Production Engineering (2021) 15:35–43
https://doi.org/10.1007/s11740-020-01000-8

PRODUCTION MANAGEMENT

A deep q-learning-based optimization of the inventory control in a linear process chain

M.-A. Dittrich[1] · S. Fohlmeister[1]

Received: 30 July 2020 / Accepted: 6 November 2020 / Published online: 23 November 2020

Abstract
Due to growing globalized markets and the resulting globalization of production networks across different companies, inventory and order optimization is becoming increasingly important in the context of process chains. Thus, an adaptive and continuously self-optimizing inventory control on a global level is necessary to overcome the resulting challenges. Advances in sensor and communication technology allow companies to realize a global data exchange to achieve a holistic inventory control. Based on deep q-learning, a method for a self-optimizing inventory control is developed. Here, the decision process is based on an artificial neural network. Its input is modeled as a state vector that describes the current stocks and orders within the process chain. The output represents a control vector that controls orders for each individual station. Furthermore, a reward function, which is based on the resulting storage and late order costs, is implemented for simulations-based decision optimization. One of the main challenges of implementing deep q-learning is the hyperparameter optimization for the training process, which is investigated in this paper. The results show a significant sensitivity for the leaning rate α and the exploration rate ε. Based on optimized hyperparameters, the potential of the developed methodology could be shown by significantly reducing the total costs compared to the initial state and by achieving stable control behavior for a process chain containing up to 10 stations.

Keywords Inventory control · Deep q-learning · Process chain · Self-optimizing control · Learning parameters

List of symbols

α	Learning rate
α_d	Decay of the learning rate
a_{random}	With possibility ε randomly chosen action at time t
a_t	Action at time t chosen by the agent
C_{late}	Late order costs
$C_{s,n}$	Storage costs of station n
cv_{dim}	Dimension of the control vector/number of the neurons of the output layer
e_{max}	Number of training episodes per training
ε	Exploration rate
ε_d	Decay of the learning rate
f_c	Total cost factor
f_{late}	Late order cost factor
f_{sc}	Storage cost factor
γ	Discount factor
$id(\ldots)$	Identity function
m	Number of simulated time steps per training episode
MAE	Mean absolute error
MDP	Markov decision process
n	Number of intermediate stations
$n_{warm-up}$	Number of warm-up episodes
$o_{late,n}$	Number of late orders at station n
o_n	Order placed by station n
$o_{open,n}$	Number of open orders within a simulated day at station n
o_q	Interval of the order quantity
o_t	Interval of the order distance
$Q(s_t, a_t)$	Quality of an action during a state s at time t
ReLU	Rectifier linear unit
r_t	Reward at time t
$s_{dis,n}$	Disposable stock at station n
s_{memory}	Number of data sets that can be stored in the replay memory
$s_{ph,n}$	Physical stock at station n
s_t	Observed state at time t

✉ S. Fohlmeister
fohlmeister@ifw.uni-hannover.de

[1] Institute of Production Engineering and Machine Tools, An der Universität 2, 30823 Garbsen, Germany

sv_{dim}	Dimension of the state vector/number of the neurons of the input layer
τ	Target model update
t_{ij}	Transportation time from station *i* to *j*

1 Introduction

Today, manufacturing companies are confronted with a multitude of challenges that make cost-efficient production significantly more difficult. These challenges result from the increasing globalization of sales markets, to which the actual production processes must be adapted. To minimize logistics costs and respond flexibly to the local demand, manufacturing processes are becoming increasingly decentralized through the setup of manufacturing sites along this local demand [1]. This trend is accompanied by a mass customization which leads to an increased customer demand for highly individualized products within the shortest possible delivery times [2, 3]. These two trends are leading to the integration of increasingly complex process chains which in turn prompts correspondingly intricate material flows both within a single company and across different companies.

The timely and cost-efficient planning and control of process chains must be considered at both a local and a global level. Local production planning and control of a single manufacturing system depends on an accurate inventory and demand planning across multiple factories. This way, local production capacities can be correctly planned and, if necessary, rescheduled on short notice [4]. On a global level, however, there is often the issue that the site-specific requirements are not known in advance or are insufficiently clear based on customer demand. As shown in the lower half of Fig. 1, this leads to the so-called bullwhip effect. Here, customer-side demand fluctuations along the associated process chain are overinterpreted and larger inventories than are necessary for processing the current demand are built up. This leads to increased capacity requirements at the local level, which must be controlled accordingly [5]. The following reasons for the bullwhip effect are listed in the literature: First, information sharing among different companies across the process chain is necessary to achieve an optimized inventory control [6]. However, the order and production requirements are usually planned decentrally for each station of the process chain. That means that information for holistic control is missing and is methodically not integrated in the used software systems, although this could be possible due to modern information technology [7]. In addition, rules for inventory control based on past demands are often developed, yet they do not correctly reflect future changes in ordering behavior [8]. Therefore, flexibility is necessary in current globalized and volatile markets. Although many control approaches exist, they are either not flexible enough or consider the process chain to be controlled only at a local level and cannot therefore implement a holistic inventory optimization. However, due to technological advances, the collaboration among distributed sites is possible [9]. Therefore, the main contribution of this article is a new method that enables a self-optimizing inventory control of a global process chain to plan and control local production capacities efficiently despite fluctuating demands. Compared to existing methods, it is not necessary to develop a mathematical model to describe the decision problem. This is especially important when a continuous adaption to changing customer behavior is necessary. Hence, this article presents a novel approach for a self-optimizing inventory control using machine learning techniques in Sect. 3, the implementation in Sects. 4 and 5 and the investigation of the developed methodology in Sect. 6.

