Annals of Operations Research (2022) 308:415–439
https://doi.org/10.1007/s10479-021-03935-2

S.I. : ARTIFICIAL INTELLIGENCE IN OPERATIONS MANAGEMENT

Artificial intelligence-based inventory management: a Monte Carlo tree search approach

Deniz Preil[1] · **Michael Krapp**[1]

Accepted: 8 January 2021 / Published online: 19 April 2021

Abstract
The coordination of order policies constitutes a great challenge in supply chain inventory management as various stochastic factors increase its complexity. Therefore, analytical approaches to determine a policy that minimises overall inventory costs are only suitable to a limited extent. In contrast, we adopt a heuristic approach, from the domain of artificial intelligence (AI), namely, Monte Carlo tree search (MCTS). To the best of our knowledge, MCTS has neither been applied to supply chain inventory management before nor is it yet widely disseminated in other branches of operations research. We develop an offline model as well as an online model which bases decisions on real-time data. For demonstration purposes, we consider a supply chain structure similar to the classical beer game with four actors and both stochastic demand and lead times. We demonstrate that both the offline and the online MCTS models perform better than other previously adopted AI-based approaches. Furthermore, we provide evidence that a dynamic order policy determined by MCTS eliminates the bullwhip effect.

Keywords Monte Carlo tree search · Supply chain inventory management · Artificial intelligence · Bullwhip effect

1 Introduction

The supply chain management literature spans a wide range of topics, such as facility location, production, scheduling, transportation, return of goods, forecasting, and inventory management, this last being the main subject of this paper. One key task of supply chain management is the integrated planning and control of the inventory of all actors in the supply chain, from the source of supply to the end user, to reduce the overall inventory costs while improving customer service (Ellram 1991). Reducing inventory costs is of major importance, as these costs account for a considerable proportion of the companies' total logistics costs (Rood-

✉ Deniz Preil
deniz.preil@wiwi.uni-augsburg.de

[1] Department of Quantitative Methods in Economics, University of Augsburg, Universitaetsstr. 16, 86159 Augsburg, Germany

bergen et al. 2015; Lancioni 2000). Particularly crucial is the coordination of order policies, specifically, the determination of an order policy in an integrated manner (Li and Wang 2007).

However, this effort is hampered by the bullwhip effect, which refers to increasing order variability when moving up the supply chain. Examples of this effect occurring in industry can be found, e.g. in Lee et al. (1997) and Fransoo and Wouters (2000). According to Lee et al. (1997), the four main causes of the bullwhip effect are (1) demand forecast updating, (2) order batching, (3) price fluctuation, and (4) rationing and shortage gaming. In addition, there are several other causes, such as stochastic lead times or behavioural aspects (Chatfield et al. 2004; Bhattacharya and Bandyopadhyay 2011). Croson and Donohue (2006) as well as Nienhaus et al. (2006) investigate human behaviour in a multi-echelon supply chain and conclude that it also contributes to the undesired amplification of order variability. An appropriate order policy should therefore not only minimise overall inventory costs, but also aim at mitigating or even eliminating the bullwhip effect, albeit these two goals are often interrelated.

In this study, we consider an exemplary serial multi-echelon supply chain with four actors: a retailer, a distributor, a manufacturer, and a supplier. In each period the exogenous customer demand arrives at the retailer, who, in turn, replenishes inventory by placing an order with the distributor. The distributor places an order with the manufacturer, who, in turn, places an order with the supplier. The latter obtains replenishments from an external source with unlimited supply. As a result, in each period it is necessary to determine whether and how much each supply chain actor orders from the respective upstream actor. The supply chain under investigation is similar to the popular beer game (Sterman 1989), a gamification that simulates the material and information flows in a supply chain. Besides stochastic customer demand, we assume stochastic lead times, whereby we allow order crossing, i.e. the orders may overtake each other. The objective is to minimise the overall inventory costs, including holding costs and backorder costs.

Numerous studies have investigated the analytical determination of optimal order policies under a variety of conditions and assumptions. We refer to Axsäter (2015) for a comprehensive review. However, these approaches can only guarantee optimal solutions up to a certain degree of supply chain complexity. That is why optimal order policies are unknown for a large number of supply chain environments (Axsäter 2015). In particular, factors such as stochastic demand or stochastic lead times increase the complexity.

Unlike analytical methods, heuristic approaches, including many from the domain of artificial intelligence (AI), are more capable of coping with complex and uncertain large-scale multi-echelon supply chain environments (Mele et al. 2006; Güller et al. 2015; Mortazavi et al. 2015), as in our setting. The vast majority of previous AI approaches determine the parameters of static (one-shot) order policies, like the classical base-stock policy. Such approaches identify the policies once at the beginning and do not alter them during the problem horizon. Consequently, it is impossible to update these policies to new observations or changes of the supply chain environment. Few state-based approaches model such changes by transitions in the state space, allowing the determination of state-dependent order policies. However, state-based approaches in particular suffer from the curse of dimensionality, as increasing the number of supply chain actors induces an enormous expansion of the state space. Previous studies thus often aggregate the states, so as to drastically reduce the state space, resulting in a loss of information. In contrast, we address the curse of dimensionality by adopting Monte Carlo tree search (MCTS), as this technique is able to cope with extremely large state spaces (Bertsimas et al. 2017). In particular, we propose an online MCTS approach, which periodically incorporates all currently available information into the process of determining and updating order policies. MCTS is a family of algorithms mainly used in the domain of AI (Browne et al. 2012), while applications in operations research are still scarce. Since

MCTS is useful not only for the supply chain under consideration, but also for other complex stochastic planning tasks, we hope to make this technique more known within the operations research community.

Our contribution is threefold. The first and foremost aim is to draw attention to the application of MCTS in the context of supply chain inventory management. To the best of our knowledge, this is the first study employing MCTS for an inventory problem in a multi-echelon supply chain. Second, we implement an MCTS online planning model which determines order polices that takes into account all available information in every period. We are hence able to model the states of the supply chain in a much more detailed way than previous approaches. Third and last, we propose a new type of dynamic MCTS-based order policy. This policy is characterised by the most downstream actor reducing the volatility of demand and, hence, eliminating the bullwhip effect. In addition, this policy is capable of compensating potentially pronounced initial shortages or overstocks.

This paper is organised as follows: in Sect. 2, we briefly review the literature on AI-based approaches to supply chain inventory management. Section 3 outlines the structure of our exemplary supply chain as well as the formal problem. In Sect. 4, the key components of MCTS are described. Additionally, we briefly present the implementation of this approach to determine order policies for the exemplary supply chain. To assess the advantages of the approach, Sect. 5 compares our results with those of two previous studies. Furthermore, we examine to what extent the MCTS-based order policy mitigates the bullwhip effect in the long run. In Sect. 6, we discuss how our results differ from those of previous studies, examine the properties of the dynamic MCTS-based order policy, and provide further implications of our results. Section 7 concludes and highlights promising directions for future research.

2 Artificial intelligence in supply chain inventory management

Even though there is no uniform definition of AI, it basically addresses the development of computer systems to understand and mimic human behaviour (Russell and Norvig 2020). Haenlein and Kaplan (2019) characterise AI as 'a system's ability to interpret external data correctly, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation.' Apart from applications such as self-driving cars or speech recognition (Haenlein and Kaplan 2019), AI also has a wide range of uses in supply chain management, such as forecasting (Carbonneau et al. 2008; Jaipuria and Mahapatra 2014; Chawla et al. 2019), routing (Simić and Simić 2012; Govindan et al. 2019), supplier selection (Chai et al. 2013), and inventory management, just to name a few. Various approaches have been proposed in the last decades for each of those fields. Some of them address not only one single domain of application, e.g. routing, but also a combination of multiple domains, such as inventory routing (Shukla et al. 2013; Sadeghi et al. 2014). However, we focus primarily on inventory management problems where AI-based methods are particularly useful when an optimal order policy is either infeasible or too expensive to implement (Min 2010).

In addition to the categorization at the application domain level, an additional categorization can be made at the method level. In this regard, we follow the categorization of Min (2010). However, we only mention the categories most important for supply chain inventory management. Furthermore, we extend the category of Genetic Algorithms to the broader category of Evolutionary Algorithms, which offer a whole family of nature-inspired algorithms contributing to artificial intelligence (Floreano and Mattiussi 2008).

Evolutionary Algorithms (EAs) consist of population-based metaheuristics adopting the principles of biological evolution. Several solution candidates form a population of individuals (De Jong 2006). In each evolutionary step, a fitness score is calculated for each individual. Then individuals with higher fitness scores are more likely to reproduce. Probably the most widespread subcategory of EAs in supply chain inventory management is the class of Genetic Algorithms (GAs), which are based on operations such as crossover or mutation. Kimbrough et al. (2002) show that GA-based order policies outperform human decision-making in their version of the beer game. In settings where optimal order policies are known, their GA approach is able to find the optimal order policies. Moreover, in another setting, where analytical solutions are not available due to the complexity of the supply chain, Kimbrough et al. (2002) present a so-called $x + y$ order policy as an alternative to the common base-stock policy. According to a base-stock policy, whenever the inventory position falls below the base-stock level S, also referred to as the order-up-to level, an order is placed to raise the inventory position back to S. This policy is also known as a $(S-1,S)$ policy (Muckstadt 2004). Köchel and Nieländer (2005) assume a (s, Q) policy and investigate a GA approach in a larger-scale environment consisting of five echelons in order to identify the reorder point s as well as the order quantity Q for all stages that minimises the overall supply chain costs. Daniel and Rajendran (2005) also propose a GA for a multi-echelon supply chain. However, they assume a periodic-review inventory system relying on base-stock levels. They show that their solution almost matches the optimal one computed by complete enumeration. In a subsequent paper, Daniel and Rajendran (2006) extend their study by proposing three new variants of GAs and one multi-objective GA. Radhakrishnan et al. (2009) pursue a similar approach, while Grahl et al. (2016) consider a more complex supply chain setting and assume differentiated service times. Apart from minimising the overall supply chain costs, the bullwhip effect is often a central object of investigation. O'donnell et al. (2006) show that the bullwhip effect can be reduced significantly by applying GAs. In a recent study, Badakhshan et al. (2020) present a multi-objective GA framework to minimise the bullwhip effect, the cash flow bullwhip as well as the overall supply chain costs.

