Int. J. Production Economics 271 (2024) 109206

Contents lists available at ScienceDirect

International Journal of Production Economics

journal homepage: www.elsevier.com/locate/ijpe

Augmenting Monte Carlo Tree Search for managing service level agreements

Masih Fadaki [a,*], Atie Asadikia [b]

[a] Department of Supply Chain and Logistics Management, RMIT University, Melbourne, Australia
[b] Faculty of Engineering and IT, University of Melbourne, Melbourne, Australia

ARTICLE INFO

Keywords:
Monte Carlo Tree Search
Reinforced learning
Service level agreement
Machine learning
Supply chain
Allocation policies

ABSTRACT

Monte Carlo Tree Search (MCTS) is an algorithmic technique utilized in reinforcement learning, a subfield of artificial intelligence, that combines tree-based search and random sampling for decision-making in uncertain environments. Although MCTS has been successfully used for playing complex games such as Chess and Go, without customizing the original algorithm using domain knowledge, it struggles to effectively solve complex supply chain problems. This study proposes several augmenting mechanisms for MCTS, tailored for managing service level agreements. Furthermore, we enhance the proposed solution for products/services where adjusting the base-stock level is feasible. The results demonstrate that even with non-stationary demand, where most optimization methods reach their limits, employing these augmentation mechanisms significantly improves MCTS performance.

1. Introduction

Artificial Intelligence (AI), a pivotal branch of computer science, is dedicated to the creation of systems capable of executing tasks that customarily require human cognition. A seminal forecast by Bughin et al. (2018) anticipates the economic output of AI technologies to approximate $13 trillion by 2030, potentially accelerating global GDP by approximately 1.2 percent annually. AI research has evolved considerably since its formative years in the 60 s, ranging from modest interest to its current accelerated expansion. Nevertheless, the tangible value of AI to businesses became prominent only in recent years. Prime exemplars include IBM Watson, utilized for medical diagnoses, and Fleet Learning, a product of Tesla that facilitates vehicles to foresee fluctuating circumstances by incessantly sharing fleet data (Pournader et al., 2021).

The main rationale behind this study stems from the observation that solving various supply chain problems is similar to playing strategic games. In such scenarios, participants need to analyze interdependent decisions, anticipate rival actions, and optimize their strategies to achieve the highest performance or score in the game. Looking at supply chain problems from this angle suggests to use the game theory as the solution approach. However, it is important to note that game theory has its limitations in numerous real business settings such as coping with the non-stationary demand. An alternative approach is to customize AI algorithms that have been developed to play complex games like Chess and Go for the specific requirements and settings of supply chain problems. In recent years, many complex supply chain problems have been revisited and solved using Reinforcement Learning approach. For example, Wang and Minner (2024) addressed a demand fulfillment problem in a multi distribution center online retail environment where demand and replenishment lead time are stochastic using Deep Reinforcement Learning (DRL). Similarly, DRL was used to formulate a new method for stochastic control in multi-action inventory management (Kaynov et al., 2024).

The key motivation of this study is to provide a solution approach for addressing allocation problems within the context of a service level agreement, even when the assumption of stationary demand is violated. Several model-based allocation mechanisms have been proposed by scholars for managing service level agreements (e.g., Kwon et al., 2008; Abbasi et al., 2022; Kloos and Pibernik, 2020; Chen and Thomas, 2018). However, maintaining the assumption of stationary demand for retailers can be untenable in the real business world. Furthermore, majority of allocation policies have been developed for managing the immediate demands, however, when the penalty of deviation from the target service level is applied at the end of a performance review period, which is generally nominated in a service level agreement, ignoring the future demands provides a suboptimal solution. Regarding the methodological contribution of this study, we propose a method to enhance the standard MCTS algorithm for solving complex supply chain problems. While the MCTS has shown promise in game design and simple supply chain problems, we demonstrate the need to customize the MCTS algorithm for tackling complex problems. The second motivation for this study arises from the fact that, although a supplier may select the base-stock model for inventory management under a service level agreement, there is no obligation to maintain a fixed base-stock level. Using a constant base-stock level when demand is non-stationary does

* Corresponding author.
E-mail address: masih.fadaki@rmit.edu.au (M. Fadaki).

