Journal of Manufacturing Systems 58 (2021) 210–230

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment, interfaces and intelligence

Kaishu Xia [a], Christopher Sacco [a], Max Kirkpatrick [b], Clint Saidy [a], Lam Nguyen [a], Anil Kircaliali [a], Ramy Harik [a,*]

[a] McNAIR Center for Aerospace Innovation and Research, Department of Mechanical Engineering, College of Engineering and Computing, University of South Carolina, 1000 Catawba Street, Columbia, SC, 29201, USA
[b] Siemens Product Lifecycle Management Software Inc., Charlotte, NC, 28277, USA

ARTICLE INFO

Keywords:
Smart manufacturing systems
Robotics
Artificial intelligence
Digital transformation
Virtual commissioning

ABSTRACT

Filling the gaps between virtual and physical systems will open new doors in Smart Manufacturing. This work proposes a data-driven approach to utilize digital transformation methods to automate smart manufacturing systems. This is fundamentally enabled by using a digital twin to represent manufacturing cells, simulate system behaviors, predict process faults, and adaptively control manipulated variables. First, the manufacturing cell is accommodated to environments such as computer-aided applications, industrial Product Lifecycle Management solutions, and control platforms for automation systems. Second, a network of interfaces between the environments is designed and implemented to enable communication between the digital world and physical manufacturing plant, so that near-synchronous controls can be achieved. Third, capabilities of some members in the family of Deep Reinforcement Learning (DRL) are discussed with manufacturing features within the context of Smart Manufacturing. Trained results for Deep Q Learning algorithms are finally presented in this work as a case study to incorporate DRL-based artificial intelligence to the industrial control process. As a result, developed control methodology, named *Digital Engine*, is expected to acquire process knowledges, schedule manufacturing tasks, identify optimal actions, and demonstrate control robustness. The authors show that integrating a smart agent into the industrial platforms further expands the usage of the system-level digital twin, where intelligent control algorithms are trained and verified upfront before deployed to the physical world for implementation. Moreover, DRL approach to automated manufacturing control problems under facile optimization environments will be a novel combination between data science and manufacturing industries.

1. Introduction

The manufacturing sector is currently reinventing itself by embracing the opportunities offered by digital transformation [1–3], industrial internet [4–6], automation [7–9], and machine learning [10–13] among other innovations. This development is commonly referred to as the Fourth Industrial Revolution (Industry 4.0) or Smart Manufacturing. To some extent, Smart Manufacturing systems can be seen as the cognitive counterparts of automation of physical processes [7]. While physical automation (e.g. robotic systems) relieves human operators of unergonomic, dangerous, repetitive, and heavy workloads, cognitive automation attempts the aiding or replacement of mental tasks that are stressful and repetitive by automatically processing large manufacturing dataflows [8]. Enabling this cognitive automation in manufacturing systems requires connected device ends such as sensors, controllers, actuators, networks, and intermediate software or frameworks, which is termed by the Industrial Internet-of-Things (IIoT).

Overall, Smart Manufacturing centers around data exchange, data acquisition, data processing/analysis, and utilization through data feeds into a continuous knowledge pipeline. Four workflow modules of data-driven smart manufacturing frameworks are identified by Tao et al. [14] : (1) manufacturing module with different kinds of manufacturing activities; (2) data driver; (3) real-time monitoring; and (4) problem processing. It has also been addressed that investigated smart manufacturing applications exploit big data analytics on real-world manufacturing data to refine manufacturing practices, increase production efficiencies or enhance product performances. Therefore, the convergence between digital model and physical manufacturing world

* Corresponding author.
E-mail address: harik@cec.sc.edu (R. Harik).

https://doi.org/10.1016/j.jmsy.2020.06.012
Received 29 October 2019; Received in revised form 16 June 2020; Accepted 24 June 2020
Available online 9 July 2020

has been pursued as an essential goal of data-driven smart manufacturing. However, smart manufacturing systems has been constrained by the lack of methods to connect factories to control processes in a more dynamic and open environment [15]. For example, previous research on data-driven smart manufacturing mainly focuses on data collected from the physical systems instead of the virtual model [16]. One most common concern with modelling physical smart manufacturing systems or products is that direct process quality measurements are often unavailable [17] or rare [12]. Moreover, there remains gaps in the applicability of currently available engineering tools towards smart manufacturing with data-driven controls. Risks such as discrepancy between virtual and physical manufacturing, or out-of-sync communications caused by hardware latency, can be significant sources of model non-convergence, which leads to isolated, fragmented and stagnant data management [18]. To cope with these challenges, it needs to be investigated that how to fully adapt current industrial tools as data drivers to derive predictive models of smart manufacturing knowledges. These knowledges can be generated both from virtual and physical spaces to enhance data integration and model convergence. Such attempt is also expected to demonstrate the easy applicability of manufacturing intelligence to practitioners without requiring specific data analytics expertise.

As an example of developed industrial digital transformation methodologies, Virtual Commissioning intends to verify manufacturing systems and associated control programs through simulation before the physical implementation by enabling the connection between a virtual plant model and a real controller [19]. However, such digital transformation method has been less attractive as it requires combined expertise and varied engineering skills, such as Computer-Aided Engineering, robotics, kinematics, control logistics, Object-oriented Programming, etc. In addition, the importance of low-level modeling (e.g. geometrical modeling, functional modeling, and electrical modeling) is further highlighted [20] alongside the basic implementation by Computer-Aided Engineering simulation tools and object-oriented databases. The combined expertise required to fill the gap between the physical systems and their digital counterparts is restricting the industrial scale applications of manufacturing digitalization.

Furthermore, enterprises, especially in small and medium scales, are also in need of the capabilities to quickly implement product changes and to produce highly variable products [21], which addresses the importance of highly responsive production control systems [22]. Meanwhile, highly autonomous automations driven by control intelligence in manufacturing industries are still at their early stages as safety, trust and efficiency concerns remain unresolved [23]. To pursue hardware autonomy, the required training process for online smart control systems will put more constraints on the industrial platforms. For example, integrating autonomous agents into hard-coded programs will need intermediate translations to interface mutually [8], hence introducing more variance in the overall system. Large amount of communication redundancy with instant feedback signals will need feasible memory management, as inefficient training approaches will lead to increased system run time, etc.

Digital twinning has been conceptually proposed as a future direction towards data-driven smart manufacturing by digitally representing physical entities and pursuing deep cyber-physical integration [14]. Compared to pure simulations, the idealization of digital twin is generally looking for dynamic optimization technologies that enable real-time system reflections, interaction between physical and virtual spaces, and automatic model evolvement with updated data feed [18]. It should be noted that the connectivity to the physical platform is a key advantage of digital twin approaches over a purely simulation approaches. Through live signals, data collected from external sensors, or knowledge derived from the physical cell can be fed into the digital twin. Therefore, dynamic connections to factory control processes should be underlined for digital twin implementations of Smart Manufacturing systems, as the industrial controls components are inevitable within the systems. In another word, the convergence of physical and virtual factories control process should be pursued. The digital twin should be capable of running event-driven simulations or models to perform control logics and predict system behaviors in responses to events. Moreover, digital twin of smart manufacturing systems, as an interactive and data-driven tool, should be developed towards powerful computer-aided modules, which contribute to generating manufacturing strategies, integrating virtual and physical data feeds into decision-making process, and predictive system monitoring based on the live signal feedbacks. However, considering the complexities of building the digital equivalence in virtual spaces [24] and various roles with inconsistent definitions [25], digital twin applications in recent years have yet made inadequate progress in production systems.

In this work, a novel approach is proposed to establish continuous interfaces with a virtual environment accommodated by industrial computer-aided applications to overcome aforementioned bottlenecks towards data-driven digital manufacturing systems. The proposed method to pursue digital twinning based on current virtual commissioning applications is to employ large-scale simulations, prompt system indicators, and computation technologies to establish a life-like digital manufacturing platform, where dynamics of live manufacturing cells can be captured, represented, and predicted. Moreover, this work attempts to demonstrate that the implementation of virtual commissioning, as one step to system-level digital twinning, will accelerate the training, testing, and validation of smart control systems. To that end, a use case of such modular digital twin is presented as surrogate models for physical manufacturing systems during the primary manufacturing intelligence development phases. By training a dynamic scheduler, the additional features provided from external sources can be envisioned to autonomously convey key information regarding process optimality. As a result, the conceptual outline towards an AI-driven robotic manufacturing cell proposed by our preliminary proof of concept [26] is further completed by illustrating more detailed implementations and key technologies along with some preliminary results. The ideation of such a platform optimization tool as a concept of digital engine coupled with a true virtual commissioning platform fits directly under the Smart Manufacturing umbrella and has the potential to develop into a number of advances in smart systems including improved throughput, safe human intervention, prompt self-monitoring, and highly autonomous operational control.

The remainder of this paper is organized as: Section 2 provides a review of existing research topics on digital twin of production systems. Section 3 introduces the system environment; construction of the virtual cell and its capability to simulate manufacturing problems. Section 4 presents the interfaces between the systems and demonstrates near real-time communications using the implemented interfaces. Section 5 proposes an implementation of this Digital Engine using Deep Reinforcement Learning algorithms as dynamic scheduling agents. Section 6 presents some primary training results with a specific case study within the virtual environment, which is generally applicable to certain manufacturing problems. The last Section 7 concludes the main contributions of this work and proposes future research directions.

2. Literature review

Negri et al. summarized the roles of the Digital Twin (DT), which are still mostly applied in product predictive maintenance and condition-based monitoring related research in aeronautics and space fields [25]. However, it is worth noticing that DT usage is emerging in the fields of manufacturing and robotics, where the emphasis on Virtual Commissioning and automation system optimizations become in demand. In 2013, the first work reporting research on DT in advanced manufacturing sector considered DT to be the virtual counterpart of production resources, and not only of the product [27]. In addition, with its core technologies in big data analytics and cloud platforms, recent Industry 4.0 developments help expand the application of DT to more

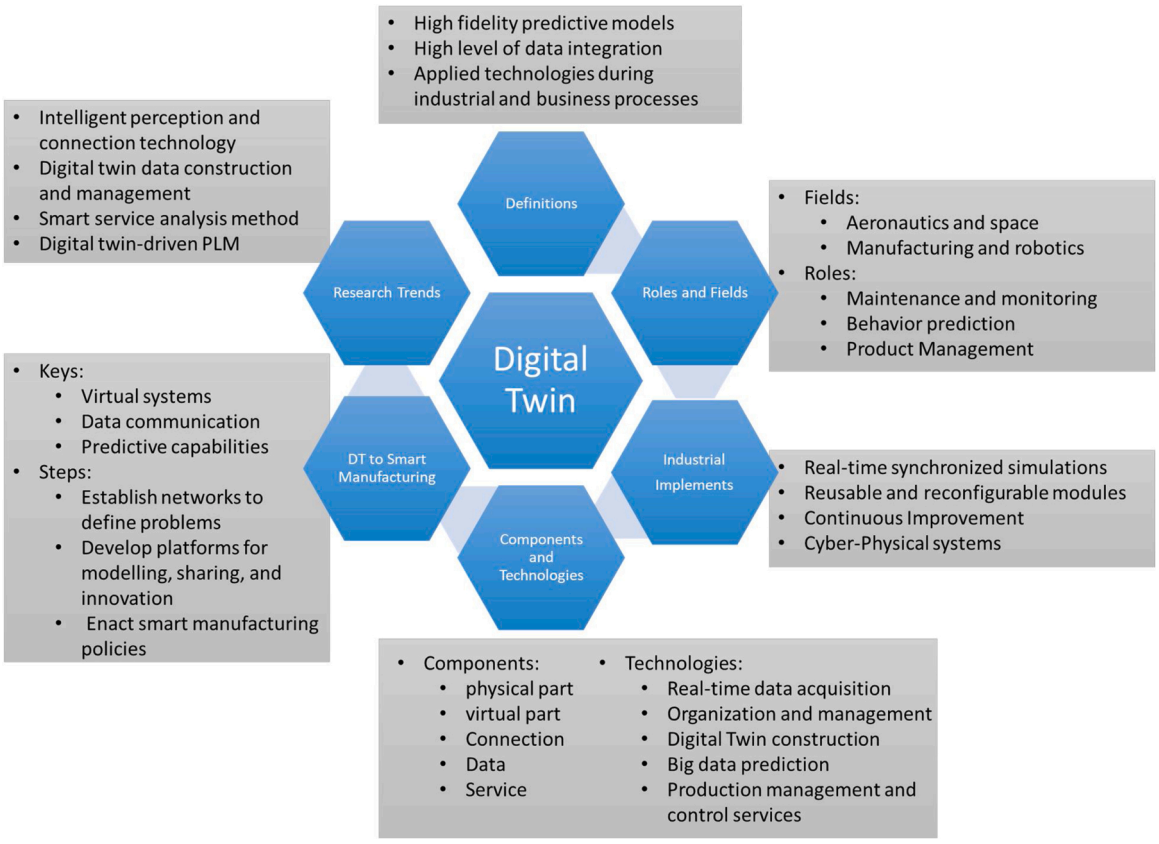


Fig. 1. A structured summary of investigated digital twin literatures.

data-driven, virtually augmented, and smart integrated manufacturing systems.

The definition of Digital Twin has been non-uniform and ambiguous depending on the application areas. Scientific researchers tend to look for high-fidelity reference models to improve predictive capabilities of digital twins, where interactive optimization can occur on both physical and virtual parts. Schleich et al. proposed a concept of utilizing Skin Model Shape [28], an abstract model of the physical interface between a workpiece and the environment, as a reference model to predict the product physical properties [29]. Although the scope of this work focuses on the digital twinning of product development, the idea of using digital twin for geometrical variation management serves as the basis of production system twinning such as industrial process control. Another work by Knapp et al. proposed to utilize a transient, three-dimensional mathematical model for Additive Manufacturing process as the blocks for digital twin [30]. Meanwhile, industrial implementations of Digital Twins focus more on implanting developed and tested technologies to manufacturing systems, communicating about gained manufacturing knowledge, and continuously optimize by data-driven decision-making. Zhuang et al. concluded five current key technologies: real-time data acquisition, organization and management, DT construction, big data prediction, production management and control services [24]. Konstantinov et al. demonstrated uses of Cyber-Physical System enabled virtual engineering tools within a practical workflow to complement existing engineering tools and methods [31].

Virtual Commissioning is another well-developed technology for testing systems through simulations to evaluate the safety and feasibility of scheduling and manufacturing approaches before physical deployment. An overview by Hoffmann et al. [20] demonstrated the implementation of Virtual Commissioning necessitates a Computer-Aided Engineering (CAE) simulation tool environment and object-oriented databases containing simulation models of manufacturing system components, by which several recent attempts [32,33] were made in realizing Virtual Commissioning with the same philosophy using different tools to construct virtual plant, hardware/software architecture, communication pathways, etc. The employment of virtual counterparts of manufacturing systems was further validated in [31,34]. However, several deficits and problems of currently employed DT technologies were summarized [35] in production systems towards near real-time optimizations, including manual motion data acquisition process, insufficient coupling between simulation and optimization resulting in slow-down in parameter generation, lack of simulative investigations, high expense to use real-time locating system technology such as RFID on the tracked objects, and concerns in data security. Machine vision technology as image recognition is proposed to identify products at large production machine. Built on the emerging technologies, complete components of DT in manufacturing are proposed by Tao et al. to include five parts: physical part, virtual part, connection, data, and service. They also generalize the characteristics of DT systems to be real-time reflection, interaction and convergence between virtual and physical space; along with self-evolution, real-time updates, and continuous improvement through comparing virtual space with physical space in parallel [36].

