ManufactSim: Manufacturing line simulation using heterogeneous distributed robots

Benoit Piranda, Ishan Gautam, Jerome Meyer, Anass El Houd, Julien Bourgeois

Abstract The creation of current assembly lines can benefit from the new advances made in the fields of Computer Science and the Internet of Things (IoT) to increase their flexibility and improve their reliability. There are assembly line simulators developed for this purpose. However, these simulators have been designed to model every detail of the line and take hours to be done. The aim of this paper is to introduce a faster and more accurate computer-based solution - *ManufactSim*- allowing the simulation of a real production system. This implementation derives from a behavioral modular robots simulator enhanced with a 3D display option. The results show that *ManufactSim*'s performances are above the standards with an execution time less than 11 seconds for 8 hours shift running on a CAD computer. Our developed solution is able to face this challenge with an highly accurate and efficient simulator without compromise. The performed benchmarks show that we obtain a robust and agile tool needed for a global future solution based on Machine Learning. The benefits of this contribution will permit to automate the generation of industrial assembly lines while caring on multiple optimization criteria.

Key words: Manufacturing, Industry 4.0, Digital Twin, Behavioral Simulator, Distributed Algorithms, Operational Research, Optimization Methods

1 Introduction

Industrial production is a constant adaptation process whose aim at increasing quality, speed or cost of the production, to cite only a few. Automation science has

Benoit Piranda, Ishan Gautam, Jerome Meyer, Anass El Houd, Julien Bourgeois
Univ. Bourgogne Franche-Comte, FEMTO-ST institute, CNRS, France,
e-mail: `{first}.{last}@femto-st.fr`

Jerome Meyer, Anass El Houd
Faurecia, France, e-mail: `{first}.{last}@faurecia.com`

developed numerous theoretical tools which have been used to optimize the manufacturing process. However, designing a new manufacturing line still faces lots of challenges. A product is made of different parts assembled all together and the assembly order can vary, there are multiple choices of robots, some tasks can be made by humans and storage can placed between assembly units. The number of combination is therefore huge. Furthermore, the cost function of each solution has also many parameters like the employee cost, the machine cost or the production rate. Aside from these challenges, deep learning is becoming a prevalent solution in many fields of optimization. Its capacity to digest huge amount of data and to solve complex problems have broadened its scope of use. In manufacturing, deep learning has been successfully applied to vision, Automated Guided Vehicle (AGV), and in many other domains but rarely to the design of manufacturing lines. We propose to develop a complete environment for helping manufacturing line designers.

A machine learning algorithm will be used to generate the scenario and to optimize the different parameters (cf. Figure 1). To do this, it will need to quickly evaluate the efficiency of the scenario. There exists plenty of manufacturing line simulation which could be used to do like [12] or [7], except that these simulators have been designed to model every details of the line and a simulation could take hours to finish. What we need, in this context, is a different type of simulator which does not exist to the best of our knowledge. That is why, we propose in this article to develop *ManufactSim*: a behavioral discrete event simulator able to simulate a real manufacturing line in a few seconds. The simulation and visualization engine is a