https://doi.org/10.1016/j.ijpe.2024.109206
Received 14 June 2023; Received in revised form 3 March 2024; Accepted 6 March 2024
Available online 7 March 2024

not yield the best results for the supplier, making it constantly infeasible to meet the target service level without overstocking. Therefore, we propose a dynamic and adaptive solution to adjust the base-stock level based on forecasted demand for future periods. Our proposition aligns with previous studies in which scholars attempted to employ a two-stage approach, whereby in the first stage, the demand is forecasted and then fed into the second stage, where an optimization model is utilized to make a decision under uncertainty (Islam et al., 2021).

The primary contribution of this study is the customization of a powerful AI algorithm called Monte Carlo Tree Search (MCTS) to effectively manage service level agreements in situations where a supplier's capacity constraint necessitates the use of an allocation policy. The proposed customization includes augmenting mechanisms aimed at enhancing the original MCTS algorithm through:(i) improving the precision of the MCTS search process by reducing the solution space through the identification of valid allocations under service level agreement settings, (ii) improving the MCTS selection process by prioritizing allocations with a lower distance from the realized demands, and (iii) improving allocation decisions by giving priority to demands when the current deviation from the target service level is the largest. Furthermore, in order to provide a comprehensive solution for managing the service level agreement using MCTS, we propose an adaptive system that adjusts the base-stock level. This system utilizes the demands of future performance review periods to drive a fine-tuning mechanism for updating the base-stock level.

2. Background

This study aims to propose a solution for allocation decisions within the context of Service Level Agreements (SLA) by employing the MCTS algorithm. To provide a comprehensive background for this problem, it is necessary to review two key perspectives: (i) the Monte Carlo Tree Search algorithm, and (ii) service level agreements as well as allocation mechanisms.

MCTS algorithm has been extensively studied in the fields of artificial intelligence and game design, offering valuable insights and techniques (Senington, 2022). However, its application to solve supply chain problems remains relatively limited (Edelkamp et al., 2016). Therefore, exploring the existing literature on MCTS within the supply chain domain is crucial to identify its potential and shed light on its effectiveness in addressing allocation decisions under SLAs.

In addition to understanding the MCTS algorithm, it is essential to review the various solution approaches proposed for allocation decisions in the context of service level agreements. The literature on SLAs provides valuable insights into the different mechanisms and strategies employed to allocate resources and meet performance targets. By examining the existing approaches, this study aims to build upon the existing knowledge and contribute to the development of effective allocation solutions within the framework of SLAs.

Overall, a comprehensive review of the literature on both the MCTS algorithm and service level agreements with allocation mechanisms will provide a strong foundation for addressing the allocation decision problem under SLAs using the MCTS algorithm in this study.

2.1. Monte Carlo tree search

Monte Carlo Tree Search (MCTS) is a powerful method for finding optimal decisions in a given domain by sampling the decision space and constructing a search tree based on the outcomes. The foundation of this method was laid by Abramson (1986) in his PhD thesis, where he extensively explored its application in games such as Tic-Tac-Toe. Subsequently, in 1992, Brügmann utilized MCTS for the first time in a go-playing program, marking a significant milestone.

In 2006, Coulom (2006) further developed the Monte Carlo method for game-tree search and introduced the term "Monte Carlo tree search". This advancement gained substantial traction when Kocsis and Szepesvári (2006) proposed the UCT algorithm, which enhanced the effectiveness of MCTS. Notably, the UCT algorithm was successfully implemented in the renowned go-playing program called MoGo by Sylvain Gelly, along with Yizao Wang, Rémi Munos, and Olivier Teytaud. By 2008, MoGo had achieved a "dan" (master) level in 99 go, and the Fuego program demonstrated victories against strong amateur players in 99 go.

The allure of MCTS lies in its statistical nature, making it an anytime algorithm where increased computational power generally leads to improved performance. It can be employed with minimal domain knowledge, making it applicable in various problem domains. MCTS has proven its worth by solving complex problems that were previously deemed challenging for other techniques (Qiu et al., 2022; Senington et al., 2021; Carvalho et al., 2019). It has emerged as a versatile and powerful approach capable of finding optimal solutions, demonstrating its effectiveness across a range of applications where other methods have struggled.