The implementation of Digital Twin is centered around data and interfacing communications since mass customization and flexible production emphasize the need for an easier high-level data storage and model exchange between different systems connected to the DT [37]. In this work, an object-oriented paradigm (XML/JSON) and IoT middleware are used for an easier exchange of data [37]. The transient data feed between objects requires more efficient and safer communication protocols to one of the DT key technologies, such as MQTT [38], MTConnect [8], and OPC-UA [39] attempt to address this. OPC-UA in particular has a more general adoption in industy. In the field of system optimization, a comparison between the conventional process optimization tool Value Streaming Mapping and Digital Twin was made and hence the potential near real-time data acquisition and simulation

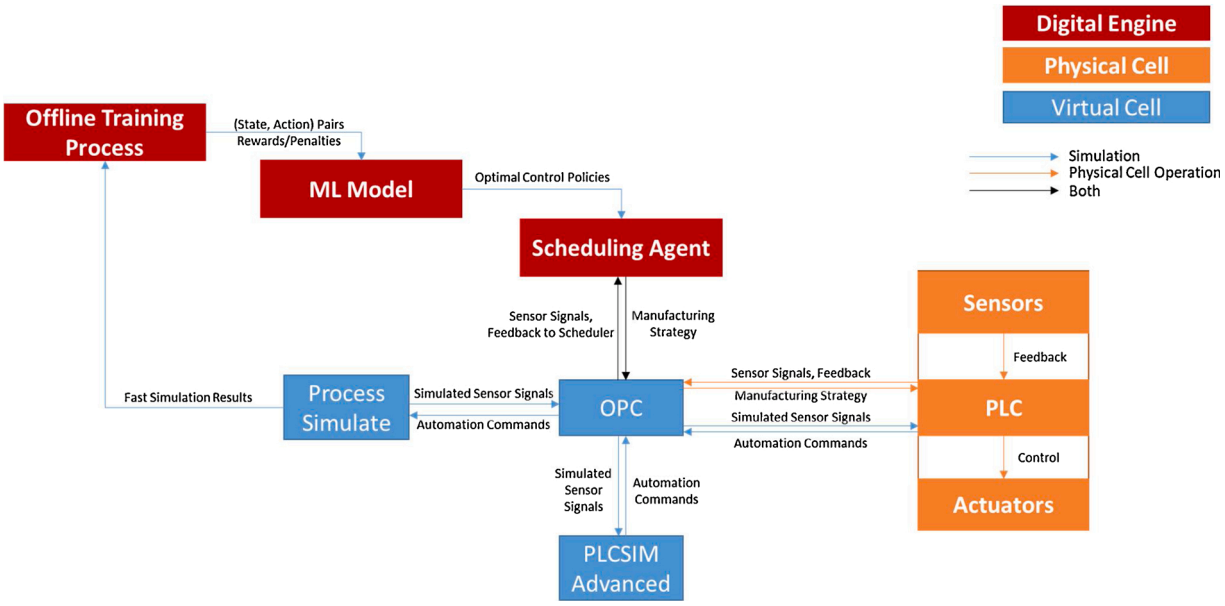


Fig. 2. Data Flow between proposed Virtual-Physical-Scheduler System Ends.

capabilities of DT were demonstrated [16]. An interesting application of applying real-time synchronized simulation of the production system as a part of highly responsive and modular production control system is proposed, named *Synchro-push* [22], which continuously updates inventory status and performs adaptive scheduling of production orders and transfer management in prompt response to the changes in the production mix. Cyber-Physical Systems (CPS) are further envisioned to help developing small and medium enterprises manufacturing applications by dividing the physical system into reusable and reconfigurable modules. Derived templates or modules have shown improved design flexibility and productivity for highly-customized robotic manufacturing systems [21].

Kritzinger et al. attempts to distinguish recent applications by the level of data integration between digital and physical objects [40]. *Digital Model* implements bi-directional manual data flow, while *Digital Twin* enables bi-directional automatic data flow. *Digital Shadow* only feeds one-way automatic data flow from the physical object into the digital object, while from digital object to physical object manually. Based on this categorical method, majority of the investigated publications are classified as Digital Shadow and Digital Model. The case studies that fit the definition of two-way data streaming *Digital Twin* applications are scarce. Another comment provided in this literature study is that the *Digital Model* helps to find particular components of a product faster. In the context of Smart Manufacturing, desired components of production systems can also incorporate optimal strategies to finish manufacturing tasks. The advantages of DT are also highlighted [16] in continuous data acquisition, automated and repeated derivation of optimization measures and capturing of motion data, etc. Particularly, communicating with DT system can access augmented data for optimization process. For example, a use case of DT is described in a large assembly shop floor, where it serves to predict and prevent potential disturbances, anomalies, and problems to the production procedures in advance. Three kinds of data are being exchanged: real-time perception data, production process data and production activity plan data. These are highlighted and their linkage to resource deployment and optimization are depicted under an event-driven assembly line [24]. To that end, three levels of DT components need to be modelled: element, behavior and rule, which can be interpreted as: system components, predictive responses, and control policies.

It is identified that the future research trends on DT-driven design and manufacturing to be: (1) Intelligent perception and connection technology, (2) Digital twin data construction and management, (3) Smart service analysis method based on digital twin data, and (4) More applications on digital twin-driven PLM [18]. The data-driven approach to Smart Manufacturing was further outlined with three essential steps: (1) Establish networks to define problems; (2) Develop platforms for modelling, sharing, and innovation; (3) Enact smart manufacturing policies. A structured summary of this literature review is shown in Fig. 1 to highlight surveyed topics regarding the digital twin concept when identifying current research gaps.

3. Simulation environment of the digital twin

In this work, the Digital Twin is based on one of the technologies of digital transformation, Virtual Commissioning. Current implementations of Virtual Commissioning still require manual construction of the digital system, definition and tuning of system components. However, the development of industrial software solutions to Virtual Commissioning has greatly improved the accuracy and user-friendliness of offline programming robotic systems and verifying control logic over the traditional commissioning process. The Virtual Commissioning solution used to build the virtual cell for this work was *Siemens Tecnomatix Process Simulate*. In this section, system conceptual outline proposed in the author's previous work is revisited; the construction of the virtual cell and some useful features, such as robot offline programing, collision detection and robot reachability, are further discussed to support digital twin predictive capabilities. Finally, event-based simulations are explored to resolve signal-based control problems.

3.1. System overview

At the level of system integration, this smart data-driven digital twin implementation workflow unfolds from three aspects. First, a virtual platform is constructed within industrial software to simulate real-life manufacturing cell behaviors. Second, towards a real-time, two-way implantation of digital twin, the control loop digitalization and near-real-time data communications are furtherly realized. Third, the virtual and physical system integration is driven by an intelligent scheduler while training machine learning models for scheduling optimization. This dynamic scheduler agent, termed the Digital Engine (DE), is developed as a smart process optimization tool utilizing integrated platform data and applicable machine learning algorithms. To solve

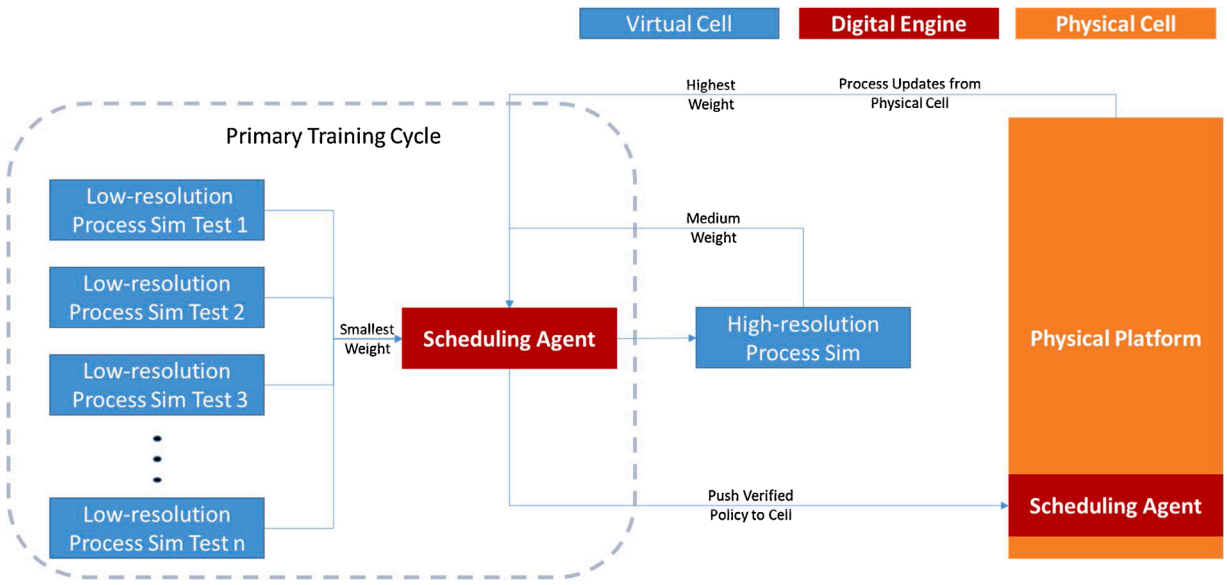


Fig. 3. Digital Engine Development with proposed Virtual-Physical-Scheduler System.

real-life manufacturing problems, the virtual environment need to be constructed in high fidelity that able to employ technologies in simulations, sensors, communication and computation to receive cell states and reflect responses to process disturbances, to which extent meet the criteria of a *Digital Twin* [40].

The proposed system (Fig. 2) consists of: **(1)** Machine Learning (ML)-based dynamic scheduling agent *Digital Engine* (components in red) linked with both **(2)** the physical manufacturing cell (components in orange) including sensors, PLC controllers, middleware control components and other actuators, and **(3)** the virtual manufacturing cell (components in blue) accommodated by selected industrial simulation software, enabling the testing and commissioning of control logics and programs to be pushed to the physical plant. The communication of the proposed system requires information flow between three ends as in Fig. 2. The detailed implementation of control loops and data communications will be introduced in Section 4.

In the field of process system engineering, Reinforcement Learning has been applied to better solve some challenging optimal control problems like the dual adaptive control and more general nonlinear stochastic optimal control [41]. Therefore, the Digital Engine is proposed to use Reinforcement Learning with deep neural networks, namely Deep Reinforcement Learning, to solve stochastic optimal control problems with uncertainties from either the highly complicated signals or processes [11]. In these real-world problems, Reinforcement Learning has been found to outperform current other optimal control techniques, such as the model predictive control (MPC) [42]. Implementation of the Digital Engine is further described in Section 5 and Section 6.

At the level of data integration, the proposed digital twin implementation can be characterized by a data-driven approach as it augments real-world data with virtual data to derive heuristic algorithms for varied analytical use cases. The heuristic algorithms are induced by data samples generated from either simulations or physical processes and in return drive decision-makings. With the capabilities of near real-time data monitoring and inference, manufacturing decisions can be autonomously made by the proposed digital twin to generate adaptive actions with statistical significances and/or empirical evidences, instead of by conventional hardcoded control logics with deterministic conditions.

Compared to real-world manufacturing data, data generated by the virtual model are repeatable, inexpensive and clean. For some of the analytical models, virtual data can be utilized to show some initial results and mine some hidden relationships, as they tend to be abundant and easier to be inferred with, given some degrees of convergence between virtual and physical data. Meanwhile, if high-dimensional and messy real-world manufacturing data is directly used to automatically feed into the industrial control processes as forms of short and dense signals, they must be preprocessed and inferred to higher-level manufacturing knowledges, namely by semantic communications [8]. To conclude, this work intends to provide data-driven decision-making solutions to the current industrial applications with a high-level data integration from virtual and physical systems.

As shown in Fig. 3, the training process for the dynamic scheduler of Digital Engine is designed to be augmented by developed virtual environment and its interfaces. Training Digital Engine is proposed to perform under three phases: (1) primary training cycles by fast iterative low-resolution simulations; (2) virtually verification training cycles with high-resolution simulations; (3) physical verification training cycles with physical platform. Each of these training phases relies on increasing data fidelity, with the primary training cycle taking place virtually with a coarse time step, the next phase of training with a higher fidelity simulation, and finally generating networks updated from the physical cell. The Deep Reinforcement Learning networks are trained through a gradient descent process, thus requiring finite learning iterations. The importance of a learning iteration, and thus the degree to which a resulting training sample is weighted in the networks, is adjusted depending on the data fidelity of the training loops. In the field of machine learning, such importance weighted transfer of Reinforcement Learning samples is explored and envisioned to be improved by fully exploit possible similarities between different tasks [43]. The idea of such transfer of learning is to utilize prior trained models to resolve more specific, unexplored and furtherly complicated datasets instead of training the large models from scratch, as past experiences can inductively affect learning and performances in a new situation. The transfer of learning has been addressed as an important approach in the machine learning community, since this integrated learning transfer can be facilitated by reusing of trained neural networks.

3.2. Construction of virtual cell

In Reinforcement Learning, smart scheduling agents must be given an environment to interact with and learn from. The clear limitations of performing these training cycles with a physical platform, simply for safety reasons alone, necessitates digital simulations and a means to validate manufacturing strategies ahead of implementation on a

Table 1
Advantages of Siemens Process Simulate as a digital twin environment for smart manufacturing systems [44].

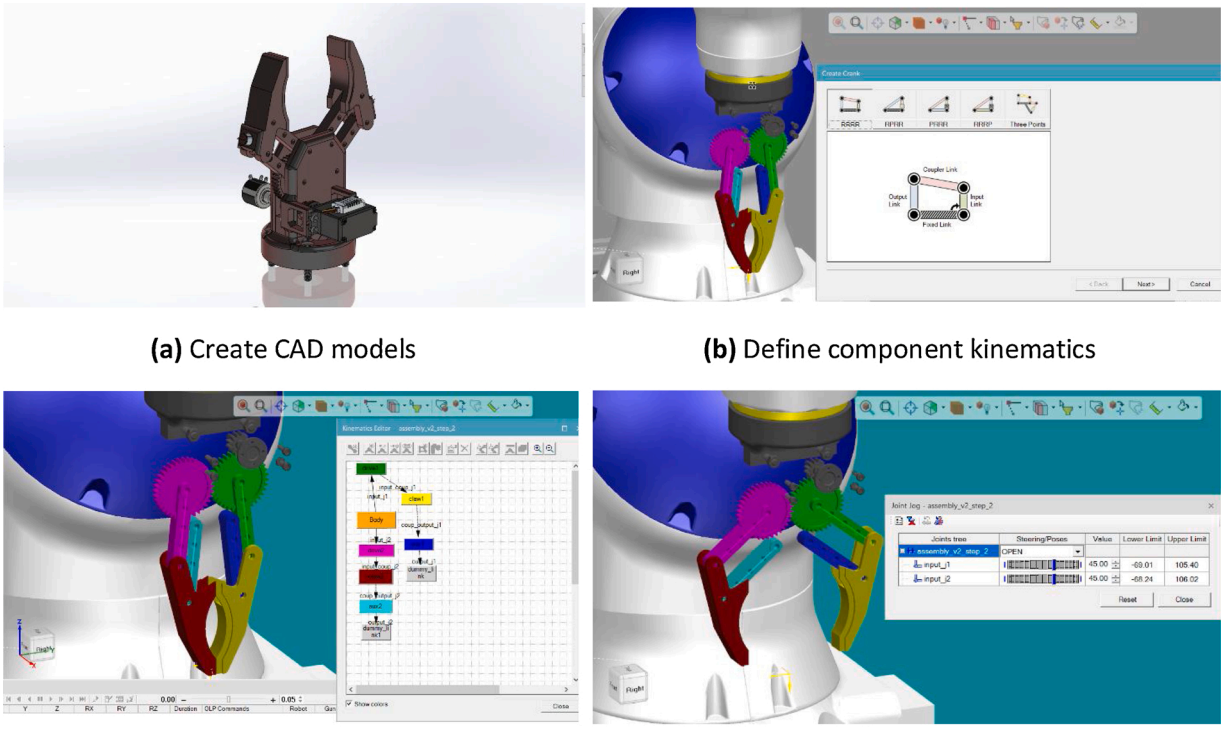
Advantages	Underlined Features
Early detection of product design issues Physical prototypes reduction by upfront virtual validation Reduce cost by re-using standard tools and facilities	2D and 3D sections; 3D measurements; resource modeling (3D kinematics)
Minimize productions risk by simulating several manufacturing scenarios Early validation of production commissioning in a virtual environment Increase process quality by emulating realistic processes throughout the process lifecycle	Line and workstations design; static and dynamic collision detection; sequencing of operations; discrete and continuous process simulation such as: projection of welds on parts, Gun search wizard, project arc seam, Torch alignment, Weld gun validation, Design/modify weld gun and tooling geometry and kinematics, robot reach test; Robot reach test; Robot smart placement; Robotic simulation editing
Early validation of the mechanical and electrical integrated production processes (PLC and robotics) Optimize cycle times through simulation	Assembly and robotic path planning and offline programming; integrated simulation using actual PLC code and control hardware via signals exchange over OPC interface; Controller-specific command recognition; Event-driven simulations with internal resource logic
Ensure ergonomically safe processes	Human tasks simulation such as: reach envelopes; vision window; postures; auto grasp wizard; ergonomics analysis

physical platform. The integration of the Digital Twin into the training and testing process (See Fig. 3) provides a virtual platform where cell states and actions can be retrieved from, in order to simulate potential physical outcomes and reduce potential risks on a physical counterpart.