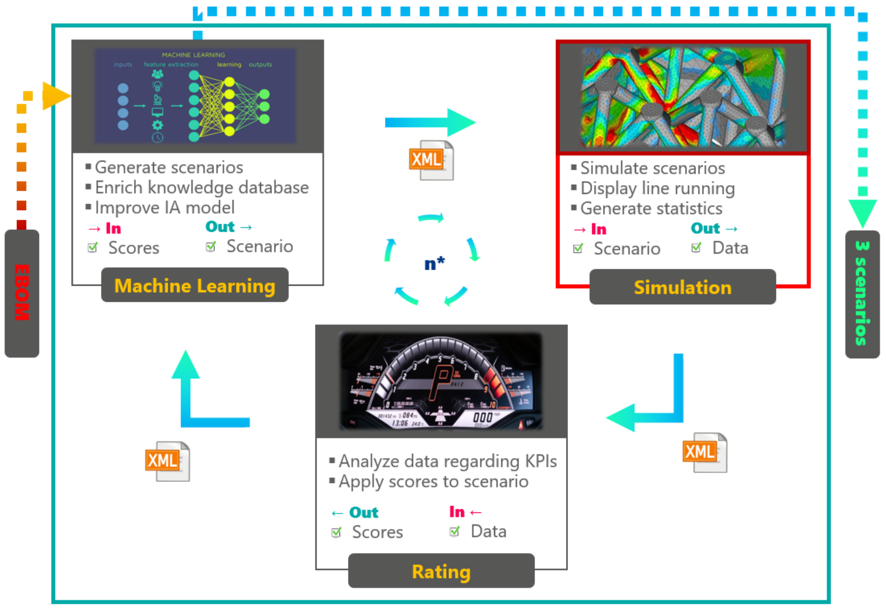

Fig. 1: Optimization solution architecture

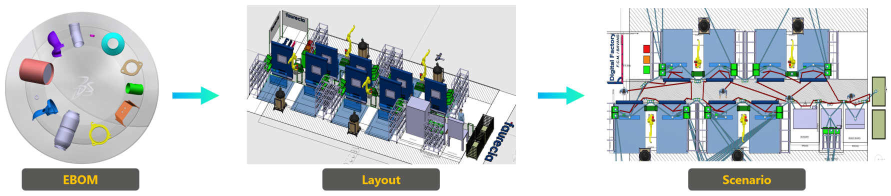

Fig. 2: Manufacturing environment

fork of *VisibleSim* [10], a modular robots behavioral simulator and as a bonus, this visualization engine is able to display and animate the manufacturing line in 3D.

In Section 2, we will present some background information about the industrial context of this study, followed by a state of the art in manufacturing line simulation in Section 3. Section 4 will present our approach to enable high-speed simulation of manufacturing lines and Section 5 will report our results showing that our environment is fast, precise and adaptable. Section 6 will conclude the article and will present some future work.

2 Industrial context

Three components are identified in order to design a manufacturing process: the product's *Engineering Bill Of Material* (EBOM), the manufacturing environment and the scenario as presented in Figure 2. The EBOM contains the list of parts to assembly with their constraints and parameters. The line layout sums up the physical manufacturing environment while the manufacturing scenario is mapping the entities interactions organized into process.

Today, each manufacturing scenario is designed manually by a *Process Architect*. This expert has to dispatch the assembly operations into a line layout and deal with an explosion field of possibilities. Then the scenario is presented to the *Plant Manufacturing Leader* who is in charge of carrying on the manufacturing process. Next the scenario is updated with the last feedback and presented to the job experts. Once all feedback are collected, the scenario get a last optimization for a final validation by the *Program Manufacturing Leader* who is in charge of respecting the engagement contracted with customers. The Figure 3 illustrates three states of the combinatorial explosion for a seven parts EBOM assembled in four stations which results on five operations. So that for each state respectively loading number N_L and sub-assemblies number N_S we compute the pair $\{N_L, N_S\}$. We get the optimal solution $\{11,5\}$, all possible loadings $\{64,5\}$ and all possible loadings and sub-assemblies $\{64,27\}$.

A assembly line is the result of a set of entities enumerated below, combined to describe how to assemble the pieces, manage the interactions between robots, workers, and finally define evaluation parameters.

