Parametrization of a demand-driven operating model using reinforcement learning

Louis Duhem, Maha Benali, Guillaume Martin

Parametrization of a demand-driven operating model using reinforcement learning☆

Louis Duhem [a,*], Maha Benali [a], Guillaume Martin [b]

[a] Département de mathématiques et génie industriel, Polytechnique Montréal, 2500 Chemin de Polytechnique, Montréal H3T 1J4, Québec, Canada
[b] Département de génie industriel, Ecole des mines d'Albi-Carmaux, All. des sciences, Albi, 81000, France

A B S T R A C T

Nowadays, production and supply planning are more complex than ever before with low customer tolerance, complicated bill-of-materials, and high product variety. Most of the time, conventional approaches such as MRP and Lean approaches are not efficient. To overcome these issues, Ptak and Smith (2019) introduced Demand Driven Material Requirements Planning (DDMRP). This methodology relies on a Demand-Driven Operating Model (DDOM) which uses actual demand in a combination of strategic buffers to protect critical parts. For the past decade, research around DDMRP has been focused on proving and advancing the methodology in different industrial environments, while neglecting its parametrization. Indeed, the authors suggested general rules to set the DDOM's key parameters, yet no learning approach has been developed to set them. This present paper is the first that proposes to use machine learning to parametrize a DDOM facing unknown demand, and particularly to adjust dynamically the order spike threshold and the order spike horizon. A reinforcement learning algorithm with three different reward functions is coupled to a DDMRP flowshop simulation model facing an atypical demand including spikes. Besides studying the learning ability of the algorithm, we evaluate the performance of the model which is compared to a DDOM without parameter adjustment. The analysis shows that it is possible to drive the order spike thresholds to increase the performance of the production system, regarding customer satisfaction and stock level optimization. The findings of this paper point out the possibility to drive DDOM parameters with an automatic method using reinforcement learning.

1. Introduction

The context of production management in the early twenty-first century has been marked by a new way of consuming, which has induced ambiguity and volatility. Moreover, the supply base changed, by becoming more competitive and by creating some uncertainty around lead times, prices and demands. Consequently, the number of complexity sources has greatly increased. This congregation of phenomena in the industrial field is referred to as VUCA: Volatility, Uncertainty, Complexity, and Ambiguity (Ptak and Smith, 2019). Atypical events, such as demand spikes, are increasing, thus threatening the smooth execution of day-to-day operations. Conventional production methods, which rely on demand forecasts, fail to deal with this new complex environment.

In this context, Ptak and Smith (2019) developed a new method called Demand Driven Materials Requirements Planning (DDMRP), which aims to reduce stock levels and delivery time. The method emphasizes the information and material flow and focuses on "actual demand" to "forecast better" (Ptak and Smith, 2019). DDMRP relies on good aspects and principles of different production methods and heuristics, such as MRP (Material Requirements Planning) and Distribution Requirements Planning (DRP). It also uses the "pull and visibility emphasis" of Lean and Theory of Constraints and the "variability reduction" of Six Sigma. It brings all these production principles together into a new method and sets new production rules to reduce stock levels and increase customer service. This is made possible by putting decoupling points called buffers along the supply chain, which induces independence in the system and prevents variability effects from occurring. Early research works have started to prove that DDMRP is more efficient than traditional methods in specific industrial contexts (Azzamouri et al., 2021; Miclo et al., 2019).

DDMRP follows the principle of "position, protect and pull"

☆ This document is the result of the research project funded by Polytechnique Montréal, Canada.

* Corresponding author.
E-mail address: louis.duhem@polymtl.ca (L. Duhem).

(Pekarčíková et al., 2019). This principle relies on 5 five components that allow any Demand-Driven Operation Model (DDOM) to focus on actual demand and prioritize flow-based orders. The five components are listed below:

1. Strategic inventory positioning (position): this component defines where buffers are placed;
2. Buffer profiles and levels (protect): this component sets how much the decoupling points will be protected;
3. Dynamic adjustments (protect): this component allows to adjust the amount of protection following parameters and external factors;
4. Demand-driven planning (pull): this component enables to generate supply orders;
5. Visible and collaborative execution (pull): this component deals with the management of open supply orders.

The buffer parametrization is part of the DDMRP dynamic adjustment component: the production parameters need to be adjusted to adapt itself to the actual demand. For instance, a buffer must be able to detect an order spike that could threaten its integrity. An order spike is defined by two parameters: an order spike threshold (OST) and an order spike horizon (OSH). (Ptak and Smith, 2019) recommend general rules to calculate them, such as using a proportion of buffer levels or the average daily usage. Nevertheless, these rules do not rely on consistent and objective algorithms (Bahu et al., 2019). Only Damand et al. (2022) recently proposed using an automatic method to adjust 8 parameters (including OST and OSH) of a DDOM with a genetic algorithm. However, their algorithm does not consider a learning method and presents a weak responsivity to uncertain demand.

Next to that, reinforcement learning-based algorithms have become more and more popular to deal with supply chain management problems. Reinforcement learning is an area of machine learning relying on trial and error (Morales, 2020) that has been widely used in many industrial problems. For instance, Xanthopolous et al. (2018) present different reinforcement learning uses in industrial contexts. To our knowledge, only one article proposes integrating a reinforcement learning entity in a DDOM. Cuartas Murillo and Aguilar (2022) introduce an agent that intervenes on general parameters that are not specific to the DDOM (i.e., the inventory levels and the setup time). However, little attention has been paid to setting up DDOM parameters such as OST and OSH.

The present article deals with the management of atypical demand profiles (including spikes), which is a rarely addressed problem in the literature. An innovative approach allows the adjusting of two DDOM parameters, which are usually calculated with simplistic formulas advised by Ptak and Smith (2019), by using reinforcement learning. The structure of the article is the following: the second section present the literature related to the DDMRP and the reinforcement learning themes, and underlines the article's contribution. Then, Section 3 exposes the case study, the methodology, and the experimentation, while the results are presented and analyzed in Section 4. Finally, we conclude and offer research perspectives in Section 5.

2. Related literature review

In this section, we first present a short DDMRP literature review, along with a functional description of the method. Then, we build a reinforcement learning literature related to our specific context. Finally, we evoke the article's contribution.

2.1. DDMRP literature review

Ptak and Smith (2011) introduced the DDMRP in 2011. Their two books establish the first documentation type of the method (Ptak and Smith, 2011, 2019), and induce some research around it. Consequently, DDMRP projects have been realized in the past years, although not many were documented.

One type of studies that can be found in DDMRP literature is comparative studies. Because DDMRP is partly based on the MRP, those two production methods are often compared. The authors show that, in specific contexts, the DDMRP is better than the MRP in a quantitative way (Miclo et al., 2015, Miclo, 2016; Shofa et al., 2018). Miclo et al. (2015) show that a DDMRP can reduce the Work in Process (WIP) of an MRP by 26 %, while Shofa et al. (2018) achieved a reduction of 11 % in terms of inventory volumes compared to an MRP system. The authors also seek to realize a qualitative analysis to compare both methods in industrial environments (Kortabaria et al., 2018; Shofa and Widyarto, 2017; Ihme and Stratton, 2015). The results show a stock level decrease (up to a 52.53 % inventory level decrease for Kortabaria et al., 2018) and shorter delivery times (for example, Shofa and Widyarto managed to compress the lead time from 52 to 3 days). The results are a better visibility in the supply chain for the workers and fewer instabilities in the system. DDMRP is also often compared to other production methods like Kanban/Lean (Miclo et al., 2019) or Optimised Production Technology, a method of production flow management (Thürer et al., 2022). In all cases, it reveals itself more performing in the studied environments and more attractive to academic study.

DDMRP has also been documented by case studies, which use company data to observe the effects of DDMRP in specific industries. For instance, DDMRP has been studied in the automotive industry (Shofa and Widyarto, 2017) and ink manufacturing (Ihme and Stratton, 2015). Across 30 case studies, Bahu et al. (2019) proved that a DDOM is adapted to different industrial contexts. On the same note, Velasco Acosta et al. (2019) showed that a DDOM can be developed to run in a complex manufacturing environment. Then, one strand of research has been to improve DDMRP and its performance. For instance, Jiang and Rim (2017) suggest a way to reduce lead time and inventory costs by keeping work-in-progress stocks at specific stations. On another note, Lee and Rim (2019) present a safety stock formula for DDMRP replenishment, while proving it outperforms traditional DDMRP results. It is worth noticing that the DDMRP literature mostly relies on case studies, including simulations of real data or implementation methods in companies (Damand et al., 2022). Even though some authors already addressed some problems like the positioning of the buffers (Abdelhalim et al., 2021), the literature on implementation methods is not quite developed (Butturi et al., 2021).