Besides GAs, there are several other subcategories of EAs. For example, Deshpande et al. (2011) propose a Bacterial Foraging Algorithm in combination with Fuzzy Goal Programming to determine each actor's review period R and order-up-to level S in a multi-objective decision problem. Keshari et al. (2018) apply a Bacterial Foraging Algorithm as well. They assume a multi-echelon supply chain with various transportation modes between each echelon and identify mode-specific base-stock levels. Duan and Liao (2013) present a different framework, based on Differential Evolution and Harmony Search, to determine near-optimal (s,S) policies in a capacitated supply chain. Güller et al. (2015) propose a multi-objective Particle Swarm Optimisation Algorithm in a supply chain assuming (s,Q) policies. Devika et al. (2016) simultaneously analyse both the bullwhip effect and the net stock amplification. They suggest a hybrid algorithm consisting of population-based multi-objective Simulated Annealing and a multi-objective Electromagnetism Mechanism Algorithm. EAs are not only adopted to determine order policies in supply chains, but are also of major importance in several other supply chain management disciplines, cf. Govindan (2016) for a comprehensive overview.

To take into account uncertainties in supply chains, concepts of Fuzzy Logic are often used. They are frequently combined with other methods to determine order policies. For example, Petrovic et al. (1999) present an approach based on Fuzzy Set Theory to model uncertain demand and supply with the aim of identifying cost-minimising base-stock levels. A model based on Fuzzy Set Theory and echelon-stock is presented by Giannoccaro et al. (2003). It aims at minimising the average overall costs over an infinite time horizon. Furthermore,

Wang and Shu (2005) develop a fuzzy supply chain model in combination with a GA to determine cost-minimising base-stock levels subject to fulfilling the product's target fill rate.

A field employing a completely different concept is reinforcement learning (RL). Most RL-based approaches use state-based models, where the states capture the inventory of all supply chain actors. The main idea of RL is to make decisions that depend on signals caused by the interaction with the environment (Sutton and Barto 2018). Over the course of time, a policy is learned autonomously according to these signals. Giannoccaro and Pontrandolfo (2002) study a single product setting under uncertain demand and lead times in a serial system with three echelons. They present an RL approach managing inventory decisions at all echelons of the supply chain in an integrated manner. Chaharsooghi et al. (2008) consider a beer game scenario in an uncertain environment: they compare their RL-based results with the GA-based results of Kimbrough et al. (2002). Jiang and Sheng (2009) apply an RL approach to a supply chain setting with multiple retailers and multiple suppliers to investigate (s,Q) policies as well as (R,S) policies. The values of s and S are learned, based on past experience, with the aim of satisfying a given service level. An RL approach to a four-echelon supply chain with non-stationary customer demand is described by Mortazavi et al. (2015). Most RL order policies can be considered as state-dependent base-stock policies. However, the base-stock levels are not explicitly determined, but result from the best action chosen in each of the states. Unfortunately, previous RL approaches suffer from the curse of dimensionality, requiring a drastic reduction of the state space (Giannoccaro and Pontrandolfo 2002; Chaharsooghi et al. 2008; Mortazavi et al. 2015).

Table 1 compares the studies mentioned above by contrasting their methods, objectives, and proposed order policies. With the exception of the policy proposed by Priore et al. (2019), all order policies in Table 1 share an important characteristic: they are determined once only at the beginning of the planning horizon. In contrast, all available information at each echelon and especially in each period should be considered in the determination of the order policies to improve their performance by adjusting to the dynamics of the supply chain. Previous studies do not (or only to a very limited extent) exploit this potential. Admittedly, RL approaches take into account changes in the environment or new available information by their use of time-varying states. However, the number of states is considerably limited due to the curse of dimensionality. We overcome this problem by applying an online MCTS approach, which periodically incorporates all available information into the process of determining and updating order policies.

Hardly any of the existing AI approaches to inventory management are based on online models. Keshari et al. (2018) therefore emphasise the need for such models. In a recent study, Priore et al. (2019) apply inductive learning (based on a C4.5 classification algorithm) to determine an online as well as a static offline (one-shot) order policy. Moreover, they compare both policies and show the superiority of the online version. In their model, at each echelon and in each period one out of four different options can be chosen. Our approach differs considerably from theirs in many ways. In our model, the options for the periodic update are neither limited nor fixed. Instead, they are based on the precisely recorded state of the supply chain environment. Furthermore, the periodic decisions of our proposed order policy are not independent: since a decision made in the last period affects the current state (due to delays in flows of material and information), it also affects the current decision.

Thus, the following research questions arise. How powerful is our dynamic MCTS approach compared to existing AI approaches? How do MCTS order policies affect the overall supply chain costs? And to what extent can they counteract the bullwhip effect as well as undesirable situations with major shortages or overstocks?

Table 1 Review of AI approaches to supply chain inventory management with multiple echelons

Study	Method	Objective	Order policy
Petrovic et al. (1999)	Fuzzy set theory	Min. overall costs	$(S-1,S)$ policy
Giannoccaro and Pontrandolfo (2002)	Reinforcement learning	Min. average overall costs	State-dependent $(S-1,S)$ policy
Kimbrough et al. (2002)	Genetic algorithm	Min. overall costs	$x+y$ policy
Giannoccaro et al. (2003)	Fuzzy set theory	Min. average overall costs	$(S-1,S)$ policy
Daniel and Rajendran (2005)	Genetic algorithm	Min. overall costs	$(S-1,S)$ policy
Köchel and Nieländer (2005)	Genetic algorithm	Min. overall costs	(s,Q) policy
Wang and Shu (2005)	Fuzzy set theory, genetic algorithm	Min. overall costs	$(S-1,S)$ policy
Daniel and Rajendran (2006)	Genetic algorithm	Min. overall costs	$(S-1,S)$ policy
O'donnell et al. (2006)	Genetic algorithm	Min. overall costs	$x+y$ policy
Chaharsooghi et al. (2008)	Reinforcement learning	Min. overall costs	State-dependent $(S-1,S)$ policy
Jiang and Sheng (2009)	Reinforcement learning	Max. service level	(s,Q) policy, (R,S) policy
Radhakrishnan et al. (2009)	Genetic algorithm	Min. overall costs	$(S-1,S)$ policy
Deshpande et al. (2011)	Bacterial foraging algorithm	Min. overall costs, min. weighted fulfilment time, max. service level	(R,S) policy
Duan and Liao (2013)	Differential evolution, harmony search	Min. overall costs	(s,S) policy
Güller et al. (2015)	Multi-objective particle swarm optimisation	Min. overall costs, max. service level	(s,Q) policy
Mortazavi et al. (2015)	Reinforcement learning	Min. overall costs	State-dependent $(S-1,S)$ policy
Devika et al. (2016)	Multi-objective electromagnetism mechanism algorithm, population-based multi-objective simulated annealing	Min. bullwhip effect, min. net stock amplification	Base-stock related policy based on Mosekilde et al. (1991)
Grahl et al. (2016)	Genetic algorithm	Min. overall costs	$(S-1,S)$ policy

Table 1 continued

Study	Method	Objective	Order policy
Keshari et al. (2018)	Bacterial foraging algorithm	Max. overall profit	Transportation mode specific $(S-1,S)$ policy
Priore et al. (2019)	Inductive learning based on C4.5	Min. bullwhip effect	Online and offline versions of base-stock related policies based on Mosekilde et al. (1991)
Badakhshan et al. (2020)	Genetic algorithm	Min. overall costs, min. bullwhip effect, min. cash flow bullwhip effect	Base-stock related policy based on Mosekilde et al. (1991)

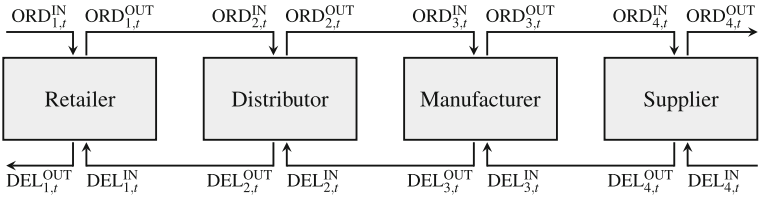

Fig. 1 Supply chain structure of the beer game

3 Problem description and modelling

We now explain in detail the supply chain under consideration and present an appropriate formal sequential decision-making model. The problem statement corresponds to those in Kimbrough et al. (2002) and Chaharsooghi et al. (2008). However, these studies lack full mathematical formulations of the supply chain, in particular of the flows of information and goods. Using the same problem statement as Kimbrough et al. (2002) and Chaharsooghi et al. (2008) enables us to compare our results (obtained by MCTS) with theirs (obtained by GA and RL, respectively).

3.1 Description of the supply chain

We consider a serial multi-echelon supply chain similar to the beer game with the following four echelons, who are also actors, indexed by $i = 1, \ldots, 4$: a retailer, a distributor, a manufacturer, and a supplier, cf. Fig. 1. In each period $t = 1, \ldots, T$, stochastic demand D_t by an exogenous customer arrives at the retailer, who, in turn, replenishes inventory by placing an order with the distributor. The distributor places an order with the manufacturer, who, in turn, places an order with the supplier. The latter obtains replenishments from an external source with unlimited supply. For notational simplicity we index the external source with $i = 5$ and the external customer with $i = 0$.

