches 0.97, which corresponds to 99% of the proposed sequential

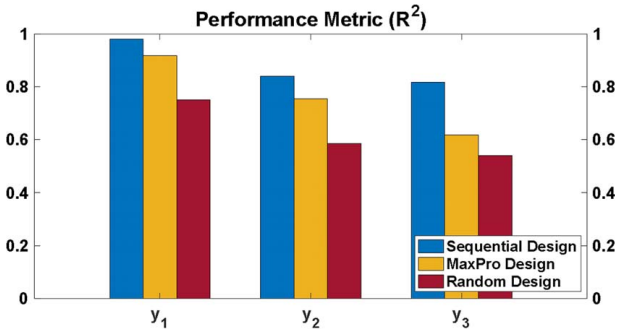

Fig. 7 Performance comparison of statistical metamodeling measured by R^2, among sequential design, MaxPro design, and Random design for three responses (y_1: utilization of AGVs, y_2: number of jobs waiting for AGVs, and y_3: number of jobs in Queue-In) under the fixed simulation budget. The budget requires 13 additional simulation runs for y_1, 9 additional simulation runs for y_2, and 1 additional simulation run for y_3.

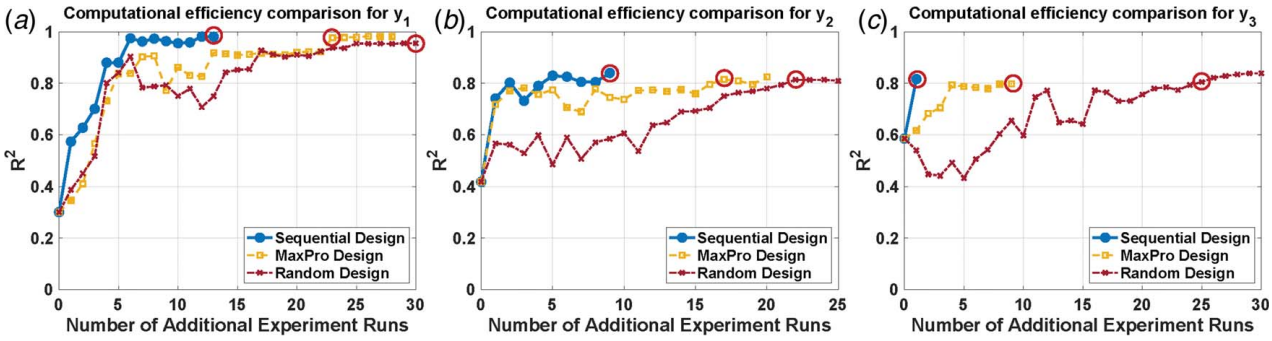

Fig. 8 Computational efficiency analysis of sequential design, MaxPro design, and Random design for three responses (y_1: utilization of AGVs, y_2: number of jobs waiting for AGVs, and y_3: number of jobs in Queue-In) to achieve comparable accuracy levels as the sequential design

design's R^2 value of 0.981. The results of the comparative analysis for y_1 are depicted in Fig. 8(*a*). The proposed sequential design requires 13 additional points, resulting in a dataset of a total of 21 simulation runs to learn the statistical metamodel. In contrast, both comparative design approaches need more additional runs to achieve an R^2 value of more than 0.98. In particular, the MaxPro design attains the R^2 value of more than 0.97 with 23 additional points, while the random design fails to achieve comparable accuracy, even with 30 additional points.

The cases for y_2 involve collecting additional simulation points until the R^2 value surpasses 0.80, which is 95% of the R^2 value of the sequential design, 0.840. As shown in Fig. 8(*b*), the sequential design necessitates 9 additional points, which leads to a dataset with a total of 17 simulation runs to train the statistical metamodel. However, neither comparison design method achieves comparable accuracy with only 9 additional points. The MaxPro design requires at least 17 additional points to attain the R^2 value of more than 0.80, while the random design achieves it with 22 additional points.

In the case of y_3, additional simulation results have been collected until the R^2 value exceeds 0.80, which corresponds to 99% of the R^2 value obtained with the sequential design, 0.817. As illustrated in Fig. 8(*c*), only one additional point is necessary to effectively predict the response surface, resulting in a dataset comprising a total of 9 simulation results to train the statistical metamodel. On the other hand, comparative design approaches require a greater number of supplementary simulation results to achieve an R^2 value of more than 0.80. Specifically, the MaxPro design necessitates 9 additional points to reach the desired R^2 value, while the random design requires 25 additional points.