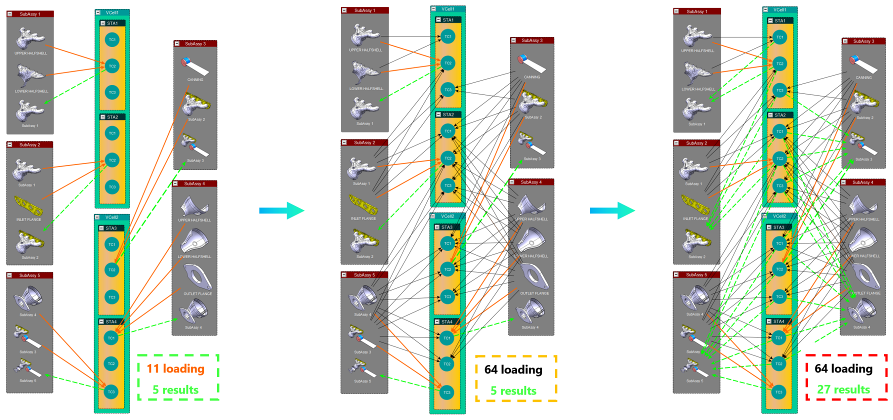

Fig. 3: Combinatory explosion

An EBOM is a list of sub components which are composing a manufactured product as illustrated in the figure 2. The assembly is following a defined process resulting from a multiple tasks succession. Many parameters and constraints are defining the EBOM such as a volume to produce per year, a target assembly cycle time, some customers datum.

A layout is a production line footprint describing the physical manufacturing environment aspects. Equipment and humans workers also called entities are listed in order to design the assemblies flows. The parameters and constraints of a layout can be defined by a list of equipment including cells, machines robots and tooling, a sum-up of storage and buffers, a way of working (clockwise, anti clockwise), an amount of workers.

A manufacturing scenario is a list of actions which defines the interactions over the line's layout entities. The scenario is a macro view of every steps needed to complete an assembly.

A flow is a continuity of actions performed by each resources. In our case we've got three types of interdependent flows such as human operator, robot and product. The KPIs for Key Indicators Performance are used in the industry for evaluating the efficiency rate. In an highly concurrency environment this criteria is required in order to keep an eye on the competitiveness. Our main KPI is focused on the actions duration performed by the line's layout resources. The load balancing is the way to evaluate the workload balancing between resources.

3 Related work

Digital tools are currently used to manage the new processes creation such as "*3DExperience* platform" [2] from *Dassault Systems*. These tools are offering a wide range of services for industries by covering the product life cycle management. Considering the aim of this tools is to bring a complete overview of the process without generating optimized manufacturing scenarios, we assume that an extra automation tool might be helpful.

So that, several works were done by using Machine Learning methods and more precisely Reinforcement Learning. After a selection on the most accurate research, a first approach from 2018 was turned around the application of Google's Deep Q-Learning architecture by Waschneck et al. to optimize production scheduling in a multi-agent environment. Their method is applied to a dynamic and complex job shop environment consisting of work centers with different constraints, multiple machines of different types and multiple products [11]. Then a second research was done in 2019 on "Design, implementation and evaluation of reinforcement learning for an adaptive order dispatching in job shop manufacturing systems" [6]. In 2021, a work is covering the topic "Designing an adaptive production control system using reinforcement learning" [5]. In another contribution to scenario simulation and statistics generation, Kampa et al. [4] proposed an approach that allows a better representation of real production processes and assembly lines by using discrete event simulation methods.

A survey published in 2020 suggests that 75% of possible research areas in the application of machine learning to improve production planning and control are barely explored or not addressed at all [3], which is the case for the item "Smart Design of Products and Processes" in which we propose to develop this work.

4 Our proposal

In order to simplify the complexity of the possible operations combinations (see Fig. 3), we have developed a manufacturing optimized scenario generation tool. We focus on the simulator which is a single part of a whole solution.

The goal is to output multiple manufacturing data for different cases as the volume of part produced or the shift duration. Three components are necessary to perform a simulation: a product, a line layout and a manufacturing scenario. Each pair of single manufactured product combined with different line layout will propose multiple manufacturing scenarios. Each of them will have to be simulated in order to evaluate them efficiency. In a addition to the simulation, a second API will analyze and score the generated data for each scenario. Then global solution will be able to perform a efficiency benchmark over each manufacturing scenario.