Authors have also contributed to the DDMRP literature by building reviews. Azzamouri et al. (2021) present a literature review of the evolution of the methods and the research advancement. The main point that can be drawn from their literature review is that the method is still not enough studied and validated. Most of the studied cases remain theoretical and lack concrete applications. They also manage to show that DDMRP is adapted to the industrial field and is not limited to only one area by aggregating its literature. The authors (Azzamouri et al., 2021) tend to give methodological and practical comments since the method is still new (Pekarčíková et al., 2019). Nevertheless, DDMRP integration results are rarely published. Furthermore, DDMRP is mostly used in discrete-processed industries, while neglecting continuous-processed industries such as gas, cement, or mines (Azzamouri et al., 2021). The reviews also point out that the DDOM can be modeled in different ways. It is possible to simply use Excel (Shofa and Widyarto, 2017; Ihme and Stratton, 2015), but we note that discrete-event simulation is widely used (Martin, 2020; Miclo et al., 2015; Kortabaria et al., 2018; Velasco Acosta et al., 2019).

Finally, little attention has been paid to DDOM parametrization (Butturi et al., 2021). Ptak and Smith (2019) leave only general recommendations about the setting inputs, although Miclo et al. (2015) insist on the importance of the parametrization. Only recently has there been some research about it. Lee and Rim (2019) suggest some parametrization elements for the lead time factor and the variability factor. Dessevre et al. (2019) propose to implement a dynamic adjustment method for the setup time. Nevertheless, Bahu et al. (2019) and Damand

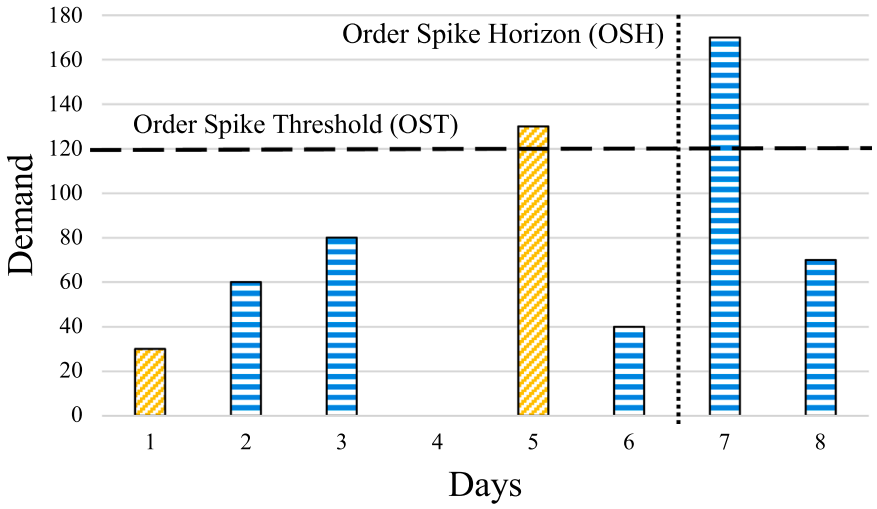

Fig. 1. Example of a qualified demand calculation using the order spike threshold (OST) and the order spike horizon (OSH).

et al. (2022) point out the lack of automatic algorithms to set up the DDOM parameters. We suppose it is due to the high number of software that is compliant with a DDMRP implementation (Demand Driven Institute, 2022). Even though some softwares are compliant with a DDMRP implementation, such as Asprova or SAP, some practitioners can struggle to integrate general methods of DDOM parametrization in their software.

The next section focuses on DDMRP mechanisms and more precisely the DDOM parametrization.

2.2. DDMRP parametrization

The DDOM revolves around the net flow position equation (Ptak and Smith, 2019). It is defined in Eq. (1).

$$Net\ Flow\ Position\ (NFP) = On\ hand + on\ order - qualified\ demand \quad (1)$$

The qualified demand is the sum of today's demand, qualified spikes, and eventual backlog (Ptak and Smith, 2019). An order is a pike if it is over the order spike threshold (OST). We illustrate a qualified demand calculation in Fig. 1. The qualified demand is 30 (day 1) + 130 (order spike at day 5) = 160 products (sum of the bars with diagonal hatches in Fig. 1).

The buffers are decoupling points allowing us to maintain some reference stock. They are represented by 4 zones (green, yellow, red base, and red safe), which are calculated with 4 different equations. By following the NFP evolution in the buffer zones, the production manager can easily make decisions along the supply chain. If the NFP is below the top of the yellow zone (TOY), a supply order is generated to bring back the NFP on top of the green zone (TOG). This equation also replaces a decision made on an actual stock with a decision made on an equivalent of the stock position.

Besides the OST and OSH, other parameters are used to compute the buffer zones:

- the Average Daily Usage (ADU);
- the Decouple Lead Time (DLT): the longest cumulative time between the buffer and the previous references;
- the Lead Time Factor (LTF): the uncertainty factor of lead time;
- the Variability Factor (VF): the uncertainty factor of demand;
- the Minimum Order Quantity (MOQ).

These factors are used to compute the levels of the four buffer zones, presented in Eqs. (2)–(5).

$$Green\ zone = Max\ (ADU * DLT * LTF,\ MOQ) \quad (2)$$

$$Yellow\ zone = ADU * DLT \quad (3)$$

$$Red\ base\ zone = ADU * DLT * LTF \quad (4)$$

Table 1
Analyzed parameters in DDMRP literature.

Authors	ADU	Spikes	DLT	LTF	VF	Demand variability	Internal variability	Buffer levels	MOQ	OST and OSH	Used approach
Cuartas Murillo and Aguilar (2022)				X		X		X	X		RL
Damand et al. (2022)	X	X		X	X	X		X	X	X	Genetic algorithm
Dessevre et al. (2019)	X		X			X	X				Adjustment module
Ihme and Stratton (2015)								X			Simulation
Jiang and Rim (2017)			X				X	X			Genetic algorithm
Kortabaria et al. (2018)	X					X		X			Qualitative study
Lee and Rim (2019)	X	X	X	X	X	X		X			Optimization
Miclo et al. (2015)				X				X			DES
Miclo (2016)		X	X	X	X	X		X			DES
Miclo et al. (2019)	X					X		X			Simulation
Pekarčíková et al. (2019)	X					X		X			Simulation
Shofa and Widyarto (2017)	X					X		X			Simulation
Shofa et al. (2018)	X					X		X			Simulation
Velasco Acosta et al. (2019)	X		X			X		X			DES
The present paper		X				X	X	X		X	RL and DES

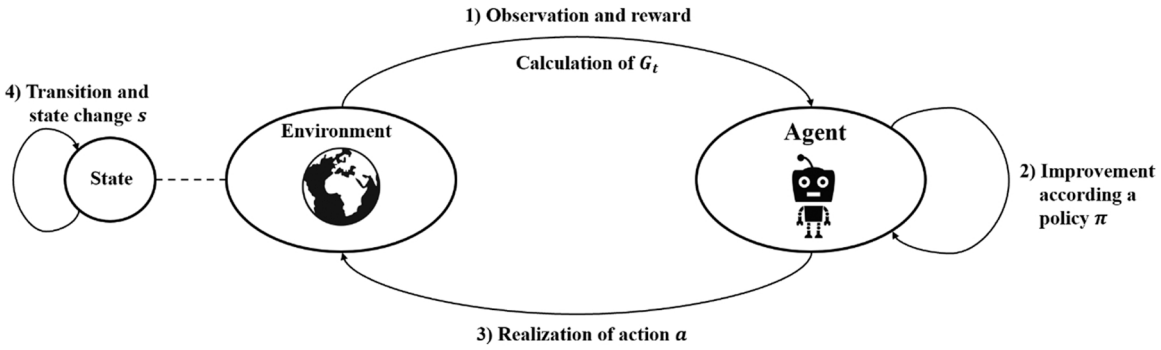

Fig. 2. One timestep of reinforcement learning.

$$Red\ safe\ zone = ADU * DLT * LTF * VF \tag{5}$$

In addition to the buffer levels, the buffer parameters intervene in the OST and OSH calculation methods. Ptak and Smith (2019) recommend setting the OSH above 1.0 ×DLT, and they propose to use three different methods to initialize the OST:

1. At 50 % of the top of the red zone (TOR);
2. At the top of the red safe zone;
3. Using ADU, $OST = 3 \times ADU$.

Nevertheless, the DDMRP authors state that it is possible to calculate those parameters using historical data. Yet no calculation method with historical data has been developed.