Although MCTS has been widely used in the AI domain, its applications in supply chain management are still scarce. In their study on applying MCTS to inventory management, Preil and Krapp (2022) showed that both the offline and the online MCTS models outperform several adopted AI-based approaches and they provided evidence indicating that a dynamic order policy determined by MCTS eliminates the bullwhip effect. Another example of applying the MCTS in the operations research field is related to the study of Saqlain et al. (2022) in which they showed that MCTS-scheduler outperformed various baseline planning algorithms and provided superior performance for solving Flexible Job-Shop Scheduling problem. A MCTS algorithm based solution was also proposed by Li et al. (2021) for the same problem. In a related context, MCTS has been utilized to solve the multi-Commodity Pickup and Delivery Vehicle Routing Problem which aimed to optimize the pickup and delivery of multiple unique commodities using a fleet of several agents with limited payload capacities Goutham and Stockar (2023). Finally, Barletta et al. (2023) employed the MCTS algorithm to solve the hybrid fleet capacitated vehicle routing problem.

2.2. Service level agreements and allocation mechanisms

A service level agreement is a performance-based contract that prioritizes outcomes over processes. This type of agreement outlines the target service level expected by the buyer, the duration of the performance review period (PRP), and includes financial incentives or penalties to encourage the attainment of the specified performance level. SLAs are extensively utilized in various industries, with a survey conducted by Oblicore Inc. revealing that more than 90% of businesses employ SLAs to effectively manage their suppliers (Oblicore, 2007). While the most commonly used industries for managing contracts with SLAs are IT and supply chain, SLAs have also been effectively utilized in various other sectors. Examples include allocating airport terminal gates to airlines (Jahani et al., 2021), in online travel-booking systems, realtor services to home sellers, and by insurance and financial service providers (Bianco et al., 2008).

In business-to-business (B2B) settings, buyers typically determine the minimum required service level, while in business-to-consumer (B2C) settings, suppliers set target service levels based on customer segments, taking into account perceived value (Ellram et al., 2004). Once the SLA is agreed upon, it becomes the supplier's responsibility to fulfill the contract terms and achieve the desired service level by investing in inventory or capacity. Different measures have been proposed in the literature to assess the supplier's service level, with the fill rate and ready rate being the most commonly used metrics. The fill rate, which is the average fraction of immediate demand satisfaction, is particularly relevant in our study (Silver et al., 1998; Axsäter, 2015).

Researchers have investigated the impact of service level measures and the length of the performance review period on the performance and stocking policies of both buyers and suppliers, considering both finite and infinite time horizons. For instance, in a single buyer supply chain model, Thomas (2005) examined the distribution of the finite-horizon fill rate and found that shorter review periods are generally preferred for meeting target levels. Similarly, Katok et al. (2008) conducted a laboratory study to observe decision-makers' behavior and concluded that longer review periods may be more effective when aiming to stimulate higher order quantities, especially in SLAs with a double-marginalization problem.

Having an effective allocation policy is crucial for effectively managing a service level agreement (Abbasi et al., 2022). The allocation policy outlines the rules and guidelines for distributing resources and fulfilling customer demands, ensuring that the target service level is met. With an efficient allocation policy in place, organizations can optimize their resource allocation, minimize stockouts, and maximize customer satisfaction (Jahani et al., 2021). By strategically allocating resources based on demand patterns, product characteristics, and service level requirements, organizations can strike a balance between inventory costs and meeting customer expectations. An effective allocation policy enables businesses to allocate resources dynamically, adjusting allocations based on real-time demand fluctuations and inventory availability. This proactive approach ensures that customer orders are fulfilled promptly, minimizing backorders and reducing the risk of penalties for not meeting the agreed-upon service levels (Fadaki et al., 2022a). Overall, an effective allocation policy is instrumental in optimizing inventory management, enhancing customer service, and maintaining a strong performance record in SLA-driven environments.