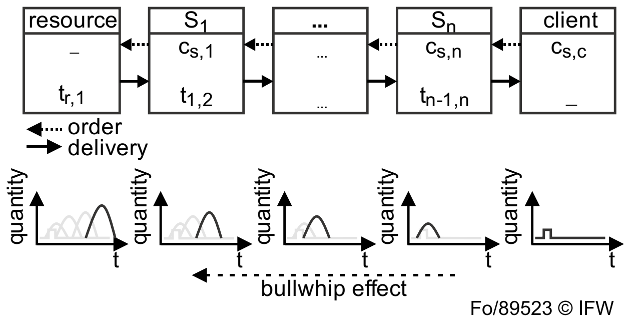

Fig. 1 Description of a linear process chain and the bullwhip effect

2 Related work

Inventory control in cross-company process chains includes the order of necessary raw materials or semi-finished products for each station, the actual production process and delivery to the next processing station. Minimizing the inventory while guaranteeing the capability to deliver is of crucial importance [10]. This is often done with rule-based algorithms. These rules are derived from past demand patterns [11]. However, due to their low flexibility, these methods are not suitable for strongly varying demand patterns.

From a mathematical perspective, the decision problem in the context of the inventory control of a process chain can be described as a Markov decision process (MDP). Accordingly, the probability that a certain subsequent system state will occur depends exclusively on the current system state. Decision problems of this kind can be modelled by the method of reinforcement learning [12]. In contrast to supervised learning, there is no need for a description of the training data by labels. Instead, control decisions are made by a software agent and the resulting subsequent state is evaluated. This evaluation is based on a reward function and the discounted summation of the best possible subsequent state evaluations. The repeated application of the algorithm leads to an iteratively adaption of the stored evaluations, which in turn results in self-optimized decision-making [13, 14]. The reinforcement learning approach was applied by Giannoccaro et al. to control a process chain. The decision problems were modelled as an MDP and solved by a SMART algorithm [15]. Similar approaches were investigated by Valluri et al. and Mortazavi et al. They implemented a q-learning agent for local elements of the process chain, which controlled the inventory of the respective station by triggering orders based on a cost-oriented reward function. In both cases, the complexity of the resulting decision problem could only be handled to a limited extent. Even after 150,000 training episodes, Valluri et al. did not achieve sufficient learning progress to ensure a stable control behavior by the agent [16]. This problem was solved by Mortazavi et al. [17] by limiting the state space for the local agent to eight possible system states. The corresponding action space was limited to seven possible control decisions. In these two approaches, the limits of previous reinforcement learning attempts regarding the control of complex process chains have become clear.

Through the approach of deep q-learning, a converging learning behavior can also be realized for more complex decision problems [18]. The characteristic feature of this approach is a replay memory. As shown in Fig. 2, for each learning sequence a random data set consisting of the current system state s_t, the selected action a_t, as well as the resulting subsequent state s_{t+1} and the calculated reward r_t is selected [19]. As a consequence, data sets of successive and thus strongly correlating states are not used in subsequent learning steps. That way, an over-adaptation of the algorithm for the respective decision situation is avoided [20]. Furthermore, the function approximator is not implemented by a single artificial neural network as in previous approaches. Instead, an action network is used to approximate the action-value function. In the next step, a second target network specifies the target value for the gradient descent step within the optimization. This network is derived with a factor τ from the action network. This ensures stable target values and thus a stable learning behavior for complex decision problems. As shown in Fig. 2, the selection of an action a_t is mainly based on the action network. However, to avoid local optimization, an ε-greedy policy is implemented. Thus, actions are selected randomly with probability ε to explore previously unknown solutions [18].