As we evoked in the introduction, the research around DDOM parametrization is concise. Table 1, which is an updated version of the literature of Azzamouri et al. (2021), gathers the DDMRP articles dealing with parameters analysis. Only a few recommendations are made to establish the DDOM parameters, so it can be uneasy for a production manager to drive them. Only Miclo (2016), Lee and Rim (2019), and Damand et al. (2022) suggest some parametrization elements with an algorithm. Miclo (2016) presents two different approaches to optimize some parameters: one with a meta-heuristic formula, and the other one with a simulated annealing algorithm, aiming at optimizing the buffers' LTF, VF, and DLT. On another note, Lee and Rim (2019) use a heuristic formula to adjust the LTF and VF. Other authors, like Dessevre et al. (2019) and Martin (2020) suggest some parametrization elements but do not use an algorithm to adjust them.

Among those authors, only Damand et al. (2022) studied OST and OSH adjustment, as can be seen in Table 1. They developed a multi-objective genetic algorithm to adjust 8 different DDOM parameters, including the OST and OSH. However, in Damand et al. (2022), a "strong assumption" was made: demand was assumed to be known in advance. Besides, the proposed genetic model has only been used for optimization purposes and does not include a learning process, in contrast to the present study. We deal with a realistic context and feed the RL algorithm with demand data daily. Thus, as in real life, we assume that we know only demand (i.e., customer orders) received up to the present day. This paper aims at incorporating a learning entity that enables the DDOM to deal efficiently with unknown demand by adjusting the parameters OST and OSH. The learning entity will take the shape of a reinforcement learning agent.

2.3. Reinforcement learning-related literature

Reinforcement learning (RL) is a machine learning area based on a trial-error algorithm. An autonomous entity called an agent decides for actions to take in an environment, which is a set of states and values. The agent aims to reach a goal. This goal is translated in the algorithm by a reward function, which returns a numerical value for a state. During the training process, the agent learns from its actions to create an optimal policy, i.e., a function that prescribes the best action to proceed considering a specific state. A state change, called a timestep, includes the following steps:

1) Observation and reward: the agent observes the environment through a space state and assigns a numerical value called a reward;
2) Improvement of the policy π: the agent improves the policy to prescribe better actions;
3) Realization of the action a: the agent selects an action a in the action space and prescribes it to the environment;
4) Transition and state change s: the environment realizes the action a and makes the state change s.

The composition of a timestep is represented in Fig. 2. The algorithm stops if a terminal state is reached. If there is no terminal state, the algorithm is limited by a time restraint. A set of timesteps from an initial state to a terminal state is called an episode (Morales, 2020).

RL proceeds by using trial and error. This means that it needs to find a compromise between exploration, which consists to try new actions, and exploitation, which attempts to use what the agent already learned. When the agent is exploiting, it seeks to maximize the action-value function or Q-function. This function is used to calculate the sum of the collected rewards in a single episode, considering a decision policy noted π. A decision policy is a function that prescribes an action to do for a non-terminal state. The Q-function is defined in Eq. (6) if the agent considers a policy π and does the action a at state s. The gain at a timestep t (G_t) is the sum of the rewards from the timestep $t+1$ to the final timestep T ($G_t = R_{t+1} + R_{t+2} + \ldots + R_T$, where R_t is the reward of the timestep t).

$$q_\pi(s,a) = \mathbb{E}[G_t | S_t = s, A_t = a] \tag{6}$$

In practical applications of RL, the exact value of that function cannot be calculated. Therefore, an approximated value is used, which can be calculated with a neural network. A neural network is a set of processors called neurons, which are activated by an activation function. Real values are injected into an input layer and travel through the network to produce an activation sequence. They go through hidden layers and come out of an output layer. During the learning process, the neural network aims to determine the neuron weights to reproduce the desired behavior (Schmidhuber, 2015).

Neural networks have been widely used in RL problems; this type of algorithm is called Deep Q-learning. Dittrich and Fohlmeister (2020) use a neural network to develop a cooperative multi-agent system. Neural networks can also be used to create a shared decision system, allowing one to take independent actions on different degrees of liberty, such as the movements of a robot (Tavakoli et al., 2018). In the present work, the neural network is used to reproduce the behavior of the Q-function (Eq. 6).

Reinforcement learning relies on concrete, well-defined, and simple tasks (Morales, 2020). However, it has some drawbacks, like the need of

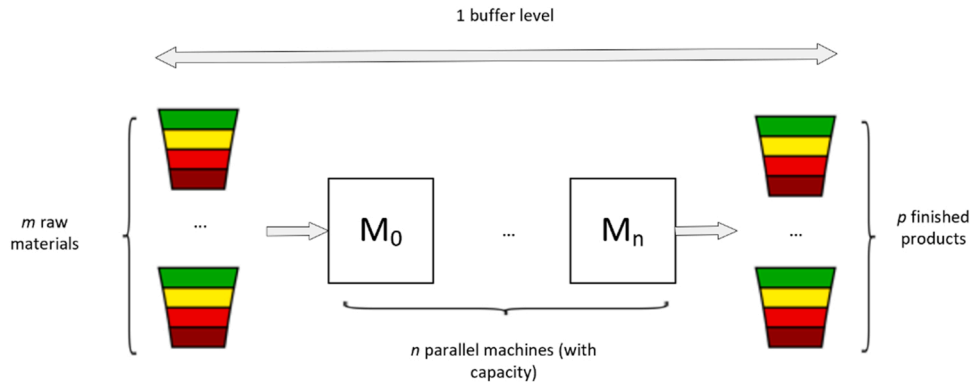

Fig. 3. DDMRP hybrid flowshop model.

a lot of experiments, and the potential difficulty to understand and interpret the obtained rewards. One of the stakes of this algorithm is temporal credit attribution. The algorithm must be able to identify which states and actions should be rewarded after a certain amount of timesteps.

As will be illustrated, there exist many different reinforcement learning applications in the industry (Xanthopolous et al., 2018). For example, Tsai et al. (2020) show that it is possible to control an articulated arm with a reinforcement learning agent in a shoe factory. The methodology that they used is for comparing RL with traditional heuristics and for showing that RL can bring better results (Wang et al., 2017). RL is often used in discrete optimization to limit the associated cost to production constraints (Stockheim et al., 2003; Cao et al., 2003). It shows that reinforcement learning is not limited to the optimization of the production line, while it is necessary to well define the problem and adapt the RL elements, such as the space state, the reward function, and the action state.

The literature around reinforcement learning is quite developed and many of its drawbacks have already been addressed, such as overfitting (Lai et al., 2021), and the exploration-exploitation compromise (Stockheim et al., 2003; Dittrich and Fohlmeister, 2020). On another side, Zhou et al. (2021) solved the curse of dimensionality by showing that it was possible to use RL on large databases, which is an important stake if reinforcement learning tends to spread in the industry.

Finally, reinforcement learning gathers a large number of algorithms (Kemmer et al., 2018), which is convenient when it comes to adapting an algorithm to a problem. Nevertheless, it is still possible to enrich its study and literature (Neves et al., 2021).

To our knowledge, only Cuartas Murillo and Aguilar (2022) integrated a DDOM and an RL agent. They use a hybrid algorithm using DDMRP methodology and RL to optimize the time to buy a product and to determine the quantity they need; a strategy that is totally different from that of this paper, which focuses on DDOM parameters. While creating a Q-learning algorithm, they come up with three approaches concerning the reward function: one which uses the inventory levels, another one that evaluates the distance between the inventory level and the optimal level, and one that involves the inventory levels and their distance to the optimal levels. They train their algorithm on various case studies, contexts, and demand profiles. Their agents show promising results in different scenarios.

2.4. Article contribution

The DDMRP literature presents some gaps concerning the parametrization of its Demand-Driven Operating Model. While it is known for its dynamic adjustments, it seems that the buffer parameters adjustment is not studied deeply enough. In particular, it lacks automatic methods to drive the DDOM parameters. Instead of acting on the stock levels (Cuartas Murillo and Aguilar, 2022), we propose to adjust the OST and OSH. While it has already been done by using a multi-objective genetic algorithm for optimization purposes (Damand et al., 2022), we attempt to include a machine learning entity to solve problems with unknown demand. No study has been done with this methodology on the adjustment of those parameters. By using a reinforcement learning algorithm, we develop an innovative calculation method for these parameters while evaluating the impact on stock and service levels. This approach fits in an automating process of the DDOM parameters adjustment.

