Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 107 (2022) 1198–1203

www.elsevier.com/locate/procedia

55th CIRP Conference on Manufacturing Systems

Reconfigurable manufacturing system scheduling: a deep reinforcement learning approach

Jiecheng Tang, Yousef Haddad, Konstantinos Salonitis*

[a]*Sustainable Manufacturing Systems Centre, Cranfield University, Bedford, MK43 0AL, UK*

* Corresponding author. Tel.: +44 (0) 1234 758347. *E-mail address:* k.salonitis@cranfield.ac.uk

Abstract

Reconfigurable Manufacturing Systems (RMS) bring new possibilities toward meeting demand fluctuations while, at the same time, challenges scheduling efficiency. This paper presents a novel approach that, for the scheduling problem of RMS on multiple products, finds a dynamic control policy via a group of deep reinforcement learning agents. These teamed agents, embedded with a shared value decomposition network, aim on minimising the make-span of a constant updating order group by guiding a group of automated guided vehicles to move modules of machine, raw materials, and finished products inside the system.

Keywords: Reconfigurable Manufacturing System; Multi-agent System; Deep Reinforcement Learning; Flexible Job-shop Scheduling Problem

1. Introduction

The reconfigurable manufacturing system (RMS) paradigm was introduced in the last decade of the 20th century [1, 2]. An RMS is typically designed for balancing fluctuating demand levels [2]. However, the roll-out in the industry is rare [3, 4] in the past two decades due to a number of challenges. Unlike a conventional production line with predefined machines and fixed architecture, an RMS consists of several kinds of adjustable components, such as several reconfigurable machine tools, or even a flexible material handling system. These flexible parts make designing an optimal control system [5] for an RMS an arduous task. An adequate control system wire-walks among production scheduling, process planning, reconfiguration control, and task assignment, amongst others.

To observe an RMS system and optimise its objectives, multi-agent architecture is one of most often used control systems of RMS [6]. Under this architecture, the RMS scheduling problem is considered as a variant of the flexible job-shop scheduling problem [7]. Recently, reinforcement learning (RL) applications on finding close-to-optimal scheduling policy is becoming more powerful and versatile. Different from mathematical programming or meta-heuristic algorithms [8], RL tackles this NP-hard [8, 9] problem by formulating the problem into a Markov decision process. RL asks an agent / agents to keep interacting with the environment to gain transitions and rewards. With adequate experience on interacting, an RL agent / agents become(s) “far-sighted” and “resilient” to a dynamic environment like RMS. Under multi-agent environment, RL agents could also gain cooperation ability [10] for maximising the global rewards.

RL for RMS scheduling has been previously investigated in [11] where the authors demonstrated its great potential. This paper builds on the work of [11] and presents an upgraded RMS control policy training framework based on a deep reinforcement learning method and a multi-agent discrete event simulation environment. The framework is able to reduce the complexity of the control system by assigning limited action space to every agent. These agents self-organise and provide an optimal policy based on an artificial neural network.

The rest of this paper is organised as follows: section 2 briefly reviews the relevant research on multi-agent RMS control system and multi-agent RL on scheduling optimisation. Section 3 describes the information flow among RL agents and

10.1016/j.procir.2022.05.131

RMS environment, and explains how agents train a universal policy inside a neural network. Section 4 presents two numerical case studies based on the proposed framework chasing the optimal on-time delivery, first one, although not entirely successful, helped develop the modelling approach whereas the seconds one provides acceptable outcome. The final section presents the concluding remarks and future research directions.

2. Related work

Problems pertaining to reconfigurable manufacturing systems (RMS) design, flexible job-shop scheduling, and reinforcement learning on scheduling have been extensively investigated in the contemporary literature. This section will, therefore, briefly highlight key papers in the extant literature.