The sequential design method significantly reduced the number of experiments required to train the statistical metamodel efficiently. Furthermore, the proposed design approach implemented an iterative procedure, continuing until the next additional data point no longer contributed to improving the accuracy of the response surface. This methodological approach offers clarity on the number of simulation experiments necessary to predict response surfaces using statistical metamodels and allows for a reduction in computational overhead.

5.3 Case Study: Optimal Number of Automated Guided Vehicles. The experimental studies conducted in the 20-machine jobshop yielded three types of output responses: the utilization of AGVs (y_1), the number of jobs waiting for AGVs (y_2), and the number of jobs in Queue-In (y_3). The statistical metamodeling approach was employed to construct response surfaces with a reduced number of simulation runs. Figure 9(*a*) illustrates the quadratic relationship between y_1 and control factors. The utilization of AGVs is observed to increase with higher job arrival rates, indicating a positive correlation.

In contrast, the relationship between y_2 and the control factors is more complex, as demonstrated in Fig. 9(*b*). The number of jobs waiting for AGVs shows a negative correlation with the number of AGVs, while the relationship with the job arrival rate exhibits varying trends. In scenarios with fewer than three AGVs, over 100 jobs tend to wait for AGVs. When there are four AGVs in the system, the value of y_2 remains relatively constant regardless of the job arrival rate. However, with more than five AGVs, the response shows a quadratic relationship with the control factors. In cases where the number of AGVs exceeds five, the number of AGVs no longer acts as a bottleneck in the process flow. Instead, the process flow is hindered by the capacity limitations of machines and queues.

As shown in Fig. 9(*c*), scenarios involving high job arrival rates and a large number of AGVs result in an accumulation of jobs in the Queue-In, indicating that the process flow is impacted by the number of jobs present in the Queue-In. It is worth noting that

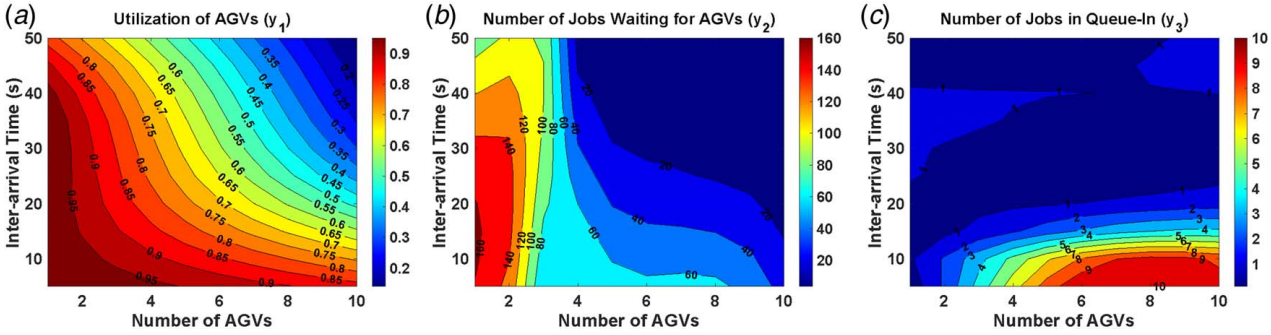

Fig. 9 Response surfaces of three responses (y_1: utilization of AGVs, y_2: number of jobs waiting for AGVs, and y_3: number of jobs in Queue-In), which are predicted by the proposed statistical metamodeling for the sequential design of computer experiments

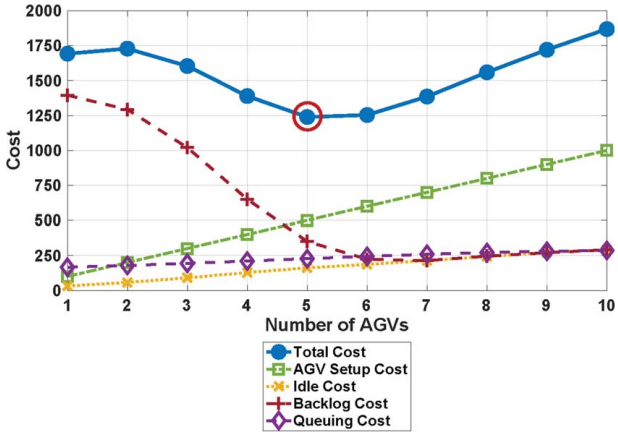

Fig. 10 Cost-effectiveness analysis with respect to the number of AGVs under job uncertainty

the capacity of each Queue-In is set to 10, further highlighting the influence of job accumulation on the process flow in such scenarios.