We also point out that the proposed method is adapted to a multi-product environment. Based on Branching Dueling Q-Network presented by Tavakoli et al. (2018), we manage to create a shared representation system that is able to make independent decisions between different products. It widely limits the effects of complexity and potential dimension concerns.

3. Case study and methodology

In this section we first present the case study and the model hypothesis. Then, we explain the agent integration and the experimentation.

3.1. Case study

We consider a DDMRP hybrid flowshop (see Fig. 3), which uses a workflow pulled by the demand and a pushed flow of production orders. Based on the model developed by Martin (2020), a set of p products are made by n parallel machines using m raw materials. The stocks of raw materials and finished products are controlled by DDMRP buffers.

We choose to use a hybrid flowshop, first because it is a common manufacturing environment that can cover many case studies (Martin, 2020). Second, because a hybrid flowshop possesses some advantages including flexibility and avoidance of bottlenecks due to the redundancy of machines (Zhou et al., 2019). The environment is modeled with a Python discrete-event simulation (DES), in which different entities communicate to realize the simulation events. The events are treated according to a priority list (see below), which allows us to set the execution order, in case two events are to be realized at the same moment. The priority list is the following:

1. serve the orders due;
2. update the ADU;
3. update the DLT;
4. update the LTF;
5. update the zones;
6. decide the quantities to product.

Demand generation is inspired by the model of Dessevre and Benali (2020). The interarrival time between two orders follows an exponential distribution of a 2-days mean. We draw uniformly an order at ± 20 % of

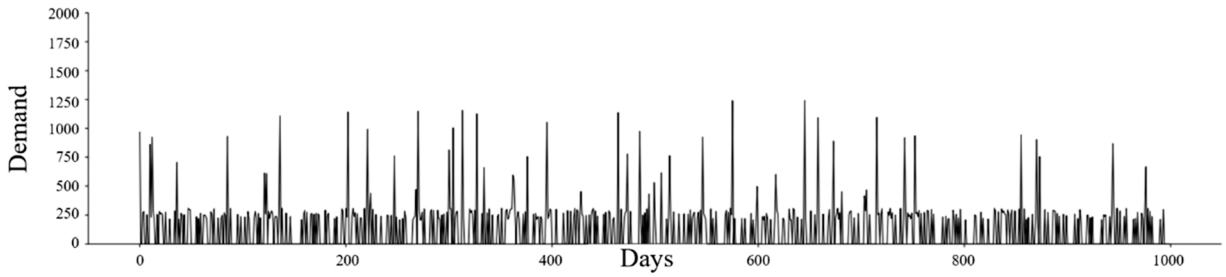

Fig. 4. Demand profile of a product.

the mean size of the demand. This mean size is drawn uniformly between 50 and 200 products. On average, the order spikes appear each 5 or 20 days, following an exponential distribution. The size of an order spike is uniformly drawn at 2, 3, 4, or 5 times the size of a regular order. An example of a demand profile is available in Fig. 4, with a spike frequency set at 20 days. We study a rarely addressed problem in the literature: a demand profile presenting regular spikes. Our work does not apply to stable demands (stationary or seasonal), but can be easily incorporated in environments presenting changing and regular spikes, which matches with atypical demand contexts.

The model developed by Martin (2020) allows us to change the flowshop workload. This is made by modifying the number of machines by production line. Thereby, we can use a workload of 50 %, 80 %, and 95 %; a liberty degree that is used in the analysis of the results.

3.2. *Agent integration*

The agent integration consists of adding an autonomous entity called an Agent to the model developed by Martin (2020). First, this entity needs to be adapted to the discrete-event simulation. Consequently, we need to create agent events, which are recurrent processes involving the agent. In a traditional RL algorithm, the agent timesteps would set the actions and the updates. In this paper, timesteps are already set by the DDOM. This means that we must consider timesteps for the agent and timesteps for the DDOM. These two notions overlap, while they are very different. Therefore, we replace the "agent timesteps" with the "agent interactions": when the agent interacts with the system, it changes the state and adjusts the parameters.

Secondly, the agent entity needs to communicate with the other entities, such as the buffers and the environment. Consequently, it needs to be able to receive and send messages, which are: "do an agent interaction" and "update the neural network". Details about the intervention of a neural network will be given further down. Because two events were added, we had to create a new event priority list, which allows us to respect the DDMRP principles while giving the agent information about the state updates. The new event priority list is now the following:

1. serve the orders due;
2. **do an agent interaction**;
3. update the ADU;
4. update the DLT;
5. update the LTF;
6. update the zones;
7. decide the quantities to product;
8. **update the neural network**.

We use an algorithm with two neural networks called Double Deep Q-Network. Compared to traditional RL algorithms (e.g., Monte Carlo, Q-learning, or State-action-reward-state-action (SARSA)), Deep Q-Network is the only type of RL algorithm that can work with a discrete action space and a continuous state space while overcoming some dimensional issues (Morales, 2020). Unlike traditional RL algorithms, a Deep Q-Network algorithm uses a neural network to reduce the complexity and detect underlying links between the values of the Q-function in a way to improve learning (Tavakoli et al., 2018). When using Double Deep Q-Network, both neural networks can be updated without creating a positive bias. Such an algorithm is simultaneously sequential (it considers the consequences of delaying the actions), evaluative (it considers the exploration-exploitation compromise), and sampled (it limits the experiments if the history is too large) (Morales, 2020). More precisely, we use a variant of the Double Deep Q-Network, called Branching Dueling Q-Network (BDQ) inspired by Tavakoli et al. (2018). This algorithm has the advantage of splitting the Q-function in the neural network into a "state-value" part (shared between all the

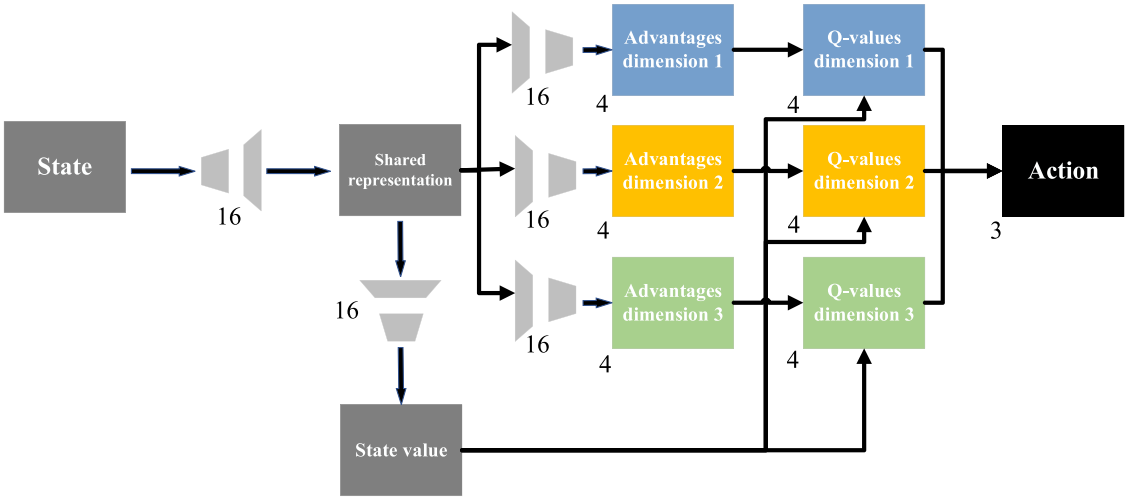

Fig. 5. Branching Dueling Q-Network (BDQ) architecture.

actions) and an "action-advantage" part (specific to each action). Moreover, it allows us to work on several products at the same time which are treated independently (Tavakoli et al., 2018). The BDQ architecture is presented in Fig. 5. The BDQ algorithm is available in Appendix A. We also provide the main pseudocode of the proposed DDMRP simulation in Appendix B.

The model includes 3 finished products. We consider that an episode covers 1000 days and that a replication contains 100 episodes. A scenario assumes the same configuration, which means the same parameters and external factors. Each scenario is repeated 3 times. To bring consistency to the analysis of the results, we suppose that two replications with the same configuration also have the same demand. Only the setup times and the manufacturing times change from one replication to another.