Siemens Process Simulate provides these functions such as importing CAD, kinematics definition and simulation, collision checking, and teach-type robot programming. Detailed advantages of selecting this industrial software as potential smart manufacturing system digital twin platform are listed in Table 1. It is summarized by authors that the advantages of *Siemens Process Simulate* fall into four groups of features: (1) computer aided engineering; (2) task-specific manufacturing process simulations; (3) data exchange enclosed in an industrial package, where PLC and OPC servers work with live signals for process control; (4) Ergonomics analysis. In this work, the emphasis will be put on the first three groups of features with an exception of ergonomics analysis. With a working knowledge of the first three groups, a functioning virtual robotic cell can be easily created and defined; from which, possible robot configurations can be derived, simulated and translated to robot programming languages, given accurate robot calibrations and critical frames work definition such as the Tool Center Point (TCP). These procedures are normally referred as off-line programming.

In Process Simulate, a functional virtual cell depends on accurate definitions of system components. While CAD models and definitions of well-developed products (such as most industrial robot models) can be directly retrieved from manufacturers with kinematic definitions to ensure accurate simulation, some cell components still require manual definition before importing to Process Simulate. For example, the kinematics of an in-house manufactured robot gripper from our stage 1 platform had to be defined as in Fig. 4.

In defining the kinematics of this device, first a CAD model was



(a) Create CAD models

(b) Define component kinematics

(c) Link components

(d) Jog joints to verify kinematic definitions

Fig. 4. Gripper kinematics definition in Process Simulate.

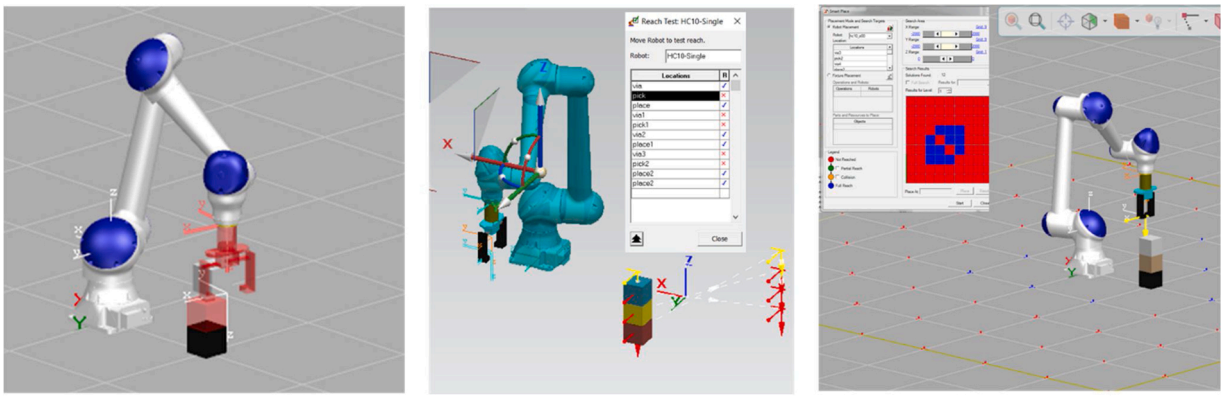


Fig. 5. Predictive capabilities of proposed digital twin.

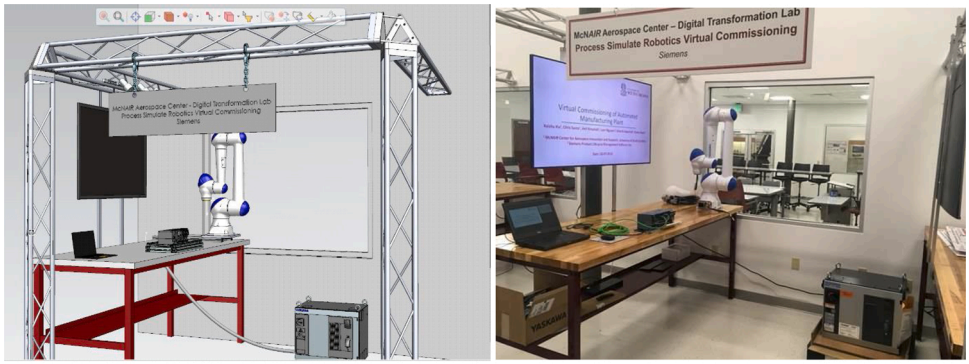


Fig. 6. Stage 1 platform setups.

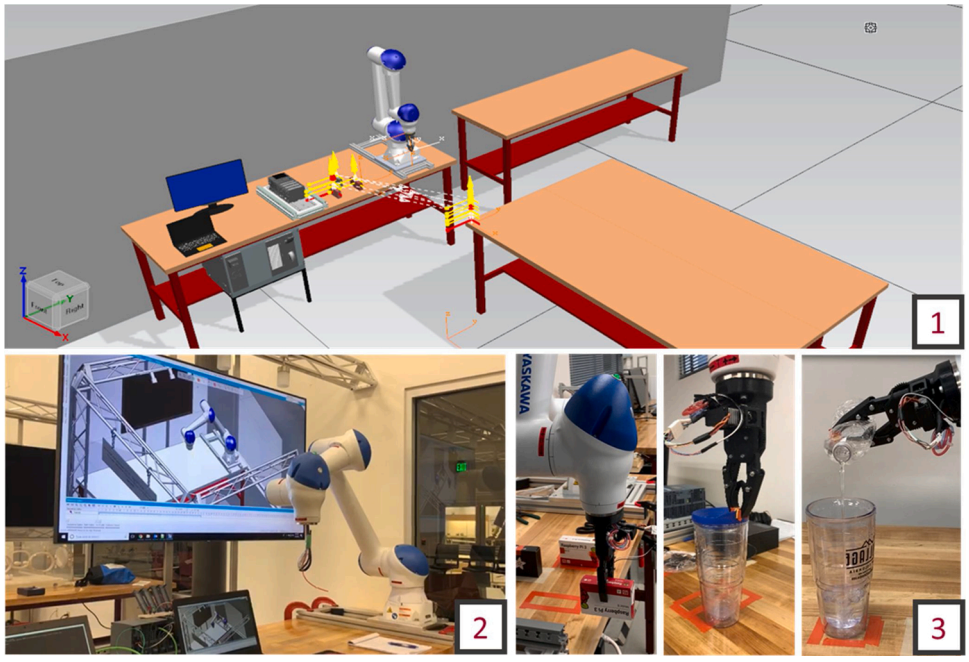
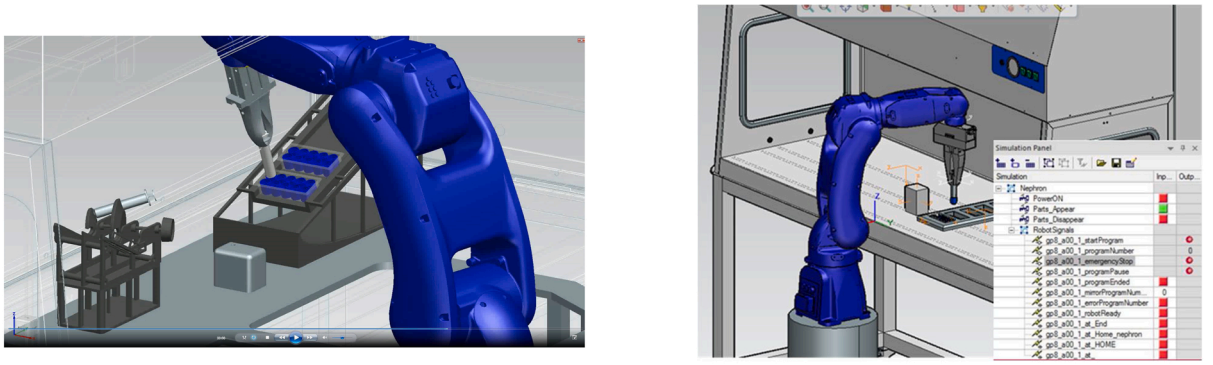


Fig. 7. Stage 1 platform experiments: (1) Virtual path planning; (2) Offline programming with near-synchronous virtual robot motions; (3) Virtual commissioning tasks: pick and place, capping and assembly, and pouring by continuous movement.



(a) Simulation for automated syringe filling process **(b)** Signal-based control under line simulation

Fig. 8. Collaboration with Nephron Pharmaceuticals on their robotic manufacturing application.

created in a CAD software such as Siemens NX, Catia or SolidWorks and exported with all the components in an assembly folder. Process Simulate supports importing of CAD files from almost all sources with valid formats. As in Fig. 4a, the robot gripper was designed using Siemens NX and output as an assembly of kinematic units. Second, the kinematics of the gripper base, gear rod, connecting rod and gripper finger were grouped as a crank, which consisted of a fixed link, input link, output link and coupler link. As in Fig. 4b, the gripper was composed with double crank mechanisms (parallelogram linkages) coupled with a motor driving gear, which can be easily defined in Process Simulate by designating each unit's role. Third, the defined components were grouped and linked, and their relative translations or rotations were specified. In Fig. 4c, the diagram linking each unit specifies their kinematic dependencies, as the relative translations and rotations can be numerically defined by arrowing two units. Finally, the kinematic definitions were tested by jogging the joints as in Fig. 4d. The gripper positions and join values were configured as the OPEN, CLOSE, SEMIOPEN poses for simulation uses.

Beyond object kinematics definitions, the successful virtual cell construction also requires accurate calibrations for object placement and postures, mounting or attachment information about grippers or any other end effector on robot ends, object collision detection to verify there are no dynamic intrusions between objects, robot reachability validation to see if all the path locations are within the possible configured reach, etc. Process Simulate provides a simple user interface by using the virtual cell, replicated from the physical cell, to simulate object dimensions to detect collisions (Fig. 5a), robot path validations with reverse kinematics calculations (Fig. 5b), and design potential physical cell setups by smart locating objects (Fig. 5c). Benefiting from aforementioned *Process Simulate* features, our stage 1 platform setup were constructed (Fig. 6a) with a high fidelity to the physical cell (Fig. 6b) ensuring all the robot paths are valid and safe within reachable and collision-free regions.

3.3. Simulation of robot motions

To ensure the feasibility of proposed system, the path accuracy of Process Simulate offline programming was investigated. Virtually programmed paths were tested on Stage 1 platform physical setup (Fig. 6 right), which includes a Human-collaborative robot Yaskawa Motoman HC-10, the YRC1000 OEM robot controller, and a SIMATIC S7-1516 F Siemens Programmable Logic Controller (PLC). The program generated by *Process Simulate* offline programming was directly transferred to YRC1000 and successfully interpreted by HC10 with their proprietary Yaskawa programming language, *INFORM III*.

Stage 1 platform experiments (Fig. 7) were performed in the following procedures: (1) virtual path planning in *Process Simulate*; (2) offline programming, which generates the *INFORM III* code within *Process Simulate* platform; (3) physical experimentations with virtually generated robot programs. Three different robot tasks were virtually commissioned and then physically tested: pick and place, water pouring, capping and assembly. These robot tasks were accomplished with high precisions given correct calibrations, defined kinematics, path locations and robot configurations. Moreover, under the option of 1:1 real-time simulation speed, the simulation and real robot programs are executed near synchronously. However, some tuning of the simulation fidelity must be attempted to reach near zero lag. Depending on the computing performance of the PC hosting *Process Simulate*, a slight latency delay might be generated when *Process Simulate* attempts to calculate dynamic inverse kinematics in real-time. This latency can be effectively mitigated by either upgrading the computing capability of the host PC or downgrading the simulation resolution to reduce the amount of kinematic calculations. Simulation resolution in *Process Simulate* is controlled by simulation update intervals. While smaller simulation intervals lead to computational latency, higher path planning precision can be obtained, and hence potentially increases digital twin predictive capabilities, e.g. more keen object collision detections. Recall in Fig. 3, controlled training phases under both low-resolution and high-resolution simulations need to be performed during Digital Engine development. The functionality of *Process Simulate* can easily achieve controlled simulation resolutions depending on the needs of both training phases. Properly reducing simulation resolutions will avoid redundant training time and help to identify some process indications faster. With the above experiments, it is concluded that *Process Simulate* is a powerful tool for the simulation-based virtual commissioning of robot motions and paths. Given the simulation accuracy, robot manufacturing virtual cell proposed in this work is based on *Process Simulate* virtual environment and its communication pathways.

3.4. Event-based simulations

Time-based simulation is typically used in digital models when planning deterministic robot operations during off-line programming. In a time-based simulation, the sequence of events that occurs in the cell is fixed and specified before the simulation runs. In contrast, an event-based simulation does not require manual sequencing of events, and instead events are determined dynamically at runtime. Event-based controls are more useful since they are more robust against disturbances, such as change of lead time, altered task, equipment failure, etc. Zhuang et al. highlighted one of the Digital twin-based production management and production service to be event-driven real time decision-making subject to rescheduling or plan adjustments due to such production disturbances [24]. Conventionally, control engineers had to manually program the logic behaviors in the physical plant, which can be time consuming processes and difficult to debug with. As controls programs become larger, unexpected bugs such as communication

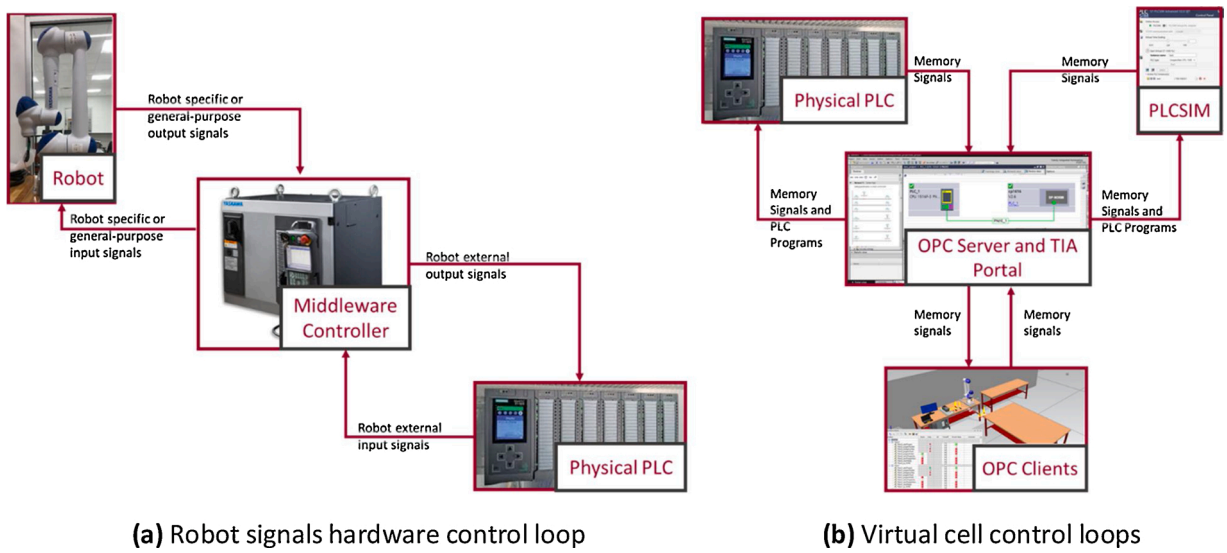


Fig. 9. Digitalization of control loops in physical and virtual robot platforms. (a) Robot signals hardware control loop. (b) Virtual cell control loops by Hardware-in-the-loop (PLC as controller) and Software-in-the-loop (PLCSIM as controller).

interlock failures occur, which can cause cascading delays or damages to the physical cell. To reduce the risk of these problems, virtual commissioning using event-based simulation, also named as Line Simulation or Cyclic Event Evaluator (CEE) simulation in *Process Simulate*, allows control engineers to program, debug and test their signal-based control logics.