In each period, each actor decides whether and how much to order from the respective upstream actor. Any order that cannot be immediately satisfied is backlogged. Furthermore, in each period, each actor fulfils the sum of the newly received and backlogged orders if possible. Otherwise, the missing amount will be backlogged. Due to stochastic lead times, deliveries do not necessarily arrive immediately. We assume that both the lead times L_t and the demand D_t in each period t are independent and uniform distributed with the following probability mass functions:

$$P(L_t = l_t) = \begin{cases} \frac{1}{5}, & \text{if } l_t \in \{0; 1; \ldots; 4\} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$P(D_t = d_t) = \begin{cases} \frac{1}{16}, & \text{if } d_t \in \{0; 1; \ldots; 15\} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

According to the classical beer game (Sterman 1989), the event sequence at each echelon i in every period t is as follows (starting from the supplier $i = 4$ to the retailer $i = 1$):

1. Actor i receives new deliveries $\text{DEL}^{\text{IN}}_{i,t}$ (according to the respective lead times) from the upstream actor $i + 1$.
2. Actor i receives an order $\text{ORD}^{\text{IN}}_{i,t}$ from the downstream actor $i - 1$.

3. Actor i fulfils the received order plus backlogged orders, if possible. This outgoing delivery is denoted by $\mathrm{DEL}_{i,t}^{\mathrm{OUT}}$.
4. Actor i decides how much to order from the upstream actor $i+1$ to replenish stock. This order is denoted by $\mathrm{ORD}_{i,t}^{\mathrm{OUT}}$.

The information lead time is equal to one, i.e. orders placed by an actor arrive at the upstream actor with a delay of one period. Therefore,

$$\mathrm{ORD}_{i,t}^{\mathrm{OUT}} = \mathrm{ORD}_{i+1,t+1}^{\mathrm{IN}} \quad \forall i = 1, \ldots, 4 \tag{3}$$

holds in every period t. Note that the external customer's demand arrives immediately at the retailer. Therefore, realised demand d_t equals $\mathrm{ORD}_{1,t}^{\mathrm{IN}}$. In contrast to that, the arrival of outgoing deliveries depends on the lead times: a delivery sent in period t will arrive at the downstream actor in period $t + l_t$, i.e. with a delay of l_t periods. As a consequence, at the time of placing an order, it is uncertain when it will be fulfilled. Whether an order of actor $i-1$ can actually be completely fulfilled depends on the inventory level $\mathrm{INV}_{i,t}$ of actor i at the beginning of period t and on the incoming delivery $\mathrm{DEL}_{i,t}^{\mathrm{IN}}$. Let $[m]^+$ denote $\max\{0, m\}$ and $[m]^-$ denote $|\min\{m, 0\}|$, whereby both expressions are nonnegative numbers. Then, the outgoing delivery of actor i in period t is given by

$$\mathrm{DEL}_{i,t}^{\mathrm{OUT}} = \begin{cases} \mathrm{ORD}_{i,t}^{\mathrm{IN}}, & \text{in case 1} \\ [\mathrm{INV}_{i,t}]^+ + \mathrm{DEL}_{i,t}^{\mathrm{IN}}, & \text{in case 2} \\ [\mathrm{INV}_{i,t}]^- + \mathrm{ORD}_{i,t}^{\mathrm{IN}}, & \text{in case 3} \\ \mathrm{DEL}_{i,t}^{\mathrm{IN}}, & \text{in case 4} \end{cases} \tag{4}$$

where:

$$\begin{array}{lllll} \text{case 1:} & \mathrm{INV}_{i,t} \geqq 0 & \wedge & \mathrm{INV}_{i,t} + \mathrm{DEL}_{i,t}^{\mathrm{IN}} \geqq \mathrm{ORD}_{i,t}^{\mathrm{IN}} \\ \text{case 2:} & \mathrm{INV}_{i,t} \geqq 0 & \wedge & \mathrm{INV}_{i,t} + \mathrm{DEL}_{i,t}^{\mathrm{IN}} < \mathrm{ORD}_{i,t}^{\mathrm{IN}} \\ \text{case 3:} & \mathrm{INV}_{i,t} < 0 & \wedge & \mathrm{DEL}_{i,t}^{\mathrm{IN}} \geqq \mathrm{ORD}_{i,t}^{\mathrm{IN}} + [\mathrm{INV}_{i,t}]^- \\ \text{case 4:} & \mathrm{INV}_{i,t} < 0 & \wedge & \mathrm{DEL}_{i,t}^{\mathrm{IN}} < \mathrm{ORD}_{i,t}^{\mathrm{IN}} + [\mathrm{INV}_{i,t}]^-. \end{array} \tag{5}$$

Cases 1 and 2 summarise all scenarios with no backlogged orders, while in cases 3 and 4, there are backlogged orders. In case 1, the new order $\mathrm{ORD}_{i,t}^{\mathrm{IN}}$ can be satisfied immediately, but not in case 2. More precisely, in case 2, only the on-hand inventory $[\mathrm{INV}_{i,t}]^+$ of actor i and the delivery arriving at actor i can be sent to actor $i-1$, which is, however, less than $\mathrm{ORD}_{i,t}^{\mathrm{IN}}$. Case 3 comprises all scenarios in which the deliveries $\mathrm{DEL}_{i,t}^{\mathrm{IN}}$ arriving at actor i are large enough to satisfy both the new order $\mathrm{ORD}_{i,t}^{\mathrm{IN}}$ and the backlogged orders $[\mathrm{INV}_{i,t}]^-$, while in case 4 only the new deliveries $\mathrm{DEL}_{i,t}^{\mathrm{IN}}$ can be sent to the downstream actor $i-1$.

As lead times are stochastic and vary between zero and four, it is possible that two or more deliveries arrive simultaneously at the same actor. We even allow order crossing. Hence, the incoming deliveries $\mathrm{DEL}_{i,t}^{\mathrm{IN}}$ at actor i in period t are given by

$$\mathrm{DEL}_{i,t}^{\mathrm{IN}} = \sum_{j \in J} \mathrm{DEL}_{i+1,t-j}^{\mathrm{OUT}}, \tag{6}$$

where $J = \{j | j = l_{t-j}\}$. Regarding inventory costs, let $c_{i,+}$ denote the inventory holding costs of actor i per unit per period and $c_{i,-}$ the corresponding backorder costs. The common goal of the actors is to minimise the overall costs of the supply chain over the planning horizon T, i.e.

$$\sum_{t=1}^{T} \sum_{i=1}^{4} c_{i,+}[\mathrm{INV}_{i,t}]^+ + c_{i,-}[\mathrm{INV}_{i,t}]^-, \tag{7}$$

by coordinating their orders and, hence, by determining an order policy in an integrated manner. However, this requires a sequential decision-making model, such as a Markov Decision Process.

3.2 Markov decision processes

Formally, a Markov Decision Process (MDP) is a 4-tuple (S, A, p, r) where S is a set of states s describing the environment and A is a set of feasible actions or decisions. Furthermore, A_s denotes the subset of A comprising all actions that could possibly be chosen in state s. When action $a \in A_s$ is chosen at time t, the environment changes from state s at time t to state s' at time $t+1$ with the transition probability $p(s'|s,a)$. The last component $r(s,a,s')$ is the reward function, which returns the immediate reward. It is, in our case, the sum of inventory holding costs and backorder costs caused by taking action a in state s at time t and then transitioning to state s' at time $t+1$.

A decision rule δ_t is a mapping from states to actions, i.e. δ_t specifies the action $a \in A_s$ to be taken in state s at time t. Further, a policy $\pi = (\delta_1, \delta_2, \ldots, \delta_{T-1})$ is a sequence of such decision rules over the planning horizon T. Our goal is to minimise the overall supply chain inventory costs over the planning horizon by determining an appropriate order policy.

We now specify an MDP that is suitable for the supply chain structure outlined before. The state variable s representing the whole supply chain is given by the following (6×4) matrix:

$$\begin{bmatrix} \mathrm{INV}_{1,t} + \mathrm{DEL}^{\mathrm{IN}}_{1,t} - \mathrm{ORD}^{\mathrm{IN}}_{1,t} & \ldots & \mathrm{INV}_{4,t} + \mathrm{DEL}^{\mathrm{IN}}_{4,t} - \mathrm{ORD}^{\mathrm{IN}}_{4,t} \\ \mathrm{DEL}^{\mathrm{IN}}_{1,t+1} & \ldots & \mathrm{DEL}^{\mathrm{IN}}_{4,t+1} \\ \mathrm{DEL}^{\mathrm{IN}}_{1,t+2} & \ldots & \mathrm{DEL}^{\mathrm{IN}}_{4,t+2} \\ \mathrm{DEL}^{\mathrm{IN}}_{1,t+3} & \ldots & \mathrm{DEL}^{\mathrm{IN}}_{4,t+3} \\ \mathrm{DEL}^{\mathrm{IN}}_{1,t+4} & \ldots & \mathrm{DEL}^{\mathrm{IN}}_{4,t+4} \\ \mathrm{ORD}^{\mathrm{IN}}_{1,t} & \ldots & \mathrm{ORD}^{\mathrm{IN}}_{4,t} \end{bmatrix}.$$

Each column is assigned to one actor. The first row represents the inventory levels immediately before step 4 in the event sequence, i.e. before placing orders, while rows two to five contain the incoming deliveries in the next periods, i.e. the deliveries that are already in transit. Since the respective time indices might be confusing, some more explanation is needed in this regard: due to stochastic lead times, it is random at time t which deliveries will actually arrive in period $t+k$ (with $k \in \{1; \ldots; 4\}$). Therefore, at time t, $\mathrm{DEL}^{\mathrm{IN}}_{i,t+k}$ is not the sum of all deliveries that *will* arrive at time $t+k$, but the sum of all deliveries in transit, i.e. those that are known so far and hence *might* arrive at time $t+k$. The last row of s captures the incoming orders. These entries are only required for the determination of the action space A_s, see (9).