The authors in [5] highlighted two main categories of drivers of responding to change when designing an RMS. The first being demand changing drivers, which include changes in product, manufacturing process, lead time, volume, quality level and product price. While the other is production system environment changing drivers, which include changes of supply, resources, legislative environment, and readjustment cost. Key reviews of RMS [3, 12, 13] suggested volume fluctuation is one of the most important attributes to investigate. Architecture of RMS is another key feature to be considered while modelling such a production environment.

The authors in [14] modelled an RMS based on complex adaptive system theory as a multi-layer multi-agent system. In the aforementioned model, a workshop manager agent manages several workshop agents. Through a rule-based system (i.e. a series if-else conditions), each workshop agent can control several resource agents. Based on an agent-communication language, the author in [15] designed a multi-robot system that shares knowledge inside the system. With RL-based optimisation, this system separates a global plan into local tasks and provides a better solution than meta-heuristic algorithms, such as ant colony optimisation and particle swarm optimisation. Other papers [9, 16–18] suggest similar results that RL normally outperform meta-heuristic approaches in complex systems. For example, two decades ago, the authors in [16] used Q-III reinforcement learning method to choose the best real-time system operating mode among 3 kinds of fixed dispatching rules. The authors in [11] developed a framework which can provide an optimal online task-launching policy based on a trained Deep Q Network (DQN) in a simplified RMS. The authors in [19] modelled a flexible manufacturing systems (FMS) as a Petri net then used multi-agent RL to outperform conventional heuristics and meta-heuristics on finding the best dispatching rule on every single machine.

A material handling system (MHS) is one of major components of an RMS which help transporting materials and finished product from point to point [3]. Automatic guided vehicles (AGVs) are widely used in FMS [20, 21] and some novel production lines [21] as MHS. The authors in [22] solved an AGV task scheduling problem in an FMS by using collaborative evolutionary genetic algorithm which belong to meta-heuristic approaches. The authors in [21] used Q-learning, an RL approach, to solve a multi-AGV scheduling problem on a production line. The result suggest that their approach has more advantage on minimising the average makespan with more complex systems. After the game of Go is mastered in 2013 [23], deep reinforcement learning (DRL) gained significant attention. Several advanced techniques had been invented for helping agent converge more quickly and for keeping the policy stable after convergency such as Double DQN [24] and prioritised experience replay [25]. Discrete-event simulation (DES) helps transferring a production environment into a Markov process. The authors in [26] speeded up the optimisation process of a sim-heuristic framework by applying a genetic algorithms on a DES. The authors in [27] optimised the schedules of a linear production line, a parallel production line, and a re-entrant production line by combining DES and DRL. The result massively outperforms the fixed dispatching rules in all three systems, while showing robustness to time randomness. Considering the RMS and DRL's unique features and challenges, this paper presented a framework bridges these two areas together on a simplified scheduling problem.

3. A novel framework of RMS scheduling

Compared to a flexible manufacturing system aims on providing generalized flexibility, reconfigurable manufacturing systems (RMSs) with structural adjustable components bring great potential on handling fluctuating and customised demands while a proper mechanism for scheduling is needed to release such potential. This section describes a scheduling problem in a simplified grid-world shape RMS discrete-event simulation (DES) and a deep reinforcement learning-based scheduling agent training process for gaining a scheduling policy. The general idea behind reinforcement learning (RL) is asking an agent with limited action space to interact with a predefined environment. With the reward feedbacked from the environment, an RMS-DES in this paper, the agent can constantly update its prediction on the environment and gain higher long term accumulated reward.