There have been proliferate number of studies in which myopic allocation policies are investigated (e.g., Fadaki et al., 2022b). When capacity binds, retailers compete to secure higher capacity in order to maximize their profit. Various capacity allocation mechanisms have been proposed in the operations research domain such as Proportional, Uniform, and Linear allocation policies. Although each policy uses different approaches to solve the allocation problem, Fadaki et al. (2022b) demonstrated that the performance of conventional allocation policies is identical when no strategic reasoning is applied. Therefore, in this study, we select the Proportional allocation policy as the baseline for comparing the performance of the MCTS algorithm. The Proportional allocation policy is widely used in resource allocation for distributing resources in proportion to the demands of each entity. By using the Proportional allocation policy as a benchmark, our aim is to evaluate the effectiveness and superiority of the MCTS algorithm in optimizing allocation problems.

3. Notations

A list of notations of this study is provided as below:

R	The set of retailers.
S	The base capacity level of supplier in a base-stock policy.
d_{rt}	Demand of retailer r in time t.
a_{rt}	Allocation of retailer r in time t.
$\mathcal{A}(.)$	Decision rule. Allocation mechanism function that assigns a feasible allocation(s) (actions) for a state vector.
p_s	Supplier's profit for selling one unit of product.
π_t	Profit of supplier in time t.
$\beta_{r,[t_0,t_0+\tau-1]}$	The fill rate of retailer r computed at the end of performance review period starting at t_0 and ending at $t_0+\tau-1$.
$\widehat{\beta}_r$	Target fill rate of retailer r.
λ_r	Penalty cost for any unit deviation from the target fill rate.
τ	Performance review period.
C_e	Exploration parameter in UCT.
C_d	Deviation parameter.
Ψ	Set of states in MDP. ψ is a state (node) in MCTS decision tree.

4. Model and results

Before introducing augmenting mechanisms, we provide a brief explanation of the original MCTS algorithm, which is presented in Algorithm 1. Original MCTS algorithm comprises four key stages that are executed iteratively: *Selection*, *Expansion*, *Simulation*, and *Backpropagation*. In the Selection stage, the algorithm traverses the search tree by selecting nodes based on a selection strategy that balances exploration and exploitation. The Expansion stage involves adding new nodes to the tree by generating all possible moves from the selected node. In the Simulation stage, random playouts or simulations are performed from the newly expanded nodes to estimate their potential outcomes. Lastly, in the Backpropagation stage, the results of the simulations are backpropagated through the tree, updating the statistics of the visited nodes. This iterative process continues until a predefined stopping condition is met, leading to the identification of the most promising move or decision based on the accumulated knowledge within the search tree.

With respect to the Selection strategy of MCTS, various methods can be used, and among them, UCT (Upper Confidence Bounds for Trees) stands out as one of the most effective approaches (Eq. (1)). Developed by Kocsis and Szepesvári (2006), UCT strikes a balance between exploration and exploitation by utilizing a formula that considers both the exploitation of the current knowledge about move values and the exploration of unexplored moves. By dynamically adjusting the exploration–exploitation trade-off, UCT efficiently explores the search space and focuses on promising moves. Its effectiveness in selecting high-quality moves has made UCT a popular and powerful choice in MCTS algorithms, contributing to its success in various domains, including games and decision-making problems.

$$\mathbb{UCT}_j = \overline{\mathcal{R}}_j + C_e\sqrt{\frac{Log N_j}{n_j}} \tag{1}$$

In the UCT, n_j and N_j are the number of times node j and its parent have been visited, respectively. $\bar{\mathcal{R}}_j$ is the empirical mean of the rewards (profit) that have been obtained by trajectories going forward from that node.

4.1. Markov decision processes

To model the sequential decision process of this problem, a Markov Decision Process (MDP) framework is employed with the following general assumptions: (i) time is discrete and organized in a numerable and infinite succession of equally spaced instants; (ii) demands are exogenous and non-negative integers; (iii) demand during a period is fulfilled with the on-hand stock at the beginning of that period; (iv) the available inventory is pooled to fulfill the demand of all retailers; (v) the supplier uses a base-stock policy with zero lead time; (vi) unused capacities are not carried over to the following periods (leading to lost sales); and (vii) the product is sold only in whole units. Below are the details of the MDP components for the allocation problem under a service level agreement.