Although MCTS is able to cope with extremely large state spaces, large action spaces are often intractable. We therefore need a reduced action space instead of an action space that includes all possible action vectors. A means to achieve this goal is the so-called '$x+y$ rule', cf. Kimbrough et al. (2002). It states that each order placed by an actor i consists of the order x just received from actor $i-1$ plus an additional value y which has to be determined for each actor and time separately. Following our notation and adopting the $x+y$ rule, we define an order $\mathrm{ORD}^{\mathrm{OUT}}_{i,t}$ placed by actor i at time t as follows:

$$\mathrm{ORD}^{\mathrm{OUT}}_{i,t} = \mathrm{ORD}^{\mathrm{IN}}_{i,t} + y_{i,t}. \tag{8}$$

To reduce the action space, we restrict $y_{i,t}$ to $\{-3; -2; \ldots; 4; 5\}$. Several simulations have proven that in our model, this is a suitable range. Consequently, the set of available actions in state s at time t is given by:

$$A_s = \{a = [y_{1,t}; y_{2,t}; y_{3,t}; y_{4,t}] | y_{i,t} \in \{-3; -2; \ldots; 4; 5\} \wedge y_{i,t} \geq -\mathrm{ORD}_{i,t}^{\mathrm{IN}} \quad \forall i\}. \tag{9}$$

As orders have to be nonnegative, we further restrict $y_{i,t}$ to ensure the nonnegativity of (8). After action $a \in A_s$ has been chosen, the environment changes from state s to state s' with probability $p(s'|s, a)$. The subsequent state s' thus depends not only on the action a, but also on the probability function p and, hence, both on customer demand and lead times. It is straightforward to determine the subsequent state s'. $\mathrm{INV}_{i,t+1} = \mathrm{INV}_{i,t} + \mathrm{DEL}_{i,t}^{\mathrm{IN}} - \mathrm{ORD}_{i,t}^{\mathrm{IN}}$ denotes the inventory level of actor i at time $t + 1$, i.e. in state s'. For the incoming orders of actors $i \in \{2; 3; 4\}$ in state s', the following holds true: $\mathrm{ORD}_{i,t+1}^{\mathrm{IN}} = \mathrm{ORD}_{i-1,t}^{\mathrm{IN}} + y_{i-1,t}$. As already mentioned, the external customer demand arrives immediately at the retailer, i.e. $\mathrm{ORD}_{1,t+1}^{\mathrm{IN}} = d_{t+1}$, while the orders of actors $1, \ldots, 4$ need one period to arrive at the respective upstream actor. The incoming deliveries in state s' and the in-transit shipments already known up to this state, i.e. rows two to five in the state matrix s', can be determined by Eq. (6). Finally, the last row of s' is given by the action a taken in state s. This transition causes the following immediate reward:

$$r(s, a, s') = \sum_{i=1}^{4} c_{i,+}[\mathrm{INV}_{i,t+1}]^{+} + c_{i,-}[\mathrm{INV}_{i,t+1}]^{-}. \tag{10}$$

Consequently, the expected reward is given by:

$$\sum_{s' \in S} r(s, a, s') p(s'|s, a). \tag{11}$$

Note that in our case, 'reward' actually means 'costs' (comprising inventory holding costs and backorder costs). For this reason, we strive to find a policy that minimises the sum of the expected rewards over the planning horizon. From a theoretical point of view (supposing unlimited memory and time), it is conceivable to exactly determine an optimal policy in a finite horizon MDP using backward induction (Puterman 2014). However, the large action space and the extremely large state space render backward induction infeasible in the case at hand. Therefore, an MCTS-based approach is chosen to solve our MDP heuristically.

4 Monte Carlo tree search

Over the last couple of years, MCTS has achieved widespread use in AI, particularly in sequential decision problems that can be represented with tree structures (Browne et al. 2012). Regarding the categorization of AI-based methods (cf. Sect. 2), MCTS is related to RL, as they have various similarities, such as observing rewards, memorising feedback, finding a good policy, and assigning values to actions (Vodopivec et al. 2017). The most popular area of application of MCTS so far is that of games, such as Othello (Robles et al. 2011) or Go (Gelly and Silver 2011; Gelly et al. 2012; Silver et al. 2016), the latter of which made MCTS widely accepted in the AI-community (Browne et al. 2012). This is due to the fact that many games can be represented as sequential decision problems. Moreover, especially in games, AI-based algorithms can be easily compared against human intelligence to evaluate their performance.

Although the application of MCTS to general planning problems, such as MDPs, was suggested early on (Kocsis and Szepesvári 2006), such applications are still far less common in the operations research literature than are the applications of MCTS focusing on solving games in the computer science literature. This observation was recently confirmed by Jiang et al. (2020), who propose an optimistic tree search called primal-dual MCTS and apply it to optimise a ride-sharing network. In particular, they stress that the operations research community has not taken advantage of MCTS so far, except for a small number of studies. Bertsimas et al. (2017) apply MCTS to two large-scale dynamic resource allocation problems dealing with tactical wildfire management and queuing network control. An MCTS approach to find efficient patrolling schemes on graphs is proposed by Karwowski and Mańdziuk (2019). Neto et al. (2020) use MCTS for a harvest scheduling problem. In the following, we give a short introduction to MCTS and then apply it to our supply chain planning problem. For more in-depth information and a comprehensive overview of MCTS, we refer to Browne et al. (2012).

4.1 Basic idea of Monte Carlo tree search

MCTS is a best-first search method that approximates the value $Q(s, a)$ of an action a in a state s by a randomised exploration of the action space. It incrementally constructs a partial search tree based on the outcome of the preceding exploration. In turn, the search tree is used to estimate the values of actions, with estimates (in particular of the most auspicious actions) becoming more precise as the tree grows in size.

The basic version of MCTS rests on four steps (cf. Fig. 2) that are carried out consecutively in each iteration (Browne et al. 2012):

1. Selection: Starting from the root node and using a recursively applied child selection policy, the search tree is traversed until an expandable node is reached. A node is said to be expandable if it is a non-terminal node and has unvisited (i.e. unexpanded) child nodes.
2. Expansion: One or more child nodes are added to the search tree according to some expansion policy.
3. Simulation: Starting from the expanded node(s), a so-called playout (or rollout) is run which consists of a simulation until a terminal node is reached. This simulation generates a value estimate.
4. Backpropagation: The estimated value is propagated backwards from the last selected node to the root node of the search tree.

The term ‘tree policy’ is often used for the combination of child selection and expansion policy, while the term ‘default policy’ refers to the criteria for executing the rollout. The nodes of the search tree are equivalent to the states in our MDP. That is why we use the term ‘state’ instead of ‘node’ in the following.

Constructing a search tree of depth T, from the initial state (at time $t = 1$) to a terminal state (at time $t = T$), capturing the whole length of the planning horizon, is too time consuming and requires too much memory. We therefore implement a rolling horizon procedure instead (Powell 2007) which extends the tree up to length $h < T$. MCTS then only generates an (order) policy for the horizon from t to h. According to this order policy, the best action in the current state is executed, i.e. δ_t. Depending on this action as well as on the realizations d_{t+1} and l_{t+1} of the random variables D_{t+1} and L_{t+1}, the system transitions to the next state. In this state, MCTS is applied again for the subsequent horizon from $t + 1$ to $t + 1 + h$, i.e.

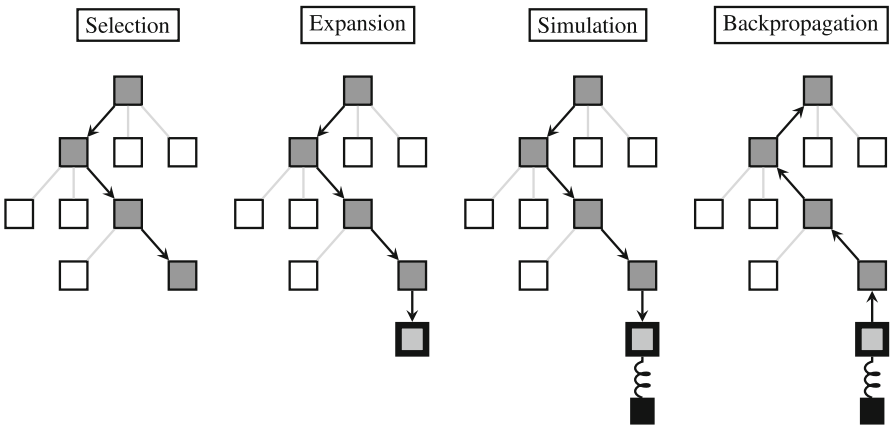

Fig. 2 Sequence of a Monte Carlo tree search

after incrementing t. We repeat this procedure until the last component δ_{T-1} of the order policy is reached. Next, we provide a pseudocode description of our MCTS implementation.

4.2 Application to supply chain inventory management

Starting from the current state s of the supply chain environment, numerous simulations are executed to a depth of h in order to update the estimated state-action value function $Q(s, a)$. This process (cf. the pseudocode description of Algorithm) is implemented recursively and comprises the four steps already mentioned: selection, expansion, simulation, and backpropagation, which are explained in more detail below. The sequence of MCTS is repeated until a stopping criterion, e.g. a maximum number of simulations (i.e. number of playouts) or a runtime limit, is met. After that, the action that minimises $Q(s, a)$ is executed (as $Q(s, a)$ represents costs). We repeat this algorithm to obtain the next action in the following state.

Selection: If the current state s is in the set of expandable states M_s, we choose an action that minimises a selection criterion. Otherwise, we move on to the expansion step. We will discuss the selection criterion in more detail below. To begin with, note that if one or more actions have never been selected in state s, one of these actions needs to be chosen. To this end, if the set K_s of all actions previously selected in state s is smaller than the set A_s of all possible actions in state s, a new action is added to K_s according to GetNextAction(s) and (as will be justified below) a large negative initial value is assigned to this action. GetNextAction(s) samples and returns a valid action $a \in A_s$ that has never been selected before in state s.

One of the most widely-used selection criteria stems from the the Upper Confidence Bounds for Trees (UCT) algorithm (cf. Kocsis and Szepesvári 2006). In state s, it selects an action a which (in our case) minimises

$$Q(s, a) - C\sqrt{\frac{\log N(s)}{N(s, a)}}, \tag{12}$$

where $C > 0$ is an exploration constant, $N(s, a)$ is the total number of simulations in which action a was selected in state s, and $N(s)$ is the total number of simulations run from state s. If an action a has never been selected before in state s, i.e. $N(s, a) = 0$, a large negative value (instead of the UCT value (12)) is assigned to this action. This ensures that all actions

chosen in state s are considered at least once before any subsequent state is expanded further. As soon as an action has been selected, state s transitions to state s' according to a sample of a generative model G. This model comprises the stochastic components (i.e. customer demand and lead times) of our supply chain. The selection step is applied recursively until a state is reached that has not yet been expanded.