Another advantage of adopting virtual commissioning with signal-based control is that it simulates the system response of the virtual cell from physical devices interactions such as PLCs and Human Machine Interfaces (HMI) by enabling signal exchanges between the physical devices and virtual cells. For example, operators can easily push commands to the whole virtual platform using the far simpler HMI screens; simultaneously, the same HMI screens and logics verified in *Process Simulate* can be then safely implemented in the physical cell. In addition, within the virtual cell in *Process Simulate*, users can easily create virtual versions of sensors and link with any signals inside physical PLCs or HMIs using defined control logics. These control logics can be defined either within the scope of an object, namely resource logic behaviors, or over the whole cell, namely control modules. One example of logic behaviors is that the robot servo signal needs to be on before setting a master job. This precondition is usually specified by the robot manufacturer and need to be implemented in the simulation within the robot itself. Resource logics can then be duplicated and reused when another same robot model is added to the virtual cell. The control modules are where a sequence of programs get scanned and executed when the simulation starts. They are programed on a higher level and can be seen as a main entrance of cell events, which resembles PLC main programs. These completes the digital counterparts of signal-based control systems (more details in Section 4), which is a crucial element of automated system-level of digital twin. Beyond our Stage 1 platform, the proposed digital twin implantation is also expanding to small and medium-sized enterprises robotic manufacturing solutions (Fig. 8a) to create signals-based control logics inside a virtual cell (Fig. 8b) that is easier and safer to experiment with than inside a physical cell.

4. System control loops and communication pathways

Beyond simulation-based virtual cell construction, data communication between systems is one of the other major topics in creating an interactive digital twin. Depending on the types of controllers and interacting environments in the control loop, system commissioning is categorized as real commissioning, hardware-in-the-loop commission ing, reality-in-the-loop commissioning and constructive commissioning [19]. In particular, Virtual Commissioning control loops, under the assumption of interacting with virtual environments, are classified as "hardware-in-the-loop" and "software-in-the-loop" depending on whether physical components such as PLCs and HMIs or their virtual counterparts are connected to the simulations.

In this section, our stage 1 implementation of both "hardware-in-the-loop" and "software-in-the-loop" are introduced in corresponding to prosed data flow between *Process Simulate* and either a physical PLC or a PLC simulator *Siemens PLCSIM Advanced*. Then, a customized application program interface (API) with *Process Simulate Tecnomatix. NET*, is designed and utilized to connect Machine Learning (ML) frameworks to the virtual cell for fast offline training process. When the ML-based scheduler is trained to be sufficiently intelligent, reliable and robust, the scheduling agent is deployed to communicates with both virtual cell and physical cell through a shared information hub *Open Platform Communications* (OPC) server, through which high resolution training cycles will be performed on both digital and physical cells. This completes the proposed system data flow in Fig. 2 and training phases in Fig. 3, and implies that proposed digital twin of production systems can be used as an augmentation tool to train decision-making intelligence with significantly reduced safety and cost concerns.

In this digital twin, the fusion of data from physical and virtual sources is proposed to be realized in two manners. First, with the philosophy of Virtual Commissioning being the capability to virtually validate system engineering, an intuitive data fusion occurs in a sequential manner, which means the digital twin, as a surrogate system, to upfront check system data resides in object dimensions, robot dynamics, signals, control logics and executed programs before they flow into the physical system implementations. This approach is described in this work as an importance-weighted data integration process (see Fig. 3). Second, beyond the conventional virtual commissioning approach, our proposed system, which is driven by machine learning modules, enables a pathway to convert unprocessed, complex and unclean real-word data to semantic communications among PLC control logics. The classification and pattern recognition capabilities of machine learning algorithms will be further utilized in the industrial decision-making process in a timely manner. To that end, specific data inference models will need to be developed, trained and validated by different datasets that can be potentially amplified by virtual data. This manner of data fusion by hybridizing physical and virtual datasets for specific manufacturing processes will be further pursued in our

subsequent work, which is enabled by the data communication scheme in this proposed digital twin implementation.

4.1. Physical control loop and its digital counterpart

The main industrial control loop in a physical cell is administrated via a PLC, which centralizes all the control logic flow between lower level components. Advancements in control paradigm typically necessitate reworks in current physical configurations, including redesgin, rewiring and reprogramming of physical PLCs [22]. To cope with this, we implement a similar philosophy of cyber-physical system based modular factory for the goal of easily customizable and reconfigurable control modules [24]. To evaluate control feasibility and effectiveness, control scheme simulations as a digital counterpart of the physical control loop need to be developed. The use case of control digitalization can be the closed-loop optimizations for shop-floor mangement and control services [24].

Our physical cell components of stage 1 platform (Fig. 6b) include sensors, actuators, middleware controllers and a *S7-1500* PLC. Middleware controllers are chosen to control specific acurators, end effectors or lower-level control objects, for example, *YRC1000* controller is the master of the HC10 robot. The communication between *S7-1500* and *YRC1000* robot controller is achieved through a Siemens *CP1616* PROFINET board. *Siemens TIA Portal* as the automation software platform to program Siemens PLCs, including modules as *WinCC* and *Step7*, can also create HMI screens and allow access to the OPC server from a PC. The physical control loop over a robot is presented in Fig. 9a.

During the control process, downloaded programs are executed by PLC cyclically scanning and compiling sequenced rungs, usually as ladder diagrams. When these programs are directly compiled and tested on physical setups, their debug process is often difficult and time consuming, as the PLC is potentially giving or receiving faulty commands to the physical system. Hence, Virtual Commissioning provides a methodology that can interact with the digital twin not only by performing process simulations, but also by virtualizing the control loops.

The control loops of Virtual Commissioning components are described in Fig. 9b. By which means, programmers are able to expect the system responses from the digital twin by downloading untested logics to either physical PLCs (“hardware-in-the-loop”) or the simulated PLC (“software-in-the-loop”). The “hardware-in-the-loop” implementation consists of the following components: physical PLC, OPC server and OPC clients. OPC server/client pairs are software interface standard enabling PC to communicate with industrial hardware devices. OPC server converts the hardware communication protocol used by PLCs to OPC protocols. OPC server is accommodated in *S7-1500* and can be accessed by OPC clients such as *Process Simulate*, which connects directly to digital cell signals. On the other hand, a “software-in-the-loop” implementation in Fig. 9b presents a software-only control loop that includes the virtual counterparts of the physical components: simulated HMI, PLC simulator, OPC server and OPC clients. The difference between “software-in-the-loop” and “hardware-in-the-loop” lies in whether simulated PLC and HMI are used instead of physical PLC and HMI. “Software-in-the-loop” excludes the usage of hardware components in the loop of two-way communications between physical and digital counterparts by routing the signals through the OPC server and the PLC simulator, where the programs can be executed within a software environment that matches the behavior of a real PLC. By this route, control safety and feasibility can be evaluated in the virtual environment before downloading to physical PLC. Hence, the digital counterpart of the control loops is achieved by simulating both PLC functions in *PLCSIM Advanced* and Human Machine Interfaces in *WinCC Runtime*. Therefore, proposed digital twinning is realized not only in system modelling and simulations, but also in the digital transformation of control and connection pathways, which completes the definitions of *Digital Twin* [40].

Table 2
Streaming script between Process Simulate API and Digital Engine in one training episode. The Process Simulate performs different simulations under two modes: Standard Mode with time-based simulations and Line Sim Mode with event-based simulations. Hence, cyclic data exchanges via API are differently implemented. Meanwhile, the Digital Engine has pipeline end to exchange data with API and AI model to continuously train the scheduler.

Process Simulate Tecnomatix. NET		Data Flow Pipeline	*Digital Engine*	
Standard Mode	Line Sim Mode		Pipeline End	AI model
1. Create a snapshot of initial status	**1.** Control module initialization (reset entries and perform initial scan)		Listen on pipeline	Sample buffer memories and train AI model
Loop until episode finished signal				
2. *Push* plant time-based simulation status	**2.** *Push* plant signal-based status	→		
			3. *Pull* current status and previous simulation results from pipe	Feed current status to AI model. Memorize last simulation results
Listen on pipeline				
		←	**4.** *Push* optimal operation to pipeline	Evaluate all action values for optimal operation
5.*Pull* optimal operation command from pipe				
6. Set optimal operation related program; play forward simulation to update plant status	**6.** Set PLC output signals to actuators and trigger program execution; and update plant status		Listen on pipeline	Sample buffer memory and train ML model
7. Apply snapshot if episode finished signal true	**7.** Reset control modules and simulation player if episode finished signal true			
Repeat				

4.2. Integrating the digital engine

The interface structure between digital and physical twin has been discussed thoroughly at this point. Based on this, our work extends the communications to an AI-driven scheduler *Digital Engine*, where AI algorithms such as Deep Reinforcement Learning and Machine Vision can be trained and integrated into both digital and physical twins, so that the trained dynamic scheduler can be reliably working on both ends. The language that is being used to develop *Digital Engine* is selected to be *Python*, where there are many AI frameworks built by data scientists and imported and deployed with limited data science expertise required. To enable communications between customized programming ends and digital platform in *Process Simulate*, a pipeline streaming script (Table 2) is created using the *Process Simulate* API named *Tecnomatix. NET*, which provides an object-oriented programming interface with *Tecnomatix* products backend database. It is developed to interact with products as *Process Simulate* by accessing all the defined objects and operations in virtual cells, creating new objects and operations, controlling playback of simulations, creating customized commands in the user interface, etc. Particularly, in our application, the ability to retrieve online signals and program control modules are essential because the proposed digital twin

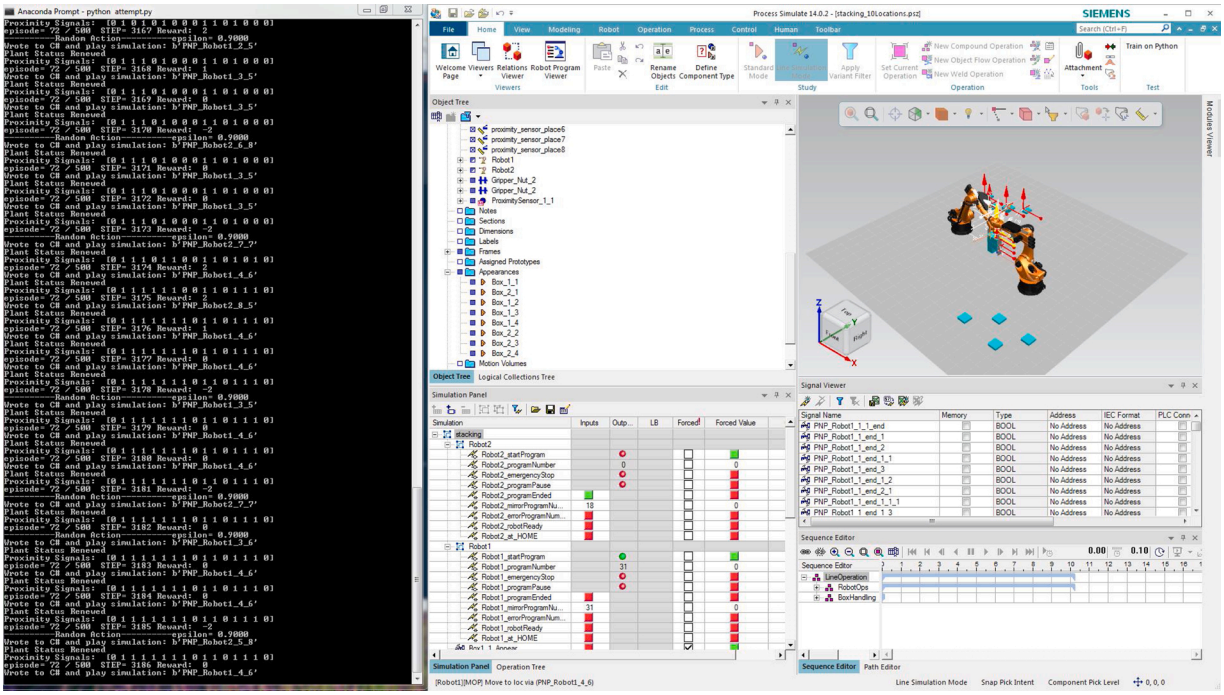


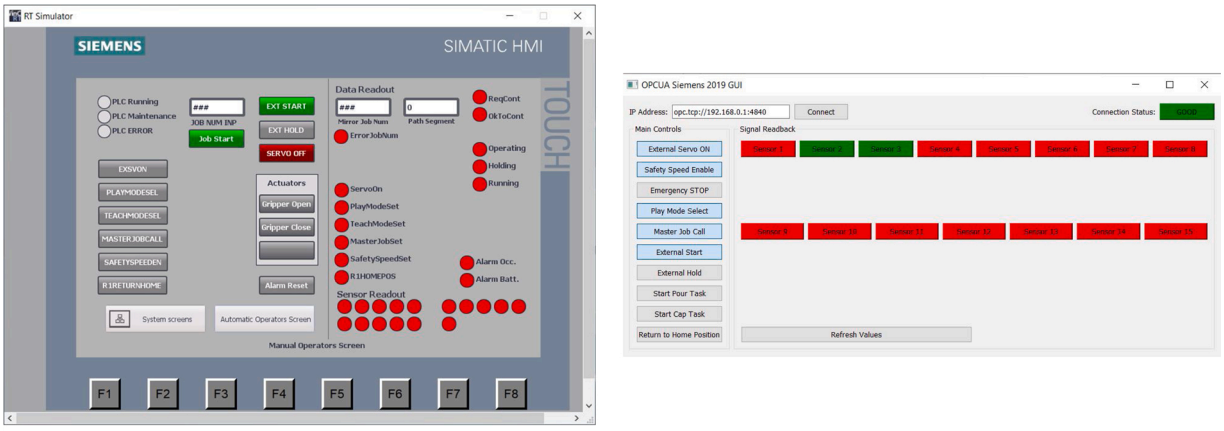
Fig. 10. The virtual cell interface with Digital Engine under Line Simulation Mode.

of production process communicate through live signals on OPC server. As will be discussed in Section 5, the system-embedded manufacturing intelligence needs to be initially derived from real-time monitoring over the virtual cell, and then later apply the identical interface to the physical platform. Thus, the online signals are preferably fed from the *Process Simulate* Line Simulation Mode, where event-based simulations can be controlled by signals from PLC/PLCSIM in a timely manner.

