VIRTUAL COMISSIONING OF MANUFACTURING SYSTEM INTELLIGENT CONTROL

Kaishu Xia[1], Christopher Sacco[1], Max Kirkpatrick[2], Ramy Harik[1] and Abdel-Moez Bayoumi[1]

[1] McNAIR Center for Aerospace Innovation and Research, Department of Mechanical Engineering, College of Engineering and Computing, University of South Carolina
1000 Catawba St., Columbia, SC, 29201, USA

[2] Siemens Product Lifecycle Management Software Inc., Charlotte, NC, 28277, USA

ABSTRACT

Smart manufacturing systems seek to provide adaptive intelligent strategies to manufacturing practitioners in response to environmental changes, system prognosis, and optimal solution identification. For large scale manufacturing processes, their control methodologies are expensive to train, test and develop. Virtual Commissioning, as a digital transformation method, offers a data-driven approach to automate manufacturing system knowledge so that a digital twin can be developed to visually represent manufacturing plants, numerically simulate robot behaviors, predict system faults and adaptively control manipulated variables. In this work, integrating a Machine Learning agent into the Virtual Commissioning platform Siemens Technomatix Process Simulate further expands the usage of the digital twin by training and verifying intelligent control algorithms before pushing them to the physical world for implementation. This is accomplished by transferring data between the Siemens Process Simulate Software Development Kit and Google's TensorFlow framework to implement a Reinforcement Learning-based dynamic scheduling algorithm on a virtual manufacturing cell. For future development, control via an industrial controller will allow communication between the digital world and physical manufacturing plant so that synchronous control can be achieved. The developed control algorithms are expected to assign tasks, schedule work, generate optimal path solutions and demonstrate control robustness.

1. INTRODUCTION

1.1 **Purpose**

The ever increasing complexity and uncertainty of the modern manufacturing environment [1] [2] leads to a number of potential problems in the rapid production of a given part that require analysis of manufacturing indicators. For example, underperforming suppliers can cause significant disruptions in the manufacturing process and have cascading effects requiring manual decision-making and assessments [3]. In the modern manufacturing environment, there are potentially thousands of individual machine movements and commands that cumulatively affect the time required for assembly or finishing.

The notion of optimization in such a combinatorically dense environment would historically be challenging enough, but with the addition of disruptions and part arrival dynamics, finding optimal manufacturing policy solutions for a given process is effectively impossible with traditional means

in the context of a live production environment. Historically, this was primarily attempted by human intervention in the scheduling process; gut feelings and educated guesses driving the allocation of potentially millions of dollars of equipment and materials. Such rationales have clearly been of value to manufacturers with a set of conditions that produces potential uncertainties in the production process of a part. Disruptions, uncertain part arrival times, and managing scarce resources are incidents that appear not only in an individual manufacturing cell, but also throughout a modern factory environment. The scalability of any solution is an additional factor to consider in implementation. With these outstanding challenges, this document aims to outline a conceptual and experimental framework for optimal scheduling of manufacturing operations with the aid of digital transformation. We intend to propose a novel merging of a virtual commissioning platform with reinforcement learning-based scheduling techniques to develop an optimal policy agent to inform the automated manufacturing of a given part. Section 2 will represent an overview of the Digital Engine concept we have developed. Section 3 gives detailed explorations of the problems involved with defining a solution involving manufacturing simulation and Reinforcement Learning (RL).

1.2 **Literature Review**

Virtual Commissioning (VC) is a concept for testing systems through simulations to evaluate the safety and feasibility of scheduling and manufacturing approaches before physical deployment and realization. An overview [4] demonstrated the implementation of VC necessitates a Computer-Aided Engineering (CAE) simulation tool environment and object-oriented databases containing simulation models of manufacturing system components. The authors also highlighted the importance of low-level modelling in VC, for example, geometrical modelling, functional modelling, and electrical modelling. Several recent attempts [5][6][7] were made in realizing VC with the same philosophy using different simulation tools. The employment of virtual counterpart of manufacturing systems was furtherly validated in [8] [9]. The concept of *Digital Twin* (DT) was discussed in many areas as the term for virtual counterparts. Complete components of DT in manufacturing are proposed [10] to include five parts: physical part, virtual part, connection, data, and service. An overview [11] generalizes the characteristics of DT to be (1) real-time reflection, (2) interaction and convergence between virtual and physical space, (3) self-evolution, real-time updates, and continuous improvement through comparing virtual space with physical space in parallel.