We propose to calculate two factors α_{OST} and α_{OSH}, with $\in \alpha_{OST} \in \{0.25; 0.5; 0.75; 1.0\}$ and $\alpha_{OSH} \in \{0.0; 0.15; 0.35; 0.5\}$. Each couple of α_{OST} and α_{OSH} represents an action of the agent. We hypothesize that those two DDOM parameters can have a cross-effect and can both be improved using reinforcement learning with peak demand profiles. We mix the two strategies since the OST and the OSH both apply on order spikes. Since our experiments are time-consuming, we choose not to use a continuous action space, because it will have greatly increased the algorithm complexity and the execution time (Morales, 2020). The OST and OSH are modified after an action is taken so that Eqs. (7) and (8) are fulfilled. Thus, our initial action space contains 16 actions.

$$OST = \alpha_{OST} * TOR \tag{7}$$

$$OSH = (1 + \alpha_{OSH}) * DLT \tag{8}$$

First, the choices of α_{OST} and α_{OSH} values follow Ptak and Smith (2019)'s recommendations. To preserve the buffer's integrity (defined as the penetration rate of the buffer, the ratio between the available flow and the total size), Ptak and Smith (2019) recommend setting an OST smaller or equal to TOR (i.e., $\alpha_{OST} \leq 1$) and an OSH higher or equal to DLT (i.e., $\alpha_{OSH} \geq 0$). Second, we proceed by trial-and-error to set an upper bound for α_{OSH} and concluded that a high value of α_{OSH} is irrelevant for learning (some actions are quickly ignored by the agent). Besides, Ptak and Smith (2019) mention that it is useless to consider a too big value of OSH, unless for "finished items that have large and known dependent demand orders [...] in which customers have agreed to take a significant amount of stock within a short window". Finally, we proceed by trial and error on how many values we should consider in order to achieve proper learning (i.e., with an observable growth of the Average Accumulated Reward (AAR) function). Fewer values do not offer enough possibilities to learn. In contrast, too many values lead to considerable computational time to explore all actions.

The state space is composed of the moving demand average of the 30 last days and the NFP of each product. The interaction frequency of the agent is set at 25 days, while the update of the neural network is done every 500 days. Therefore, the neural network is updated twice by episode. This decision regarding the frequency of the event has been made by trial and error while balancing a compromise with the neural network size.

Finally, we established three reward functions. Each of these reward functions is characterized by a goal and a numerical function associated with the goal. They are made to reach the stakes of the DDMRP: optimize the stock levels and reduce delivery time. The three agents are described below:

Agent 1 attempts to optimize the stock levels. It uses the NFP position of each product in its corresponding buffer. Its reward function is defined in Eq. (9).

$$R_1 = \begin{cases} -\left(1 + \dfrac{NFP - TOG}{NFP}\right) \text{ if } TOG < NFP \\ -1 \text{ if } TOY < NFP \leq TOG \\ 3 \text{ if } TOR < NFP \leq TOY \\ 0 \text{ if } 0 \leq NFP \leq TOR \\ -1 \text{ if } NFP < 0 \end{cases} \tag{9}$$

Agent 2 aims at maximizing customer satisfaction by decreasing the delivery time. The On-Time Delivery (OTD) value is defined as the ratio between the number of complete delivered orders and the number of received orders. It is calculated on a time window. The reward function of agent 2 is defined in Eq. (10).

$$R_2 = OTD = \frac{number\ of\ complete\ delivered\ orders}{number\ of\ received\ orders} \tag{10}$$

Finally, agent 3 conciliates the two first reward functions by summing them. It considers at the same time the stock levels and customer satisfaction. Its reward function is defined in Eq. (11).

$$R_3 = R_1 + R_2 \tag{11}$$

These three reward functions allow us to study different goals and the different possibilities of reinforcement learning.

3.3. Experimentation

The experimentation deals with two subjects: the validity limits of reinforcement learning and its capacity to react to external factors. We use four key performance indicators to support our analysis:

1. The average service level, which reflects the ratio between the number of complete delivered orders and the number of received orders.
2. The finished goods inventory mean, which indicates the mean level of goods inventory for the last 200 days of an episode (an episode is 1000 days long).
3. The Average Accumulated Reward (AAR). We define it in Eq. (12), where N is the number of episodes and r_t represents the reward at episode t. This indicator allows us to check the growth of the rewards and so the agent efficiency. The AAR is calculated on a time window of the last 30 episodes. Furthermore, each AAR is normalized to be contained on a [0,1] interval.

Table 2
Experimentation.

Experiment	Experiment 1: OST and OSH adjusting	Experiment 2: Comparison of different reward functions	Experiment 3* : Effects of external factors on the learning process
Variable OST	Yes / No	Yes	Yes
Variable OSH	Yes / No	No	No
Reward Function	R2	R1 / R2 / R3	R3
OST Initialization	50% / RS / ADU	50% / RS / ADU	50% / RS / ADU
OSH Initialization	1.0 / 1.25 / 1.5 * DLT	1.0 * DLT	1.0 * DLT
Spike Frequency	1 per 5 days	1 per 5 days	High (1 per 5 days) / Low (1 per 20 days)
Workload	80%	80%	50% / 80% / 95%
# of scenarios	27	9	18

* : Experiment where a comparison with a Baseline and a Know-It-All is done

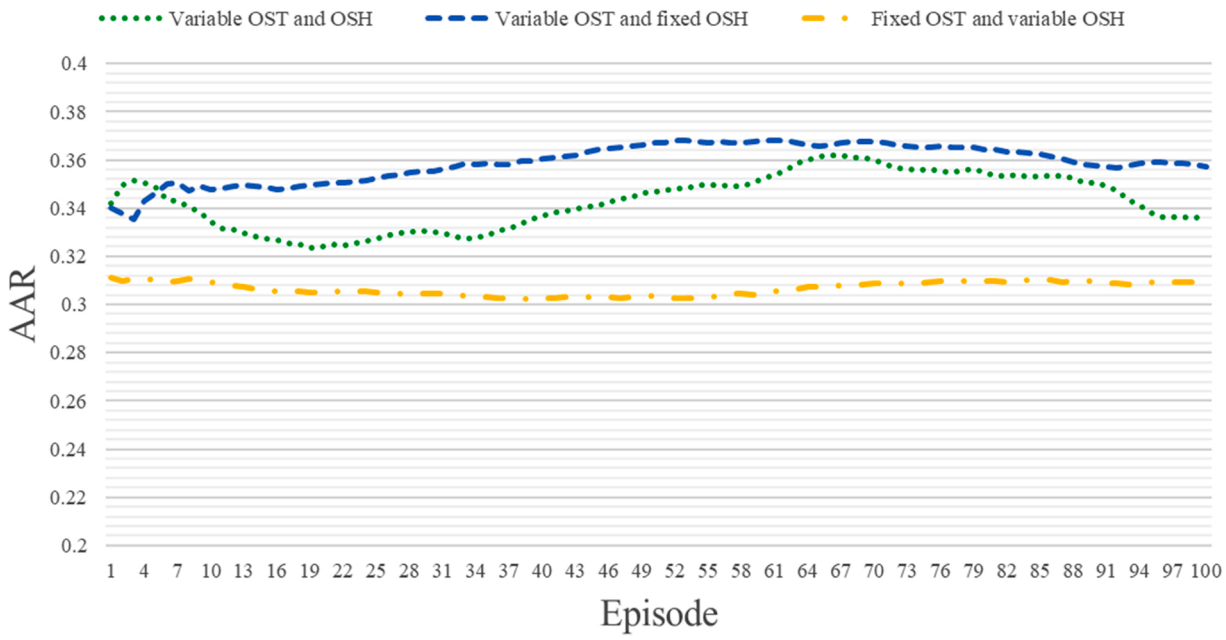

Fig. 6. Average accumulated rewards (AAR) of three scenarios with the same characteristics except the OST and OSH adjusting.

$$AAR = \frac{r_1 + r_2 + \ldots + r_t}{N} \tag{12}$$

The reward gain, formulated at Eq. (13), where $AAR_{100}$is the Average Accumulated Reward of the 100th episode, and (AAR) is the lowest value of the Average Accumulated Reward. It calculates the global growth of the AAR function.

$$RG = AAR_{100} - \min(AAR) \tag{13}$$

During the experimentation, each replication is conducted with and without reinforcement learning. The replication without RL is called Baseline, which is the behaviour of the simulation without an agent, and so without the parameter adjustment. It allows us to compare the RL performance with a free-agent simulation. We also propose to compare the RL agent to a Know-It-All agent which knows all demands in advance. The Know-It-All is inspired by Damand et al. (2022)'s study which assumes demand to be known, but without including a learning process. Furthermore, we propose to initialize the OST with each of the three methods of Ptak and Smith (2019): 50 % of TOR (50 %), at the top of the red safe zone (RS) and using ADU (ADU). Likewise, we use three calculation methods for the OSH initialization: 1.0 × DLT, 1.25 × DLT, and 1.5 × DLT. The experimentation is presented in Table 2 and has 3 objectives.