Expansion: When an unexpanded state s has been reached, this state is added to the set of expandable states M_s. Then, we proceed to the simulation step.

Simulation: Starting from an unexpanded state, a simulation is run to traverse the tree recursively until depth h is reached. The simulation is executed according to a rollout policy π_{rollout} which either is completely random or contains domain-specific knowledge. The latter alternative offers the possibility of focusing the search on promising branches. However, there is also the risk of not exploring the search space sufficiently. In our model, we use a random rollout policy which samples the subsequent state s'. This allows us to calculate the costs $r(s, a, s')$ as well the sum q of these costs on the path from the state at time t to the state at time $t + h$.

Backpropagation: Finally, $Q(s, a)$ is updated from the state newly added in the expansion step back to the root state at time t by backpropagating q. In order to avoid considering all previous simulations, we formulate this as a stepwise update by incrementing $N(s, a)$, and thereby also $N(s)$, and afterwards updating $Q(s, a)$ with respect to q and $N(s, a)$.

5 Experimental results

In this section, we describe the initial state of the beer game, i.e. the initial inventory levels as well as the flows of information and goods. We then investigate how MCTS-based policies perform compared to policies generated by GA and RL. Finally, we consider whether MCTS-based policies are able to counteract suboptimal initial inventory levels as well as the bullwhip effect in the long run.

5.1 Initial setting

According to the original rules of the beer game, the inventory levels are initially equivalent to the on-hand inventories and equal 12 units per actor (Sterman 1989). Each actor has already (at time $t = 0$) placed a 'pre-order' of 4 units with the respective upstream actor before actually placing the first 'real' order (at time $t = 1$). Due to the information lead time, these pre-orders are received in the first period. Then, the corresponding quantities are placed in transit and will arrive one period later (at time $t = 2$) by assumption. Additionally, 4 units are in transit to each actor and arrive at time $t = 1$. Hence,

1. $\text{INV}_{i,1} = 12 \quad \forall i = 1, \ldots, 4$,
2. $\text{ORD}^{\text{OUT}}_{i,0} = 4 \quad \forall i = 1, \ldots, 4$,
3. $\text{DEL}^{\text{IN}}_{i,1} = 4 \quad \forall i = 1, \ldots, 4$

results in the initial state:

$$\begin{bmatrix} 12+4-d_1 & 12+4-4 & 12+4-4 & 12+4-4 \\ 4 & 4 & 4 & 4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ d_1 & 4 & 4 & 4 \end{bmatrix}.$$

Algorithm 1 Monte Carlo tree search

procedure MCTS(s,h)
  repeat
    SELECT(s,h)
  until stopping criterion is met
  return $\arg\min_{a\in A_s} Q(s,a)$
end procedure

procedure SELECT(s,h)
  if $h=0$ **then**
    return 0
  end if
  if $s\notin M_s$ **then**
    return EXPAND(s,h)
  end if
  if $|K_s|<|A_s|$ **then**
    $a\leftarrow$ GETNEXTACTION(s)
    $K_s\leftarrow K_s\cup\{a\}$
  end if
  $a\leftarrow \arg\min_{a\in A_s}\left\{Q(s,a)-C\sqrt{\frac{\log N(s)}{N(s,a)}}\right\}$
  $s'\sim G(s,a)$
  $q\leftarrow r(s,a,s')+$ SELECT($s',h-1$)
  BACKPROPAGATE(s,a)
  return q
end procedure

procedure EXPAND(s,h)
  $M_s\leftarrow M_s\cup\{s\}$
  return SIMULATE(s,h)
end procedure

procedure SIMULATE(s,h)
  if $h=0$ **then**
    return 0
  end if
  $a\sim\pi_{\text{rollout}}(s)$
  $s'\sim G(s,a)$
  return $r(s,a,s')+$ SIMULATE($s',h-1$)
end procedure

procedure BACKPROPAGATE(s,a)
  $N(s)\leftarrow N(s)+1$
  $N(s,a)\leftarrow N(s,a)+1$
  $Q(s,a)\leftarrow Q(s,a)+\frac{q-Q(s,a)}{N(s,a)}$
end procedure

Note that at the time of placing the first orders, customer demand d_1 is known at the retailer echelon, cf. the event sequence in Sect. 3. To ensure the comparability of our results with those of Kimbrough et al. (2002) and Chaharsooghi et al. (2008), we make the same assumptions $c_{i,+} = 1$ and $c_{i,-} = 2$ regarding the cost parameters. Furthermore, we set $C = \sqrt{10^{-3}}$ and $h = 10$ as this parametrization has proven to be suitable in our case within numerous simulations.

5.2 Comparison with other AI techniques

We compare the results generated by our MCTS approach with those of Kimbrough et al. (2002) and Chaharsooghi et al. (2008). All studies employ the same initial setting as described previously. For the comparison, we first applied MCTS to the four test problems $TP_1, \ldots, TP_4$ by Chaharsooghi et al. (2008). Table 2 provides the associated realizations of customer demand and lead times. Note that these lead times refer to 'real' orders (not pre-orders), i.e. orders placed in transit beginning with period $t = 2$. Since we believe that having only four test problems is insufficient to capture the diversity of possible scenarios, we extended the number of test problems to 100 after evaluating $TP_1, \ldots, TP_4$. In all test problems, the demand was randomly drawn from a (discrete) uniform distribution spanning from 0 to 15, cf. (2). Analogously, the lead times were drawn from a (discrete) uniform distribution spanning from 0 to 4, cf. (1).

The training of our MCTS model, implemented in Python, was run on an Intel Core i7 quad-core CPU with 3.40 gigahertz, 16.0 gigabytes of RAM, and Windows 10 x64 operating system. The runtime for each component δ_t of the order policy $\pi = (\delta_1, \delta_2, \ldots, \delta_{T-1})$ was 5 minutes. Table 3 reports the overall inventory costs associated with the order policies generated by MCTS, GA, and RL, where the costs related to the latter two are quoted from Kimbrough et al. (2002) and Chaharsooghi et al. (2008).

Table 2 Realizations of customer demand and lead times in the test problems, cf. Chaharsooghi et al. (2008)

	Customer demand (35 periods)
TP_1	15,10,8,14,9,3,13,2,13,11,3,4,6,11,15,12,15,4,12,3,13,10,15,15,3,11,1,13,10,10,0,0,8,0,14
TP_2	5,14,14,13,2,9,5,9,14,14,12,7,5,1,13,3,12,4,0,15,11,10,6,0,6,6,5,11,8,4,4,12,13,8,12
TP_3	15,10,8,14,9,3,13,2,13,11,3,4,6,11,15,12,15,4,12,3,13,10,15,15,3,11,1,13,10,10,0,0,8,0,14
TP_4	13,13,12,10,14,13,13,10,2,12,11,9,11,3,7,6,12,12,3,10,3,9,4,15,12,7,15,5,1,15,11,9,14,0,4
	Lead times (35 periods)
TP_1	2,0,2,4,4,4,0,2,4,1,1,0,0,1,1,0,1,1,2,1,1,1,4,2,2,1,4,3,4,1,4,0,3,3,4
TP_2	2,0,2,4,4,4,0,2,4,1,1,0,0,1,1,0,1,1,2,1,1,1,4,2,2,1,4,3,4,1,4,0,3,3,4
TP_3	4,2,2,0,2,2,1,1,3,0,0,3,3,3,4,1,1,1,3,0,4,2,3,4,1,3,3,3,0,3,4,3,3,0,3
TP_4	4,2,2,0,2,2,1,1,3,0,0,3,3,3,4,1,1,1,3,0,4,2,3,4,1,3,3,3,0,3,4,3,3,0,3

Table 3 Inventory costs generated by the different approaches

	TP_1	TP_2	TP_3	TP_4
GA	2555	3109	4156	4330
RL	2417	3169	4038	4205
MCTS	2115	1716	1962	2034
$MCTS_{offline}$	2162	1863	2665	2486

MCTS outperforms all other approaches in each of the test problems $TP_1, \ldots, TP_4$. This is not particularly surprising, as our MCTS approach adjusts the order policy according to all available information in every period, while the GA and RL approaches determine only one-time order policies. Consequently, comparisons between this MCTS approach and the GA or RL approaches might be deceptive. In order to address this issue, we also include an 'offline' version $MCTS_{offline}$ in the comparison. $MCTS_{offline}$ indeed follows an 'online' (i.e. rolling horizon) procedure. However, the initial state in each period is completely random: lead times and costumer demand are drawn from (1) and (2). Therefore, $MCTS_{offline}$ has no informational advantages over GA and RL. Nevertheless, it generates smaller inventory costs, cf. Table 3.

In the 100 additional test problems, both $MCTS_{offline}$ and MCTS achieve far better results than the other methods: $MCTS_{offline}$ generates lower costs than GA and RL in 86 out of 100 cases, while MCTS generates the lowest costs in all 100 cases. Figure 3 shows a box whisker plot of the overall inventory costs generated by the four approaches.

5.3 Long-term behaviour of MCTS

If inventory levels in the initial state are unfavourable, i.e. if substantial shortages or overstocks occur, compensating for them requires a certain period of time. Consequently, a sound order policy should be able to overcome such suboptimal inventory levels at all echelons, at least in the long run. Since all random distributions are stationary in our model, MCTS-based policies are supposed to meet this requirement.

