Dynamic Scheduling in a Flow Shop Using Deep Reinforcement Learning

Maria Grazia Marchesano(✉), Guido Guizzi, Liberatina Carmela Santillo, and Silvestro Vespoli

University of Naples Federico II, 80125 Napoli, NA, Italy
{mariagrazia.marchesano,guido.guizzi,santillo,
silvestro.vespoli}@unina.it

Abstract. Machine Learning (ML) techniques and algorithms, which are emerging technologies in Industry 4.0, present new possibilities for complex scheduling methods. Since different rules can be applied to different circumstances, it can be difficult for the decision-maker to choose the right rule at any given time. The purpose of the paper is to build an "intelligent" tool that adapts its choices in response to changes in the state of the production line. A Deep Q-Network (DQN), a typical Deep Reinforcement Learning (DRL) method, is proposed for creating a self-optimizing scheduling policy. The system has a set of known dispatching rules for each machine's queue, from which the best one is dynamically chosen, according to the system state. The novelty of the paper is how the reward function, state, and action space are modelled. A series of experiments were conducted to determine the best DQN network size and the most influential hyperparameters for training.

Keywords: Reinforcement learning · DQN · Scheduling · Flow shop

1 Introduction

The Industry 4.0 and emerging technologies offer new possibilities for dynamic scheduling strategies using Machine Learning (ML) techniques and algorithms [19]. This brings with it a number of significant new challenges and planning opportunities. Since different rules can be applied to different situations, it can be difficult for the decision-maker to select the best rule at any given time. Moreover, dispatch rules are limited to their local information horizons, so, there is no rule that exceeds others by different goals, scenarios and conditions of the system [15]. Many approaches have been taken in the literature to address the problem of scheduling and production optimal control, and [3] presents a very comprehensive literature review. In the literature [7] suggests that real-time knowledge of the production system can lead to significant improvements in dynamic scheduling performance. Even a decentralized scheduling approach can bring benefits to the performances of a production line [4]. To overcome problems

Published by Springer Nature Switzerland AG 2021
A. Dolgui et al. (Eds.): APMS 2021, IFIP AICT 630, pp. 152–160, 2021.
https://doi.org/10.1007/978-3-030-85874-2_16

that might occur when dealing with problems in which there is no complete knowledge of the system dynamics the methods and the algorithms of the ML, the deep learning (DL) [8] and reinforcement learning (RL) [9] can be exploited. In [10] convolutional and generative adversarial neural networks are employed in manufacturing and in dynamic scheduling, also Heger in [5] proposes a ML method that adjusts the parameters of the dispatching rule based on the system state. The method of Deep Reinforcement Learning (DRL) is the proper way to produce a self optimization scheduling policy, so that the accurate simulation and high performance data provided by a simulation tool can be used. DRL has recently been studied and used in the manufacturing systems because its characteristics allow it to address decision-making problems that can be difficult in today's complex and changing manufacturing systems environment [21]. A RL-based task-assigning strategy to enable multi-project scheduling in the Cloud Manufacturing perspective is addressed in [2]. The aim of the present paper is to create an "intelligent" tool that updates its choices in response to changes in the production line's situation. It can have a potential practical use because the data from the production line can be sent to a controller as inputs, and a decision can be made in the event of a disruption or a sudden change in the line. In [11] there is a similar approach to the one here proposed, but their goals are different (minimize the makespan), and even the design of the RL model, as in their case is implemented with Petri Net. In [12] there is the implementation of the RL using DQN combined with edge computing framework, for the scheduling of a job shop. Otherwise the approach proposed here is validated with a flow shop and a series of experiments is carried out to evaluate the best hyperparameters and network structure to be used by the DQN to achieve the best performance values. Besides the manner in which state, reward function, and action space are modelled is what distinguishes this approach from other presented in literature.