To identify the optimal number of AGVs in the case study defined in Eq. (7), the values of cost coefficients are set to $C_{\text{AGV}} = 100$, $C_{\text{Idle}} = 500$, $C_{\text{Backlog}} = 10$, and $C_{\text{Queuing}} = 10$. Also, response functions y_1, y_2, and y_3 are approximated with respect to the number of AGVs based on the response surfaces in Fig. 9. Figure 10 illustrates the result of cost analysis in the case study.

The analysis result reveals that as the number of AGVs increases, there are corresponding increases in AGV setup costs and idle costs, while the backlog costs decrease. With a greater number of AGVs deployed in the jobshop, the idle costs increase due to a quadratic drop in AGV utilization. The backlog cost caused by AGVs exhibits a decreasing trend when the number of AGVs increases within the range of 1–6. However, beyond six AGVs, the backlog cost starts to increase. This finding suggests that the delay in job processing is not solely attributable to AGVs as a bottleneck when the number of AGVs exceeds six. Other underlying factors, such as the low processing speed of machines, limited machine availability, or insufficient queue capacity, may contribute to the backlog of jobs.

As illustrated in Fig. 9(*c*), the Queue-In length tends to be longer when the market demand becomes higher, or there are more AGVs deployed in the jobshop. This result emphasizes that the processing and queuing capacities of machines have a significant impact on the process flow when there is high market demand. Based on the uncertainty of market demand and the given constraints of the model and configuration of the digital factory, 6 is determined as the optimal number of AGVs in this case study. With $x = 5$, the total cost related to the number of AGVs amounts to 1239.21, which is lower than the cost of 1253.99 when $x = 6$.

6 Conclusions

This paper presents a novel data-driven DT approach designed to simulate and optimize manufacturing process flows in small and medium-sized manufacturing jobshops. The proposed methodology offers attractive features that contribute to its effectiveness and applicability, including:

(1) *Multi-agent simulation model* that comprehensively captures the complexity and dynamics in manufacturing systems. The model represents manufacturing things as agents, capturing their characteristics, actions, and states. The interaction between these agents is facilitated through message transactions and action triggers, providing a detailed representation of the manufacturing systems.
(2) *Statistical metamodeling* for the sequential design of computer experiments that serves as a surrogate model for the complex simulation model. This approach enables efficient optimization by reducing the computational overhead associated with running extensive simulations. By constructing statistical metamodels, the proposed methodology offers a more computationally efficient alternative for decision-making and optimization in manufacturing jobshops.
(3) *Graph models* of job flow and AGV traveling that enhance real-time monitoring of jobshop's statuses, facilitating a deeper understanding of the dynamics of the manufacturing process. The job flow graph captures the sequence of operations and dependencies between tasks, while the AGV traveling graph illustrates the paths followed by AGVs as they transport jobs between machines.

The proposed methodology fetches an unprecedented opportunity for SMM to fully utilize the high technology of Industry 4.0. This paper also focuses on DT-enabled optimization of complex manufacturing jobshops that involve an interactive network of manufacturing things and further reduces the computation overhead via the sequential design of computer experiments. Experimental results showed that the proposed methodology brings intelligence to a manufacturing system with DT modeling and simulation optimization. Also, the proposed experimental design method efficiently reduced the required computation resources through the sequential design approach, while effectively predicting response surfaces by statistical metamodeling.

Our future work will investigate the integration of multi-agent models with artificial intelligence techniques for manufacturing planning, scheduling, and maintenance. By incorporating advanced AI methodologies, DT can achieve greater efficiency, effectiveness, and adaptability in managing complex manufacturing systems. This research direction allows for the exploration and development of intelligent decision-making mechanisms that leverage advanced sensing, big data, simulation modeling, and digital twinning. Consequently, DT models can be further refined and optimized to address the intricate challenges associated with planning, scheduling, and maintenance tasks within manufacturing environments. Providing a novel and flexible digital twin model to small and medium-sized manufacturing enterprises leads to competitive advantages in the global manufacturing industry.