VisibleSim [10] is a behavioral simulator designed to simulate distributed programs running in modules composing distributed robots. This tool is useful to define large sets of modules placed in a lattice that are all running the same code. Each module is able to communicate with its neighbors, i.e. the modules that are placed in a direct neighbor cell.
This simulator really executes `C++` codes that are ran by each module (we call them *Block Codes*) and simulates the real world, including point to point communications

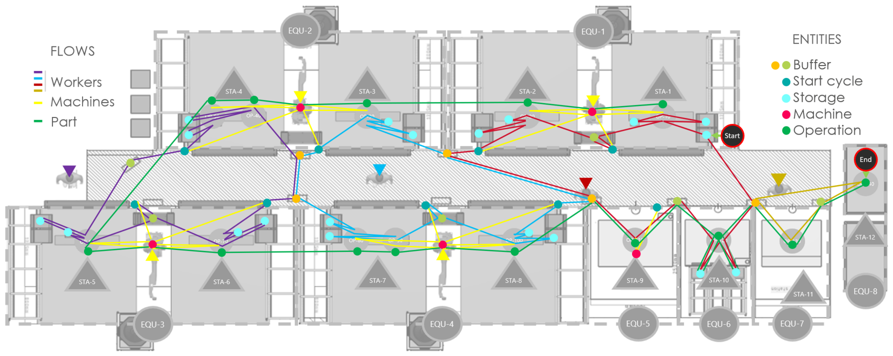

Fig. 4: Manufacturing line assembly flows example

taking into account the delay of transmissions, sensors and actuators events due to the interaction with the simulated world.
VisibleSim has been evaluated with many kinds of robots, some of them exist in real and then allow to compare real and simulated value. For example, experiments in previous works [9] shows the very good capability of *VisibleSim* to simulate real distributed programs running in real *Blinky Blocks* robots.

We propose here to adapt *VisibleSim* in order to simulate precisely the production lines in order to extract many KPI values. The update concerns three items: the replacement of the lattice which allows to define which modules are connected, the way of writing the *Block Code* and the graphic representation of the system.
Concerning the definition of the graph, a connection must be defined between two elements that exchange parts of the conception during the assembly. These connections are bidirectional that allows to transmit a piece from a module to another one and receive an acknowledgment in the same canal. The list of connections that correspond to the edges of the flow presented Figure 4 is enumerate in an XML files. As a consequence, the number of connections is much less important than the one defined by classic robots used in *VisibleSim* (they may have up to 12 neighbors by module). As the consequence, it reduces the number of resolution steps necessary to simulate the distributed system.
Programming robots with *VisibleSim* consists in writing a `C++` code that describes what the module must do at starting and when receiving a message from a neighbor module or an event from a sensor. We need here define a more flexible method, adapted to the global workflow.
To do so in *ManufactSim*, we divide the treatment in two parts: first in the internal *Block Code* we describe the behavior of the agent using `C++` for a list of actions, it mainly consists in sending messages or waiting for messages to communicate with the neighborhood. This part corresponds to the knowledge of the agent, for instance when a worker need to perform an action into a station, he has to first check if the door is open otherwise he has to wait. The second part is stored in an XML file, it consists in a succession of calls of the previous actions detailed for each agent.

Agent	Layer	Description	Color
Station	0	Base component where parts are assembled.	white
Buffer	1	Storage for sub assembly next to station.	green
Worker	2	Human workers interacting with elements.	gray
Robot	3	Welding machine which performs operations.	blue

Table 1: List of agents running on *Block Codes*, colors in the last column are the color of the corresponding object in Figure 5

Then in *ManufactSim*, the *Block Code* executes the list of actions which are stored in the XML file. This list is initially copied in the local memory of each module and then consumed respecting the order. This decomposition of the *Block Code* allows to evaluate different scenarios, described in the XML file without recompiling the simulator.

Concerning graphical representation of the configuration, *ManufactSim* shows the modules freely placed in several overlapping layers. For example a *robot* can be placed inside a *station* just by placing it in a higher layer than the one used by the *station*. Table 1 shows a part of the list of agents with their color and layer used in Figure 5. This Figure shows a screenshot of graphical interface of *VisibleSim*, where each agent is represented by a distinctive 3D model and a specific color. This scene is animated according to the message and events and can be freely observed during the simulation. Two simulation modes are available, one respecting the real time showing the actions played with the real duration and the other running the simulation as fast as possible to get the simulation results quickly.