Intelligent controllers with Deep Reinforcement Learning initially succeeded in video games through a proper design of reward and deep learning mechanisms [12][13][14]. The exploration of implementing Deep Learning (Neural Network) in robot controllers has also been developing [15] [16]. However, training such intelligent controllers by real commissioning on manufacturing plants could be costly and dangerous, moreover, the complexity of a manufacturing plant leads to more demanding designs of the reward-prediction system than on a single robot actuator. This work and its subsequent research aims to develop such intelligent controllers with the aid of VC or, furthermore, DT.

2. CONCEPTUAL OUTLINE

2.1 The Digital Engine

The conceptual bedrock for the system outlined in this document is a virtual platform capable of testing numerous iterations of a control policy on manufacturing process simulations while interfacing with a physical platform. This software interface is termed the *Digital Engine* for the platform presented herein after. The Digital Engine consists of the dynamic scheduling agent linked with both the physical manufacturing cell and advanced simulation software for training and commissioning of policies to be pushed to the physical plant. The Digital Engine is a path forward for automation of manufacturing with limited intervention from human managers.

The implementation of the proposed Digital Engine requires an information flow between different platforms, in particular the Siemens Software (the virtual cell and its interfaces), Physical Cell, and Machine Learning (ML)-based scheduler systems (See Figure 1). For the Virtual-Physical communication, the signal exchange between physical and digital platforms occurs on industrial controllers such as Programmable Logic Controllers (PLC). The PLC receives sensor signals from physical sensors in the Physical Cell and simulated sensor from Siemens Process Simulate. Meanwhile, the PLC, programmed by users' devices through another set of Siemens software named Totally Integrated Automation (TIA) Portal, sends commands to the automation in the manufacturing cell. Communication between the PLC, or its simulation, and the virtual cell is achieved through an Open Platform Communications (OPC) server where simulated/real signals and hardware commands can be converted to corresponding communication protocols. The OPC server, as an information hub, exchanges and stores data from all three systems. For communication between the ML-based scheduler system end and either virtual/physical cell, data collected from Process Simulate and the real world will be sent to ML systems to train scheduler models. An integrated software development kit in Process Simulate named Technomatix.NET enables controlling the virtual cell with object-oriented programs in C#, which communicates with ML scheduler. Subsequently, via OPC server, ML scheduler writes virtually verified control commands to the physical PLC so that a Virtual Commissioning process is performed.