2 Problem Formulation

In this section we will discuss about the problem of scheduling in a flow shop with DRL as the paper focuses on the construction of an DRL-supported method that employs the DQN to choose the best rule for scheduling jobs on machines of a flow shop production line. Reinforcement Learning (RL) is a mathematical formalization of a problem involving decision-making. Since it focuses on goal-directed learning from feedback, RL is distinct from other Machine Learning approaches. Instead of being told what actions to take, the learning agent must find out for itself which actions result in the greatest reward, by putting them to the test by "trial and error". The learning agent refers to the entity that acts and learns from the environment (simulation model) through observations, performs specific actions, and receives a reward. The virtual environment is the world in which the learner works. The observation is state-related: the learning agent receives knowledge about the current state of the system. Every piece of information contained in a system is almost impossible to know, therefore only a selected subset of real information is provided for the learning agent in the form of an

observation. Deep Reinforcement Learning (DRL) is a method of Reinforcement Learning that employs Deep Neural Networks (DNN) to approximate the value function and the representation of the state and the action space. The issue is that dispatching rules are not always the best ones; depending on the configuration of the problem and the conditions, one rule may be superior to another. In the literature the decision making problem related to the best dispatching option to choose in a flow shop has been assessed by [14]. The authors consider as a point of strength not to require training but only the optimization of the network's weights to achieve good performances. Otherwise, we investigate how develop the network of a suitable dimension with certain values of hyperparameters to accomplish this task. Simulation has been extensively used to investigate the performance of dispatching rules. Aside from some general and common findings, one widely accepted conclusion is that on a global scale, no dispatching rule is superior to the others. Certain ones, such as shortest processing time (SPT), perform well on some performance measures, such as mean flow time (the amount of time jobs spend in the system), but poorly on others, such as maximum job lateness. As a result, their performance is highly dependent on the studied system's configuration, operating conditions, and the performance criterion used to test the rules. As a result, decision-makers can have difficulty determining which dispatching rules are best suited to their problem. Simulation samples are used as training sets in the learning approaches that have been proposed. These sets provide examples of dispatching rule choices that have resulted in good or poor results, allowing an automated system to learn. The production line simulated is based on the work of Hoop and Sperman. The authors in [6] investigated the behavior of a CONWIP production line under various conditions. The scenarios depict the best and worst possible performance of a CONWIP production line, respectively. They studied the action of a production line in a real-world scenario (the Practical Worst Case, PWC) in which the processing times of job are exponentially distributed through the workstations, in a balanced line (the average working time is the same for each phase of work). Since system entities do not respond autonomously to environmental changes in a CONWIP [1] we propose a system/model that dynamically identifies, with the use of DQN, the best rule to use for job processing according to the system condition.

3 The Proposed Method

In recent years, a new algorithm known as deep Q-network (DQN) has been designed; it combines a classic RL algorithm known as Q-Learning with a deep neural network (DNN). Mnih in [13] proposed this algorithm. DQN is an RL tool and an extension of the Q-learning approach in which a deep neural network replaces the state-action tables. The DQN's learning of the value function is affected by weight changes based on the loss function: $L_t = (E[r + \gamma\ max_a\ Q(s_{(t+1)}, a_t)] - Q(s_t, a_t))^2$ in which E[r + max_a Q($s_{(t+1)}, a_t$)] represents the optimum predicted reward associated with the transition to the state $s_{(t+1)}$; r is the reward associated with the action a_t and the state s_t; is the

discount factor used to balance immediate and potential reward; and $Q(s_t, a_t)$ is the network's approximate value. Back-propagation is used in the network to propagate the errors calculated by the loss function, which fits the principle of gradient descent. The policy's behaviour is determined by a ε -greedy approach to strike a balance between exploring new states and manipulating existing good policies. Regarding the method proposed, the reward function is formulated in accordance with the work of Hoop and Sperman, so the function will be linked to the throughput (TH) of the line. The function have its intersection with the x-axis, representing the TH_{mobile} of the line, at the TH relative to the PWC, TH_{PWC}; it is asymptotic to zero, because the TH is a quantity that cannot take negative values; and it has its maximum in the TH relative to the best case scenario. The TH corresponding to the best case is compensated with the highest reward, and values greater than this are evenly and equally weighted.