The tasks list is the XML formalism of a manufacturing scenario attached to a line layout. Every action is mapped to a *Block Code* which can be an agent so that

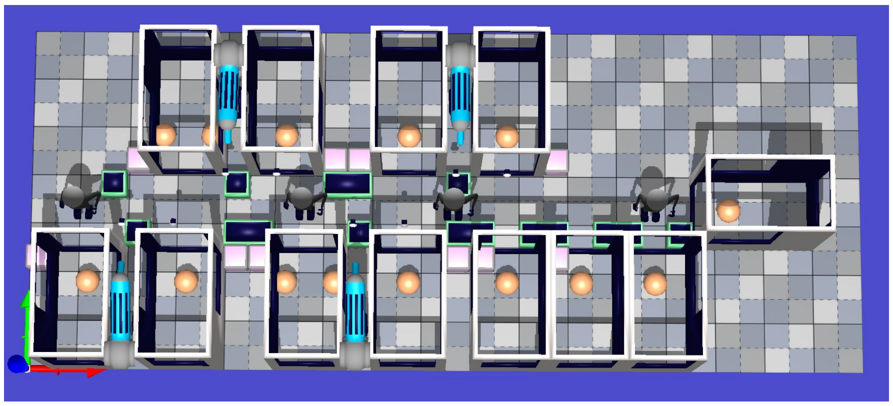

Fig. 5: *ManufactSim* graphic interface describing the arrangement of the manufacturing line.

simulation can be performed. The XML code is organized with global parameters, such as the line details and the list of parts with them corresponding welding.

The second part of the file describes the manufacturing scenario through a set of instructions according to the manufacturing equipment. Each equipment contains one or two stations which are assigned to a human operator. The file structure is respectively made of XML tags, i.e. 'vcell' containing 'station' and 'robot'. In addition, the tag 'action' defines and assigns the tasks to each resource by mapping it with the *Block Codes*.

The problem of finding optimal manufacturing scenarios is classified as NP problem: we can easily and quickly check if a candidate solution is indeed a possible scenario but we cannot prove if it is the unique best solution [1]. Our search for an optimal production scenario can be assimilated to a combinatorial optimization problem which consists in finding one of the best solutions in a set of feasible manufacturing scenarios [8].
A scenario is modeled by a set of parameters/variables presented by the following feature matrix M_s.

$$M_s = \begin{bmatrix} a_{11} & a_{12} & \cdots \\ \vdots & \ddots & \\ a_{K1} & & a_{KK} \end{bmatrix}$$

The feature vectors a_{ij} are chosen in such a way that they directly affects the KPIs, which means that a well-chosen combination of these parameters will result in optimal production performance.

5 Experiments

In this section, we are performing several experiments between our simulator and the *3DExperience* tool. This demonstrates the high speed and the accuracy as well