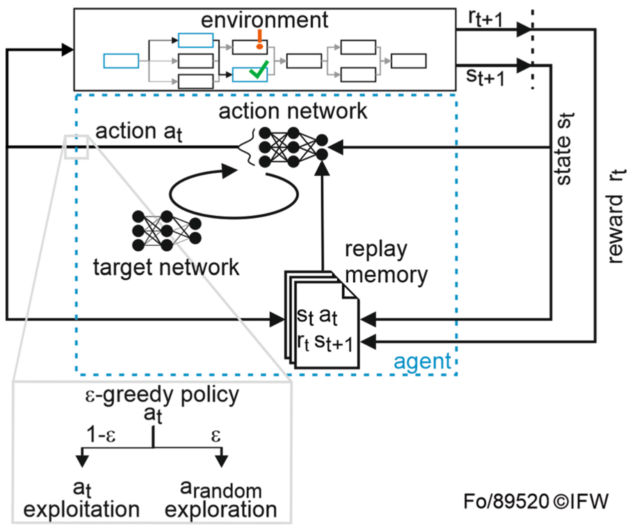

Fig. 2 Methodology of the deep q-learning

Oroojlooyjadid et al. [21] have used the deep q-learning algorithm to implement a self-optimizing method for the inventory control of a single station within a linear process chain. The remaining stations were controlled by predefined rules. They were able to show that deep q-learning is suitable for achieving convergent learning behavior even for complex systems. However, the disadvantage of this approach is that control optimization is methodically only provided locally for a single station. The possibility to acquire operating data over the entire process chain and thus, optimize the inventory of the entire process chain is not provided.

The presented methods show that it is possible to control the inventory of a linear cross-company process chain. However, due to current requirements caused by globalized and volatile markets, a holistic and flexible method for inventory control, which can efficiently model and solve complex system states, is necessary.

3 Methodology for self-optimizing inventory control

The overall method designed for a self-optimizing inventory control based on a deep q-learning agent is shown in Fig. 3.

The developed method is initially divided into four substeps. During initialization, the control agent is linked to the corresponding process chain by parameterizing the state and control vectors as well as the reward function according to the respective application and by setting up the function approximator as described in Sect. 3.1. Before the actual training phase, in the warm-up phase, $n_{warm\text{-}up}$ episodes are performed without using the implemented learning behavior. Instead, random control decisions are chosen. The resulting data of the respective system states are used to initially fill the replay memory.

The actual training phase is subdivided into e successive training episodes. A training episode describes a cycle in which control decisions are made and implemented repeatedly while considering the resulting stock levels and costs. As a result, each training episode begins with the

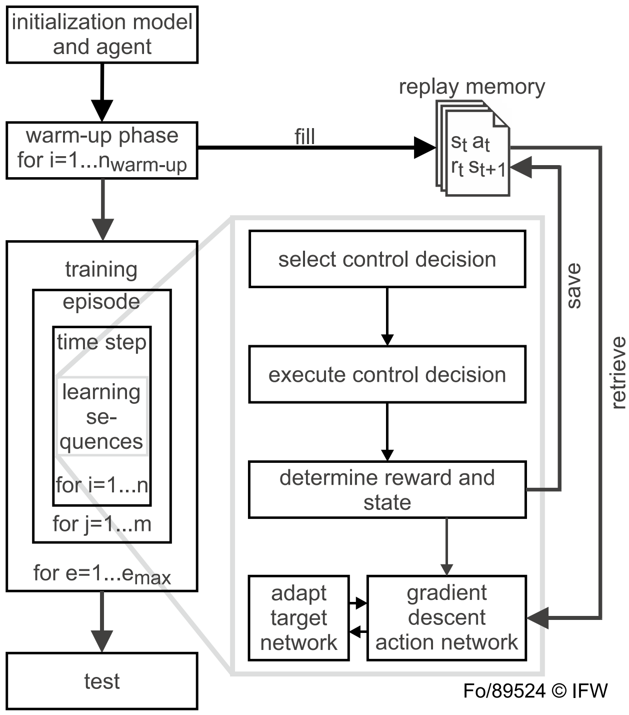

Fig. 3 Overall method for self-optimizing inventory control

same initial system state s_0. This also ensures that long-term control decisions are considered by the developed method. If a training episode only included one control decision and its direct effects, the decision-making would be optimized based on the control decisions which would lead to the best possible subsequent state s_{t+1} without considering the resulting subsequent system states s_{t+2} to s_m.

Depending on the specific use case, a training episode is subdivided into m time steps. On each of these time steps, n control iterations are performed according to the number of intermediate stations in the process chain. This ensures that a control decision can be made within one time step for each processing station. Alternatively, it would have been possible to parameterize an output layer whose neurons describe a set of control decisions for each of the stations. However, for complex process chains in particular, this procedure would lead to a correspondingly complex function approximator and consequently to significantly increased computing time. In each of these iterations, the actual learning process is integrated, after which a control decision is selected and executed. The recorded operating data of the state of the process chain is stored in the replay memory in the same way as during the warm-up phase. Subsequently, a random data set is selected from the replay memory, based on which the action network is adapted using the gradient descent optimization following the optimization process described in Sect. 3.2.

The final step of the method is the test and application of the previously trained deep q-learning agent. A further training episode is simulated in the same way as the procedure in the previous step. The difference is that no further optimization of the decision-making process is conducted. Instead, the optimized control behavior is investigated over a longer period so that conclusions can be drawn about the optimization quality with regard to the bullwhip effect. Regarding a practical application of the developed method, this procedure allows the control network to be trained on a simulation basis in the first step. It is then applied for the control of the real system without the danger of an unwanted change in the decision behavior due to a further adjustment of the action network.