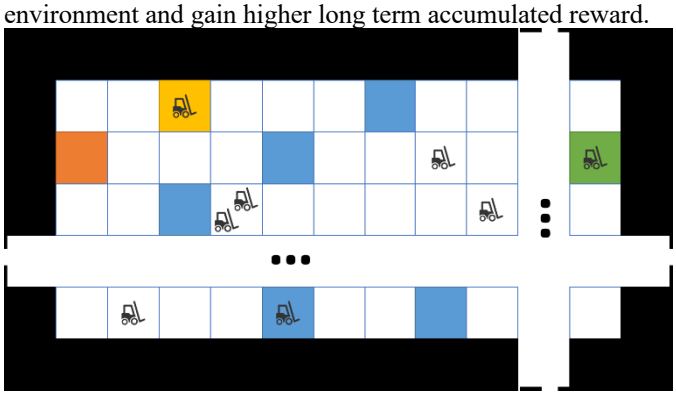

Fig. 1. Grid-world shape RMS-DES

3.1. A Grid-world shape RMS

Fig. 1 presents the RMS-DES used in this paper. This simplified $m \times n$ grid-world-shaped RMS, with several automated guided vehicles (AGVs) presented as fork-lift icons in Fig. 1, is used to perform discrete-event simulation (DES). These AGVs, controlled by a centralised-trained agent, move "freely" inside the RMS. Several reconfigurable machine tools

(RMTs), presented as blue blocks in Fig. 1, are randomly assigned at first place. All of these machine tools are fixed in place after initialisation. The agent asks AGVs to bring modules and material to the machine tools, which themselves are reconfigurable. These RMTs in the RMS-DES model react passively when AGVs attempt to interact with them. This RMS has an order list which contains several randomly generated orders to mimic fluctuating demand. Each order requests one kind of product with a random batch size and a due date.

The distributed executing action space for an AGV is 6, which include 4 kinds of movement: up, down, left, and right. The other two are waiting and interacting. Inside the RMS, AGVs are allowed to spend a tiny negative local reward, to move in any direction until they hit a wall. If an AGV is asked to hit a wall, this AGV will be forced to stay in front of the wall until next turn and receive a big local negative reward. If the agent asks an AGV to stay at the current location and wait until next event, this AGV will receive a small local negative reward from the RMS-DES. AGVs can also interact with RMTs and 3 kinds of warehouses across the RMS. The first one is a Module Storage (MDS) showed as an orange block at left edge in Fig-1. MDS provide certain kind of module that can be installed to RMTs. RMT can only manufacture raw material with modules installed. A raw material delivers from the second kind of warehouses, Material Storage (MTS). AGVs can only deliver raw material to RMT installed correct module then trigger a manufacturing process event. After this process, the material in an RMT change to a finished good. AGV now need to transfer this product to the third kind of warehouses, a Finished Good Storage (FGS), where RMS check stock level when an order is due. After every AGV interact with RMS-DES, the RMS-DES will check whether there is an order need to deliver. If there is a due order and FSG has sufficient inventory, RMS will broadcast a huge global positive reward to all AGVs inside the system. On the other hand, if an FSG does not have enough stock while an order is due, all AGV agent will receive a big negative global reward. The order met due date would be replaced by another random generated order regardless of it has been fulfilled or not. All possible event due to interactions between AGV and RMS-DES are listed in Table 1.