Acknowledgment

The authors gratefully acknowledge the support from National Science Foundation under the grant IIS-2302834. Any opinions, findings, or conclusions found in this paper are those of the authors and do not necessarily reflect the views of the sponsor.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The authors attest that all data for this study are included in the paper.

Nomenclature

ABM = agent-based model
AGV = automated guided vehicle
DT = digital twin
GP = Gaussian process
IoT = Internet of Things
MAS = multi-agent system
MHA = material handling agent
MHS = material handling system
RFID = radio-frequency identification
SMM = small and medium-sized manufacturer
WIP = work-in-process

References

[1] Yang, H., Kumara, S., Bukkapatnam, S. T., and Tsung, F., 2019, "The Internet of Things for Smart Manufacturing: A Review," IISE Trans., **51**(11), pp. 1190–1216.

[2] Negri, E., Fumagalli, L., and Macchi, M., 2017, "A Review of the Roles of Digital Twin in CPS-Based Production Systems," Proc. Manuf., **11**, pp. 939–948.

[3] Cimino, C., Negri, E., and Fumagalli, L., 2019, "Review of Digital Twin Applications in Manufacturing," Comput. Ind., **113**.

[4] Kampe, N., 2019, "Technology-in-Industry Report," Automation Alley.

[5] Mittal, S., Khan, M. A., Romero, D., and Wuest, T., 2018, "A Critical Review of Smart Manufacturing & Industry 4.0 Maturity Models: Implications for Small and Medium-sized Enterprises (SMES)," J. Manuf. Syst., **49**, pp. 194–214.

[6] Leng, J., Wang, D., Shen, W., Li, X., Liu, Q., and Chen, X., 2021, "Digital Twins-Based Smart Manufacturing System Design in Industry 4.0: A Review," J. Manuf. Syst., **60**, pp. 119–137.

[7] Zhang, M., Tao, F., and Nee, A., 2021, "Digital Twin Enhanced Dynamic Job-Shop Scheduling," J. Manuf. Syst., **58**, pp. 146–156.

[8] Ye, Z., Si, S., Yang, H., Cai, Z., and Zhou, F., 2021, "Machine and Feedstock Interdependence Modeling for Manufacturing Networks Performance Analysis," IEEE Trans. Ind. Inform., **18**(8), pp. 5067–5076.

[9] Zhu, R., Aqlan, F., Zhao, R., and Yang, H., 2022, "Sensor-Based Modeling of Problem-Solving in Virtual Reality Manufacturing Systems," Expert Syst. Appl., **201**(117220).

[10] Mittal, U., Yang, H., Bukkapatnam, S. T., and Barajas, L. G., 2008, "Dynamics and Performance Modeling of Multi-stage Manufacturing Systems Using Nonlinear Stochastic Differential Equations," 2008 IEEE International Conference on Automation Science and Engineering, Arlington, VA, Aug. 23, IEEE, pp. 498–503.

[11] Yang, H., Bukkapatnam, S. T., and Barajas, L. G., 2013, "Continuous Flow Modelling of Multistage Assembly Line System Dynamics," Int. J. Comput. Integr. Manuf., **26**(5), pp. 401–411.

[12] Resman, M., Protner, J., Simic, M., and Herakovic, N., 2021, "A Five-Step Approach to Planning Data-Driven Digital Twins for Discrete Manufacturing Systems," Appl. Sci., **11**(8), p. 3639.

[13] Lidberg, S., Aslam, T., Pehrsson, L., and Ng, A. H., 2020, "Optimizing Real-World Factory Flows Using Aggregated Discrete Event Simulation Modelling," Flexi. Ser. Manuf. J., **32**(4), pp. 888–912.

[14] Dorri, A., Kanhere, S. S., and Jurdak, R., 2018, "Multi-agent Systems: A Survey," IEEE Access, **6**, pp. 28573–28593.

[15] Ryu, K., Son, Y., and Jung, M., 2003, "Modeling and Specifications of Dynamic Agents in Fractal Manufacturing Systems," Comput. Ind., **52**(2), pp. 161–182.

[16] Macal, C., and North, M., 2010, "Tutorial on Agent-Based Modelling and Simulation," J. Simul., **4**(3), pp. 151–162.