3.1 Parameterization of the control agent

Figure 4 displays the artificial neural network that is used as the function approximator within the framework of the implemented deep q-learning algorithm. The input neurons of the artificial neural network represent the state vector, which describes the current state of the process chain as a basis for decision-making. The observation vector quantifies the physical and disposable stock $s_{ph,n}$ and $s_{dis,n}$ for each station regarding the costs and the open and late orders $o_{open,n}$ and $o_{late,n}$ for the orders for each of the intermediate stations. Thus, the dimension of the observation vector depends on the number of intermediate stations within the process chain. In addition, the open and backorders of the customer are quantified. Since a direct utilization of the order is assumed, no storage costs are considered methodically for the customer. The dimension of the input vector and thus the number of input neurons of the artificial neural network are calculated according to the following equation:

$$sv_{\text{dim}} = 4 \cdot n + 2 \quad (1)$$

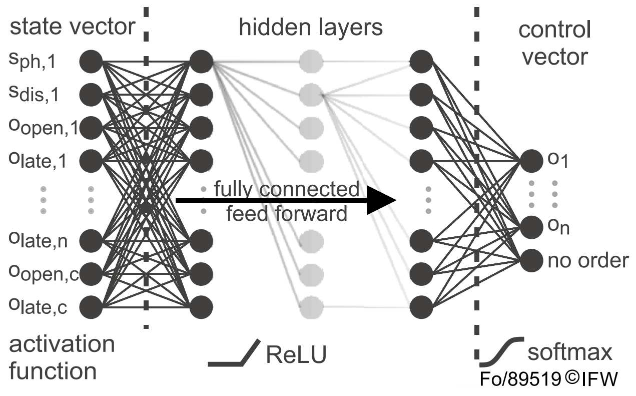

Fig. 4 Function approximator for self-optimizing control decisions

The output neurons represent the control vector. This describes the set of all possible decisions that can be made by the self-optimizing process chain control method based on the observation vector. The set of possible decisions contains the orders that can be placed by each of the stations. In addition, there is the possibility that none of the stations triggers an order. Accordingly, the number of output neurons or the dimension of the control vector is calculated according to the following equation:

$$cv_{\dim} = n + 1 \tag{2}$$

The methodical linking of the observation and control vectors is done by three hidden layers, which are linked by a fully connected and feed forward structure across all layers. This structure was chosen due to its high approximation capability for pattern recognition and decision-making [22, 23]. More intricate structures like convolutional layers are not necessary because of the lower complexity of the input data compared to, for instance, image recognition. In particular, complex structure recognition is not necessary. The dimension of each hidden layer is identical to the dimension of the input layer.

A rectifier linear unit (ReLU) was selected as the activation function for both the input layer and the following hidden layers. The reason for this is the high robustness towards the vanishing gradient problem [24]. In contrast, the softmax function was selected for the output neurons of the action vector to obtain result values normalized to the interval [0;1] for a high comparability.

3.2 Optimization of the decision process for inventory control

The optimization of the decision behavior during inventory control is achieved by the systematic adaptation of the edge weights of the artificial neural network. The necessary basis for optimization is formalized by the following error function:

$$error = r_t + \gamma \max_a Q'\left(s_{t+1}, a'_t\right) - Q\left(s_t, a_t\right). \tag{3}$$

The value $Q(s_t,a_t)$ is calculated by the control network. $Q'(s_{t+1},a'_t)$ is the corresponding result of the target network. The central element of reinforcement learning is the feedback on the consequences of a control decision for the observed environment. For this reason, the definition of a reward function r_t is of particular importance. The decision criterion for controlling the existing process chain is formed by the resulting costs. These are subdivided into storage and backorder costs. The storage costs result from existing stocks due to excessive order quantities at the intermediate stations. The backlog costs, in turn, result from insufficient order quantities. Too prevent an over-stimulation of the learning algorithm, the interval of the value of the reward function is limited to [− 1;1]. This is achieved by the following function:

$$r_t = \max\left[-1;1 - f_c \cdot \left(f_{sc} \cdot C_s + f_{late} \cdot C_{late}\right)\right] \tag{4}$$

According to this function, the value of the reward at a time t is calculated as a function of the storage and backlog costs C_s and C_{late}. Using the weighting factors f_{sc} and f_{late}, the relative weights of these two variables can be varied as a function of the observed learning behavior and the case-specific target variables. Depending on the size of the system, the total costs are calculated using the total cost factor f_c so that they lie within the defined interval of the reward function. In addition, f_c can be used to adjust the weighting of the reward function within the gradient descent optimization by considering higher costs to a greater or lesser extent. For the optimization process, stochastic gradient descent (SGD) was selected as the optimizer and the error was calculated as the mean average error (MAE).

4 Experimental setup

For the purpose of validation, the developed method was implemented by software. First, the linear process chain was modelled according to the underlying model assumptions. For this purpose, SimPy, a framework for a discrete-event simulation, was used. The exact parameters and characteristics are shown in the bottom part of Figs. 5, 6, 7, 8.