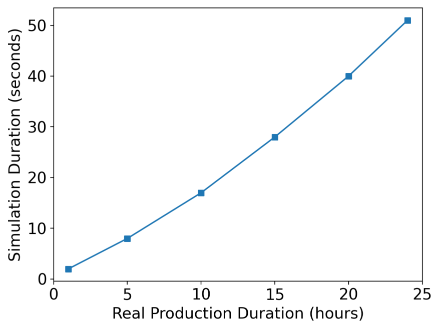

Fig. 6: Simulator execution time

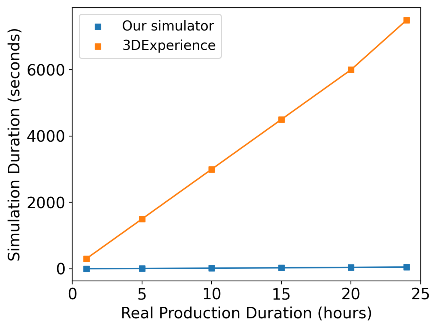

Fig. 7: Simulator versus *3DExperience*

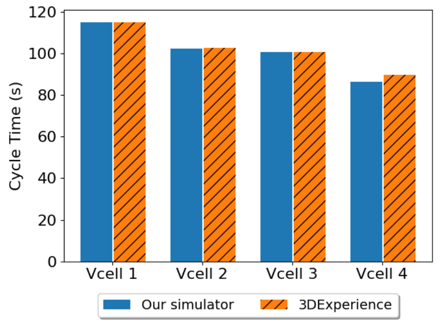

Fig. 8: Accuracy between simulator and *3DExperience*

as agility for future investigations. The experimental environment is powered by an Intel i7 CPU, 2.60 GHz 64-bit, Windows 10 operating system embedding 32 Gb of RAM memory.

This first experiment concerns the performance of our solution. Figure 6 highlights the fact that our simulator is fast regardless of the number of production hours simulated while Figure 7 compares the simulation duration for both solutions. Regarding our experimental environment, we simulate 24 hours of production in 46 seconds while *3DExperience* takes upper than 2 hours. Therefore, our simulator proves to be at least 200 times faster than *3DExperience*.

In order to evaluate the accuracy of the proposed solution, Figure 8 compares the results for both solutions. The Mean Square Error (MSE) is around 3,04 seconds resulting from updates applied on the tasks duration from the original *3DExperience*'s scenario.

Finally we can notice the agility of our method, Figures 9 and 10 are showing the simulation results for 3 different line layout configurations by varying the amount of Vcells or operators. The first figure shows the impact on the workload rate while

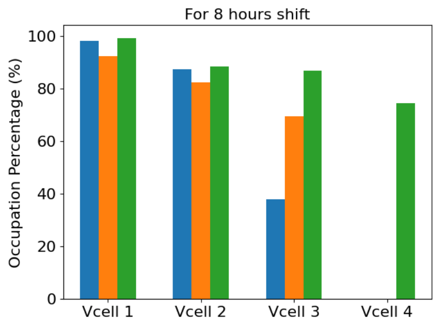

Fig. 9: Layout workload comparison

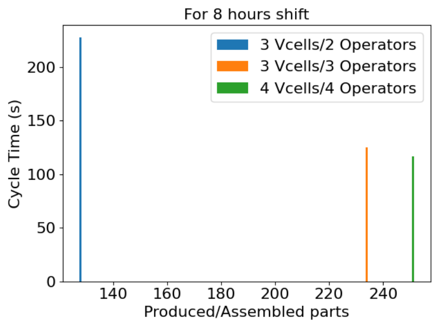

Fig. 10: Layout cycle time comparison

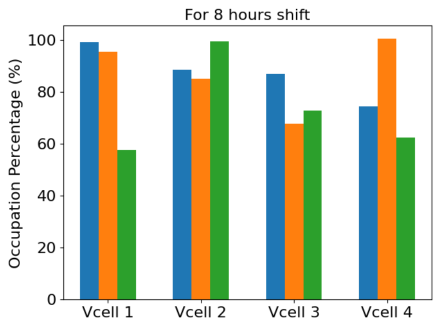

Fig. 11: Scenario workload comparison

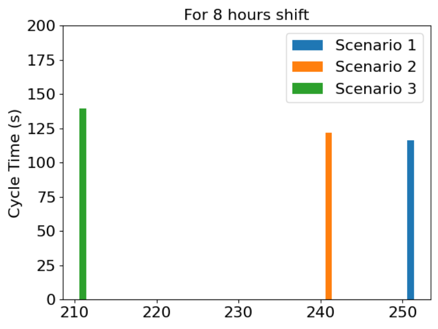

Fig. 12: Scenario cycle time comparison

the second one is focused on the cycle time. Figures 11 and 12 are showing the simulation results for 3 different manufacturing scenario configurations by changing the job scheduling arrangement. The first figure presents the impact on the workload rate while the second one focuses on the cycle time.

So that we show that our simulator is able to handle for the same EBOM, different layouts and scenarios to be generated by the artificial intelligence.