The first experiment (second column of Table 2) studies the interest of adjusting the OST and OSH. It contains 27 scenarios, which set the OST and OSH variable or fixed. We also propose considering different OST and OSH initialization to respect the general recommendations of Ptak and Smith (2019). Therefore, we fix the frequency of the spikes at 1 per 5 days and the workload at 80 % to limit the effect of external factors. Finally, we use a fixed reward function (R_2) because we do not want to study the impact of the reward function. This experiment allows us to check if it is relevant to adjust either only the OST, or only the OSH, or both. The two parameters have different roles and shapes, so it is reasonable to assume they do not act the same way when they are adjusted using reinforcement learning.

The second experiment (third column of Table 2) aims at identifying the best reward function. The objective is to create an optimal agent, which can correctly learn in different contexts. We study the three reward functions defined in Eqs. (9)–(11). Like the first experiment, we limit the effects of external factors by fixing the spike frequency and the workload. We initialize the OSH at 1.0 × DLT and consider different methods of initialization for OST. This experiment points out the importance of the goal in a reinforcement learning algorithm. For an RL agent to learn, it needs to have a well-defined and simple goal, which is entirely translated through the reward function. Therefore, a bad reward function can induce bad learning. To choose the goals of our reward function, we naturally think about the goals of the DDMRP: stock optimization and customer satisfaction. We can easily evaluate that by using the NFP and the local service levels. The two first experiments tend to create a consistent agent for the third experiment.

The third and last experiment (fourth column of Table 2) studies the effects of external factors on the learning process. Therefore, we consider two levels of the spike frequency (low and high) and three levels for the workload (50 %, 80 %, or 95 %) of the workshop. As we did for the second experiment, the OSH is initialized at 1.0 × DLT, while the OST is initialized with three different methods. This experiment checks if reinforcement learning is more efficient than the calculation methods proposed by Ptak and Smith (2019). If it is not, we identify the problematic contexts and try to bring an explanation.

4. Results and discussion

In this section, we present and discuss the results of the experimentation phase.

4.1. *Effects of the adjusted parameters on the learning process*

The first objective of the experimentation part is to check if the agent can learn by adjusting the OST and OSH. Fig. 6 shows the AAR of three comparable scenarios, for which we chose to adjust the OST and OSH or not. The curves for the scenario with variable OST and OSH, and the scenario with variable OST but fixed OSH are visibly crescent. Nevertheless, it is not the case for the scenario with fixed OST and variable OSH: the growth of the AAR is not remarkable. The results show us that the OSH variation has none or little effect on reinforcement learning. This disproves our hypothesis, according to which it is possible to learn by adjusting both OST and OSH. However, we proved that we can learn by focusing only on adjusting the OST.

We point out that the interaction frequency, set to one interaction each 25 days, has been chosen by trial-and-error. Changes in the OSH interaction frequency (from 1 to 100 days) did not bring any conclusive

Table 3
Reward gains by reward function and OST initialization.

	R1	R2	R3
50 %	0.014	0.022	0.031
RS	0.012	0.024	0.040
ADU	0.018	0.035	0.046

results (growth of the reward function). To conclude, according to the results from experiment 1, it seems that OSH adjusting is not relevant since it has too little effect on the environment.

Finally, we point out that while using reinforcement learning, small-sized action spaces are preferred to improve the learning process (Morales, 2020). In this first experiment, we noticed that a 4-state action space achieved better learning (evaluated by the growth of the AAR function) than a 16-state action space. We also want to reduce execution time by exploring a 4-state action space rather than a 16-state action space: a scenario with a 16-state action space presents a time execution of 3 h, while a scenario with a 4-state action space is limited to 1.5 h. The bigger the action space, the more difficult it is to explore all the actions and possibilities. Though during one episode, we only work with 50 agent interactions, this is too little for a 16-states actions space. Consequently, for the rest of the study, we chose to only adjust the OST and set the OSH fixed at 1.0 × DLT (which is reflected in Table 2 by "No" for variable OSH in columns 3 and 4). It brings us up to a 4-state action space.

4.2. Choice of an agent

The second experiment compares the three reward functions R_1, R_2, and R_3. Each of them has a different goal, and then a different learning process. We seek to use a reward function allowing the growth of the AAR. Therefore, we expose the reward gains expressed by Eq. (13) of the second experiment scenarios in Table 3. This table enables us to evaluate the learning efficiency. Since the R_3 agent has the best reward gains, it seems to be the most efficient reward function. Using production management terms, this proves that reinforcement learning is adapted to improve customer satisfaction and minimize inventory levels.

To find the best trade-off, we plot the service levels (y-axis) and the finished goods inventory mean (x-axis) for the scenarios of the second experiment in Fig. 7. We consider these two industrial KPI at episode 100 (the last episode) since the agent is supposed to be well-trained at this step.

As we expect, the scenarios using R_1 as a reward function have lower stock levels than scenarios using R_2. In return, there is not a significant improvement in service level with R_2. Scenarios using R_3 (we remind that $R_3 = R_1 + R_2$) present the benefits of R_1 and R_2 regarding industrial KPI (i.e., the best trade-off with lower stock levels and higher service levels). This demonstrates that R_3 is the best reward function to use if an industrial implementation may be considered.

To conclude, R_3 is the best reward function regarding industrial and learning KPI. Consequently, we chose to pursue the study with this reward function. The first two experiments enabled us to build a consistent agent which is now adapted and optimal for reinforcement learning in a DDOM.

4.3. Effects of external factors

The last step of the experimentation aims at studying the effects of external factors such as spike frequency and workload. To do so, we use R_3 as the reward function with fixed OSH at 1.0 × *DLT* (see Table 2, fourth column) and we analyze the industrial KPI (finished goods inventory mean and service level) of the last episode by comparing the RL results with those of the Baseline and the Know-It-All.

As mentioned in Section 3.3, we consider two levels for the Spike Frequency (SF) and three levels for the workload (WL) of the workshop. Low SF and High SF correspond respectively to 1 per 20 days and 1 per 5 days. WL takes 50 %, 80 %, or 95 %. We still consider the three OST initialization methods recommended by Ptak and Smith (2019).

In what follows, we present average results on the three OST initializations. We made this choice for clarification purposes, since the use of boxplots revealed that there is no noticeable impact (see Appendix C). The results gathered in Fig. 8 display the industrial KPI achieved respectively by the RL agent, the Baseline, and the Know-It-All agent at episode 100 in different contexts (i.e., different SF and WL levels).

First, we point out that the performance of the RL agent is close to the performance of the Know-It-All agent. This proves the efficiency of the learning process and demonstrates that an RL agent is adapted for industrial cases when demand is unknown or difficult to predict. Second, we try to identify when it is beneficial to use RL rather than the Baseline. We notice that with high SF (square markers in Fig. 8), Baseline outperforms RL only for WL = 50 %, while with low SF (circle markers in Fig. 8), Baseline outperforms the RL agent for WL = 50 % and WL = 80 %. The RL agent seems to learn better if the spikes are frequent (i.e., high SF) since it can detect more spikes. It also seems to be more useful for workshops presenting high workloads (beyond 80 % in our case). Consequently, we recommend using reinforcement learning with

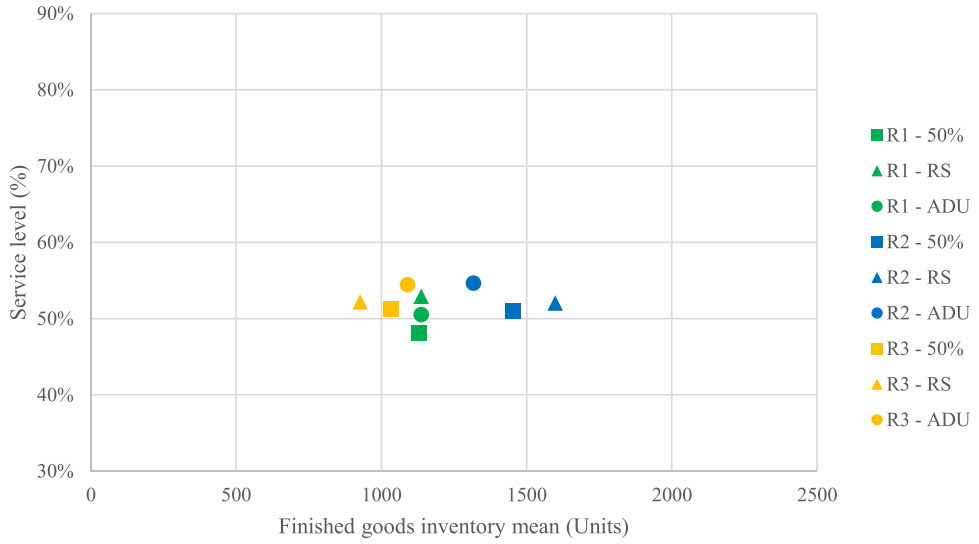

Fig. 7. Industrial KPI for the scenarios of experiment 2 at episode 100.