In Table 2, two interfaces from the *Process Simulate* API to the *Digital Engine* are developed, either feeding the status data from Standard Mode or Line Simulation Mode. In *Process Simulate*, depending on whether the simulation is performed under Standard or Line Simulation Mode, interaction with the digital twin operates differently. Under Standard Mode, since plant signals are not evaluated, the *Digital Engine* can only access plant state indications from object-oriented database backends. For example, if object collisions are to be detected, one indicator will be the distance between geometric centers of two objects. Although these plant state indications from the database backend can easily obtain an omniscient perspective of the digital twin, they are not directly sensible or accessible from the physical cell without usage of additional monitoring techniques, which means they are not ideal system indicators that can convey a convergence between the digital and physical twins. Hence, training the algorithms under Standard Mode appears to be less meaningful, as physical counterparts of these real-time knowledges are at best partially observable and cannot be the common basis on which decisions are made for both virtual and physical cells. Moreover, under Standard Simulation Mode, only scheduled operations can be set on simulation player to predict system responses (such as object collisions) by playing the preset simulations. Instead of simulating manually selected operations in a predefined sequence, each event-driven Line Simulation starts only one infinite time-based simulation and activate operations based on events, such as signal transitions triggered by HMI inputs. This is more closed to the environment of real-life manufacturing cells. For the above two reason, the authors choose to focus on developing signal-based smart dynamic scheduler under Line Simulation Mode.

Line Simulation Mode with cyclic event evaluator accurately replicates how the cell interacts with the PLC by cyclic evaluating a sequence of control modules (resemble PLC ladder programs) on a scan frequency. The scan frequency is a parameter in physical PLCs to determine the time intervals between which the sequence of ladder logics is repeatedly executed. Equivalently, in *Process Simulate* Line Simulations, simulation time interval, or update interval, is configured to emulate this behavior of physical PLCs, which is adjustable down to 10 ms, and up to multiple seconds. Output signals are evaluated by control modules only once in every time interval, by which means cell status signals, such as collision indicators, are output with a certain scan frequency. Scan frequencies determined by the choices of time intervals will potentially affect the simulation results, as the sensible cell status, in the form of signals, are only calculated and updated once every scan cycle. Hence, transient rising edge and falling edge in a short signal transition that lasts within a single scan cycle will not be detected by the control modules and PLCs. For sufficiently long-time intervals, this can cause problems with collision detection and motion accuracy. As an example, if an object moves through a transient collision that lasts less than a time interval, then the collision detection engine will not recognize this collision. Sufficiently small time intervals will avoid this problem, at the expense of computing performance as higher simulation resolution requires substantially more processing power. This is why we need to adjust the time intervals for low and high accuracy training cycles described in Fig. 3. Many low accuracy training cycles can be performed very quickly by a high time interval, and then higher resolution trainings can be performed later, with a lower time interval. In this interface, a medial scan time interval of 100 ms is selected to perform the event-based simulations under the assumption that disturbances or signal transitions that last within 100 ms can be ignored. The scan time interval can also be specified as needed either manually in *Process Simulate* or via the *Process Simulate* API by coding.

This streaming program aims to create an agile and repetitive training environment that can easily set/reset simulations between training episodes and enables a two-way data pull/push mechanism over a pipeline between C#. NET framework and python script. The C# pipeline end is pushing plant status from *Process Simulate API* while listening on *Digital Engine* pipeline end for a command based on current plant status. Meanwhile, the *Digital Engine* pipeline end is listening on the pipe until receives current signals and previous simulation results. Trained AI model is able to choose the optimal operation from the action



(a) Local HMI screen **(b)** Remote human intervention screen

Fig. 11. Human intervention interfaces.

list as the command by calculating each action advantage and store the previous simulation results in a buffer memory for future training use. Training AI models will be furtherly discussed in Section 5. As presented in Table 2 and Fig. 10, under Line Simulation Mode, the *Process Simulate API* and its interface with *Digital Engine* (DE) to select actions based on feedback signals are cyclically programmed as following:

1) Virtual PLC initial scan
2) Send robot operations and live signals to DE
3) Wait for DE to store simulation results and train ML models
4) Wait for DE to command the best operation at the current status
5) Receive operation scheduler orders from DE
6) Generate virtual PLC programs and play simulation to update virtual platform
7) If an episode is finished, end the current line simulation to reset to the initial plant status

4.3. Remote human interface via OPC server

Although adaptive intelligence demonstrated its control capability for process that follows a sequence of predefined steps in a fairly controllable environment, human still remain superior at adapting unforeseen changes in complex environment [23]. Supporting cognitive "social human-in-the-loop" is identified as a manufacturing control architecture for future smart factories [45]. Currently at an early stage of manufacturing intelligence, human interventions must be reliably enabled for automation systems considering limited prognostic knowledges of unexpected incidents such as equipment failure, or manufacturing strategy changes ordered from multiple stakeholders. Besides the characteristics such as autonomy, fully automation and proactivity, it was determined by Mittal et al. that context awareness, interoperability and compositionality are more commonly used to classify a system as a Smart Manufacturing system [46]. The integration of heterogeneous and independent systems as a network for a common goal of robustness, performance or cost is also defined as "System--of-system engineering" [47]. Supporting technologies such as cyber-physical systems and Industrial Internet (*IIoT*) are emphasized in this context. In this work, interfaces developed in proposed implementation also concern remote human interventions and monitoring over automated systems following current industrial practices and protocols.

Typically, in automation systems, safety logics and signals such as emergency stops designed to immediately terminate machine operations are not preferably administrated by users. As they are engineered to effectively prevent system damage, one should not allow their remote access and always remain the same settings by original equipment manufacturers. As network delay and potential unreliability raise some potential concerns. For this reason, the remote access applications are urged to exclude any signals and logics related to safety. Such as, the emergency stop must always only rely on a local physical HMI machine, even if it is possible to remotely control e-stop buttons. For a single robot system, external robot signals that operators should be safely allowed to interact with are: External Servo on/off, Safety Speed Enable, Play/Teach Mode Select, Master Job Call, External Start, External Hold, Job Start, Robot Return Home, etc. Hence, a local customized virtual HMI is designed (Fig. 11a) using *SIMATIC* HMI simulator within *TIA portal*. Each of the robot signals is mapped to a memory registered inside OPC Server that can be written to and read from by HMI simulator. To enable a remote-control pathway, OPC server is directly accessed online by extending a python implementation of OPC client *FreeOpcUA* [48]. A PC end GUI is designed as Fig. 11b. Connecting these signals with an online space can serve as an initiative *IIoT* platform application available to different user ends. Further efforts will be made by our subsequent work to provide features such as smart interactions, enhanced cyber security with hierarchical log in and management authorities, and online database maintenance. For instance, concerning an application of automation security, the physical HMI should override changes made by any HMI simulator locally or remotely. Meanwhile, a local HMI should override any changes made by remote HMI.

Upon completion, proposed digital twin system will expand to the integration of autonomous decision-making with secure remote human interventions. By enabling a local scheduler and web-based access to the physical cell, stakeholders will be able to design, simulate, program, and control their own distributed virtual cells, followed by remote integration of safely commissioned programs in physical cells. At a manufacturing intelligence level, integrated programs will need to be controlled and administrated by adaptive scheduling algorithms given real-time plant predictions.

5. Deep Reinforcement Learning augmented by digital twinning of smart manufacturing systems

The proposed smart manufacturing systems seek to provide adaptive, intelligent strategies to manufacturing practitioners in response to environmental changes, policy changes, system prognosis, and optimal solution identification. The success of developing such explorative algorithms highly depends on the trial expenses of applications. Due to minimum trial expenses, artificial intelligence by Deep Reinforcement Learning (DRL) initially succeeded in Atari games through a proper design of reward and deep networks [49]. Efforts are also made in

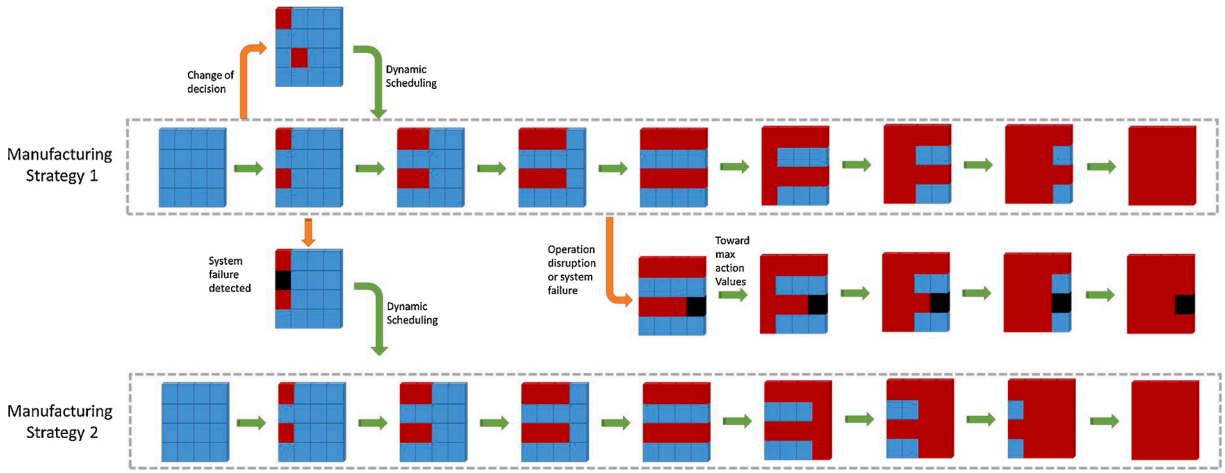


Fig. 12. Using Reinforcement Learning as dynamic scheduler to promptly respond to manufacturing process abnormalities.

robot-sensor feedback control systems [50] and became well-known when well-trained AlphaGo agent defeated the human expert *Go* Player [51].However, for large scale manufacturing processes, training such intelligent scheduler by system commissioning on manufacturing plants could be costly and dangerous, considering the expensive system data collection process and the complexity of a manufacturing plant leading to more demanding designs of the reward-prediction mechanism. This work and its subsequent efforts aim to develop such a smart agent with the augmentation of Digital Twin technologies, including simulation, sensor, industrial control, and real-time communications. Therefore, the role of proposed digital twin in bringing machine learning algorithms into the industrial manufacturing optimization process is significant. Essentially such adaptive learning not only occurs from positive experiences but also from failed experiences, for which reason, a digital twin, as a surrogate system, can ensure the affordability from system-level trials, tunings, and adaptions. More importantly, this digital twin is ready to be deployed into industrial process due to an ideal compatibility of Siemens software and hardware products. With demonstrated fidelity between the digital and physical twins, derived manufacturing intelligence on the digital twin can be easily reused and transferred to learn real-word problems.

This section discusses the implementation of a smart scheduler during manufacturing process, where high-level task scheduling problems are usually addressed. First, the use cases of dynamic scheduler are proposed. Then, applicable algorithms are reviewed in the Deep Reinforcement Learning community. At last, three performance indicators are introduced to interpret the training efficiencies.

5.1. *Artificial intelligence driven operation scheduling*

Deep Reinforcement Learning control on robots can be either low-level or high-level. The low-level controls are deployed over small controllable units such as joint motors, while high-level controls can be the scheduling of lower-level actions. For example, training robot movement to accomplish low-level tasks from scratch with sparse rewards were explored with Scheduled Auxiliary Control [50], where the agent has a set of general auxiliary tasks to be trained to perform a task in the presence of reward signals. However, training robots from scratch such as opening a door or grabbing an object for manufacturing applications appears to be unnecessary and time-consuming. The scheduler developed in this work aims to train robot to accomplish higher-level manufacturing tasks by autonomously planning a sequence of lower-level actions, which thanks to Virtual Commissioning, can be directly programmed inside the constructed virtual cell without intermediate translations of programming languages.

Fig. 12 describes some use cases where this smart scheduler demonstrates its applicability during a continuous real-time optimization process: First, the scheduler is able to accomplish the task by different strategies. Here, a manufacturing strategy is referred to a path towards manufacturing task completion. The scheduler is trained to cognize multiple strategies to achieve the goal. Second, the scheduler shows robustness due to the abnormalities during the process. Case 1: when operators decide to change the current operation, scheduler will alter the subsequent operations accordingly to ensure the successful accomplishment of the task. Case 2: when a system abnormality disables the current operation, scheduler is able to switch to another strategies to ensure the rest of the task is accomplished as concurrent human interventions can be made to fix the system abnormality. Case 3: When an inevitable system failure exists but not causing any cascading effect on the sequential operations, the scheduler is able to identify and skip the failure part while still working towards its original sequence.

5.2. *Related work*

The manufacturing knowledge consists mainly of complex, unprocessed, high-dimensional, and sensory inputs. Although Reinforcement Learning is not widely applied in manufacturing [52], the family of DRL has evolved through the past decade to handle non-static input data. One of the most fundamental DRL algorithms is Deep Q-Networks (DQN) [53]. It is a value-based method where all the accumulated future rewards (action values) are estimated by neural networks based on input observations. The estimation networks are trained either instantly after each interaction with the environment (online) or by buffered experience replay mechanism. A major drawback of natural DQN is that its difficulty to converge caused by dynamic learning targets and instability caused by its sensitivity to hyper parameters. The influence of memory buffer size to approximate convergence of different DQN reward functions was discussed and experimented [54]. While a default value of 10e6 is used in DQN community, the results show that for tabular function representations, a smaller buffer size results in a faster convergence rate as each state-action pair is revisited infinitely many times. Prioritized Experience Replay (PER) [55] is proposed to prioritize the memories in the buffer based on their temporal difference errors, which means that the more “surprising” the predictions are, the more likely it is that they will be sampled to train the networks. PER accelerates DQN convergence at an additional $O(nlogn)$ memory inquiry cost if an advanced data structure, such as sum tree, is implemented as the memory buffer. Proper design of reward function is also critical to the training process. An unbalanced memory buffer dominated by single data distribution will result in a lack of exploration and finally trapped at a local minimum under one strategy. Double DQN [56] is another successful attempt to increase the stability of training process as it

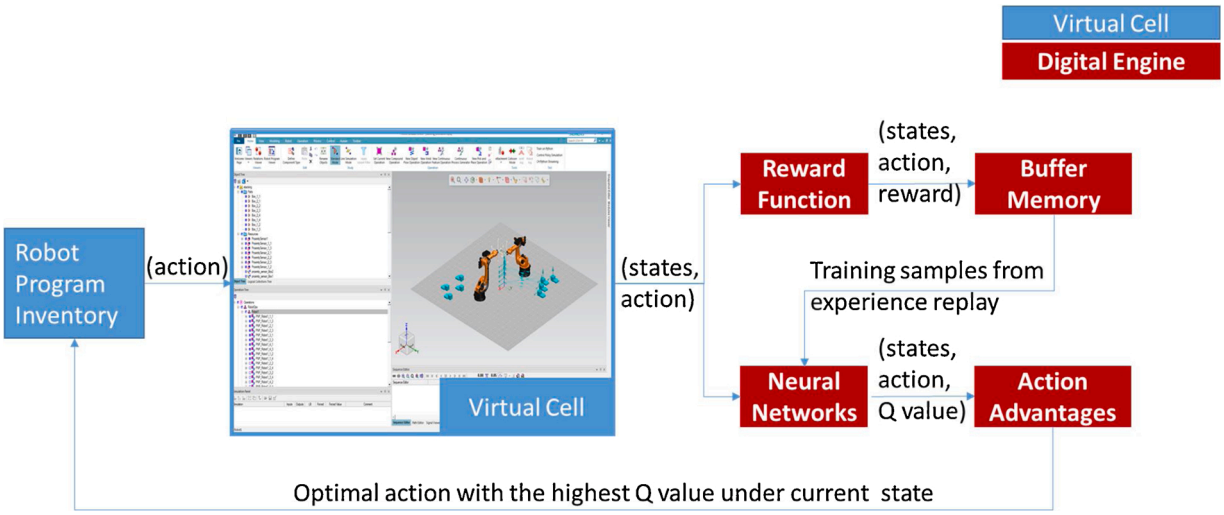


Fig. 13. DQN-based scheduler offline training process.

proposed to train evaluate network based on the decision made by itself. This will overcome the Q value overestimation caused by the difference between target and evaluate networks. Dueling networks proposed a method to separately estimate the value of states and action advantages [57]. This is particularly useful for model-free Deep Reinforcement Learning where learning across actions can be generalized at each state. Furthermore, although value-based methods such as DQN are qualified to handle high-dimensional observation spaces with discrete action spaces, their capabilities are limited when the action spaces are high-dimensional or even continuous, which causes countless combinations of strategies and significant difficulty to explore efficiently. To use DRL in continuous action domains, Deep Deterministic Policy Gradient [58] was designed with an Actor-Critic network system, which is featured as policy gradients.