6 Conclusion

In summary, we can conclude that our manufacturing line simulation approach provides a stable, fast and accurate computer-based alternative ensuring a good yield for complex manufacturing flows.
The performed experiments show that our solution delivers accurate results for 8 hours shift during less than 11 seconds running on a standard CAD computer comparing to the current tools that can take more than 2 hours for equal results. In remains to be noted that the cycle time is only a study case that could be generalized to other KPIs without ensuring changes and adaptations. It should also to be noticed that an advantage of our solution is its configurability and its agility. Integrating other functionalities and methods will modify the simulator's parameters in order to adapt and optimize the whole system.

In our future work, we plan to implement additional technical features: First, embedding more KPIs for the assembly line scenarios evaluation. In this case, we might use a multi-objective evaluation that takes into account the economic constraints extending the cycle time: i.e. for the high cost labor countries, we will mainly seek to minimize the number of operators and vice versa. A other item concerns the automatic generation of scenarios to select the best ones using artificial intelligence techniques. This method should have the learning ability without relying constantly on experts. And as we will also implement a method for visualizing post-simulation results. We suggest to generate at the end of the simulation a complete report containing diagrams that allow a better analysis of the performance indicators of the scenario.

7 Acknowledgemnts

This work was done as a part of a CIFRE (2018-0927) project with Faurecia, funded by the Ministry of Higher Education and Research of France, managed by the Association Nationale de la Recherche et de la Technologie (ANRT).

References

[1] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.
[2] Andreas Barth. 3d experiences–dassault systèmes 3ds strategy to support new processes in product development and early customer involvement. In *IFIP International Conference on Digital Product and Process Development Systems*, pages 24–30. Springer, 2013.
[3] Juan Pablo Usuga Cadavid, Samir Lamouri, Bernard Grabot, Robert Pellerin, and Arnaud Fortin. Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0. *Journal of Intelligent Manufacturing*, pages 1–28, 2020.
[4] Adrian Kampa, Grzegorz Gołda, and Iwona Paprocka. Discrete event simulation method as a tool for improvement of manufacturing systems. *Computers*, 6(1):10, 2017.
[5] Andreas Kuhnle, Jan-Philipp Kaiser, Felix Theiß, Nicole Stricker, and Gisela Lanza. Designing an adaptive production control system using reinforcement learning. *Journal of Intelligent Manufacturing*, 32(3):855–876, 2021.
[6] Andreas Kuhnle, Louis Schäfer, Nicole Stricker, and Gisela Lanza. Design, implementation and evaluation of reinforcement learning for an adaptive order dispatching in job shop manufacturing systems. *Procedia CIRP*, 81:234–239, 2019.
[7] Senem Kursun and Fatma Kalaoglu. Simulation of production line balancing in apparel manufacturing. *Fibres & Textiles in Eastern Europe*, 17(4):75, 2009.
[8] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.
[9] Benoit Piranda, Paweł Chodkiewicz, Paweł Hołobut, Stéphane P. A. Bordas, Julien Bourgeois, and Jakub Lengiewicz. Distributed prediction of unsafe reconfiguration scenarios of modular robotic programmable matter. *IEEE Transactions on Robotics*, pages 1–8, 2021.

[10] Pierre Thalamy, Benoît Piranda, André Naz, and Julien Bourgeois. Visiblesim: A behavioral simulation framework for lattice modular robots. *Robotics and Autonomous Systems*, page 103913, 2021.

[11] Bernd Waschneck, André Reichstaller, Lenz Belzner, Thomas Altenmüller, Thomas Bauernhansl, Alexander Knapp, and Andreas Kyek. Optimization of global production scheduling with deep reinforcement learning. *Procedia CIRP*, 72:1264–1269, 2018.

[12] Seyed Mojib Zahraee, Saeed Rahimpour Golroudbary, Ahmad Hashemi, Jafar Afshar, and Mohammadreza Haghighi. Simulation of manufacturing production line based on arena. In *Advanced Materials Research*, volume 933, pages 744–748. Trans Tech Publ, 2014.