To investigate the long-term behaviour of MCTS, we extended the planning horizon from 35 to 300 periods and assumed an initial state of major shortage:

1. $\mathrm{INV}_{i,1} = -50 \quad \forall i = 1, \ldots, 4\,,$
2. $\mathrm{ORD}^{\mathrm{OUT}}_{i,0} = 10 \quad \forall i = 1, \ldots, 4\,,$
3. $\mathrm{DEL}^{\mathrm{IN}}_{i,1} = 0 \quad \forall i = 1, \ldots, 4\,.$

All other parameters, distributions, and training conditions remained unchanged. Figure 4 depicts the resulting paths of all individual inventory levels. It shows that MCTS is able to stabilise the inventory levels at each echelon (at least as of approximately $t = 50$). The most time is needed in the case of the retailer. This is due to the retailer's being the echelon in

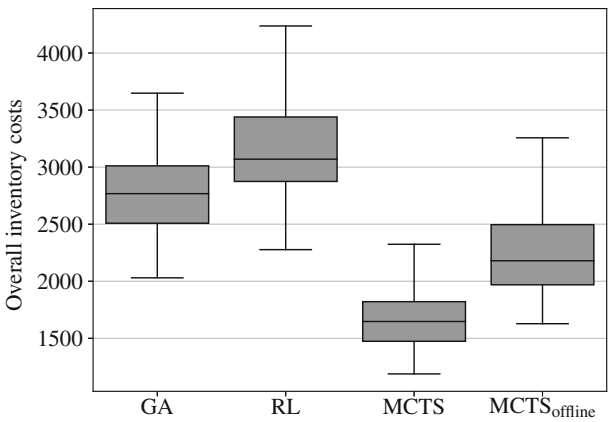

Fig. 3 Box whisker plot of the overall inventory costs generated by the different approaches

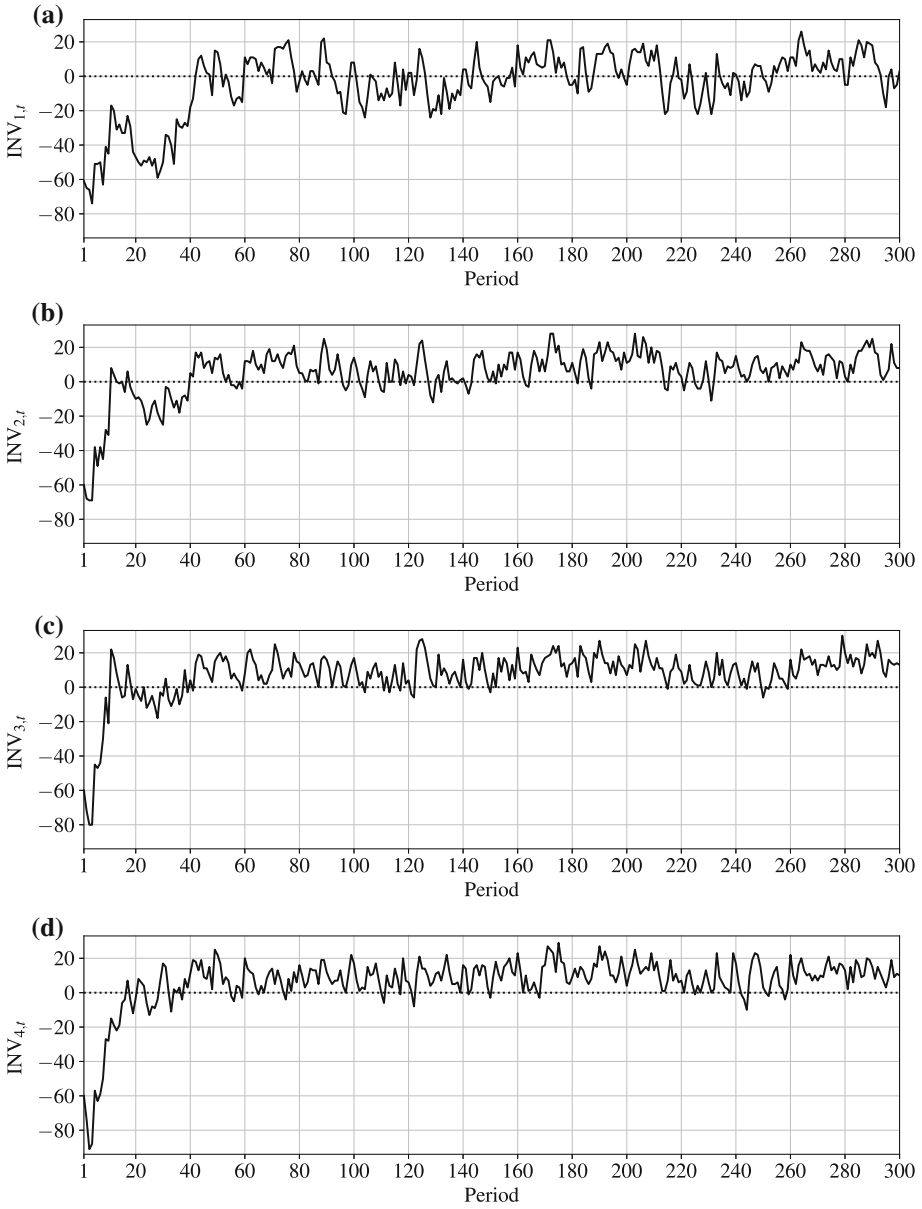

Fig. 4 Paths of the inventory levels of **a** the retailer, **b** the distributor, **c** the manufacturer, **d** the supplier

the supply chain at the largest distance from the origin of the goods. Additionally, note that inventory level fluctuations are substantially larger on the retailer side than on the supplier side. This effect is a consequence of the order policy determined by MCTS, in which the retailer reduces the volatility of customer demand.

This reduction in volatility is particularly noticeable when examining the frequencies of the various incoming order quantities $\mathrm{ORD}_{i,t}^{\mathrm{IN}}$ at echelons $i = 1, \ldots, 4$. Figure 5 shows

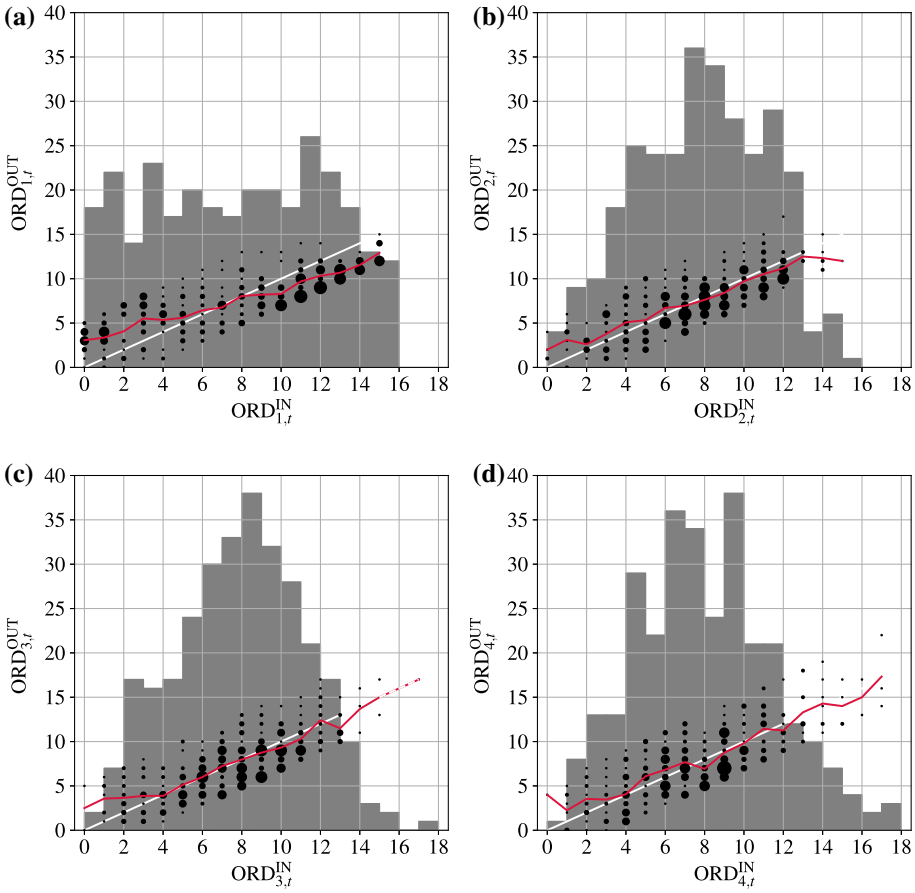

Fig. 5 Frequencies of incoming orders and comparison of incoming and outgoing orders for **a** the retailer, **b** the distributor, **c** the manufacturer, **d** the supplier

the histograms of these incoming orders in grey. Since customer demand is drawn from the stationary uniform distribution (2), all possible values $0, \ldots, 15$ have almost the same frequencies, cf. Fig. 5a. The red lines represent the average order quantities placed by each actor depending on the received orders. For example, if the external customer places no order with the retailer, the retailer, in turn, orders 3.06 units from the distributor on average. For purposes of comparison, the white lines indicate where $\mathrm{ORD}^{\mathrm{IN}}_{i,t}$ equals $\mathrm{ORD}^{\mathrm{OUT}}_{i,t}$. The black dots depict the frequencies of individual orders: the thicker the dot, the more often this order was placed. Remarkably, MCTS generates an order policy where the retailer tends to place larger orders than incoming if customer demand is low and tends to place smaller orders than incoming when customer demand is high. This reduces the volatility of demand for the upstream actors, which, in particular, results in the bell-shaped histograms in Fig. 5b–d.

In addition to the graphical indication of this phenomenon, we quantify it by measuring the bullwhip effect of the whole supply chain. This effect is captured by the coefficient of variation of $\mathrm{ORD}^{\mathrm{OUT}}_{4,t}$ divided by the coefficient of variation of customer demand. A quotient greater than one indicates the presence of the bullwhip effect (cf. Fransoo and Wouters 2000). In our scenario, with the extended planning horizon of 300 periods, there results a

quotient of 0.49 / 0.61 = 0.80. The MCTS approach thus eliminates the bullwhip effect in our supply chain. This is a remarkable finding, as our objective function only aims at the goal of minimising supply chain costs, but not eliminating the bullwhip effect. Nevertheless, the MCTS approach determined an order policy that achieves both goals.