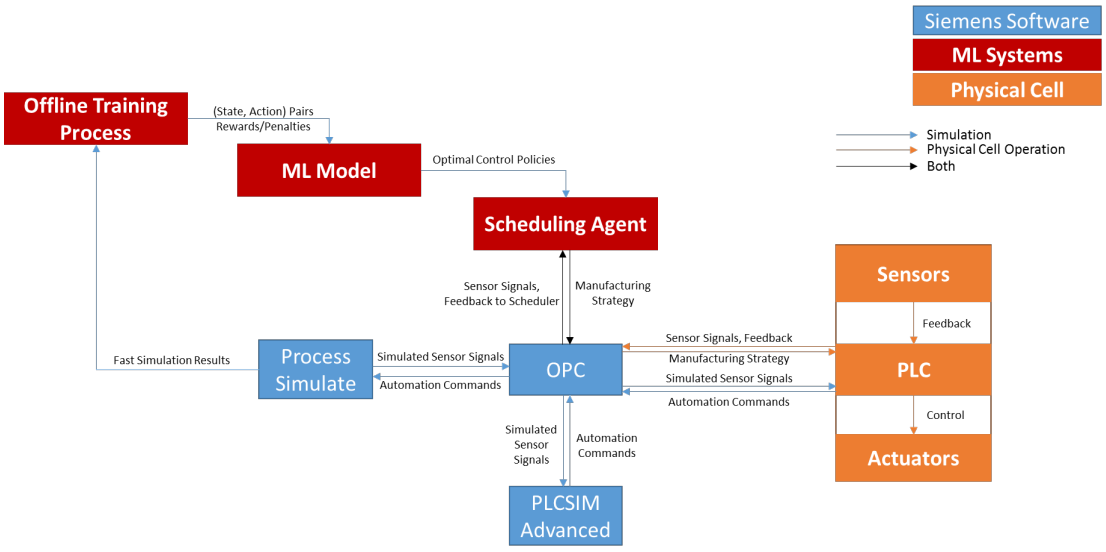

Figure 1. Data Flow between proposed Virtual-Physical- Scheduler System Ends

The ML-based scheduler is developed as an embedded component of the proposed Virtual-Physical-Scheduler system (See Figure 2). First, scheduling algorithms are tested using low-resolution simulations on a virtual cell created by Process Simulate; this develops the primary off-line training cycles. The goal at this stage is to find feasible paths to successfully accomplish the manufacturing tasks without major operational failures, such as collisions, wrong displacement, etc. This is done by running an adequate amount of rapid and low-resolution virtual simulation tests to identify and remove any obviously erroneous manufacturing strategies yet cannot be constrained without simulations. This outcome will contribute to the formulation of scheduling agent at the earliest stage. Based on the solutions from the low-resolution tests, high-resolution simulation tests are performed to further justify the ML-based scheduler on Siemens Process Simulate. These higher resolution tests allow for greater simulation accuracy, and incorporate testing of more system characteristics. Finally, the schedule agent will be tested on the physical platform and its response signals will contribute to the scheduling algorithms with the highest priority to properly capture the final objective of efficient management on the physical cell.