[17] Yao, X., and Lin, Y., 2016, "Emerging Manufacturing Paradigm Shifts for the Incoming Industrial Revolution," Int. J. Adv. Manuf. Technol., **85**(5), pp. 1665–1676.

[18] Kim, Y. G., Lee, S., Son, J., Bae, H., and Do Chung, B., 2020, "Multi-agent System and Reinforcement Learning Approach for Distributed Intelligence in a Flexible Smart Manufacturing System," J. Manuf. Syst., **57**, pp. 440–450.

[19] Park, K. T., Lee, J., Kim, H. -J., and Do Noh, S., 2020, "Digital Twin-Based Cyber Physical Production System Architectural Framework for Personalized Production," Int. J. Adv. Manuf. Technol., **106**(5), pp. 1787–1810.

[20] Chen, G., Wang, P., Feng, B., Li, Y., and Liu, D., 2020, "The Framework Design of Smart Factory in Discrete Manufacturing Industry Based on Cyber-Physical System," Int. J. Computer Integr. Manuf., **33**(1), pp. 79–101.

[21] Rolon, M., and Martinez, E., 2012, "Agent Learning in Autonomic Manufacturing Execution Systems for Enterprise Networking," Comput. Ind. Eng., **63**(4), pp. 901–925.

[22] Huang, C.-J., and Liao, L.-M., 2012, "A Multi-agent-based Negotiation Approach for Parallel Machine Scheduling With Multi-objectives in an Electro-Etching Process," Int. J. Prod. Res., **50**(20), pp. 5719–5733.

[23] Giordani, S., Lujak, M., and Martinelli, F., 2013, "A Distributed Multi-agent Production Planning and Scheduling Framework for Mobile Robots," Comput. Ind. Eng., **64**(1), pp. 19–30.

[24] Adediran, T. V., and Al-Bazi, A., 2019, "Agent-Based Modelling and Heuristic Approach for Solving Complex OEM Flow-Shop Productions Under Customer Disruptions," Computers Ind. Eng., **133**, pp. 29–41.

[25] Huang, Z., Kim, J., Sadri, A., Dowey, S., and Dargusch, M. S., 2019, "Industry 4.0: Development of a Multi-agent System for Dynamic Value Stream Mapping in Smes," J. Manuf. Syst., **52**, pp. 1–12.

[26] Yang, H., Kan, C., Krall, A., and Finke, D., 2020, "Network Modeling and Internet of Things for Smart and Connected Health Systems-A Case Study for Smart Heart Health Monitoring and Management," IISE Trans. Healthcare Syst. Eng., **10**(3), pp. 159–171.

[27] Kan, C., and Yang, H., 2017, "Dynamic Network Monitoring and Control of In Situ Image Profiles From Ultraprecision Machining and Biomanufacturing Processes," Q. Reliab. Eng. Int., **33**(8), pp. 2003–2022.

[28] Yang, H., and Liu, G., 2013, "Self-Organized Topology of Recurrence-Based Complex Networks," Chaos: Interdiscip. J. Nonlinear Sci., **23**(4), p. 043116.

[29] Liu, R., Xie, X., Yu, K., and Hu, Q., 2018, "A Survey on Simulation Optimization for the Manufacturing System Operation," Int. J. Modell. Simul., **38**(2), pp. 116–127.

[30] Du, D., Yang, H., Ednie, A. R., and Bennett, E. S., 2015, "Statistical Metamodeling and Sequential Design of Computer Experiments to Model Glyco-Altered Gating of Sodium Channels in Cardiac Myocytes," IEEE J. Biomed. Health Inform., **20**(5), pp. 1439–1452.

[31] Kan, C., Yang, H., and Kumara, S., 2018, "Parallel Computing and Network Analytics for Fast Industrial Internet-of-Things (IIOT) Machine Information Processing and Condition Monitoring," J. Manuf. Syst., **46**, pp. 282–293.

[32] Jones, D. R., 2001, "A Taxonomy of Global Optimization Methods Based on Response Surfaces," J. Global Optim., **21**(4), pp. 345–383.

[33] The Pennsylvania State University, 2023, "Factory for Advanced Manufacturing Education Lab., https://www.ime.psu.edu/research/facilities-and-labs/fame.aspx.

[34] Joseph, V. R., Gul, E., and Ba, S., 2015, "Maximum Projection Designs for Computer Experiments," Biometrika, **102**(2), pp. 371–380.