6 Discussion

A sound order policy is crucial, in particular, in complex stochastic environments where no optimal policy is known. This requires modelling the state space as precisely as possible. In this regard, the curse of dimensionality is unfortunately a limiting factor. Our results reveal that MCTS is an adequate method to tackle this problem. It significantly reduces costs, eliminates the bullwhip effect, and is able to compensate for initial shortages or overstocks. In the following, we highlight the main differences between MCTS and the prevalent approaches and provide reasons why MCTS is superior.

The vast majority of AI approaches in supply chain inventory management are used to determine one-shot order policies at the beginning of the planning horizon (cf. Sect. 2). Over the course of time this policy continues to be executed without adjustments. Changes, for example, in the distribution of customer demand (i.e. in a non-stationary environment) therefore cause problems under such a policy. However, even in a stationary environment, several consecutive periods with low customer demand may occur. If the retailer and all other upstream actors follow a one-shot policy, e.g. a base-stock policy, such low demands progressively propagate throughout the supply chain. As a result, the in-transit shipments in the whole supply chain decrease. If, subsequently, some periods with high customer demand in combination with unfavourable major delays in delivery follow, it will take more time to entirely satisfy this customer demand. By periodically adjusting the order policy to the current state of the supply chain, MCTS reacts to such situations more dynamically, resulting in substantially lower costs. In combination with the rolling horizon procedure, MCTS proactively considers the possibility of reaching such disadvantageous states. Moreover, MCTS counteracts reaching those states by selecting appropriate actions, i.e. actions with a high probability of reaching a state associated with low costs.

In contrast to one-shot policy approaches, most RL approaches use state-based models (Giannoccaro and Pontrandolfo 2002; Chaharsooghi et al. 2008; Jiang and Sheng 2009; Mortazavi et al. 2015). Although these models are also trained at the beginning of the planning horizon, order policies that can be regarded as state-dependent base-stock policies result in most cases. Nevertheless, the associated state-dependent base-stock levels are not explicitly determined. Instead, these levels result from the state-action value function $Q(s, a)$, i.e. from the best action a in state s. Similarly, the order policies identified by MCTS can also be considered as state-dependent base-stock policies. However, unlike previous RL approaches to inventory management, our approach offers the major advantage that it is not necessary to estimate the state-action value function for each of the almost innumerable states. Two distinct features are responsible for the advantages of MCTS. First, MCTS builds an asymmetric decision tree by focussing on promising states and thus ignoring a large number of unfavourable states. Second, due to the periodic incorporation of information within the rolling horizon procedure, the system reveals the actual state. Consequently, there is no need to aggregate states as in previous approaches, (cf. Giannoccaro and Pontrandolfo 2002; Chaharsooghi et al. 2008; Mortazavi et al. 2015, which allows modelling the supply chain's state space more precisely.

The dynamic order policies generated by MCTS involve considerably lower costs than those of previous approaches, cf. Table 3. This is in line with Priore et al. (2019) who, however, pursue a completely different online approach, cf. Sect. 2. At first sight, the MCTS order policy at the retailer echelon is not entirely consistent with human intuition (Croson and Donohue 2006; Nienhaus et al. 2006) as MCTS suggests ordering slightly more on average if customer demand is low and slightly less if customer demand is high, cf. Fig. 5a. Although this behaviour increases the volatility of the retailer's inventory level, cf. Fig. 4a, it decreases the volatility of the signal the retailer receives from the customer and sends upstream throughout the whole supply chain. From a holistic perspective, this is an advantage for the upstream supply chain as it helps to achieve lower overall costs as well as to eliminate the bullwhip effect. Recall that eliminating the bullwhip effect was not a part of our objective function, but a feature of the solution generated by MCTS. Further note that the retailer's behaviour is part of a state-dependent policy where all actors operate in a coordinated way. In contrast, Croson and Donohue (2006) report that the bullwhip effect occurs in the beer game with human actors. This holds true even if the scenario is less complex, the actors share their information, and aim to minimise collaboratively the overall supply chain costs (rather than the costs at their own echelons).

Implications for research

Our results suggest that AI-based dynamic order policies (such as those identified by MCTS) are promising alternatives to the prevalent one-shot order policies or those suffering from the curse of dimensionality. In addition, they provide interesting insights into complex environments and reveal new advantageous patterns of behaviour. Further extensions of the model, e.g. including non-stationary random distributions or multi-objective optimisation, can be implemented without difficulty. We hope to encourage other researchers to develop novel dynamic approaches that allow improvements compared to the static one-shot order policies often used so far. Furthermore, we would be glad if our work draws attention to the potential of MCTS in supply chain inventory management and inspires researchers in other disciplines to apply MCTS, leading to a wider dissemination of this method within the operations research community.

Implications for practice

Managers and other practitioners benefit from applying the MCTS approach not only in terms of measurable lower overall costs, but also by preventing the undesirable bullwhip effect. Furthermore, it helps to create managerial awareness of the benefits associated with reconsidering decisions and incorporating new information periodically. Even if not generated by MCTS, dynamic policies like the one suggested in our paper offer additional possibilities apart from minimising costs, e.g. adjusting order policies to reduce the volatility of orders within the supply chain and thus eliminating the bullwhip effect. The order policies common in practice do not offer such possibilities.

Moreover, MCTS can support the improvement of managerial decision skills. E.g., in a human learning environment that emulates simplified real world problems, the managers' order decisions can be evaluated by MCTS. This evaluation may employ information regarding the costs or the service levels achieved by the actual managerial decision and other possible decision alternatives. Such feedback allows managers to identify the best decision

in each situation. When transferred to real world problems, the knowledge gained in such training adds value for business practice.

7 Conclusion

This study analyses a multi-echelon supply chain consisting of four echelons, under stochastic demand and lead times. The aim is to minimise the overall inventory costs by implementing a policy that tackles the trade-off between the costs of inventory holding and of backorders. To do so, we formulate the supply chain environment as a large-scale MDP. The resulting complexity renders analytical solutions infeasible. That is why we apply a heuristic approach based on MCTS instead. In this regard, we examine both an online version which determines the order policy using real-time data, and an offline version in which this information is not available.

This paper makes the following contributions: First, we provide, to the best of our knowledge, the first MCTS-based model for supply chain inventory management. Second, this model can handle extremely large state spaces and thus allows us to capture the supply chain dynamics more precisely than previous studies. Third, the order policies derived within our framework are able to eliminate the bullwhip effect as well as to compensate for suboptimal initial inventory levels (such as pronounced shortages or overstocks).

The results demonstrate that MCTS is a promising approach to substantially reduce costs in multi-echelon supply chains. Unsurprisingly, the online version performs better than the offline version. Nevertheless, even the latter achieves superior results compared to previous studies. Furthermore, the policy determined by MCTS induces stable inventory levels and prevents the bullwhip effect.

Unfortunately, it is challenging to treat continuous order quantities by using our approach, as the basic version of MCTS cannot handle larger, especially infinite, action spaces. Attempts to address this issue can be found in the literature (cf. Couëtoux et al. 2011). However, they do not work in our setting, since our supply chain consists of multiple actors, vector-valued actions are needed, whereas those attempts only take into account scalar-valued actions. Discretising might be an option to cope with continuous order quantities.

As this study is the first application of MCTS to inventory management, it opens a wide field for further extensions and investigations. Future studies could address the application of MCTS in imperfect information supply chain environments in which the information about individual inventory levels is not shared and each actor decentrally decides on the own order quantity. Such problems could be modelled as Partially Observable Markov Decision Processes, for which MCTS has proven to be an adequate method of solution (Silver and Veness 2010). Further investigations could extend the application of MCTS to non-stationary or multi-objective inventory problems. In addition, allowing actors with different risk attitudes promises exciting new insights regarding the behavioural impacts of risk-balancing order polices generated by MCTS. From a game theoretic perspective, it is of high interest how MCTS-enabled cost savings can be allocated in a ‘fair’ way among the individual actors of the supply chain. Apart from inventory management, plenty other branches of operations research can benefit from innovative applications of MCTS. This applies in particular to problems involving complex sequential decisions. We are therefore confident to encounter more widespread use of MCTS in future research.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Axsäter, S. (2015). *Inventory control* (3rd ed.). Berlin: Springer.

Badakhshan, E., Humphreys, P., Maguire, L., & McIvor, R. (2020). Using simulation-based system dynamics and genetic algorithms to reduce the cash flow bullwhip in the supply chain. *International Journal of Production Research*, *58*(17), 5253–5279.

Bertsimas, D., Griffith, J. D., Gupta, V., Kochenderfer, M. J., & Mišić, V. V. (2017). A comparison of Monte Carlo tree search and rolling horizon optimization for large-scale dynamic resource allocation problems. *European Journal of Operational Research*, *263*(2), 664–678.

Bhattacharya, R., & Bandyopadhyay, S. (2011). A review of the causes of bullwhip effect in a supply chain. *The International Journal of Advanced Manufacturing Technology*, *54*(9–12), 1245–1261.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., et al. (2012). A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, *4*(1), 1–43.

Carbonneau, R., Laframboise, K., & Vahidov, R. (2008). Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, *184*(3), 1140–1154.

Chaharsooghi, S. K., Heydari, J., & Zegordi, S. H. (2008). A reinforcement learning model for supply chain ordering management: An application to the beer game. *Decision Support Systems*, *45*(4), 949–959.

Chai, J., Liu, J. N., & Ngai, E. W. (2013). Application of decision-making techniques in supplier selection: A systematic review of literature. *Expert Systems with Applications*, *40*(10), 3872–3885.

Chatfield, D. C., Kim, J. G., Harrison, T. P., & Hayya, J. C. (2004). The bullwhip effect–impact of stochastic lead time, information quality, and information sharing: A simulation study. *Production and Operations Management*, *13*(4), 340–353.

Chawla, A., Singh, A., Lamba, A., Gangwani, N., & Soni, U. (2019). Demand forecasting using artificial neural networks—a case study of American retail corporation. *Applications of artificial intelligence techniques in engineering* (pp. 79–89). Berlin: Springer.

Couëtoux, A., Hoock, J. B., Sokolovska, N., Teytaud, O., & Bonnard, N. (2011). Continuous upper confidence trees. *International conference on learning and intelligent optimization* (pp. 433–445). Berlin: Springer.