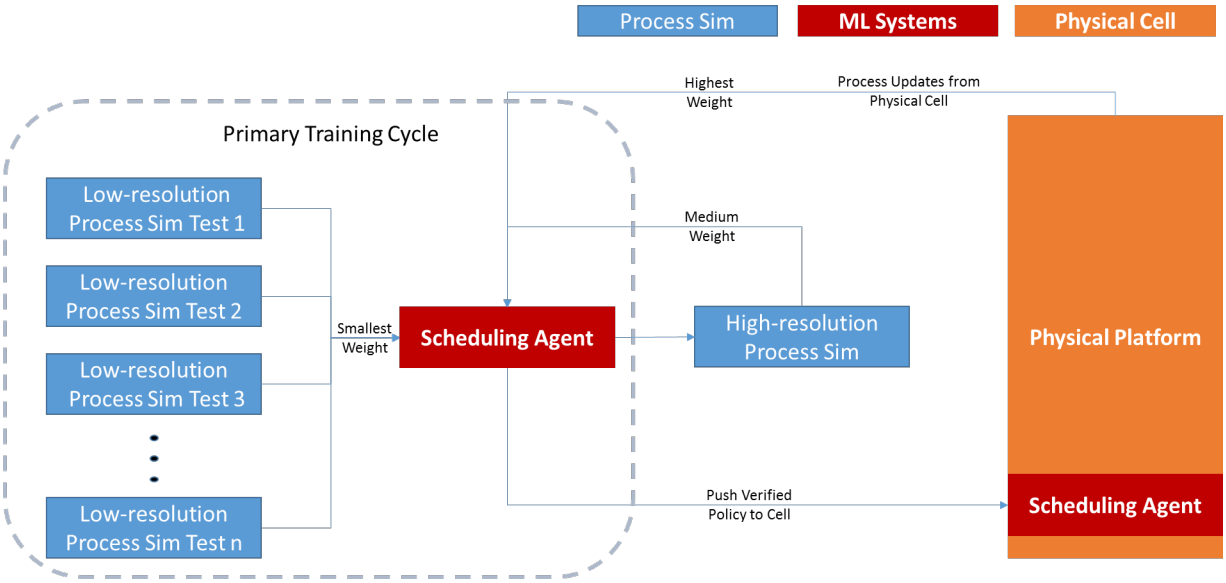

Figure 2. Controller Development with Virtual-Physical-Scheduler System

2.2 **Deep Q-Learning**

Reinforcement Learning (RL) is a foundational cornerstone of machine learning and artificial intelligence. RL is unique in its modeling of learning processes that are dependent on the return of a reward to an agent exploring an environment. This has made RL particularly useful when approaching many control problems that are traditionally difficult for hard-coded solutions such as Go. Difficulty of such control problems lies in the fact that the possible future states and actions are beyond numeration, bearing close resemblance to autonomous control system in large-volume manufacturing plants. Researchers introduce Neural Network models, namely Deep Q-Network (DQN), to predict reward functions based on the input *state-action pairs* and utilize an Experience Replay mechanism to improve data efficiency and algorithm stability [12].

Deep Q-Learning with Experience Replay described in [12] utilized deep learning techniques to accomplish deployment of optimal policies. The same approach was adapted to estimate the

action-value function (Q-function), which evaluates the highest possible future rewards by taking action a at the current state s. This estimation was made by a two-layer neural network e. Q-values Q_{eval} calculated by network e depend on the input state-action pairs (s, a).

$$Q_{eval} = e_{(s,a)} \tag{1}$$

Using another neural network t that is identically structured as e, Q_{next} is calculated to estimate the Q-values of next state s'. Target Q-values Q_{target} adds the current operation reward function value $r_{(s,a)}$ to the best future Q-value Q_{next} with a discount factor Υ. Q_{target} are the ones we are trying to predict with network e.

$$Q_{next} = t_{(s',a)} \tag{2}$$

$$Q_{target} = r_{(s,a)} + \Upsilon \max_a(Q_{next}) \tag{3}$$

Define the loss L to be:

$$L = (Q_{target} - Q_{eval})^2 \tag{4}$$

The training process constantly updates network e parameters by gradient decedent on loss L. After a certain amount of training steps, network parameters t are replaced with network e's for the a continuous improvement on Q-values estimation. Detailed algorithms see [12].

2.3 **Simulation through Siemens Process Simulate**

The RL agent must be given an environment to interact with and train on. The clear limitations of performing this training with a physical platform, simply for safety reasons alone, necessitates digital simulations and a manner in which to check the policy ahead of implementation on a physical platform. The integration (See Figure 3) of Siemens Process Simulate with DQN training and testing is to provide a digital platform where cell states and action can be retrieved from and simulated on to reduce possible risks on a physical counterpart.

Process Simulate provides functions such as importing CAD, kinematics definition, collision check, and Teach-Type Robot Programming. Under standard mode, a functioning robot cell can be easily created and defined (as in Object Tree in Figure 4). Potential robot paths to accomplish tasks can be calculated by specifying all the desired robot positions (as in Operation Tree in Figure 4). Moreover, Process Simulate provides an object-oriented programming interface with the backend software database named Technomatix.NET. This SDK is capable of accessing all the defined objects and operations in this virtual cell, creating new objects/operations, controlling playback of simulations, and creating customized commands in the user interface.