The actual deep q-learning agent was implemented in the TensorFlow framework using the Keras-rl library as part of the Keras programming interface. The algorithm was realized in the Python programming language, which ensured maximum compatibility. The learning parameters of the algorithm are again displayed in Figs. 5, 6, 7, 8. The interface between the deep q-learning agent and the environment is based on the standardized OpenAI Gym platform to ensure the compatibility to different reinforcement learning algorithms. A second agent was also implemented for the purpose of validating the developed method. After the training phase, the optimized edge weights of the control network are transferred to the agent to test the optimized control behavior over a long-term period.

For the calculation of the reward, the late order and storage costs are weighted equally with $f_{sc} = f_{late} = 1$ and the overall costs are weighted with $f_c = 0.02$ for all conducted experiments.

5 System description

For experimental investigation, a linear process chain based on the MIT beer distribution game [25] is considered, which is subdivided into the elements *resource*, *customer* and *intermediate stations*. Only one resource and one customer, but any number of intermediate stations, are allowed. The material flow is always linear from the resource via the intermediate stations to the customer. The information flow in the form of a purchase order is reversed. The individual process chain elements are described in their behavior using the following characteristics: The *customer* triggers orders at evenly distributed time intervals o_t. The order quantity is also evenly distributed in a specified interval o_q. For randomization, a fixed seed is used for the triggered orders. To avoid overfitting, each episode is assigned a specific seed based on the identity function *id* so that $seed = id(e)$. The *resource* provides the raw material and is not dependent on upstream manufacturing processes. The *intermediate stations* receive purchase orders from the upstream element and forward derived purchase orders to the downstream element. For this, fixed order quantities of 5 units are considered.

A transport time $t_{ij} = 1$ day is assumed for the transportation of the orders between the stations. At the stations, unlimited quantities of orders can be stored temporarily, raising storage costs based on a cost factor $c_{s,n} = 1$ monetary unit per unit and day during this time. The state of the process chain follows directly from the number of currently open and backorders of each intermediate station and of the customer. The value of the respective orders can be used to calculate the storage and backlog costs for each station individually and across the entire process chain. The underlying model of the process chain is shown in the upper part of Fig. 1.

6 Experimental results

As shown in Fig. 5, a converging learning behavior can be realized for a process chain of four stations. The upper left part clearly shows the plateau of the warm-up phase. According to the developed method, no optimization of the decision-making process is conducted during these episodes. The actual training phase then begins in episode 6. After about 14 episodes, the agent develops a control behavior that reproducibly leads to low normalized overall costs. Especially the storage costs are almost completely reduced. Accordingly, as shown in the lower part of Fig. 5, autonomous inventory control of the process chain is possible in the subsequent test phase. The initial increase of late orders results from the fact that the process chain is not filled at the beginning of the simulation and therefore the first orders cannot be processed directly.

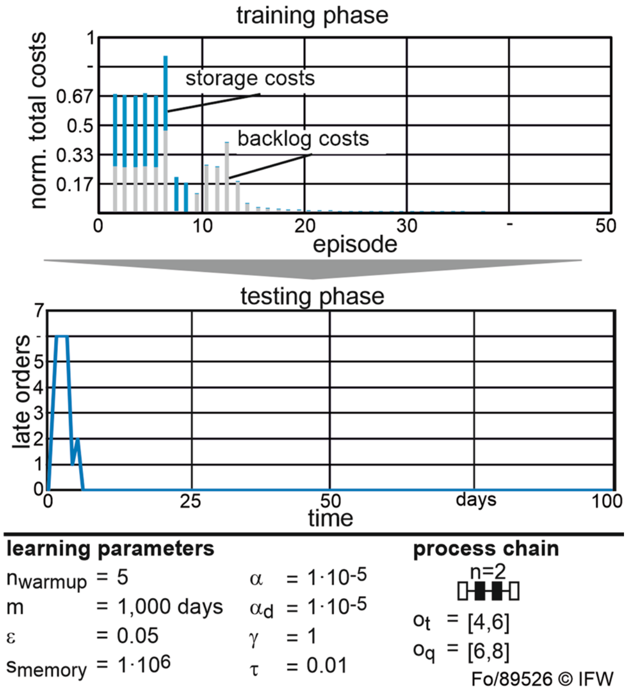

Fig. 5 Learning behavior and test application within a process chain of four stations