Table 1. All RMS-DES possible AGV interaction events

Scenario	Description
1	An AGV moves to an available place. RMS will update this AGV's location and return a tiny local negative reward.
2	An AGV moves toward a wall. RMS will freeze this AGV until the next turn and return a big local negative reward.
3	An AGV remains for one turn. RMS will freeze this AGV until next turn and return a small local negative reward.
4	An AGV with an empty cargo attempts to interact with the module storage. RMS will randomly pick one kind of modules and return a decent local positive reward.
5	An AGV carries a module and attempts to interact with the module storage. RMS will randomly unload the module from the AGV, or switch to a one kind of module and return a tiny local negative reward.
6	An AGV carries a non-module part and attempts to interact with the module storage. RMS will freeze the AGV until the next turn and return a tiny local negative reward.
7	An AGV with an empty cargo attempts to interact with the material storage. RMS will randomly pick one kind of materials and load it to this AGV and return a decent local positive reward.
8	An AGV carries a material and attempts to interact with the material storage. RMS will switch to one kind of materials randomly and return a tiny local negative reward.
9	An AGV carries a module or a finished good and attempts to interact with the material storage. RMS will freeze this AGV until next turn and return a small local negative reward
10	An AGV carries a finished good and attempts to interact with the finished good storage. RMS will unload the finished good from this AGV to the inventory and return a decent local positive reward and broadcast a global reward
11	An AGV attempts to interact with the finished good storage without a finished good. RMS will freeze this AGV until the next turn and return a small local negative reward
12	An AGV with an empty cargo attempts to interact with an RMT that has a finished good. RMS will unload the finished good from this RMT to the AGV, set this RMT to free from next turn and return a huge local positive reward.
13	An AGV carries a material attempts to interact with a free RMT with a matched module. RMS will unload the material from the AGV to this RMT, set this RMT to busy mode until it finishes production and return a huge local positive reward.
14	An AGV carries a material attempts to interact with a free RMT with an un-matched module. RMS will freeze this AGV until next turn and return a small local negative reward.
15	An AGV carries a module attempts to interact with a free RMT with a different module. RMS will switch the module from the AGV to this RMT, set this RMT to busy until it finishes reconfiguring, and return a tiny local negative reward.
16	An AGV carries a module attempts to interact with a free RMT with the same module. RMS will freeze this AGV until next turn and return a small local negative reward.
17	An AGV attempts to interact with a busy RMT. RMS will freeze this AGV until next turn and return a small local negative reward.
18	After all AGV interaction, FGS has enough stock for a due order. RMS will return a huge global positive reward.
19	After all AGV interacted with RMS-DES, FGS has not enough stock for a due order. RMS will return a big global negative reward.
20	An AGV decide not moving until next turn. RMS will return a tiny local negative reward.

3.2. Prioritised experience replay (PER)

An experience replay buffer can significant improving agent converging speed and stability after convergency [28]. However, given that moving inside the system would be the major AGV action inside the RMS-DES, an effective mechanism is needed to reduce the replay probability of the transitions due to these actions. An RL agent with Prioritised Experience Replay (PER) can learn more frequently from transitions which have larger reward expectation difference. Using the absolute value of the already-computed TD error δ as the ranking criterion, PER automatically decides which transition to store into a database with sum-tree data frame and which experiences worth replaying to the RL agent. Comparing to a common DQN transition $\{s_t, a_t, r_t, s_t'\}$, the transitions stored in a PER are in form $\{s_t, a_t, r_t, s_t', p\}$ where p represents a priority value equal to $|\delta| + \epsilon$, and ϵ is a small positive number

that prevents a transition from never being revisited if this transition have zero error.

When sampling transitions from a PER, uniform random sampling could get a lot of useless data. However, only picking up the 'useful' data could also lead to over-fitting. Stochastic prioritisation sampling, similar to the epsilon-greedy strategy in RL training, is used in this paper. The probability of a certain transition being sampled can be presented as $P(i) = p^{\alpha}_i / \sum_k p^{\alpha}_k$ where k is the ratio between total transitions in PER and the mini-batch needed to be sampled, and α is a hyperparameter to control how much prioritisation is used. To control the bias brought by stochastic prioritisation sampling, an importance-sampling weight is introduced as $w_i = (N \cdot P(i))^{-\beta}$ where β is an incrementing value from β_0, another hyperparameter, to 1. Combining the importance-sampling weight w with TD error δ, Q-network can be updated and annealing from bias.

3.3. RMS scheduling by reinforcement learning

By taking the advantages of DES and PER, the proposed RL scheduling policy can be described in Algorithm 1 below. To start training the scheduling agent, some hyperparameters are needed including prioritisation exponent, initial anneal exponent, learning step size, buffer size, and all other settings for RMS-DES. The training process is performed in Double DQN form where two separated neural networks (NNs) are used during training. The primary NN is the Q network which updates all the time for best performance and the secondary NN is the Q_{target} network which gets updated from time to time.