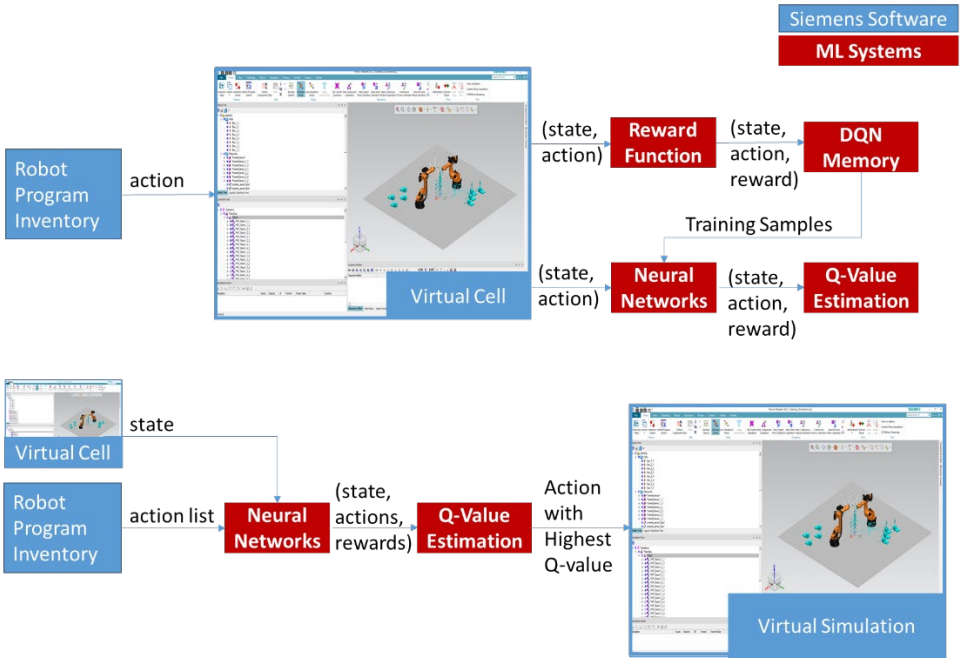

Figure 3. DQN-based operation scheduler training (Top) and testing (Bottom) procedures

3. CASE STUDY

A proposed conceptual outline has been implemented on digital platforms, which includes creating and interacting a virtual cell with Technomatix.NET, Deep Reinforcement Learning Algorithms, and policy simulation/verification. Commissioning on a physical cell is expected in future work. In this section, a case study of a two-robot system accomplishing a stacking task is discussed as an example to illustrate virtual commissioning of intelligent policies. The objectivity of this case study is to show Digital Engine connections between virtual systems, which lead to a prerequisite of proposed virtual commissioning implementation.

3.1 **Virtual Cell on Process Simulate**

A virtual counterpart of a robotic cell was created as shown in Figure 4. Three commands are created in this work to interact Process Simulate with Python shell scripts with Technomatix.NET framework. *Command_1* "Training on Python" enables the connection from Process Simulate to Python scripts by sending cell status and all predefined robot programs in Program Inventory, then initiates training cycles. *Command_2* "Control Policy Simulation" receives the operation recommendations from trained control policy and performs simulations on the virtual cell. *Command_3* "C#/Python streaming" implements recommendation-simulation process in one click. The goal of this command is to create an online intelligent control system based both on the recommendation from prior training model and an immediate response from the cell status.