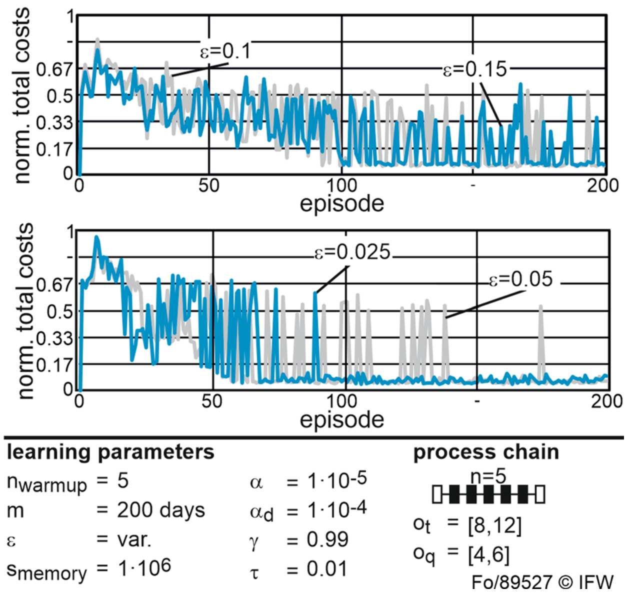

Fig. 6 Influence of the exploration rate ε

6.1 Sensitivity analysis towards changing system behavior

An increase in system complexity of up to seven stations leads to a correspondingly more complex control vector. To avoid the formation of local optima, a random control decision is made with probability ε to find previously unknown solutions. The influence of ε on inventory optimization is shown in Fig. 6. In particular for higher values of $\varepsilon = 0.1$ and $\varepsilon = 0.15$ the achieved total costs diverge clearly. It is obvious that random control decisions are selected too often, which do not lead to an improvement of the control

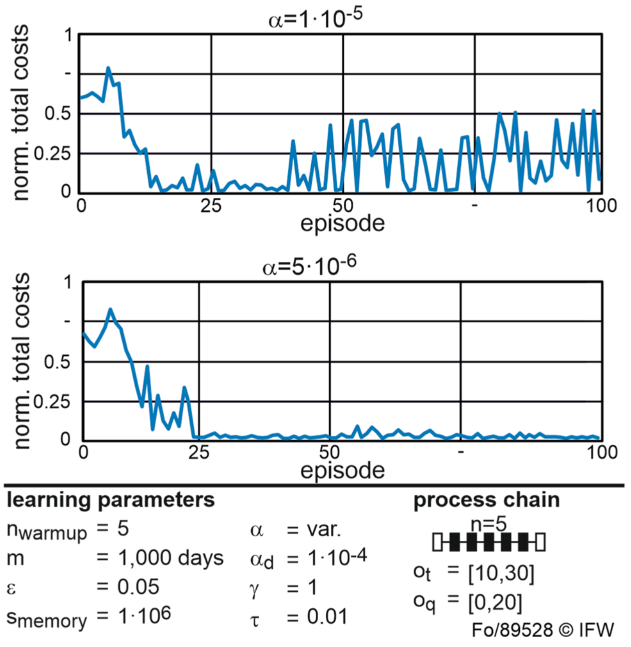

Fig. 7 Adaption of the learning rate α due to a higher uncertainty regarding order quantity and time

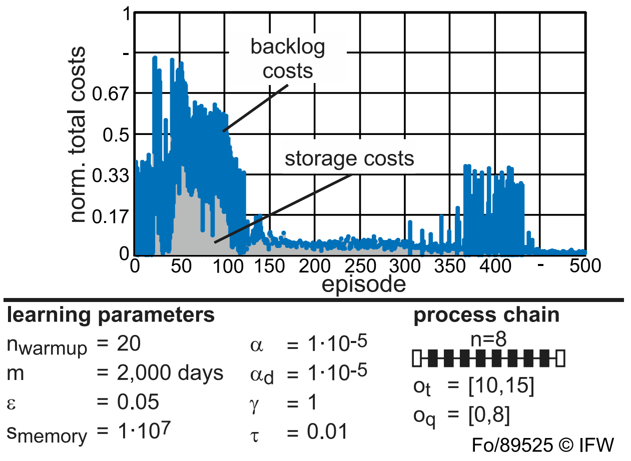

Fig. 8 Learning behavior within a process chain of 10 stations

behavior. This results in a lack of a suitable optimization data basis for the control agent to keep the inventory as low as possible. The best result was achieved with $\varepsilon = 0.05$. Here, enough random control decisions were made to develop new control strategies, and on the other hand, only so many that a sufficiently constant learning basis was provided.

To evaluate the sensitivity of the developed method regarding stochastic changes in the behavior of the process chain, the intervals of the allowed order quantities o_q and times o_t by the customer were increased to [10, 30] and [0, 20]. As shown in Fig. 7, the previously used value of the learning rate $\alpha = 1 \times 10^{-5}$ leads to a diverging learning behavior. The reason for this is the overestimation of individual outliers in the customer's ordering behavior. A convergent learning behavior can only be achieved by reducing the learning rate to $\alpha = 5 \times 10^{-6}$.

6.2 Investigation of the system stability

To demonstrate the suitability of the developed method for more complex process chains, a linear process chain with eight intermediate stations was implemented. Figure 8 shows the learning progress of the control agent. Due to the higher system complexity, the number of simulated days per training episode m was increased to 2000 days. In addition, the size of the replay memory and thus the number of warm-up episodes was adjusted. Overall, the number of necessary training episodes further increased to about 125 episodes. After the total costs could be kept constantly low for about 200 training episodes, a significant increase in backlog costs could be observed from episode 320 onwards. This could be reduced again after about 100 episodes. The reason for this could be an insufficient decay of the exploration rate ε_d. Because of that, a relatively large number of exploration steps are still carried out even after 300 episodes and thus control decisions are realized that do not correspond to the actually found optimum.