Decision Epochs: At each decision epoch, the decision maker (supplier) chooses an action (allocation) in the state occupied by the system at that time. We consider a discrete-time MDP with the following decision epochs $T = \{1, 2, \dots, H\}$, $H \le \infty$, where H represents the length of the planning horizon. Note that in each performance review period ($\tau < H$), the performance measure (fill rate) is measured at the

Algorithm 1: Generic Monte Carlo Tree Search

```
Input  : rootNode
Output: BestChildNode

while computational budget is not exhausted do
    selectedNode ← Selection(rootNode);
    expandedNode ← Expansion(selectedNode);
    simulationResult ← Simulation(expandedNode);
    Backpropagation(expandedNode, simulationResult);
end
return BestChild(rootNode);

Function Selection(node):
    while node is fully expanded and not terminal do
        node ← BestChild(node);
    end
    return node;
end

Function Expansion(node):
    unexploredActions ← Actions(node) -
     ExploredActions(node);
    randomly select an unexplored action a;
    newChildNode ← CreateNode(node, a);
    AddChild(node, newChildNode);
    return newChildNode;
end

Function Simulation(node):
    currentState ← State(node);
    while currentState is not terminal do
        randomly select an action from Actions(currentState);
        currentState ← Result(currentState, selectedAction);
    end
    return ResultValue(currentState);
end

Function Backpropagation(node, result):
    while node is not null do
        UpdateStats(node, result);
        node ← Parent(node);
    end
end

Function BestChild(node):
    return child node with the highest value according to the
     selection strategy;
end
```

end of the performance review period for all time units within the most recent period".

States: As defined by Powell (2011, p. 179), "A state variable is the minimally dimensioned function of history that is necessary and sufficient to compute the decision function, the transition function, and the contribution function. In plain English, a state variable is all the information you need to know (at time t) to model the system from time t onward". In other words, it is a *state of knowledge*. In this study, according to the selected base-stock inventory policy, the supplier's inventory level is a constant $\mathcal{S}$ number of items in each period. Therefore, although the number of available items ($\mathcal{S}$) technically serves as one of the state variables, its value remains constant across all states ($\mathcal{S}_t = \mathcal{S}, \forall t$). Thus, to achieve a minimally dimensioned state vector, it could be excluded. However, for transparency purposes, we choose to retain it.

With respect to the demand from retailers, let d_t represent the vector of demand for all retailers in the tth period, defined as $d_t = (d_{rt})_{r\in R, d_{rt}\in\mathbb{Z}}$. Regarding the sequence of events, it should be noted that d_{t+1} is an exogenous random variable representing the change in demand during the time interval between t and $t+1$. At time t, d_t is a vector of numbers, while d_{t+1}, at time t, is considered random. In other words, d_t is known at time t, when the supplier is making allocation decisions. Basically, events within a single period follow this sequence: the supplier's inventory is replenished to $\mathcal{S}$ level at the beginning of the period; exogenous demands are realized; and allocation decisions are made. Consequently, the state of the system in each period can be defined as $\psi_t = (\mathcal{S}, d_t)$, which is sufficient for making allocation decisions if a myopic allocation policy is employed (see Section 4.2). In the next Section 4.3, it will be demonstrated that using the original MCTS algorithm to solve an allocation problem does not yield an acceptable level of performance. Several augmenting mechanisms will be proposed in subsequent sections to enhance the performance of MCTS for allocation decisions. Although not all the proposed mechanisms impact the MDP structure, there are two major updates to the MDP model: (i) updating the state vector, and (ii) revising the action space. With respect to updating the state vector, we propose including the vector of average fill rates for all retailers ($\bar{\beta}_t = (\bar{\beta}_{rt})_{r\in R}$) in the state vector. Consequently, the state of this allocation problem is updated to $\psi_t = (\mathcal{S}, d_t, \bar{\beta}_t)$. Note that $\bar{\beta}_{rt}$ represents the average fill rate of retailer r for all periods from the beginning of the current performance review period up to the current period (t). Further details on how $\bar{\beta}_t$ is utilized to adjust the allocation decisions are provided in Section 4.4.3. Finally, a set of Ψ can be defined as the 'set of all possible values of the state vector ψ'.