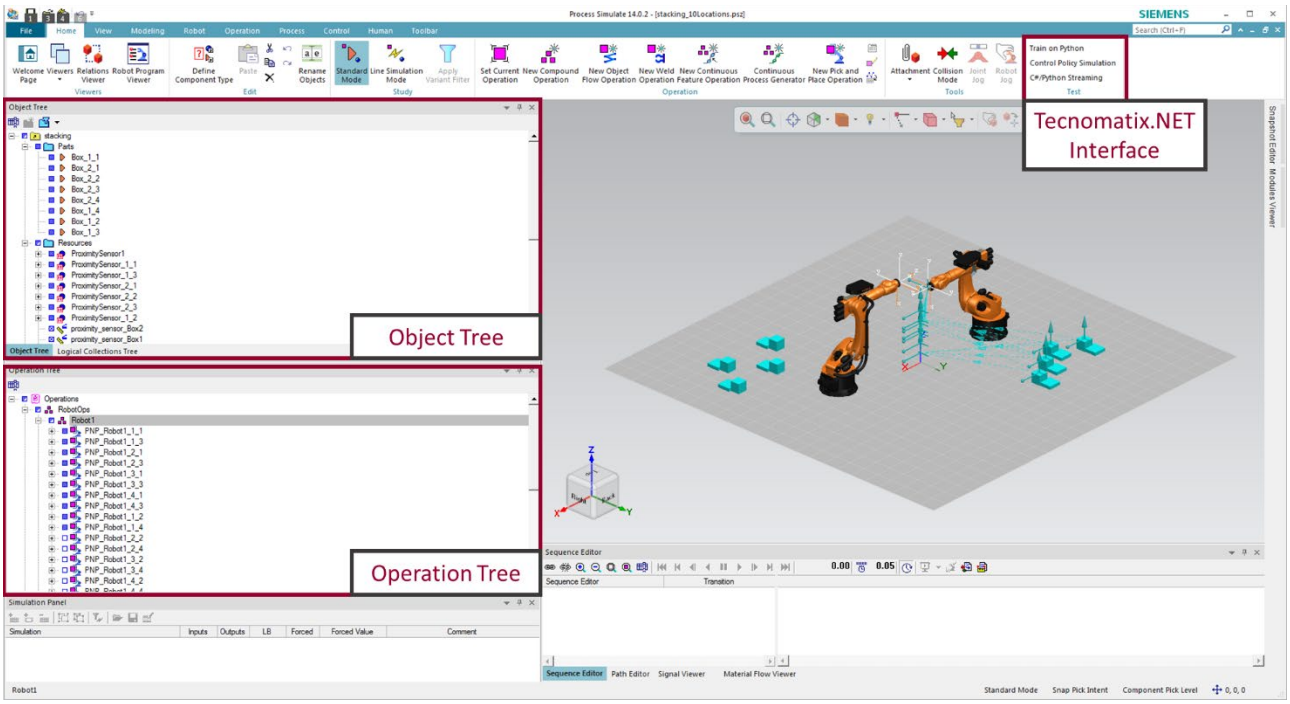

Figure 4. Virtual Cell on Siemens Process Simulate User Interface

3.2 **DQN Methodology**

In this work, as one Deep Reinforcement Learning algorithm, Deep Q-Learning was implemented as one attempt toward a proposed intelligent scheduler. As explained in *Section 2.2*, the idea of Deep Q-Learning is to use deep neural networks to model the Q values for each *state-action pair* and then find the most rewarding action. Training Neural Networks using prior memory samples will derive a multilayer estimation of rewards accumulation, which is predefined with the input of cell state s observations. By carefully shaping rewards, process failures can theoretically be managed in an automated fashion.

Explorations on reward functions were made by related work [17]. In this case study, trials of reward function $r_{(s,a)}$ were made to reduce the computational steps to find out correct stacking sequences. The Python scripts implementing DQN were written using Google's TensorFlow framework. The memory size was designed to be 5000 samples. The DQN hyper-parameters were set to be: learning rate=0.01, discount factor=0.9. The *Command_1* prompt and learning curve is shown in Figure 5. It is noticeable running an ϵ-greedy policy with ϵ=0.1, meaning approximately in every 10 steps, DQN policy randomly take one operation from Operation Inventory. This enables the DQN policy to explore different combinations of robot operations and the learning curve to greatly fluctuate.