$$reward = \begin{cases} \frac{\log(\frac{w_0+W-1}{W} \frac{TH_{mobile})}{r_b}}{\log((\frac{w_0+W-1}{W})^{\frac{1}{score_{max}}})}, & if \quad TH_{mobile} \in [0; r_b] \\ score_{max}, & if \quad TH_{mobile} \in [r_b; \infty] \end{cases} \tag{1}$$

In (1), TH_{mobile} is the throughput calculated in a time window of 240 min, w_0 is the critical WIP of the production line, W is the amount of the WIP set constant in our CONWIP flow shop and r_b is the rate of the workstation that has the highest long-term utilization that is the TH in the best case (Hopp and Sperman in [6]). The $score_{max}$ in the (1) is the max reward that the system can achieve and in this work we chose to set it at 100 in order to balance the penalization and/or rewarding and the generalization of the learning. Shi et al. in [20] looked into the issue of how to model the state in an RL approach. The aim of [20] is to schedule tasks in the production line avoiding conflicts. In this paper, we have modelled the state that takes into account the jobs' characteristics as well as the line's current parameter. The vector of observations ($S_1, ..., S_{20}$ in

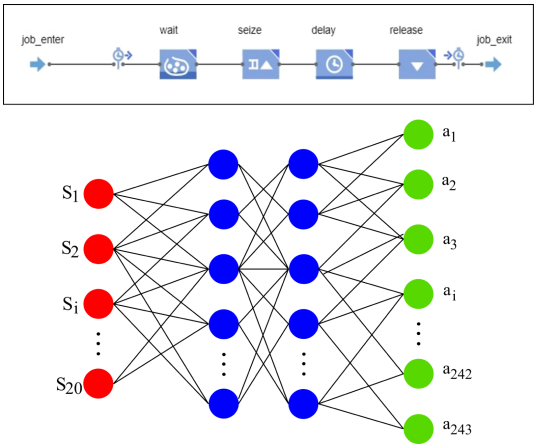

Fig. 1. The representation of the approach proposed

Fig. 1) that the DQN algorithm uses as input for training consists of the number of jobs in the queue, the sum of the processing times in the queue on each machine, the standard deviation of the processing times in the queue, and the predicted utilization on each machine. The learning agent selects the rule to use for each machine and schedules jobs for this delta T based on the chosen rule for an interval of time equal to the raw time of the line. The space of possible actions that the system can execute is linked to the scheduling rule chosen; each machine can choose between three different rules. The first rule is First In First Out (FIFO). The second is SPT, which schedules jobs with the shortest processing times. The third is longest processing time LPT. Each action represents a possible combination of rules for the five machines, e.g. [SPT, SPT, FIFO, LPT]. The number of potential choices would be 3^5, that is 243, using three rules for each of the five machines (see Fig. 1).

4 Experimental Approach

The most important aspect of this work is the development of a system, which allows us to dynamically select the processing rule for each machine and the evaluation of the DQN network's characteristics. We used Anylogic, a multi-method simulation software, and a framework called Reinforcement Learning for Java (rl4j), which is built into the DeepLearning4J library. The model is a flow shop made up of 5 workstations/machines that is simulated using Anylogic's discrete event simulation (DES) tool. At the beginning of the simulation an initial number of jobs are injected in the line, as the production system is a CONWIP, this number will be kept constant in the system. The jobs that are processed in the system are modeled as agents using a simple state-chart (queue-working-final state), and their processing times are determined by an exponential distribution with a mean of 10 min, which has a high degree of variability and is more reflective of a real manufacturing system. The required functions are added to the simulation model to enable communication between the model and the RL system. In this paper, an experimental campaign was built to determine the best network configuration in terms of size and hyperparameters. The hyperparameter's values used were chosen based on the scientific literature and the characteristics of our problem. The learning rate has been set to 0,001, which is an acceptable value since a high coefficient (e.g., 1) causes parameters to leap, while a small one (e.g., 0.00001) causes them to inch along steadily [18]. Regularization is a technique for avoiding overfitting. The "L2" regularization algorithm is used, which adds a term to the objective function that decreases squared weights. L2 improves generalization, smoothes model output as input moves, and assists the network in ignoring weights it does not need [18]. As in [16], RMSProp (for Root Mean Square Propagation) is used as a gradient-ascent algorithm, and it is a process in which the learning rate is adapted for each parameter in the network. The hyperparameter γ, that is the discount factor, is set to 0.99. The parameter defines how valuable future rewards are. This aids in demonstrating the convergence of specific algorithms. We also compare