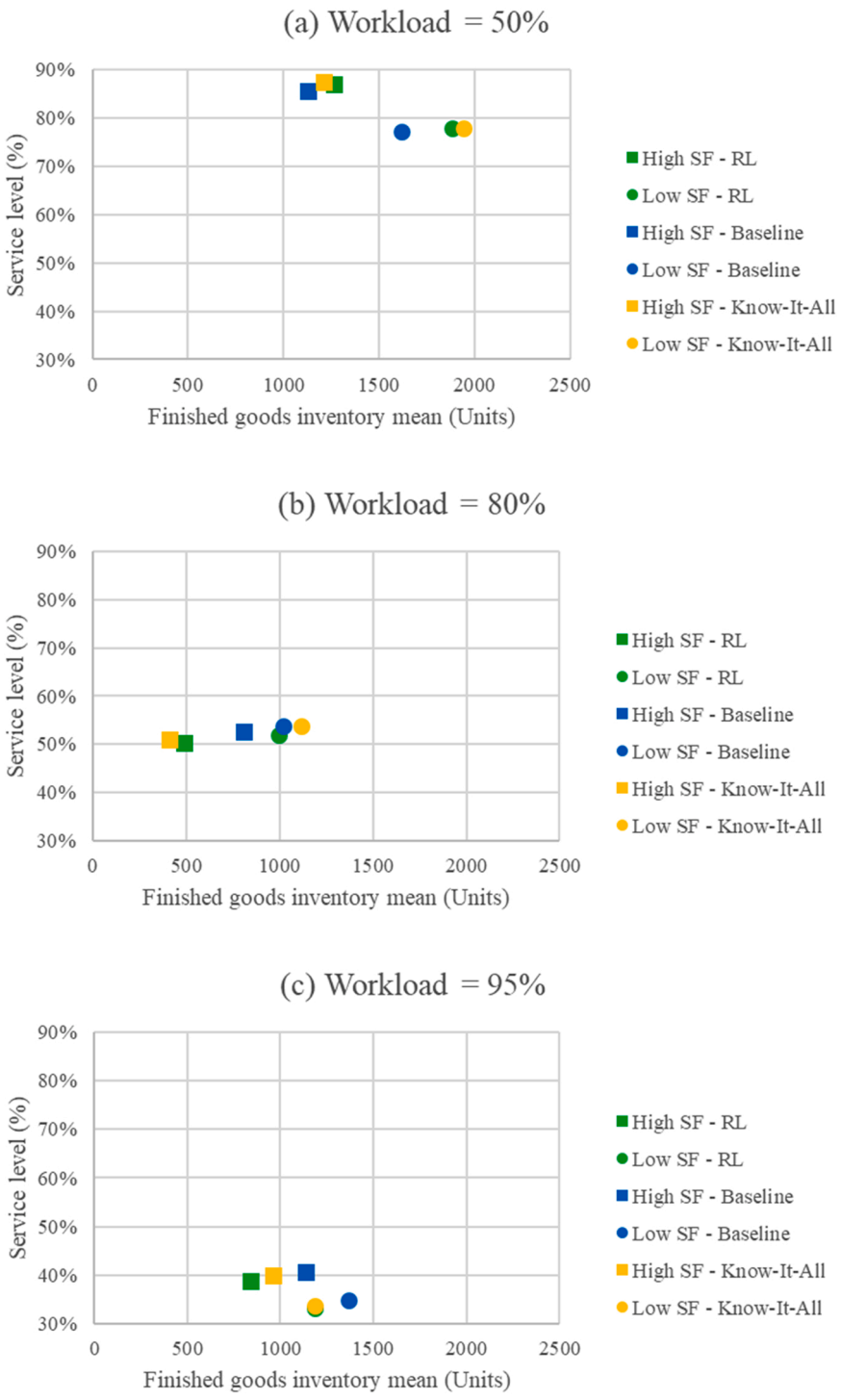

Fig. 8. Industrial KPI for the scenarios of experiment 3 at episode 100.

workshops presenting high workloads and facing frequent spikes.

5. Conclusion and research perspectives

This article introduces a new approach to parametrize a Demand-Driven Operation Model (DDOM) facing an unknown and atypical demand. A reinforcement learning agent based on a Branching Dueling Q-Network model is used and enables the DDOM to deal with demand spikes by adjusting the parameters OST and OSH.

First, we explored the interest in adjusting OST and OSH. Results demonstrate that better learning is achieved by adjusting only OST. Second, we compared the performance of the RL agent with three different reward functions and identified the one that achieved the best trade-off with low stock levels and high service levels. Finally, we challenged the RL agent and compared it to a Baseline and a Know-It-All agent in different contexts. Results show that the RL agent outperforms the Baseline (up to a 33 % stock decrease) in workshops presenting high workloads (beyond 80 % in our case) and facing frequent demand spikes. In addition, the performance of the RL agent is close to the performance of the Know-It-All agent, which demonstrates that an RL approach is adapted for industrial cases when demand is unknown or difficult to predict.

Experiments have been carried out for a specific industrial case study. Altogether, this paper provides the tool and the methodology needed to perform other experiments with different settings and in other industrial contexts. This study is the first to bring a proof of concept for the potential benefits of dynamically adjusting DDOM parameters using a learning approach.

Future research efforts concerning the DDOM parametrization may provide some new insights. First, RL or other machine learning approaches can be compared and used to adjust parameters other than OST and OSH. Second, considering different sources of uncertainty and operational constraints may be of theoretical and practical interest. Finally, RL can be used to parametrize the DDOM for different production processes, such as a divergent process rather than an assembly process.

Declaration of Competing Interest

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

Data Availability

No data was used for the research described in the article.

Appendix A. Supporting information

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.compind.2023.103874.

References

Abdelhalim, A., Hamid, A., Hsu, T., 2021. Optimization of the automated buffer positioning model under DDMRP logic. IFAC-Pap. 54 (1), 582–588. https://doi.org/10.1016/j.ifacol.2021.08.067.

Azzamouri, A., Baptiste, P., Dessevre, G., Pellerin, R., 2021. Demand-driven material requirements planning (DDMRP): a systematic review and classification. URL: http://www.jiem.org/index.php/jiem/article/view/3331 J. Ind. Eng. Manag. 14 (3), 439–456. https://doi.org/10.3926/jiem.3331.

Bahu, B., Bironneau, L., Hovelaque, V., 2019. Compréhension du DDMRP et de son adoption: premiers éléments empiriques. Logistique Et. Manag. 27 (1), 20–32. https://doi.org/10.1080/12507970.2018.1547130.

Butturi, M.A., De Rosa G., Balugani E., Gamberini, R., 2021. Understanding the Demand Driven Material Requirements Planning Scope of Application: a Critical Literature Review, in: Proceedings of the 32nd DAAAM International Symposium, Vienna, Austria. pp. 462–471. doi: 10.2507/32nd.daaam.proceedings.067.

Cao, H., Xi, H., Smith, S.F., 2003. A reinforcement learning approach to production planning in the fabrication/fulfillment manufacturing process, in: Proceedings of the 2003 Winter Simulation Conference, New Orleans, LA, USA. pp. 1417–1423. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1261584, doi:⟨10.1109/WSC.2003.1261584⟩.

Cuartas Murillo, C.A., Aguilar, J.L., 2022. Hybrid algorithm based on reinforcement learning and DDMRP methodology for inventory management. J. Intell. Manuf. https://doi.org/10.1007/s10845-022-01982-5. ⟨https://link.springer.com/article/10.1007/s10845-022-01982-5#citeas⟩.

Damand, D., Lahrichi, Y., Barth, M., 2022. Parametrisation of demand-driven material requirements planning: a multi-objective genetic algorithm. Int. J. Prod. Res. https://doi.org/10.1080/00207543.2022.2098074.

Demand Driven Institute, 2022. DDMRP compliant software. Consulted on November, 15th 2022. URL: ⟨https://www.demanddriveninstitute.com/ddmrp-compliant-software⟩.