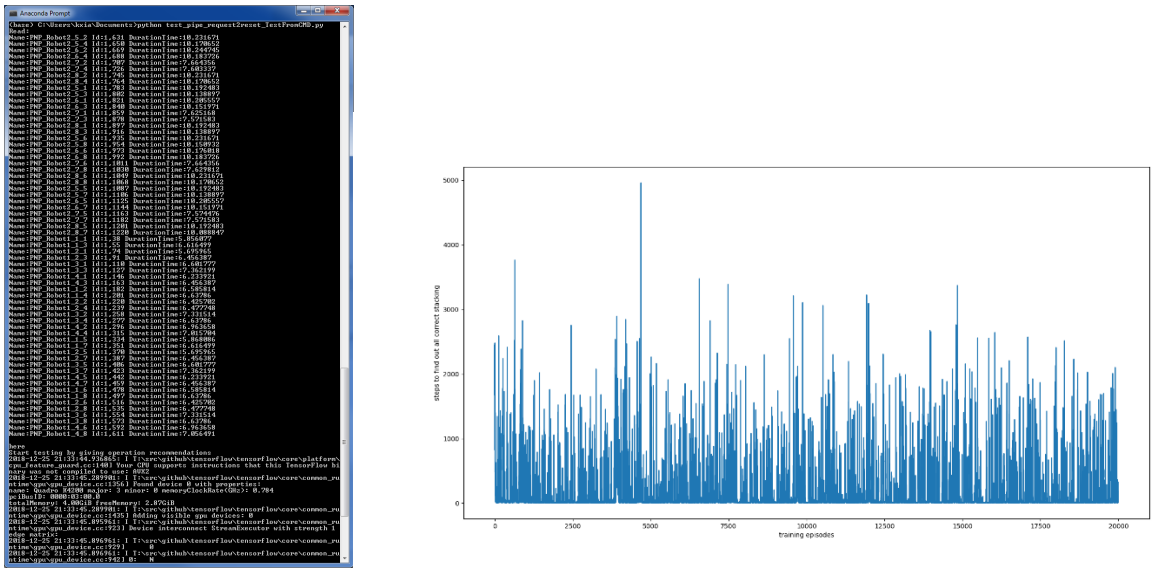

Figure 5. *Command_1* prompt with operation info (Left) and DQN training curve (Right)

3.3 **Operation Recommendation and Simulation Player**

After an adequate amount of training, the Neural Networks were eventually utilized in the decision-making process. *Command_2* "Control Policy Simulation" on Process Simulate triggers a command that restores the saved DQN model and calculates a Q value for each operation to provide scheduler recommendations (**Figure 6**). Then, it performs a policy test on the Process Simulate virtual cell. This is done by playing the simulation with the recommended stacking sequence and checking for failures such as collisions and object availabilities with current cell status.

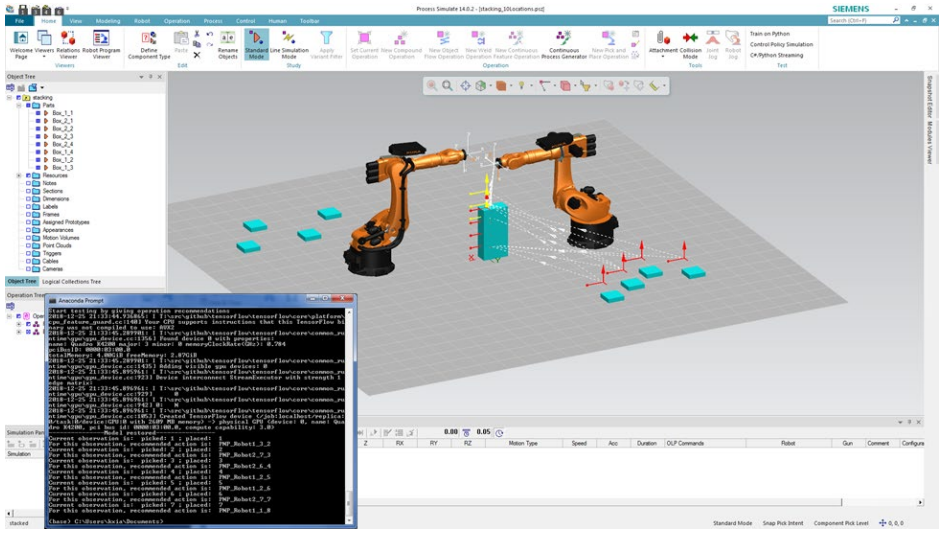

Figure 6. *Command_2* Operation Recommendations and recommended stacking sequence simulations

4. CONCLUSIONS

This work provided a proof of concept for Virtual Commissioning of an intelligent robot scheduler in manufacturing cells. The proposed VC system comprises a Virtual Cell created by Siemens Process Simulate, a ML based scheduler created by Tensorflow, and a Physical Cell. These three ends are expected to communicate with each other with instantly updated dataflow. On the basis of conventional VC methodologies demonstrated in related research, this work utilized the SDK tool Technomatix.NET in Process Simulate to interact with the virtual cell using Python scripts so that a Deep Q-Learning based scheduler model could be trained, tested, and simulated on a virtual environment. The scheduler was developed with Google's Tensorfow framework using off-line Deep Q-Learning algorithms. The fact that Technomatix.NET can stream data with Python enables instantaneous exchange of information between ML system and Process Simulate, by which online operational recommendations and simulations were furtherly performed. In Section 3, a case study was provided to illustrate the development of Virtual Cell and ML-based scheduler.