One of the biggest challenges in controlling the inventory in such a complex linear supply chain is to avoid the bullwhip effect. To validate the stability of the control method, Fig. 9 shows the physical stock $s_{ph,n}$ for the eight intermediate stations of the supply chain and the customer's late orders over a simulated period of 1000 days during the test phase. Here, too, an increase in the late orders of the customer $o_{late,c}$ can be observed, which is reduced after about 100 days. Subsequently, a constantly low level of late orders can be maintained. At the same time, the stocks of the intermediate stations are kept at a constantly low level. At stations 1, 3 and 8, a clear increase in stocks can be seen after about 700 days, but these are reduced again within a maximum of 100 days. One explanation could be the relatively high number of late orders at the customer for a longer period after about 600 days. Thus, it has been shown that the method can efficiently reduce the bullwhip effect despite fluctuating customer demand and keep stocks low or reduce them after a temporal increase within the system.

7 Conclusion and outlook

Increasing globalization and volatile markets with fluctuating demand pose high requirements for the inventory control of cross-company process chains. Hence, in this paper, a method was presented for the self-optimizing control

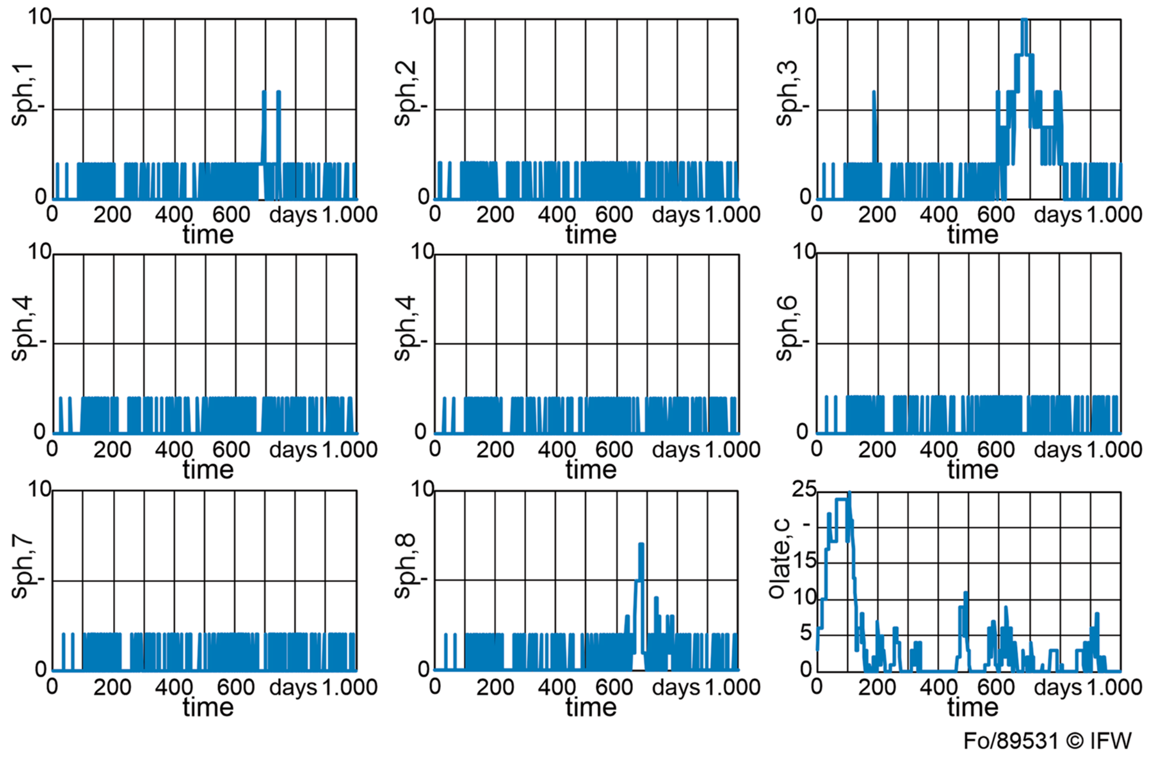

Fig. 9 Development of the stock size and late orders during the test application

of a linear process chain based on the deep q-learning algorithm. For this purpose, state and control vectors and a reward function were parameterized based on the resulting costs within the process chain. The developed method was validated based on simulation using various complex linear process chains. Consequently, it has been shown that the control of complex process chains with up to ten elements is possible. For this, the correct parameterization of the learning parameters is crucial to ensure convergent learning behavior.