Dessevre, G., Martin, G., Baptiste, P., Lamothe, J., Pellerin, R., Lauras, M., 2019. Decoupled Lead Time in finite capacity flowshop: a feedback loop approach, in: 2019 International Conference on Industrial Engineering and Systems Management (IESM), Shangai, China. URL: ⟨https://ieeexplore.ieee.org/document/8948198⟩, doi:10.1109/IESM45758. 2019.8948198.

Dessevre, G., Benali, M., 2020. Modélisation et simulation d'un module d'ajustement de la capacité d'un système DDMRP, in: 13ème Conférence Francophone de Modélisation, Optimisation et SimulationMOSIM'20, Agadir, Maroc. URL: ⟨https://hal.archives-ouvertes.fr/hal-03178098/⟩.

Dittrich, M.A., Fohlmeister, S., 2020. Cooperative multi-agent system for production control using reinforcement learning (URL). CIRP Ann. 69, 389–392. https://doi.org/10.1016/j.cirp.2020.04.005 (URL). ⟨https://www.sciencedirect.com/science/article/pii/S0007850620300263⟩.

Ihme, M., Stratton, R., 2015. Evaluating Demand Driven MRP: a case based simulated study, in: International Conference of the European Operations Management Association, Neuchatel, Switzerland. URL: ⟨http://irep.ntu.ac.uk/id/eprint/2666⟩⟨8⟩.

Jiang, J., Rim, S.C., 2017. Strategic WIP inventory positioning for maketo-order production with stochastic processing times. Math. Probl. Eng. https://doi.org/10.1155/2017/8635979.

Kemmer, L., von Kleist, H., de Rochebouët, D., Tziortziotis, N., Read, J., 2018. Reinforcement learning for supply chain optimization, in: European Workshop on Reinforcement Learning 14, Lille, France.

Kortabaria, A., Apaolaza, U., Lizarralde, A., Amorrortu, I., 2018. Material management without forecasting: From MRP to demand driven MRP. URL: http://www.jiem.org/index.php/jiem/article/view/2654 J. Ind. Eng. Manag. 11 (4), 632–650. https://doi.org/10.3926/jiem.2654.

Lai, Y.H., Wu, T.C., Lai, C.F., Yang, L.T., Zhou, X., 2021. Cognitive optimal-setting control of aiot industrial applications with deep reinforcement learning (URL). IEEE Trans. Ind. Inform. 17, 2116–2123. https://doi.org/10.1109/TII.2020.2986501 (URL). ⟨https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9072609⟩.

Lee, C.J., Rim, S.C., 2019. A mathematical safety stock model for DDMRP inventory replenishment. Math. Probl. Eng. https://doi.org/10.1155/2019/6496309.

Martin, G., 2020. Contrôle dynamique du Demand Driven Sales and Operations Planning. PhD thesis. Université de Toulouse. Toulouse, France. URL: ⟨http://www.theses.fr/2020EMAC0010⟩.

Miclo, R., 2016. Challenging the Demand Driven MRP Promises: A Discrete Event Simulation Approach. PhD thesis. Ecoles des Mines d'Albi-Carmaux. Albi, France. URL: ⟨https://tel.archives-ouvertes.fr/tel-01673811⟩.

Miclo, R., Fontanili, F., Lauras, M., Lamothe, J., Milian, B., 2015. MRP vs. demand-driven MRP: Towards an objective comparison, in: 2015 International Conference on Industrial Engineering and Systems Management (IESM), Seville, Spain. URL: https://ieeexplore.ieee.org/document/7380288, doi:10.1109/IESM.2015.7380288.

Miclo, R., Lauras, M., Fontanili, F., Lamothe, J., Melnyk, S, A., 2019. Demand driven MRP: assessment of a new approach to materials management. Int. J. Prod. Res. 57 (1), 166–181. https://doi.org/10.1080/00207543.2018.1464230.

Morales, M., 2020. Grokking Deep Reinforcement Learning. Manning Publications Co, Shelter Island, NY, USA.

Neves, M., Vieira, M., Neto, P., 2021. A study on a Q-learning algorithm application to a manufacturing assembly problem (URL). J. Manuf. Syst. 59, 426–440. https://doi.org/10.1016/j.jmsy.2021.02.014 (URL). ⟨https://www.sciencedirect.com/science/article/pii/S0278612521000509⟩.

Pekarčíková, M., Trebuňa, P., Kliment, M., Trojan, J., 2019. Demand driven material requirements planning. Some methodical and practical comments. Manag. Prod. Eng. Rev. 10 (2), 50–59. https://doi.org/10.24425/mper.2019.129568.

Ptak, C., Smith, C., 2011. Orlicky's Material Requirements Planning 3/D. McGraw Hill Professional.

Ptak, C., Smith, C., 2019. Demand Driven Material Requirements Planning (DDMRP). Industrial Press, Inc, South Norwalk, Connecticut, USA.

Schmidhuber, J., 2015. Deep learning in neural networks: an overview. Neural Networks 61, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003.

Shofa, M.J., Widyarto, W.O., 2017. Effective production control in an automotive industry: MRP vs. demand-driven MRP, in: AIP Conference Proceedings. URL: https://aip.scitation.org/doi/abs/10.1063/1.4985449, doi:10.1063/1.4985449.

Shofa, M.J., Moeis, A.O., Restiana, N., 2018. Effective production planning for purchased part under long lead time and uncertain demand: MRP vs demand-driven MRP, in: IOP Conference Series: Materials Science and Engineering. doi:⟨10.1088/1757–899X/337/1/012055⟩.

Stockheim, T., Schwind, M., Koenig, W., 2003. A reinforcement learning approach for supply chain management, in: 1st European Workshop on Multi-Agent Systems (EUMAS), Oxford, UK. URL: https://www.researchgate.net/publication/228523960_A_reinforcement_learning_approach_for_supply_chain_management.

Tavakoli, A., Pardo, F., Kormushev, P., 2018. Action branching architectures for deep reinforcement learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA. URL: ⟨https://ojs.aaai.org/index.php/AAAI/article/view/11798⟩.

Thürer, M., Fernandes, N.O., Stevenson, M., 2022. Production planning and control in multi-stage assembly systems: an assessment of Kanban, MRP, OPT (DBR) and DDMRP by simulation. Int. J. Prod. Res. 60 (3), 1036–1050. https://doi.org/10.1080/00207543.2020.1849847.

Tsai, Y.T., Lee, C.H., Liu, T.Y., Chang, T.J., Wang, C.S., Pawar, S., Huang, P.H., Huang, J. H., 2020. Utilization of a reinforcement learning algorithm for the accurate alignment of a robotic arm in a complete soft fabric shoe tongues automation process. J. Manuf. Syst. 56, 501–513. URL: https://www.sciencedirect.com/science/article/pii/S0278, doi:10.1016/j.jmsy.2020.07.001.

Velasco Acosta, A.P., Mascle, C., Baptiste, P., 2019. Applicability of demand-driven MRP in a complex manufacturing environment. Int. J. Prod. Res. 58 (14), 4233–4245. https://doi.org/10.1080/00207543.2019.1650978.

Wang, J., Qu, S., Wang, J., Leckie, J.O., Xu, R., 2017. Real-time decision support with reinforcement learning for dynamic flowshop scheduling, in: Smart SysTech 2017; European Conference on Smart Objects, Systems and Technologies, Munich, Germany. URL: ⟨https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8084561⟩.

Xanthopolous, A.S., Kiatipis, A., Koulouriotis, D.E., Stieger, S., 2018. Reinforcement learning-based and parametric production-maintenance control policies for a deteriorating manufacturing system. URL: https://ieeexplore.ieee.org/stamp/stamp.

jsp?tp=&arnumber=8114172 IEEE Access 6, 576–588. https://doi.org/10.1109/ACCESS.2017.2771827.

Zhou, R., Lei, D. and Zhou, X., 2019. Multi-Objective Energy-Efficient Interval Scheduling in Hybrid Flow Shop Using Imperialist Competitive Algorithm. in *IEEE Access,* vol. 7, pp. 85029–85041, doi:⟨10.1109/ACCESS.2019.2924998⟩.

Zhou, T., Tang, D., Zhu, H., Zhang, Z., 2021. Multi-agent reinforcement learning for online scheduling in smart factories. Robotics and computer-integrated Manufacturing 72. URL: https://www.sciencedirect.com/science/article/pii/S0736584521000855, doi:⟨10.1016/j.rcim.2021.102202⟩.