Future work on this topic includes the physical cell setup, data streaming on all three ends, ML-based scheduler algorithms improvement and optimization, user interface development, and so on. The following work will further complete the conceptual outline in Section 2. Moreover, the gap between physical and virtual cells needs to be further filled by improving the part CAD models, better defining actuator kinematics, and introducing more sensor signals to describe the cells.

5. REFERENCES

1. Harik, R., Hachem, W. E., Medini, K., & Bernard, A. (2015). Towards a holistic sustainability index for measuring sustainability of manufacturing companies. International Journal of Production Research, 53(13), 4117-4139. doi:10.1080/00207543.2014.993773
2. Saidy, C., Panavas, L., Harik, R., Bayoumi, A., & Khoury, J. (2017). Development of a Part Criticality Index in Inventory Management. *Product Lifecycle Management and the Industry of the Future IFIP Advances in Information and Communication Technology,*184-195. doi:10.1007/978-3-319-72905-3_17
3. Harik, R., Bayoumi, A. M., Panavas, L., Wilson, Z., Saidy, C., & Pinna, C. (2018). Literature review of current practices of suppliers assessment and valuation of decisions regarding underperforming suppliers. International Journal of Product Lifecycle Management, 11(3), 245. doi:10.1504/ijplm.2018.10015951
4. Hoffmann, P., Maksoud, T. M., Schumann, R., & Premier, G. C. (2010). Virtual Commissioning Of Manufacturing Systems A Review And New Approaches For Simplification. ECMS 2010 Proceedings Edited by A Bargiela S A Ali D Crowley E J H Kerckhoffs. doi:10.7148/2010-0175-0181
5. Guerrero, L. V., López, V. V., & Mejía, J. E. (2014). Virtual Commissioning with Process Simulation (Tecnomatix). Computer-Aided Design and Applications, 11(Sup1). doi:10.1080/16864360.2014.914400
6. Hoffman, P. (2016). *On virtual commissioning of manufacturing systems*. Great Britain: University of South Wales.
7. HALMSJÖ, J., & FÄLT, J. (2016). Emulation of a production cell Developing a Virtual Commissioning model in a concurrent environment.
8. Konstantinov, S., Ahmad, M., Ananthanarayan, K., & Harrison, R. (2017). The Cyber-physical E-machine Manufacturing System: Virtual Engineering for Complete Lifecycle

Support. *Procedia CIRP,63*, 119-124. doi:10.1016/j.procir.2017.02.035
9. Harrison, R., Vera, D., & Ahmad, B. (2016). Engineering Methods and Tools for Cyber–Physical Automation Systems. *Proceedings of the IEEE,104*(5), 973-985. doi:10.1109/jproc.2015.2510665
10. Tao, F., & Zhang, M. (2017). Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing. *IEEE Access,5*, 20418-20427. doi:10.1109/access.2017.2756069
11. Tao, F., Zhang, H., Liu, A., & Nee, A. (2018). Digital Twin in Industry: State-of-the-Art. IEEE Transactions on Industrial Informatics, 1-1. doi:10.1109/TII.2018.2873186
12. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M.A. (2013). Playing Atari with Deep Reinforcement Learning. CoRR, abs/1312.5602.
13. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518, 529--533.
14. Jaderberg, M., Mnih, V., Czarnecki, W., Schaul, T., Leibo, J.Z., Silver, D., & Kavukcuoglu, K. (2016). Reinforcement Learning with Unsupervised Auxiliary Tasks. CoRR, abs/1611.05397.
15. Lewis, F., Yesildirek, A., & Liu, K. (1996). Multilayer neural-net robot controller with guaranteed tracking performance. IEEE Transactions on Neural Networks, 7(2), 388-399. doi:10.1109/72.485674
16. Riedmiller, M.A., Hafner, R., Lampe, T., Neunert, M., Degrave, J., Wiele, T.V., Mnih, V., Heess, N., & Springenberg, J.T. (2018). Learning by Playing - Solving Sparse Reward Tasks from Scratch. ICML.
17. Ou, X., Chang, Q., & Chakraborty, N. (2019). Simulation study on reward function of reinforcement learning in gantry work cell scheduling. *Journal of Manufacturing Systems,50*, 1-8. doi:10.1016/j.jmsy.2018.11.005.