5.3. Training Deep Reinforcement Learning agents with developed digital twin system

As a fundamental member of the Deep Reinforcement Learning family, the Deep Q-networks (DQN) training process aided by proposed digital twin is described in Fig. 13. The sections represented in blue consist of the software system accommodating the digital twin including *Process Simulate*, the backend database and *Process Simulate* API. The DQN based agent in red, namely the Digital Engine, acquires simulation results from *Process Simulate*, which consists of an action list from robot program inventory and plant sensor signals before and after simulation for the operation chosen by the agent.

Once the descriptive data flow into the Digital Engine system, a reward r is calculated by user-defined a reward function and concatenate with the currently simulated action a, signal state before the simulation s and after the simulation s_- to form a memory transaction (s, a, r, s_-). If not implementing online training, this transaction is stored in a memory buffer as a future training sample. Through DQN, the estimated Q-values of all state-action pair (s, a) can be calculated as the prediction for cumulative future rewards if the current action a is taken, by which numerically representing the action advantage of the action a at the current state s. Target Q-values, as the training label for the estimated Q-values, are the summation of reward r in current transaction and the Q-values estimated at the next state s_-. While the estimated Q-values are directly obtained by Q-networks given the current state s. The backpropagation algorithm trains the network by minimizing the prediction error between estimated Q-values and target Q-values. Implementing memory replay mechanism, samples are taken randomly from the memory buffer to train the networks by back propagation after the network estimates Q-values and calculates target Q-values. The DQN training process initiates with randomly generated Q network values and take completely chaotic actions, which is called random exploration, to create buffer memory with transactions. As the training iterations (or episodes) proceed, probability of exploration is programmed to gradually decrease, and the agent is expected to make its own decisions as the backpropagation is tasked to minimize the Q-value prediction errors by tuning the weights and biases between each neuron layers in the Q-networks. Despite moving training targets sampled from the constantly updated buffer memories causes some convergency difficulties, it has been proved that given enough iterations and consistent feedback rewards, DQN eventually converges and able to stably make optimal decisions drive by the highest Q-values.

5.4. Performance indicators

One difficulty of implementing DRL on multiple applications is that the prior work is hardly duplicable on environment that are not experimented on before. The success in one platform remain only limited empirical values in completely new applications. For example, Natural DQN as an algorithm with multiple combination of hyper parameters, researchers will need to find a relative optimal behavior from at least parameters as reward function, action numbers and dimensions, feature numbers and dimension, network architectures, learning rates, reward decay rate, greedy policy behavior, target net update frequency, buffer memory size, training batch size, etc. If more advanced algorithms were included to improve data efficiency, the parameter list extends even longer, where each of the parameters can be sensitive to the training process. This requires expert experiences from data science. Therefore, it is needed to find general performance indicators that evaluate selected parameters to interpret the training procedures for different manufacturing applications. For instance, low-level robot tasks [50] uses the growth of intermediate Performance Indicator such as cumulative rewards to evaluate the learning speed, while the learning process of Atari games is directly recorded by each game scores earned [49]. The following performance indicators are proposed both from intuitive process evaluations that can be observed by manufacturing practitioners as well as indicator of algorithm performance that can be interpreted by data scientists:

Performance Indicator 1: The number of actions taken to complete the manufacturing task in each episode. Given the fact that the shortest route to finish a manufacturing task takes n scheduled robot operations. However, due to constantly introduced interruptions and random exploration mechanism during training, the agent initially takes

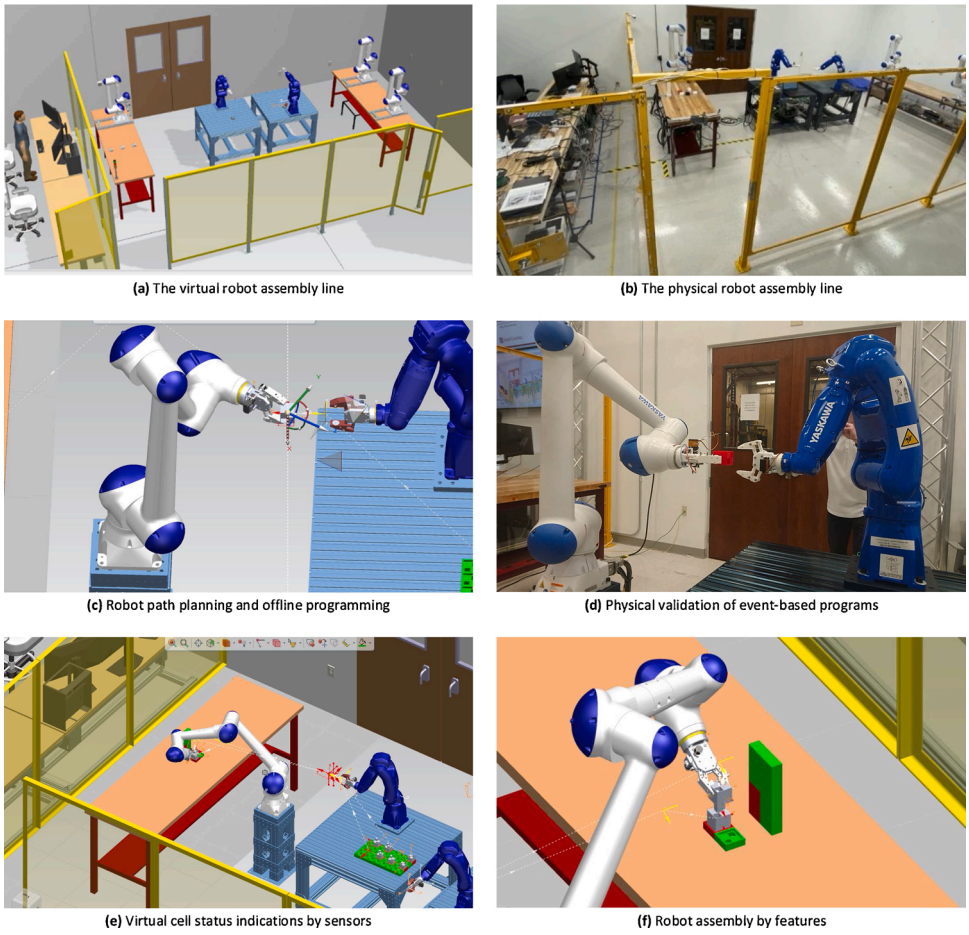


Fig. 14. Low-level offline robot programming workflow and the physical verifications.

a lot more than n steps to finish a task episode in most of the training episodes. As the agent trained to be smarter, the model is expected to predict action values, select the action with highest Q-value at the current state and generally decrease the number of wrong moves. Given enough training iterations, steps per task episode converges to the minimum number needed (n steps) despite the constantly introduced disturbances.

Performance Indicator 2: The total rewards earned during one task episode. Depending on Performance Indicator 1, Performance Indicator 2 is upper bounded by the maximum rewards can be earned when a global optimal strategy is reached with a minimum of n moves without any disturbance. As training continues, at each step t, the agent is expected to make an optimal move a_t^* following the well-trained optimal policy π^* to earn maximum rewards $r_{(s_t,a_t^*,s_{t+1})}$. The optimal policy π^* maps the current state s_t to the optimal action a_t^*. Hence, Performance Indicator 2 will converge to the maximum accumulated rewards by a discount factor Υ despite all the possible disturbances. This indicator can be formed by a Bellman optimality problem for estimating a max expected rewards accumulation $V_{(s_0)}^{\pi^*}$ (namely the value function) from the max Q-value given an optimal policy π^* and the initial state s_0 of a Markov decision process:

$$V_{(s_0)}^{\pi^*} = \max_{a_0 \in \pi_{(s_0)}^*} Q_{(s_0,a_0)}^* = E_{\pi^*}\left\{ \sum_{t=0}^{n} \Upsilon^t r_{(s_t,a_t,s_{t+1})} \middle| a_t \in \pi_{(s_t)}^* \right\} \tag{1}$$

Where $\Upsilon = 1$ for this performance indicator to directly record the earned reward summations without the effects from stepwise decaying of reward accumulation; $a_t \in \pi_{(s_t)}^*$ as the optimal policy mapping is followed.

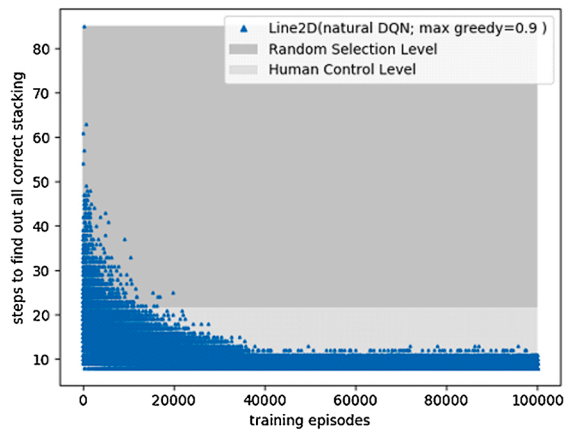
Performance Indicator 3: Q-value prediction error/loss. Performance Indicator 3 is an intermediate evaluator to indicate how good the implemented DQN as the optimal policy π^* is trained and how accurate the Q-value predictions are. Since the buffer memory keeps updated with new transactions, sampled target Q-values keep changing accordingly. Hence Performance Indicator 3 is not constantly decreasing since the agent given new state that is never explored before. In these cases, its prediction error suddenly increases and will take some time to decrease before the new transaction stored, sampled and trained by Q-networks. However, sufficiently trained agent should have explored all the possible states (in many applications this is not even possible) and Performance Indicator 3 will stably remain as a small positive value despite all the constant disturbances. Given an intermediate state s_t, an optimal policy π^* derived from the optimal Q-value Q^* evaluated by Q-networks, and the reward r earned by scheduling an optimal action a_t from the policy π^*, the prediction error/loss L at any step t is defined, and then used as an objective function in backpropagation:

$$L = \left(Q_{target} - Q_{prediction}\right)^2 = \left(r_{(s_t,a_t,s_{t+1})} + \Upsilon Q_{(s_{t+1},a_{t+1})}^* - Q_{(s_t,a_t)}^*\right)^2 \tag{2}$$

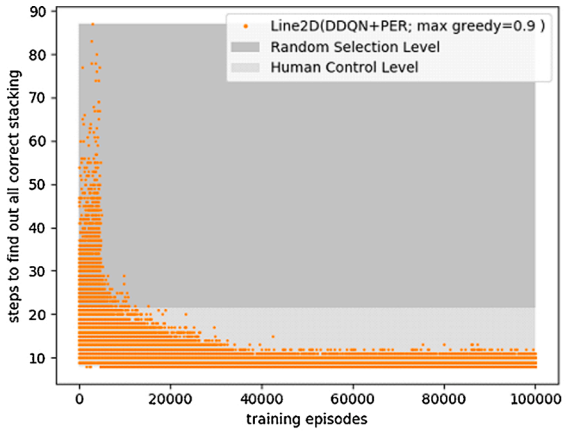
Where $a_{t+1} \in \pi_{(s_{t+1})}^*$, $a_t \in \pi_{(s_t)}^*$ as the optimal policy mapping is followed at all the steps requiring decision-makings by Q-value estimations.

6. Case study

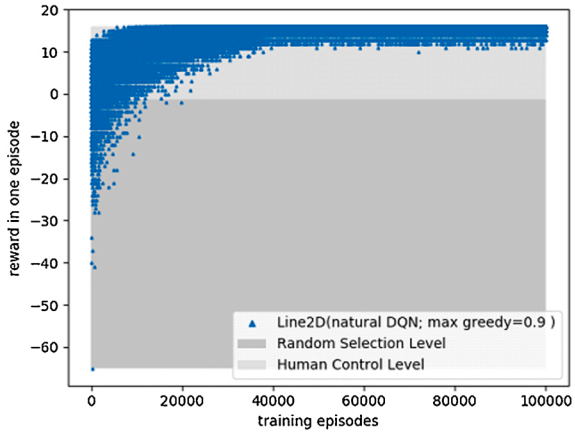
In this section, a case study is shown to train robots virtually to accomplish adaptive stacking tasks sequence collaboratively with simulated proximity sensor signals. The trained scheduler is expected to drive the virtual PLC to autonomously select robot program sequences



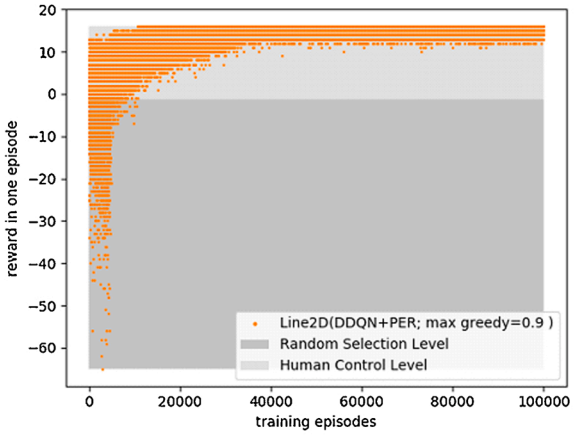
(a) Natural DQN by Performance Indicator 1



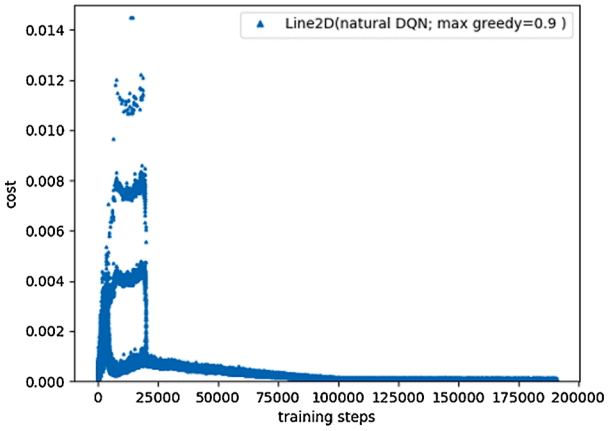
(b) DDQN with PER by Performance Indicator 1



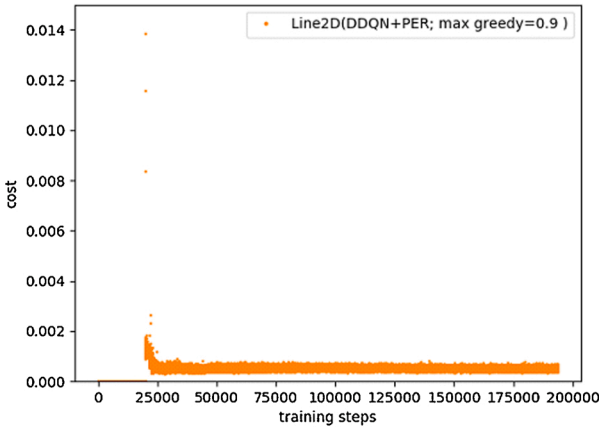
(c) Natural DQN by Performance Indicator 2



(d) DDQN with PER by Performance Indicator 2



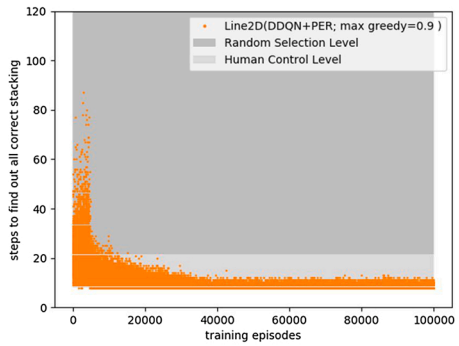
(e) Natural DQN by Performance Indicator 3



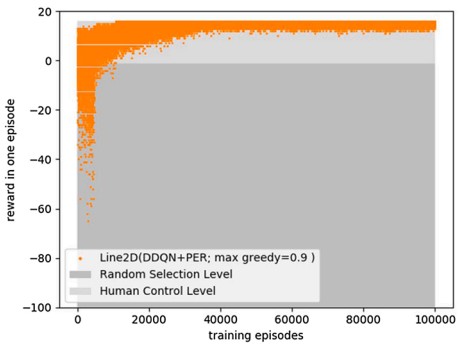
(f) DDQN with PER by Performance Indicator 3

Fig. 15. Learning Curves by natural DQN (blue) and Improved DQN (orange).