Algorithm 1. Double DQN-Based Method with PER

Input: Exponents α and β_0, mini-batch k, step-size η, minimum random probability ε, replay capacity N

Output: A Smart Scheduling Agent

Initialise a prioritised experience replay buffer D with N capacity, Δ=0

Initialise an online network Q with random weights θ and a target network Q_{target}

Initialise an RMS with n RMTs and v AGVs

for episode = 1 **to** iteration limit M **do**:

  for t = 1 **to** T **do**:

    Reset an Accumulated Global Reward R = 0

    for AGV_{01} **to** AGV_v **do**:

      Observe the current state s^{AGV}_t

      Choose an action a^{AGV}_t according to Q with ε probability for a random action

      Execute a^{AGV}_t on DES and get a local reward, r^{AGV}_t, for current AGV, a global reward, and a new state s^{AGV}_t'

      Add the global reward to R while store the transition $\{s^{AGV}_t, a^{AGV}_t, r^{AGV}_t, s^{AGV}_t'\}$ in a temporary list tl

      Store v transitions $\{s^{AGV}_t, a^{AGV}_t, r^{AGV}_t, s^{AGV}_t'\}$ from tl with maximum priority $p_t = max_{i<t}\ p_i$ to D

    end for

    for j = 1 **to** k **do**:

      Sample a transition $j \sim P(j) = p^{\alpha}_j / \sum_i p^{\alpha}i$

      Compute importance-sampling weight $w_j = (N \cdot P(j))^{-\beta} / max_i w_i$

      Compute TD-error $\delta_j = r_j + \gamma_j Q_{target}(s_j, argmax_a Q(s_j, a)) - Q(s_{j-1}, a_{j-1})$

      Update transition priority $p_j \leftarrow |\delta_j|$

      Accumulate weight-change $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q(s_{j-1}, a_{j-1})$

    end for

    Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$ then reset $\Delta = 0$

    Update Q_{target} from Q time to time

  end for

end for

The scheduling agent manipulates the AGVs inside the RMS-DES one by one. The observation for each AGV, s^{AGV}_t, contains several elements include current AGV location and carriage, the order list status, all RMT status, and all other AGV locations and their carriages. When current AGV finishes an interaction, the new observation from this AGV and a local reward well be stored in the temporary place waiting the global reward accumulate from all AGVs. Once all AGVs finishes interacting with the RMS-DES, the local rewards stored in temporary transitions would be plus/minus the accumulated global reward and form final transitions. These transitions would be then stored into the PER for training agent. This process shows in Fig. 2.

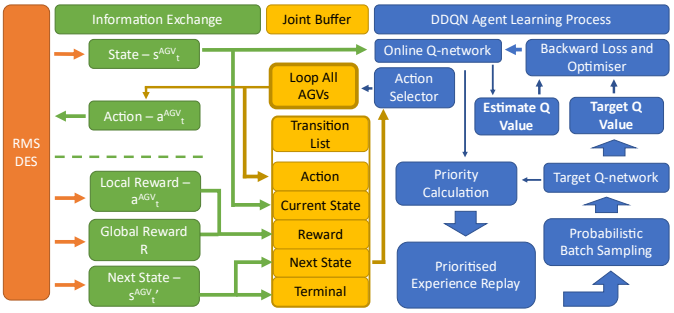

Fig. 2. Double DQN agent training interaction flow-chat

4. Case study, results & analysis

To test the effectiveness of current framework, two numerical case studies had been implemented in this paper. Both case studies request a fully automated reconfigurable manufacturing system (RMS) to produce three (3) kinds of products. Every product request one unique kind of module installed on one reconfigurable machine tool (RMT). With a correct module, an RMT can keep taking the appointed kind material and manufacturing product. The simulation time for one episode is 28800 minutes.