Croson, R., & Donohue, K. (2006). Behavioral causes of the bullwhip effect and the observed value of inventory information. *Management Science*, *52*(3), 323–336.

Daniel, J. S. R., & Rajendran, C. (2005). A simulation-based genetic algorithm for inventory optimization in a serial supply chain. *International Transactions in Operational Research*, *12*(1), 101–127.

Daniel, J. S. R., & Rajendran, C. (2006). Heuristic approaches to determine base-stock levels in a serial supply chain with a single objective and with multiple objectives. *European Journal of Operational Research*, *175*(1), 566–592.

De Jong, K. A. (2006). *Evolutionary computation: A unified approach*. Cambridge, MA: MIT Press.

Deshpande, P., Shukla, D., & Tiwari, M. (2011). Fuzzy goal programming for inventory management: A bacterial foraging approach. *European Journal of Operational Research*, *212*(2), 325–336.

Devika, K., Jafarian, A., Hassanzadeh, A., & Khodaverdi, R. (2016). Optimizing of bullwhip effect and net stock amplification in three-echelon supply chains using evolutionary multi-objective metaheuristics. *Annals of Operations Research*, *242*(2), 457–487.

Duan, Q., & Liao, T. W. (2013). Optimization of replenishment policies for decentralized and centralized capacitated supply chains under various demands. *International Journal of Production Economics*, *142*(1), 194–204.

Ellram, L. M. (1991). Supply-chain management: The industrial organisation perspective. *International Journal of Physical Distribution and Logistics Management*, *21*(1), 13–22.

Floreano, D., & Mattiussi, C. (2008). *Bio-inspired artificial intelligence: Theories, methods, and technologies*. Cambridge, MA: MIT Press.

Fransoo, J. C., & Wouters, M. J. (2000). Measuring the bullwhip effect in the supply chain. *Supply Chain Management: An International Journal*, *5*(2), 78–89.
Gelly, S., Kocsis, L., Schoenauer, M., Sebag, M., Silver, D., Szepesvári, C., et al. (2012). The grand challenge of computer Go: Monte Carlo tree search and extensions. *Communications of the ACM*, *55*(3), 106–113.
Gelly, S., & Silver, D. (2011). Monte-Carlo tree search and rapid action value estimation in computer Go. *Artificial Intelligence*, *175*(11), 1856–1875.
Giannoccaro, I., & Pontrandolfo, P. (2002). Inventory management in supply chains: A reinforcement learning approach. *International Journal of Production Economics*, *78*(2), 153–161.
Giannoccaro, I., Pontrandolfo, P., & Scozzi, B. (2003). A fuzzy echelon approach for inventory management in supply chains. *European Journal of Operational Research*, *149*(1), 185–196.
Govindan, K. (2016). Evolutionary algorithms for supply chain management. *Annals of Operations Research*, *242*(2), 195–206.
Govindan, K., Jafarian, A., & Nourbakhsh, V. (2019). Designing a sustainable supply chain network integrated with vehicle routing: A comparison of hybrid swarm intelligence metaheuristics. *Computers and Operations Research*, *110*, 220–235.
Grahl, J., Minner, S., & Dittmar, D. (2016). Meta-heuristics for placing strategic safety stock in multi-echelon inventory with differentiated service times. *Annals of Operations Research*, *242*(2), 489–504.
Güller, M., Uygun, Y., & Noche, B. (2015). Simulation-based optimization for a capacitated multi-echelon production-inventory system. *Journal of Simulation*, *9*(4), 325–336.
Haenlein, M., & Kaplan, A. (2019). A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California Management Review*, *61*(4), 5–14.
Jaipuria, S., & Mahapatra, S. (2014). An improved demand forecasting method to reduce bullwhip effect in supply chains. *Expert Systems with Applications*, *41*(5), 2395–2408.
Jiang, C., & Sheng, Z. (2009). Case-based reinforcement learning for dynamic inventory control in a multi-agent supply-chain system. *Expert Systems with Applications*, *36*(3), 6520–6526.
Jiang, D. R., Al-Kanj, L., & Powell, W. B. (2020). Optimistic Monte Carlo tree search with sampled information relaxation dual bounds. *Operations Research* (Forthcoming).
Karwowski, J., & Mańdziuk, J. (2019). A Monte Carlo tree search approach to finding efficient patrolling schemes on graphs. *European Journal of Operational Research*, *277*(1), 255–268.
Keshari, A., Mishra, N., Shukla, N., McGuire, S., & Khorana, S. (2018). Multiple order-up-to policy for mitigating bullwhip effect in supply chain network. *Annals of Operations Research*, *269*(1–2), 361–386.
Kimbrough, S. O., Wu, D. J., & Zhong, F. (2002). Computers play the beer game: Can artificial agents manage supply chains? *Decision Support Systems*, *33*(3), 323–333.
Köchel, P., & Nieländer, U. (2005). Simulation-based optimisation of multi-echelon inventory systems. *International Journal of Production Economics*, *93*, 505–513.
Kocsis, L., & Szepesvári, C. (2006). Bandit based Monte-Carlo planning. *European conference on machine learning* (pp. 282–293). Berlin: Springer.
Lancioni, R. A. (2000). New developments in supply chain management for the millennium. *Industrial Marketing Management*, *29*(1), 1–6.
Lee, H. L., Padmanabhan, V., & Whang, S. (1997). Information distortion in a supply chain: The bullwhip effect. *Management Science*, *43*(4), 546–558.
Li, X., & Wang, Q. (2007). Coordination mechanisms of supply chain systems. *European Journal of Operational Research*, *179*(1), 1–16.
Mele, F. D., Guillen, G., Espuna, A., & Puigjaner, L. (2006). A simulation-based optimization framework for parameter optimization of supply-chain networks. *Industrial and Engineering Chemistry Research*, *45*(9), 3133–3148.
Min, H. (2010). Artificial intelligence in supply chain management: Theory and applications. *International Journal of Logistics: Research and Applications*, *13*(1), 13–39.
Mortazavi, A., Khamseh, A. A., & Azimi, P. (2015). Designing of an intelligent self-adaptive model for supply chain ordering management system. *Engineering Applications of Artificial Intelligence*, *37*, 207–220.
Mosekilde, E., Larsen, E., & Sterman, J. D. (1991). Coping with complexity: Deterministic chaos in human decisionmaking behavior. *Beyond belief: Randomness, prediction, and explanation in science* (pp. 199–229). Boca Raton, FL: CRC Press.
Muckstadt, J. A. (2004). *Analysis and algorithms for service parts supply chains*. Berlin: Springer.
Neto, T., Constantino, M., Martins, I., & Pedroso, J. P. (2020). A multi-objective Monte Carlo tree search for forest harvest scheduling. *European Journal of Operational Research*, *282*(3), 1115–1126.
Nienhaus, J., Ziegenbein, A., & Schönsleben, P. (2006). How human behaviour amplifies the bullwhip effect: A study based on the beer distribution game online. *Production Planning and Control*, *17*(6), 547–557.
O'donnell, T., Maguire, L., McIvor, R., & Humphreys, P. (2006). Minimizing the bullwhip effect in a supply chain using genetic algorithms. *International Journal of Production Research*, *44*(8), 1523–1543.

Petrovic, D., Roy, R., & Petrovic, R. (1999). Supply chain modelling using fuzzy sets. *International Journal of Production Economics*, *59*(1–3), 443–453.
Powell, W. B. (2007). *Approximate dynamic programming: Solving the curses of dimensionality*. Hoboken, NJ: Wiley.
Priore, P., Ponte, B., Rosillo, R., & de la Fuente, D. (2019). Applying machine learning to the dynamic selection of replenishment policies in fast-changing supply chain environments. *International Journal of Production Research*, *57*(11), 3663–3677.
Puterman, M. L. (2014). *Markov decision processes: Discrete stochastic dynamic programming*. Hoboken, NJ: Wiley.
Radhakrishnan, P., Prasad, V., & Gopalan, M. (2009). Inventory optimization in supply chain management using genetic algorithm. *International Journal of Computer Science and Network Security*, *9*(1), 33–40.
Robles, D., Rohlfshagen, P., & Lucas, S. M. (2011). Learning non-random moves for playing Othello: Improving Monte Carlo tree search. In *IEEE conference on computational intelligence and games (CIG'11)* (pp. 305–312). IEEE: Granada.
Roodbergen, K. J., Vis, I. F., & Taylor, G. D, Jr. (2015). Simultaneous determination of warehouse layout and control policies. *International Journal of Production Research*, *53*(11), 3306–3326.
Russell, S., & Norvig, P. (2020). *Artificial intelligence: A modern approach* (4th ed.). Saddle River, NJ: Pearson.
Sadeghi, J., Sadeghi, S., & Niaki, S. T. A. (2014). Optimizing a hybrid vendor-managed inventory and transportation problem with fuzzy demand: An improved particle swarm optimization algorithm. *Information Sciences*, *272*, 126–144.
Shukla, N., Tiwari, M., & Ceglarek, D. (2013). Genetic-algorithms-based algorithm portfolio for inventory routing problem with stochastic demand. *International Journal of Production Research*, *51*(1), 118–137.
Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.
Silver, D., & Veness, J. (2010). Monte-Carlo planning in large POMDPs. *Advances in neural information processing systems* (pp. 2164–2172). Cambridge, MA: MIT Press.
Simić, D., & Simić, S. (2012). Hybrid artificial intelligence approaches on vehicle routing problem in logistics distribution. In *International conference on hybrid artificial intelligence systems* (pp. 208–220). Berlin: Springer.
Sterman, J. D. (1989). Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment. *Management Science*, *35*(3), 321–339.
Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.
Vodopivec, T., Samothrakis, S., & Ster, B. (2017). On Monte Carlo tree search and reinforcement learning. *Journal of Artificial Intelligence Research*, *60*, 881–936.
Wang, J., & Shu, Y. F. (2005). Fuzzy decision modeling for supply chain management. *Fuzzy Sets and Systems*, *150*(1), 107–127.