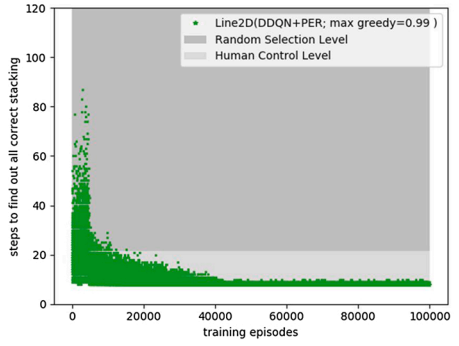
based on virtual signal feedbacks, including proximity sensor signals and robot signals. First, the virtual cell is constructed and used to generate alternative robot programs for the scheduler to select from. Virtual signals connected to the virtual cell are scanned through virtual PLC control modules under *Process Simulate* line simulations. Then, these signals are exported to the Digital Engine as process indicator to train the scheduler through a properly defined reward function. The primary training cycles proposed in Fig. 3 use the *Process Simulate* API (described in Section 4.2) considering a minimum involvement of the hardware and OPC server is preferred for the safety and power consumption at the primary training cycles. At last, some preliminary training results using different algorithms and exploration strategies are presented. As a



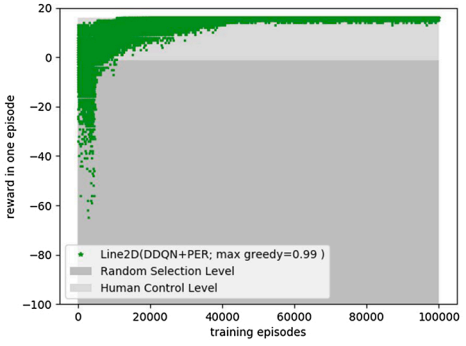
(a) DDQN+PER with ε-greedy=0.9 by Performance Indicator 1



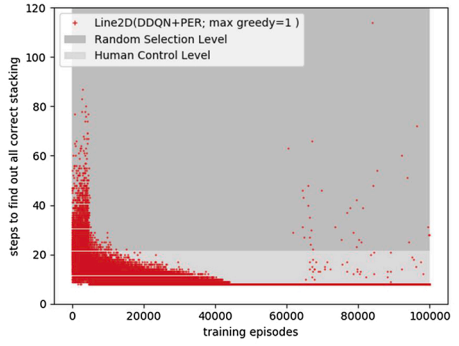
(b) DDQN+PER with ε-greedy=0.9 by Performance Indicator 2



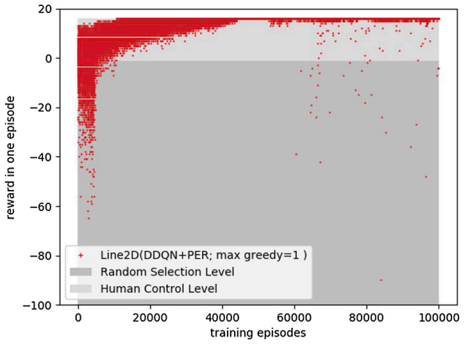
(c) DDQN+PER with ε-greedy=0.99 by Performance Indicator 1



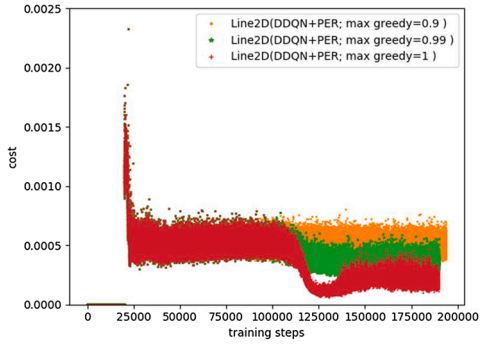
(d) DDQN+PER with ε-greedy=0.99 by Performance Indicator 2



(e) DDQN+PER with ε-greedy=1 by Performance Indicator 1



(f) DDQN+PER with ε-greedy=1 by Performance Indicator 2



(g) A comparsion of DDQN+PER with ε-greedy=0.9, 0.99, 1 by Performance Indicator 3

Fig. 16. Learning Curves by different ε-greedy = 0.9 (orange), 0.99 (green), 1 (red).

result, the *Digital Engine* primary training cycles are performed on Digital Twin system constructed by the methodologies introduced in Section 3 and 4.

6.1. Low-level offline robot programming using the digital twin

The procedures of low-level robot offline programming for a conceptual robot assembly line are presented in Fig. 14. The virtual counterpart (Fig. 14a) of the physical cell (Fig. 14b) is firstly configured and calibrated with the methodology described in Section 3.2. Then robot motions and paths (Fig. 14c) are programmed as described in Section 3.3. During this step, collision-free robot movements and complete path reachability can be verified within the virtual environment.

Then, by creating an event-based simulation, the logics enabling two robots to collaboratively pass on the manufactured parts are programmed for this case study. This event-based logics for two robot collaboration are programmed following these steps: **(1)** the HC10 (robot in white and blue) is waiting for the GP8 (robot in solid blue) to finish the robot programs where the part is delivered to the interchange location ***AND*** the gripper of GP8 at the CLOSE pose indicating part gripped by GP8. **(2)** Only if both signals are true, HC10 sets its gripper to CLOSE to grip the part. **(3)** Then HC10 sends a signal to tell GP8 to set its gripper to OPEN to let go the part and move out of the way. **(4)** When the GP8 moves to its working pose ***AND*** HC10 gripper is at CLOSE pose, HC10 moves forward to its next program to place the gripped part. This event-based programming can take a lot of trials and errors in the physical cell, however, by the aid of proposed system-level digital twin, the programming, validation and debugging processes are significantly less trivial. These event-based programs are tested in the physical cell to validate the success of robot collaboration tasks (Fig. 14d).

Assume the assembly task is to pick up parts from 8 different part pools and assemble at 8 locations, at least 64 different combinations of aforementioned event-based robot programs are needed in robot program inventory to provide enough assembly strategies for adaptive system intelligence. System-level of decision-making intelligence also requires real-time sensory feedbacks before the signal based logical evaluations in PLCs. In this case study, the proximity sensors (signal indicated by green or red in Fig. 14e) are set up at each location to indicate the pick-and-place operation status and provide the feedback to the Digital Engine. By this setup, descriptive cell states are available to us in the form of live sensor signals that are consistently accessible both virtually and physically. With the proximities of parts to the desired locations, a reward function is shaped such that real-time inputs from readable sensor signals are used to evaluate all the action advantages towards the accomplishment of the assembly task (Fig. 14f).

6.2. Training dynamic scheduler for digital engine

Since our low-level robot actions can be simplified as pick and place operations, one can correlate the success of an operation with the transition of the virtual proximity sensor signals. The identical reasoning can later be duplicated in physical cells with real proximity sensors. Specifically, if both proximity sensors at pick and place locations value changes, it indicates the current action are performed successfully without any interruption. If only the sensor at pick location changes, it is likely because the action started but did not successfully finish or got interrupted. If neither of these two signals changes, it is potentially caused by the part not being ready or safety alerts resulting in operation initiation failures. The safety alerts can be predictively generated from detected collisions or unreachable robot locations during the simulation. With the above reasoning, the following reward function for a state transition, that is from current state s taking an action a to next state s_{-}, is defined as:

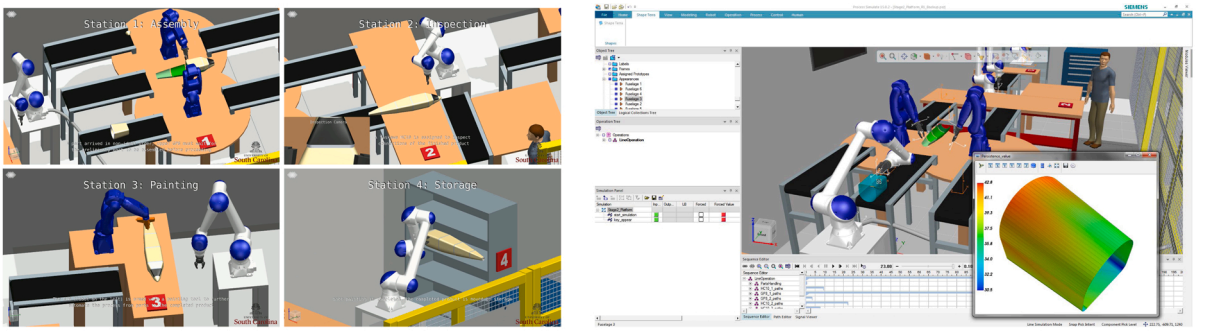
$$r_{(s,a,s_{-})} = \begin{cases} +2;\ \textit{if action a simulation completes without interruption} \\ +1;\ \textit{if action a simulation results in interruption or abortion} \\ -1;\ \textit{if action a simulation results in safety alert or bad decision} \end{cases}$$

The goal of Deep Reinforcement Learng algorithms (DRL) is to outperform human decision-making on different applications by an autonomous data-driven approach to stochastic process control problems. In particular, Markov decision processes without explicit state transition probabilities are usually solved by Reinforcement Learning. Therefore, during a training process, decision-making performance can be divided by a probabilistic region of random selection level and a human control level, which can be modelled by state transition under partly random and partly under the control of a decision maker in a Markov Decision Process. Decision intelligence is expected to be achieved at the convergence state of optimal human control performance level. In this application, we are using three Performance Indicators from Section 4.3 and these two control levels, as differently shaded in Figs. 15 and 16, to reflect DRL training states.

Two algorithms are implemented in this task, Natural Deep Q-Learning [53] and Double Deep Q-Learning [56]with a Prioritized Experience Replay (PER) [55]. The implementation of PER starts learning later than natural DQN did since it only starts training when memory buffer is filled up. This is due to the fact that it implements a sum-tree based prioritized search algorithm. It starts with filling the buffer memory with random transitions and then constructs a sum tree with all the temporal difference error of the transitions, which is used to evaluate transaction training priorities. When memory is full, the agent samples from it with the principle of the transitions with higher priority (means more prediction error) get more chances to be sampled and trained. Meanwhile, double DQN helps to solve the overestimation of Q values by Natural DQN caused by a greedy policy. Double DQN estimates the Q value of next state as a result of the same action that will be taken at current state, instead of directly select maximum Q value at the next state. Double DQN has been proved to accelerate DQN convergency by improving the exploration mechanism while PER improves the training efficiency by smarter data exploitations.

In this case study of the robot assembly cell, the shortest route to finish the task, namely Performance Indicator 1, takes 8 robot operations. The maximum accumulated reward can be earned during one task episode, namely Performance Indicator 2, is +16 when a global optimal policy is reached with 8 operations with each operation valued a reward of +2. Given adequate training, despite any process disturbances introduced from random exploration, the agent is expected to identify the shortest route to task accomplishment and make every optimal move with rewards of +2. Hence in one episode, the Performance Indicator 1 will converge to a minimum of 8 steps and Performance Indicator 2 will converge to the maximum of accumulated rewards (+16 points). This convergence under constant disturbances is shown in Fig. 15(a–d).

In Fig. 15, comparing (a) with (b), or (c) with (d), PER improved Double DQN reaches the human control level faster with less variance through the training curves at the expense of an initial stage where randomly selected actions are generated to fill the buffer. Because of this stage, the actual training for PER gets delayed compared with natural DQN. Comparing (f) with (e), one can find that once start training (as the prediction error start to be recorded at the 19960^{th} training step when the memory buffer gets fulfilled), the prediction errors for PER improved Double DQN reduce constantly and stabilize at an acceptable range much faster than natural DQN. This is because PER start training from a fulfilled memory buffer with all kinds of transitions to be sampled from. Therefore, it can be concluded that with a balanced dataset, the networks are better trained with a wide range of positive and negative memories. Hence, it needs to be realized that the distribution and abundance of data samples fundamentally affect the training of heuristic algorithms for data-driven decision makers.



(a) Digital twin of a dynamic production line **(b)** Digital twin interfacing with a reference model

Fig. 17. Stage 2 Platform in proposal. (a) Digital Twin of the production line in Process Simulate. (b) Shape Terra as a reference model to interface with the smart manufacturing scheduler.

6.3. Trade-off between exploration and exploitation in Reinforcement Learning

Within the community of Deep Reinforcement Learnings, there has been a specific challenge as the trade-off between data explorations and exploitation [52]. It underlines the need to control how much one agent is willing to compromise exploration of new experiences for faster training convergency. This can also be interpreted as a trade-off between possibilities of finding a global optimal strategy and stabilizing at a local optimal strategy. Smaller exploration rate, and hence greedier approach, helps to find a local optimal fast but are not able to adapt well when faced with new states as the agent will learn adequately under current strategy but reluctant to explore for potentially better ones.

The approach to this trade-off situation is dependent on how complex the learned environment is, that is how hard to stabilize at a local optimum or explore towards global optima. Besides, it also depends on the specific task goal we are trying to accomplish. For example, for different Atari games, if the agent is trying to avoid dangerous situations and survive as it can, less explorations while put more penalties on the dangerous memory down the current strategy will be possibly favorable. However, if the agent is trying to find the best path to finish a task and earn as much reward as possible, more explorations with different memories will be preferred to training the agent towards global optimal solution. During manufacturing process, we penalize attempted actions that result in strategy faults such as part unavailability at pick up location, placement location unavailability, object collisions, interruption, or failure during the action. These penalized actions are more undesired than a strategy that is less optimal. That is to say, when the learned environment gets too complicated to converge at a global optimum, the agent can safely settle for a stable local optimal strategy instead of keep exploring for better strategies considering damage control.