4.1. Case study one: a worthy exploration

The first case study built a 20×13 grid-world shape RMS-DES which installed 15 RMTs and 25 AGVs. The reconfiguring time from one module to another module is 5 minutes. The manufacturing time for one product is 3 minutes. The orders in the order list have a batch size from 50 to 1500 and deliver time gap is a random value from 1000 to 5000 minutes. All RMTs and AGVs initialised empty. The finished good inventory initialised with 500 stocks for all three

products. The reward setting for every scenario mentioned in last section in shown in table 2.

Table 2. Numerical case 1 reward setting

Scenario #	Local Reward	Global Reward	Scenario #	Local Reward	Global Reward
1	-5	N/A	11	-1	N/A
2	-1k	N/A	12	+10k	N/A
3	-20	N/A	13	+10k	N/A
4	+1k	N/A	14	-3	N/A
5	-1	N/A	15	-1	N/A
6	-2	N/A	16	-3	N/A
7	+1k	N/A	17	-3	N/A
8	-1	N/A	18	N/A	+10k × order
9	-2	N/A	19	N/A	-1k × order
10	+10k	+10k	20	-1	N/A

The reward quickly converges after only several rounds of training. However, there is not on-time delivery at all after the initial stock run out. To investigate the phenomenon, one AGV's movement in first training episode had been sliced out and presented in Figure 3.

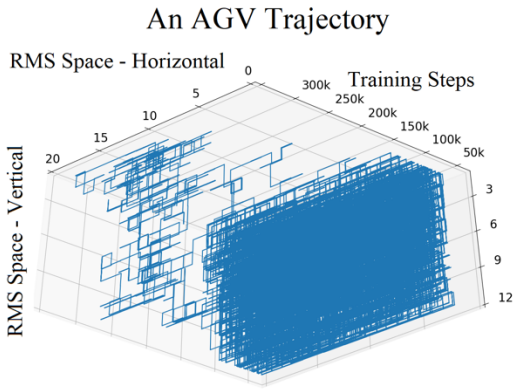

Fig. 3. Double DQN agent training interaction flow-chat

Figure 3 is a 3D plot. Alongside the grid-world shape RMS, a third axis had been added to represent the simulation time. The trajectory clearly shows this AGV randomly wandering inside the system at very beginning. However, with the greedy value reduce, agent tend to ask AGVs stay at the same place in case gaining too many negative rewards. Considering this situation, the training process cancelled. A new reward mechanism had been used for a new case study.

4.2. Case study two: a fair approach

For case study two, the size of RMS was reduced to 10×7, a quarter compared to the previous case. This RMS has 5 RMTs and 15 AGVs. The new reward mechanism, shows in Table 3, increases the negative reward an AGV will receive if it chooses to keep at current location. To increase the stimulation during the process, the local rewards generated by loading materials from material storage and deliver finished goods are increased.

Every time a simulation episode starts, two RMTs will be initialised empty, one RMT will be initialised having a finished good waiting to be carried away, one RMT will be initialised with a module wait for a material and the last RMT will be initialised with an on-going manufacturing process. Every AGV is initialised with an empty cargo, a module, a material or a finished good. These limitations and behaviour compositions are placed for helping agent experience rewarding situation more frequent and constant. Training results are shown in Figures 4 and 5 presenting the reward increasing curve and actual product an RMS could produce when testing.

Table 3. Numerical case 2 reward setting

Scenario #	Local Reward	Global Reward	Scenario #	Local Reward	Global Reward
1	**-1**	N/A	11	-1	N/A
2	-1000	N/A	12	**+1M**	N/A
3	-20	N/A	13	+10k	N/A
4	+1k	N/A	14	-3	N/A
5	-1	N/A	15	-1	N/A
6	-2	N/A	16	-3	N/A
7	**+100k**	N/A	17	-3	N/A
8	-1	N/A	18	N/A	+10k × order
9	-2	N/A	19	N/A	-1k × order
10	**+1M**	+10k	20	**-2**	N/A

The training reward trend shows in Fig. 4. It is clearly that the agent quickly learned not to hit the wall.