the performance using Adam (a more recently developed updating technique) as in [17], it derives learning rates from estimates of first and second moments of the gradients. We have investigated 8 training experiments: 2 different size of the network (1 hidden layer-150 nodes and 2 hidden layers-300 nodes); 2 values of the L2 regularization (0-0.001); and 2 types of updating technique RMSProp and Adam. The L2 regularization is what makes the learning process more general, allowing the learning agent to complete its task without overfitting. Since we want to see how the structure affects the results, we aggregate the data and look at the pattern of the performance parameters, in terms of rule chosen, when changing the number of layers and the values of hyperparameter. The learning agent frequently chooses the SPT (see Fig. 2), rule for which we have performance of high TH almost equal to the TH of the best case studied by Hopp and Sperman in [6], since the reward function is modeled in optics of maximization of the TH. Discussing the results, we can say that when the process of learning is more generic (L2 = 0.001), the policy resulting learns to achieve a good combination of throughput (TH) and cycle time due to the choice of the SPT rule which is known to be one of the better rule to solve the scheduling problem when maximise the TH. Instead, when the regularization is set to 0, the system returns a policy that chooses even the other two rules (FIFO and LPT). Overall, we can say that since we have set the goal of maximize the TH, the proposed tool has learnt how to schedule a good combination of DR. Considering the performance linked to the dimension of DNN, since the number of time the SPT rule is chosen by the learning agent is greater in the case the DNN is smaller (1 hidden layer and 150 nodes), we can say that we can also have smaller training time. Discussing the learning process it converges in the 100 epochs considered. It is true in most of the setting in terms of mean reward, as it increase over time until it reaches a maximum (Fig. 3).

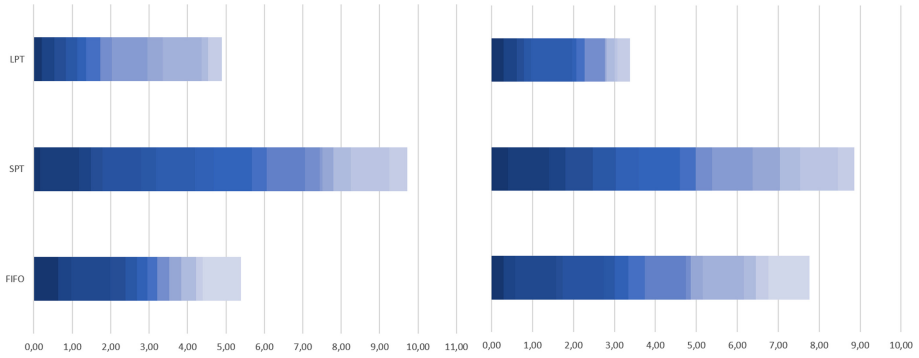

Fig. 2. Frequency of the DR choice. On the left, 1 Hidden Layer 150 nodes. On the right, 2 Hidden Layers 300 nodes.