In this work, different sets of exploration rates (ε-greedy) were experimented in the training process as in Fig. 16. ε-greedy stands for the chances of randomly chosen action out of total actions taken during training. For example, when ε-greedy = 0.9, on average, in every 10 steps, the agent takes 1 random action to explore the better strategy and the rest 9 actions are taken by the decision made by agent. As shown in the figure, comparing ε-greedy = 0.9, 0.99 and 1, less explorations help to stabilize the decisions when converge, as less variations are introduced. When ε-greedy = 1, which means the agent does not explore at all when converge, the agent decisions are deterministic under currently explored strategies. However, when the agent is faced with new states that has never been explored, when it will have a hard time recognizing the new states and unable to robustly adapt the changes. This is demonstrated by the scattered points after convergency in Fig. 16e and f of the learning curve as well as the divergent prediction errors (red) in the Fig. 16g. It can be interpreted as an over-fitting problem caused by unbalanced data sample distribution under inadequate data exploration mechanism.

7. Conclusion and future work

In this work, a novel approach is proposed to utilize digital twin simulation and communication technologies to create virtual counterparts of robot manufacturing systems, on which the intelligent scheduler based on Deep Reinforcement Learning can be safely trained. Compared to the previous attempt to introduce Reinforcement Learning into work cell scheduling [59], proposed system-level digital twinning can be expanded to complex manufacturing systems with deep neural networks to overcome the challenges due to large state and action spaces. For specific manufacturing processes, attempts [11,17] has been made to apply Deep Learning and Reinforcement Learning with the physical visual inputs. It was proposed [17] that additional work in the industrial setting is needed to demonstrate a fully integrated framework for industrial artificial intelligence, which is fundamentally discussed by this work. To that end, system embedding of artificial intelligence into commonly used industrial robots, PLC function blocks, and event-driven controls at run-time [9] is realized by this work. In addition, successful system integration of the digital twin facilitates a general architecture of semantic-aware M2M communications [8] by adding subsystems as *Digital Engine* and communication layers. Moreover, proposed intelligent digital twin has the fidelity to solve dynamic scheduling problems of highly complex flexible manufacturing models with stochastic events [11].

The contribution of this work is primarily in the following aspects: **(1)** High-fidelity Virtual Commissioning platforms created by *Siemens Tecnomatix Process Simulate* are used as virtual environments to accommodate the digital twin implementation, where system components are defined, simulated, and synchronized with live signals. Moreover, advanced tools for event-based simulation, collision detection, robot reachability test, and robot configurations by reverse kinematics are utilized to commission robot programs through this digital twin. After this offline programming process, generated robot programs can be directly transferred to physical robot systems without intermediate translations. **(2)** After construction of the virtual environment, system communications are implemented on both virtual and physical pathways. “Software-in-the-loop” and “Hardware-in-the-loop” testing methods are discussed to be the baseline of virtual commissioning control loops depending on either the virtual cell is controlled by virtual or physical controller. Moreover, intelligent scheduler communication pathway with virtual cell to rapidly perform repetitive offline training cycles is implemented by the *Process Simulate* API. Then, an application to enable *IIoT* for remote human intervention via customized OPC clients is presented. **(3)** Manufacturing Intelligence algorithm framed by Deep Reinforcement Learning is trained on the constructed digital twin. Both natural Deep Q-Learning and its improved version by Prioritized

Experience Replay and Double Q Network are implemented to improve data efficiency. As a result, a robust dynamic scheduler fed by live signals from either physical cell or its digital twin is trained as Deep Reinforcement Networks, which are reusable and transferable for other specific learning tasks. The method to embed proposed *Digital Engine* that supports scheduling in an industrial virtual commissioning platform greatly augments the power of data analytics by interfacing with industrial simulation and automation software. For which reason, this data-driven manufacturing intelligence is also ready to be deployed to specific industrial applications as a use case of Smart Manufacturing implementation.

The future work will focus on applying this methodology on more diverse manufacturing tasks and material flows in Stage 2 platform (Fig. 17a), including collaborative assembly jobs, visual inspection, optimized rework, and continuous movement tasks. Dynamic feedback signals and high-dimensional manufacturing data are automatically fed into the digital twin to train *Digital Engine*, such as analog plant signals, product part CAD feature information, and machine vision inputs. Sensor signals from force sensors, motor voltages, robot monitors, thermal cameras, and environmental condition monitoring sensors will be used to connect to such digital twin systems so that more accurate real-time plant descriptions can be collected digitally. These manufacturing knowledges can also be shared between stakeholders with the proposed *IIoT* platform for even smarter decision-makings. Meanwhile, the authors will expand the research activities to develop a high-fidelity reference model and improve predictive capabilities of the digital twin. For example, Harik *et al.* developed an algorithm to automatically extract and recognize 3D manufacturing features from product CAD models, named *Shape Terra* [60]. The authors will base the future work on the theoretical background of *Shape Terra* to develop the *Digital Engine* for high resolution training cycles, as the reference model interfacing with *Process Simulate* API (depicted in Section 4.2). Such attempts have the potential to enhance the digital twin approach towards fully automated smart manufacturing systems, delivering manufacturing intelligence driven by data from systems, processes and, even more important, product designs. Initial implementation is presented in Fig. 17b.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The presented research project is funded by the South Carolina Research Authority (SCRA), United States, with grant no. 10009353, 10009367. The authors are grateful for the software and hardware donations received from industrial partners Siemens USA and Yaskawa Motoman Robotics. The authors are also grateful for the industrial collaborations with Nephron Pharmaceuticals Corporation and the support of Lou Kennedy.

References

[1] Alkan B, Harrison R. A virtual engineering based approach to verify structural complexity of component-based automation systems in early design phase. J Manuf Syst 2019;53:18–31.
[2] Kucukoglu I, Atici-Ulusu H, Gunduz T, Tokcalar O. Application of the artificial neural network method to detect defective assembling processes by using a wearable technology. J Manuf Syst 2018;49:163–71.
[3] Lai Z-H, Tao W, Leu MC, Yin Z. Smart augmented reality instructional system for mechanical assembly towards worker-centered intelligent manufacturing. J Manuf Syst 2020;55:69–81.
[4] Liu C, Vengayil H, Lu Y, Xu X. A cyber-physical machine tools platform using OPC UA and MTConnect. J Manuf Syst 2019;51:61–74.
[5] Moghaddam M, Cadavid MN, Kenley CR, Deshmukh AV. Reference architectures for smart manufacturing: a critical review. J Manuf Syst 2018;49:215–25.
[6] Tuptuk N, Hailes S. Security of smart manufacturing systems. J Manuf Syst 2018;47:93–106.
[7] Park H-S, Tran N-H. An autonomous manufacturing system based on swarm of cognitive agents. J Manuf Syst 2012;31(no. 3):337–48.
[8] Lu Y, Asghar MR. Semantic communications between distributed cyber-physical systems towards collaborative automation for smart manufacturing. J Manuf Syst 2020;55:348–59.
[9] Adamson G, Wang L, Moore P. Feature-based control and information framework for adaptive and distributed manufacturing in cyber physical systems. J Manuf Syst 2017;43:305–15.
[10] Ren L, Sun Y, Cui J, Zhang L. Bearing remaining useful life prediction based on deep autoencoder and deep neural networks. J Manuf Syst 2018;48:71–7.
[11] Hu L, Liu Z, Hu W, Wang Y, Tan J, Wu F. Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network. J Manuf Syst 2020;55:1–14.
[12] Yun JP, Shin WC, Koo G, Kim MS, Lee C, Lee SJ. Automated defect inspection system for metal surfaces based on deep learning and data augmentation. J Manuf Syst 2020;55:317–24.
[13] Chen L, Xu G, Zhang S, Yan W, Wu Q. Health indicator construction of machinery based on end-to-end trainable convolution recurrent neural networks. J Manuf Syst 2020;54:1–11.
[14] Tao F, Qi Q, Liu A, Kusiak A. Data-driven smart manufacturing. J Manuf Syst 2018;48:157–69.
[15] Kusiak A. Smart manufacturing must embrace big data. Nature 2017;544(no. 7648):23–5.
[16] Uhlemann TH-J, Schock C, Lehmann C, Freiberger S, Steinhilper R. The digital twin: demonstrating the potential of real time data acquisition in production systems. Procedia Manuf 2017;9:113–20.
[17] Günther J, Pilarski PM, Helfrich G, Shen H, Diepold K. Intelligent laser welding through representation, prediction, and control learning: an architecture with deep neural networks and reinforcement learning. Mechatronics 2016;34:1–11.
[18] Tao F, Cheng J, Qi Q, Zhang M, Zhang H, Sui F. Digital twin-driven product design, manufacturing and service with big data. Int J Adv Manuf Technol 2018;94:3563–76.
[19] Lee CG, Park SC. Survey on the virtual commissioning of manufacturing systems. J Comput Des Eng 2014;1(no. 3):213–22.
[20] Hoffmann P, Schumann R, Maksoud TMA, Premier GC. Virtual commissioning of manufacturing systems a review and new approaches for simplification. 24th European Conference on Modelling and Simulation 2010.
[21] Zheng C, Qin X, Eynard B, Bai J, Li J, Zhang Y. SME-oriented flexible design approach for robotic manufacturing systems. J Manuf Syst 2019;53:62–74.
[22] Garetti M, Macchi M, Pozzetti A, Fumagalli L, Negri E. Synchro-push: a new production control paradigm. 21st Summer School Francesco Turco 2016 2016:150–5.
[23] Büchel B, D. Floreano and Organisation. Tesla's problem: overestimating automation, underestimating human. The conversation. 2019 [Online]. Available: http://theconversation.com/teslas-problem-overestimating-automation-underestimating-humans-95388. [Accessed 26 October 2019].
[24] Zhuang C, Liu J, Xiong H. Digital twin-based smart production management and control framework for the complex product assembly shop-floor. Int J Adv Manuf Technol 2018;96:1149–63.
[25] Negri E, Fumagalli L, Macchi M. A review of the roles of digital twin in CPS-Based production systems. Procedia Manuf 2017;11:939–48.
[26] Xia K, Sacco C, Kirkpatrick M, Harik R, Bayoumi A-M. Virtual comissioning of manufacturing system intelligent control. SAMPE 2019 - Charlotte, NC 2019.
[27] Lee J, Lapira E, Bagheri B, Kao H-a. Recent advances and trends in predictive manufacturing systems in big data environment. Manuf Lett 2013;1(no. 1):38–41.
[28] Schleich B, Anwer N, Mathieu L, Wartzack S. Skin Model Shapes: a new paradigm shift for geometric variations modelling in mechanical engineering. Comput Aided Des 2014;50:1–15.
[29] Schleich B, Anwer N, Mathieu L, Wartzack S. Shaping the digital twin for design and production engineering. Cirp Ann Manuf Technol 2017;66(no. 1):141–4.
[30] Knapp GL, Mukherjee T, Zuback JS, Wei HL, Palmer T, De A, et al. Building blocks for a digital twin of additive manufacturing. Acta Mater 2017;135:390–9.
[31] Konstantinov S, Ahmad M, Ananthanarayan K, Harrison R. The cyber-physical E-machine manufacturing system: virtual engineering for complete lifecycle support. Procedia CIRP 2017;63:119–24.
[32] Guerrero LV, López VV, Mejía JE. Virtual commissioning with process simulation (Tecnomatix). Comput Aided Des Appl 2014;11.
[33] Fält J, Halmsjö J. Emulation of a production cell - developing a virtual commissioning model in a concurrent environment. 2016.
[34] Harrison R, Vera D, Ahmad B. Engineering methods and tools for cyber–Physical automation systems. Proc IEEE 2016;104(no. 5):973–85.
[35] Uhlemann TH-J, Lehmann C, Steinhilper R. The digital twin: realizing the cyber-physical production system for industry 4.0. Procedia CIRP 2017;61:335–40.
[36] Tao F, Zhang H, Liu A, Nee AYC. Digital twin in industry: state-of-the-Art. IEEE Trans Ind Inform 2019;15(no. 4):2405–15.
[37] Schroeder GN, Steinmetz C, Pereira CE, Espindola DB. Digital twin data modeling with AutomationML and a communication methodology for data exchange. IFAC-PapersOnLine 2016;49(no. 30):12–7.
[38] Haag S, Anderl R. Digital twin – proof of concept. Manuf Lett 2018;15:64–6.
[39] Tao F, Zhang M. Digital twin shop-floor: a new shop-floor paradigm towards smart manufacturing. IEEE Access 2017;5:20418–27.
[40] Kritzinger W, Karner M, Traar G, Henjes J, Sihn W. Digital Twin in manufacturing: a categorical literature review and classification. IFAC-PapersOnLine 2018;51(no. 11):1016–22.

[41] Lee JH, Shin J, Realff MJ. Machine learning: overview of the recent progresses and implications for the process systems engineering field. Comput Chem Eng 2017; 114:111–21.
[42] Chen W, Liu H, Qi E. Discrete event-driven model predictive control for real-time work-in-process optimization in serial production systems. J Manuf Syst 2020;55: 132–42.
[43] Tirinzoni A, Sessa A, Pirotta M, Restelli M. Importance weighted transfer of samples in reinforcement learning. ICML 2018: Thirty-Fifth International Conference on Machine Learning 2018.
[44] Siemens PLM Software. Process Simulate - Manufacturing process verification in powerful 3D environment [Online]. Available:. 2011. https://www.plm.automation.siemens.com/en_gb/Images/7457_tcm642-80351.pdf.
[45] Cimini C, Pirola F, Pinto R, Cavalieri S. A human-in-the-loop manufacturing control architecture for the next generation of production systems. J Manuf Syst 2020;54: 258–71.
[46] Mittal S, Khan MA, Romero D, Wuest T. Smart manufacturing: characteristics, technologies and enabling factors. Proc Inst Mech Eng Part B J Eng Manuf 2019; 233(no. 5):1342–61.
[47] Jamshidi M. System of systems engineering - New challenges for the 21st century. IEEE Aerosp Electron Syst Mag 2008;23(no. 5):4–19.
[48] FreeOpcUa: open source C++ and Python opc-ua server and client libraries and tools. 2020 [Online]. Available: http://freeopcua.github.io/.
[49] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D. Human-level control through deep reinforcement learning. Nature 2015;518(no. 7540):529–33.
[50] Riedmiller M, Hafner R, Lampe T, Neunert M, Degrave J, Wiele TVd, Mnih V, Heess N, Springenberg JT. Learning by playing - solving sparse reward tasks from scratch. arXiv preprint arXiv:1802.10567. 2018.
[51] Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, Chen Y, Lillicrap T, Hui F, Sifre L, Driessche Gvd, Graepel T, Hassabis D. Mastering the game of Go without human knowledge. Nature 2017;550(no. 7676):354–9.
[52] Wuest T, Weimer D, Irgens C, Thoben K-D. Machine learning in manufacturing: advantages, challenges, and applications. Prod Manuf Res 2016;4(no. 1):23–45.
[53] Silver DH, Graves A, Antonoglou I, Riedmiller M, Mnih V, Wierstra D, et al. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602. 2013.
[54] Zhang S, Sutton RS. A deeper look at experience replay. arXiv preprint arXiv: 1712.01275. 2017.
[55] Schaul T, Quan J, Antonoglou I, Silver D. Prioritized experience replay. ICLR 2016 : International Conference on Learning Representations 2016 2016.
[56] Hasselt Hv, Guez A, Silver D. Deep reinforcement learning with double Q-learning. arXiv preprint arXiv:1509.06461. 2015.
[57] Wang Z, Schaul T, Hessel M, Hasselt Hv, Lanctot M, Freitas Nd. Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581. 2015.
[58] Silver D, Lever G, Heess N, Degris T, Wierstra D, Riedmiller M. Deterministic policy gradient algorithms. Proceedings of The 31st International Conference on Machine Learning 2014.
[59] Ou X, Chang Q, Chakraborty N. Simulation study on reward function of reinforcement learning in gantry work cell scheduling. J Manuf Syst 2019;50:1–8.
[60] Harik R, Shi Y, Baek S. Shape Terra: mechanical feature recognition based on a persistent heat signature. Comput Aided Des Appl 2017;14(no. 2):206–18.