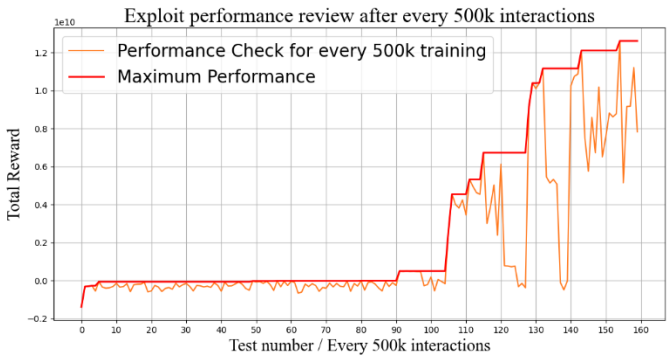

Fig. 4. Reward trend

However, the training did not progress for a very long time until a sudden leap around 45 millions trials. Figure 5 below shows the actual sum of products delivered from the RMS. Considering that the orders are randomly generated, the results look encouraging.

Fig. 5. Productivity trend

5. Conclusion

The results obtained from the case studies suggest that the proposed training framework is applicable, and efficient, when applied to medium and small-scale RMS scheduling problems. With carefully designed rewarding mechanism, some imitation demonstrations, the composition of behaviours, and most importantly, enough training episodes, this DDQN-based scheduling policy training approach can provide encouraging result. The instability can be mitigated if this framework engages with other advanced technologies, such as bagging strategy and duelling network architecture. Although the case study clearly demonstrated that the proposed framework can generate desirable results, the scalability of current framework should be further validated. The simplified grid-world RMS has been developed at high-level, in order to improve the applicability to wide RMS settings, therefore applying the framework on a real case study, with more real-life complexities is necessary for further validation. Future work on more complex real-life case studies can also further integrate and expand cooperative behaviours among agents. Another possible research direction can be to upgrade the RMS-DES to argumentation-based multi-agent system to examine the improvement in information-exchange efficiency.

References

[1] Koren Y, Ulsoy AG (1997) Reconfigurable manufacturing systems, engineering research center for reconfigurable machining systems (ERC/RMS) report# 1, the university of michigan. Ann Arbor

[2] Koren Y, Heisel U, Jovane F, et al (1999) Reconfigurable Manufacturing Systems. CIRP Ann 48:527–540.

[3] Koren Y, Gu X, Guo W (2018) Reconfigurable manufacturing systems: Principles, design, and future trends. Front Mech Eng 13:121–136.

[4] Maganha I, Silva C, Ferreira LMDF (2018) Understanding reconfigurability of manufacturing systems: An empirical analysis. J Manuf Syst 48:120–130.

[5] Benyoucef L. Reconfigurable Design to Systems: From Manufacturing Implementation. Springer Series in Advanced Manufacturing. Cham, Switzerland: Springer Nature Switzerland; 2020.

[6] Kruger K, Basson AH (2013) Multi-agent systems vs IEC 61499 for holonic resource control in reconfigurable systems. Procedia CIRP 7:503–508.

[7] Bhargav A, Sridhar CNV, Deva Kumar MLS (2017) Study of Production Scheduling Problem for Reconfigurable Manufacturing System (RMS). Mater Today Proc 4:7406–7412.

[8] He Y, Wu G, Chen Y, Pedrycz W (2021) A Two-stage Framework and Reinforcement Learning-based Optimization Algorithms for Complex Scheduling Problems. 1–11

[9] Zhang Z, Zheng L, Weng MX (2007) Dynamic parallel machine scheduling with mean weighted tardiness objective by Q-Learning. Int J Adv Manuf Technol 34:968–980.