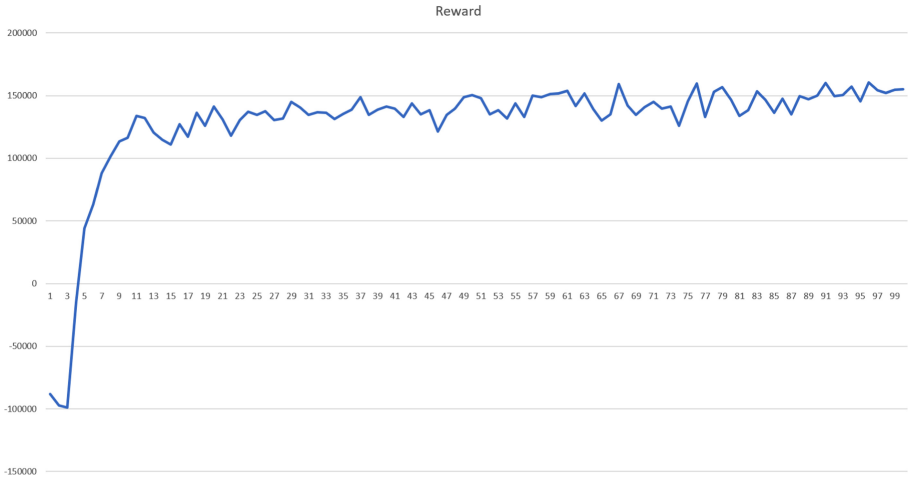

Fig. 3. The average reward per epoch (1 Hidden Layer 150 nodes, l2=0, Adam).

5 Conclusion

In this paper we presented an intelligent system consisting of machines and a learning agent developed with DQN that have decision-making autonomy and intelligence to learn about rapidly changing environments. What is new about the approach proposed is how the state, the action space and the reward are formulated. We modeled the state by taking into account the characteristics of the jobs in the queues of the working station. The action space is the rule chosen by each machine and the reward function is modelled considering the maximization of the throughput and the work of [6]. The simulation model is a flow shop made up of 5 machines. The learning agent chooses a rule for each machine. We have investigated 8 training experiments changing the size of the network, and two of the hyperparameters of the training. The method proposed is validated considering the results of the simulation, according to which, for a reward function focused to maximize the throughput, the system chooses for the most of time the SPT rule in any scenario. These results can put the foundation for a wider research, considering more complex production systems and also other dispatching rules that are much more feasible in a real manufacturing environment. In order to make the proposed tool more generic and applicable in every manufacturing configuration, it would be possible to modify some parameters and scenarios. In a practitioner point of view, this approach will generally provide managers with a method for decision-making to optimize production system control in highly complex and dynamic environments.

Acknowledgement. The authors would like Tecnologica S.r.l. for granting the research.