Based on the presented results, future work will focus on the comparison to conventional methods for inventory control and the adaptation of the developed method for more complex use cases. This applies in particular to nonlinear process chains within a local production system, as they usually occur in job shop production. Furthermore, it will be investigated how the method can be designed in such a way that transfer learning can be realized. On this basis, a simulation-based learning environment can be developed, which will enable the training of the control method and then its transfer to the actual production system. In addition, the approach will be applied to a real use case to investigate the learning and control behavior under these different conditions.

Funding Open Access funding enabled and organized by Projekt DEAL. The presented investigations were conducted within the research project DE 447/181-1. We would like to thank the German Research Foundation (DFG) for the support of this project. In addition, we would like to thank Prof. Dr.-Ing. Berend Denkena for his valuable comments and his support.

References

1. Mourtzis D, Doukas M, Psarommatis F (2013) Design and operation of manufacturing networks for mass customisation. CIRP Ann Manuf Technol 62(1):467–470. https://doi.org/10.1016/j.procir.2014.05.004
2. Daaboul J, Da Cunha CM, Bernard A, Laroche F (2011) Design for mass customization: Product variety vs. process variety. CIRP Ann Manuf Technol 60(1):169–174. https://doi.org/10.1016/j.cirp.2011.03.093

3. ElMaraghy H, Schuh G, ElMaraghy W, Piller F, Schönsleben P, Tseng M et al (2013) Product variety management. CIRP Ann Manuf Technol 62(2):629–652. https://doi.org/10.1016/j.cirp.2013.05.007
4. Pinedo ML (2012) Scheduling—theory, algorithms, and systems, 4th edn. Springer, New York
5. Forrester J (1961) Industrial dynamics. MIT Press, Cambridge
6. Cachon GP, Fischer M (2000) Supply chain inventory management and the value of shared information. Manage Sci 46(8):1032–1048. https://doi.org/10.1287/mnsc.46.8.1032.12029
7. Monostori L, Kádár B, Bauernhansl T, Kondoh S, Kumara S, Reinhart G et al (2016) Cyber-physical systems in manufacturing. CIRP Ann Manuf Technol 65(2):621–641. https://doi.org/10.1016/j.cirp.2016.06.005
8. Lee HL, Padmanabhan V, Whang S (1997) The bullwhip effect in supply chains. Sloan Manag Rev 38:93–102
9. Lanza G, Ferdows K, Kara S, Mourtzis D, Schuh G, Váncza J et al (2019) Global production networks: design and operation. CIRP Ann Manuf Technol 68(2):823–841. https://doi.org/10.1016/j.cirp.2019.05.008
10. Lödding H (2013) Handbook of manufacturing control—fundamentals, description, configuration. Springer, Heidelberg
11. Sethupathi PVR, Rajendran C, Ziegler H (2013) A comparative study of periodic-review order-up-to (T, S) policy and continuous-review (s, S) policy in a serial supply chain over a finite planning horizon. In: Pisarenko VF, Ramanathan R, Rmanathan U (eds) Supply chain strategies. Springer Issues and Models, London, pp 113–152
12. Alpaydin E (2009) Introduction to machine learning. MIT press, Cambridge
13. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. The MIT Press, Cambridge
14. Watkins C, Dayan P (1992) Q-Learning. Mach Learn 8:279–292. https://doi.org/10.1007/BF00992698
15. Giannoccaro I, Pontrandolfo P (2002) Inventory management in supply chains: a reinforcement learning approach. Int J Prod Econ 78:153–161. https://doi.org/10.1016/S0925-5273(00)00156-0
16. Valluri A, North MJ, Macal CM (2009) Reinforcement learning in supply chains. Int J Neural Syst 19(5):331–344. https://doi.org/10.1142/S0129065709002063
17. Mortazavi A, Khamseh AA, Azimi P (2015) Designing of an intelligent self-adaptive model for supply chain ordering management system. Eng Appl Artif Intell 37:207–220. https://doi.org/10.1016/j.engappai.2014.09.004
18. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG et al (2015) Human-level control through deep reinforcement learning. Nature 518:529–533. https://doi.org/10.1038/nature14236
19. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D et al (2013) Playing Atari with deep reinforcement learning. DeepMind Technologies
20. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA (2017) A brief survey of deep reinforcement learning. IEEE Signal Process Mag. https://doi.org/10.1109/MSP.2017.2743240
21. Oroojlooyjadid A, Nazari M, Snyder L, Takáč MA (2017) Deep Q-network for the beer game: a reinforcement learning algorithm to solve inventory optimization problems. In: Neural information process systems (NIPS), deep reinforcement learning symposium
22. Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Netw 4(2):251–257. https://doi.org/10.1016/0893-6080(91)90009-T
23. Yang J, Ma JA (2016) Structure optimization framework for feed-forward neural networks using sparse representation. Knowl-Based Syst 109:61–70. https://doi.org/10.1016/j.knosys.2016.06.026
24. Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. In: Proceedings of the 14th international conference on artificial intelligence and statistics (AISTATS), vol 15, pp 315–323
25. Sterman JD (1989) Modeling managerial behavior: misperceptions of feedback in a dynamic decision making experiment. Manage Sci 35(3):321–339