[10] Vinyals O, Ewalds T, Bartunov S, et al (2017) StarCraft II: A New Challenge for Reinforcement Learning

[11] Tang J, Salonitis K (2021) A Deep Reinforcement Learning Based Scheduling Policy for Reconfigurable Manufacturing Systems. Procedia CIRP 103:1–7.

[12] Napoleone A, Pozzetti A, Macchi M (2018) Core Characteristics of Reconfigurability and their Influencing Elements. IFAC-PapersOnLine 51:116–121.

[13] Yelles-Chaouche AR, Gurevsky E, Brahimi N, Dolgui A (2021) Reconfigurable manufacturing systems from an optimisation perspective: a focused review of literature. Int J Prod Res 59:6400–6418.

[14] He P, Wang QL, Yu JC (2006) A multi-agent model for reconfigurable manufacturing system based on complex adaptive system. IEEE Int Conf Intell Robot Syst 3723–3727.

[15] Oprea M (2018) Agent-based modelling of multi-robot systems. IOP Conf Ser Mater Sci Eng 444:.

[16] Aydin ME, Öztemel E (2000) Dynamic job-shop scheduling using reinforcement learning agents. Rob Auton Syst 33:169–178.

[17] Bakakeu J, Tolksdorf S, Bauer J, et al (2018) An Artificial Intelligence Approach for Online Optimization of Flexible Manufacturing Systems. Appl Mech Mater 882:96–108.

[18] Stricker N, Kuhnle A, Sturm R, Friess S (2018) Reinforcement learning for adaptive order dispatching in the semiconductor industry. CIRP Ann 67:511–514.

[19] Baer S, Bakakeu J, Meyes R, Meisen T (2019) Multi-agent reinforcement learning for job shop scheduling in flexible manufacturing systems. Proc - 2019 2nd Int Conf Artif Intell Ind AI4I 2019 22–25.

[20] Xu W, Guo S (2019) A multi-objective and multi-dimensional optimization scheduling method using a hybrid evolutionary algorithms with a sectional encoding mode. Sustain 11:.

[21] Xue T, Zeng P, Yu H (2018) A reinforcement learning method for multi-AGV scheduling in manufacturing. Proc IEEE Int Conf Ind Technol 2018-February:1557–1561.

[22] Xiao H, Wu X, Zeng Y, Zhai J (2020) A CEGA-Based Optimization Approach for Integrated Designing of a Unidirectional Guide-Path Network and Scheduling of AGVs. Math Probl Eng 2020:.

[23] Schrittwieser J, Antonoglou I, Hubert T, et al (2020) Mastering Atari, Go, chess and shogi by planning with a learned model. Nature 588:604–609.

[24] Hasselt H van, Guez A, Silver D (2016) Deep Reinforcement Learning with Double Q-Learning. Proc Thirtieth AAAI Conf Artif Intell 30:7

[25] Schaul T, Quan J, Antonoglou I, Silver D (2016) Prioritized experience replay. 4th Int Conf Learn Represent ICLR 2016 - Conf Track Proc 1–21

[26] Rabe M, Deininger M, Juan AA (2020) Speeding up computational times in simheuristics combining genetic algorithms with discrete-Event simulation. Simul Model Pract Theory 103: 102089.

[27] Shi D, Fan W, Xiao Y, et al (2020) Intelligent scheduling of discrete automated production line via deep reinforcement learning. Int J Prod Res 58:3362–3380.

[28] Long-Ji L (1992) Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine learning, 8(3-4):293–32.

Reconfigurable manufacturing system scheduling: a deep reinforcement learning approach

Tang, Jiecheng

2022-05-26

Tang J, Haddad Y, Salonitis K. (2022) Reconfigurable manufacturing system scheduling: a deep reinforcement learning approach. Procedia CIRP, Volume 107, pp. 1198-1203. 55th CIRP Conference on Manufacturing Systems 2022, 29 June - 1 July 2022, Lugano, Switzerland
https://doi.org/10.1016/j.procir.2022.05.131