References

1. Bendul, J.C., Blunck, H.: The design space of production planning and control for industry 4.0. Computers in Industry 105, 260–272 (2019). https://doi.org/10.1016/j.compind.2018.10.010
2. Chen, S., Fang, S., Tang, R.: A reinforcement learning based approach for multi-projects scheduling in cloud manufacturing. Int. J. Prod Res. **57**(10), 3080–3098 (2019). https://doi.org/10.1080/00207543.2018.1535205
3. Dolgui, A., Ivanov, D., Sethi, S.P., Sokolov, B.: Scheduling in production, supply chain and Industry 4.0 systems by optimal control: fundamentals, state-of-the-art and applications. Int. J. Prod. Res. **57**(2), 411–432 (2019). https://doi.org/10.1080/00207543.2018.1442948
4. Grassi, A., Guizzi, G., Santillo, L.C., Vespoli, S.: Assessing the performances of a novel decentralised scheduling approach in Industry 4.0 and cloud manufacturing contexts. Int. J. Prod. Res. 1–20 (2020). https://doi.org/10.1080/00207543.2020.1799105
5. Heger, J., Branke, J., Hildebrandt, T., Scholz-Reiter, B.: Dynamic adjustment of dispatching rule parameters in flow shops with sequence-dependent set-up times. Int. J. Prod. Res. **54**(22), 6812–6824 (2016). https://doi.org/10.1080/00207543.2016.1178406
6. Hopp, W.J., Spearman, M.L.: Factory Physics: foundation of manufacturing management. Irwin/McGraw-Hill, second edition edn. (2001)
7. Ivanov, D., Sokolov, B., Chen, W., Dolgui, A., Werner, F., Potryasaev, S.: A control approach to scheduling flexibly configurable jobs with dynamic structural-logical constraints. IISE Trans. **53**(1), 21–38 (2021). https://doi.org/10.1080/24725854.2020.1739787
8. Kim, H., Lim, D.E., Lee, S.: Deep learning-based dynamic scheduling for semiconductor manufacturing with high uncertainty of automated material handling system capability. IEEE Trans. Semicond. Manuf. **33**(1), 13–22 (2020). https://doi.org/10.1109/TSM.2020.2965293
9. Kim, Y.G., Lee, S., Son, J., Bae, H., Chung, B.D.: Multi-agent system and reinforcement learning approach for distributed intelligence in a flexible smart manufacturing system. J. Manuf. Syst. **57**(August 2019), 440–450 (2020). https://doi.org/10.1016/j.jmsy.2020.11.004
10. Kusiak, A.: Convolutional and generative adversarial neural networks in manufacturing. Int. J. Prod. Res. **58**(5), 1594–1604 (2020). https://doi.org/10.1080/00207543.2019.1662133
11. Lee, J.H., Kim, H.J., Lee, J.H.: Reinforcement learning for robotic flow shop scheduling with processing time variations variations. Int. J. Prod. Res. 1–23 (2021). https://doi.org/10.1080/00207543.2021.1887533
12. Lin, C.C., Deng, D.J., Chih, Y.L., Chiu, H.T.: Smart manufacturing scheduling with edge computing using multiclass deep Q network. IEEE Trans. Industr. Inf. **15**(7), 4276–4284 (2019). https://doi.org/10.1109/TII.2019.2908210
13. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015). https://doi.org/10.1038/nature14236
14. Mouelhi-Chibani, W., Pierreval, H.: Training a neural network to select dispatching rules in real time. Comput. Industr. Eng. **58**(2), 249–256 (2010). https://doi.org/10.1016/j.cie.2009.03.008
15. Oukil, A., El-Bouri, A.: Ranking dispatching rules in multi-objective dynamic flow shop scheduling: a multi-faceted perspective. Int. J. Prod. Res. **59**(2), 388–411 (2021). https://doi.org/10.1080/00207543.2019.1696487

16. Park, I.B., Huh, J., Kim, J., Park, J.: A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities. IEEE Trans. Autom. Sci. Eng. **17**(3), 1420–1431 (2020). https://doi.org/10.1109/TASE.2019.2956762
17. Park, J., Chun, J., Kim, S.H., Kim, Y., Park, J.: Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning. Int. J. Prod. Res. 1–18 (2021). https://doi.org/10.1080/00207543.2020.1870013
18. Patterson, J., Gibson, A.: Deep Learning: A Practitioner's Approach. 1st edit edn, O'Reilly, Sebastopol (2017)
19. Priore, P., Ponte, B., Puente, J., Gómez, A.: Learning-based scheduling of flexible manufacturing systems using ensemble methods. Comput. Industr. Eng. **126**(September), 282–291 (2018). https://doi.org/10.1016/j.cie.2018.09.034
20. Shi, D., Fan, W., Xiao, Y., Lin, T., Xing, C.: Intelligent scheduling of discrete automated production line via deep reinforcement learning. Int. J. Prod. Res. **58**(11), 3362–3380 (2020). https://doi.org/10.1080/00207543.2020.1717008
21. Xia, K., et al.: A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment, interfaces and intelligence. J. Manuf. Syst. 1–21 (2020). https://doi.org/10.1016/j.jmsy.2020.06.012