Actions: Regarding the allocation decisions, in the state ψ_t at time t, the action vector ($a_t = (a_{rt})_{r\in R, a_{rt}\in\mathbb{Z}, \sum_r a_{rt}\le S, a_{rt}\le d_{rt}\,\forall r}$) specifies the alternative possibilities regarding the number of items which can be allocated by the supplier to each retailer (a_{rt}), satisfying two conditions: (i) the total number of allocated items is less than or equal to the base-stock level ($\sum_r a_{rt} \le S$), and (ii) the items allocated to each retailer do not exceed their respective demands ($a_{rt} \le d_{rt}, \forall r$). Consequently, a set of A can be defined as the 'set of all possible values of the allocation vector a'.

Transition probabilities: For a general allocation problem with non-stationary demands, the transition probability is unknown. This is technically one of the reasons we recommend using MCTS as a solution approach for this type of problem. If the demand distribution were known and stationary, and if there were only one retailer in the system, then the transition probabilities could be estimated. For further details, please refer to Puterman (2005, p. 40). Even if the demands were stationary and independent, with more than one retailer in the system, considering the joint probabilities of demands introduces several complexities in estimating the transition probabilities, making the solution approach impractical for the real business environment. In cases where demands are non-stationary, estimating the joint probabilities of demands for $|R|$ number of retailers becomes analytically intractable.

Rewards: Considering that the supplier is in state ψ_t and takes action a_t, the supplier's reward or profit can be estimated using Eq. (2). Assuming the performance review period has a length of τ, the computation of profit in each period t depends on whether we are at the end of the performance review period or not. If we are not at the end of performance review period ($\frac{t}{\tau} \neq \lfloor\frac{t}{\tau}\rfloor$), the profit of supplier can be estimated as $p_s \sum_{r\in R} a_{rt}$. Conversely, if the current period (t) marks the end of a performance review period ($\frac{t}{\tau} = \lfloor\frac{t}{\tau}\rfloor$), the total penalty cost for deviating from the target fill rate ($\hat{\beta}_r$) for all retailers is deducted from the profit obtained from selling the items to the retailers: $p_s \sum_{r\in R} a_{rt} - \sum_{r\in R} \lambda_r \left[\hat{\beta}_r - \beta_{r,[t-\tau+1,t]}\right]^+$. Note that the fill rate

$\beta_{r,[t-\tau+1,t]}$ is computed for the most recent performance review period. For instance, if length of performance review period is $\tau = 10$ and we are on day 100, the fill rate is calculated from day $t - \tau + 1 = 91$ to day 100. When estimating the deviation, it is important to note that using the function $[x]^+ = max(x, 0)$ highlights the fact that any positive deviation from the target fill rate ($\hat{\beta}_r < \beta_{r,[t-\tau+1,t]}$) is desired, and only negative deviations are penalized.

$$\pi_t(\psi_t, a_t) = \begin{cases} \text{if } \frac{t}{\tau} \neq \lfloor \frac{t}{\tau} \rfloor : & p_s \sum_{r \in R} a_{rt} \\ \text{if } \frac{t}{\tau} = \lfloor \frac{t}{\tau} \rfloor : & p_s \sum_{r \in R} a_{rt} - \sum_{r \in R} \lambda_r \left[\hat{\beta}_r - \beta_{r,[t-\tau+1,t]} \right]^+ \end{cases} \tag{2}$$

To clarify the reward function of the MDP, a brief explanation of various notations of fill rate that have been used in this study is presented. The target fill rate ($\hat{\beta}_r$) for retailer r is a value indicated in the terms of a service level agreement between the supplier and the retailer. Any negative deviation from this target will lead to a penalty cost for the supplier. The actual value of the fill rate ($\beta_{r,[t_0,t_0+\tau-1]}$) for retailer r at the end of each performance review period, starting at time t_0 and ending at time $t_0+\tau-1$ is obtained using Eq. (3). Note that the fill rate is computed at the end of each performance review period for the very last time interval. For example, if $\tau = 10$, the fill rate is estimated at day 10 for the period [1,2, …, 10], and then it will be estimated at day 20 for the period [11, 12, …, 20], continuing in this manner. For simplicity of notation, the time-frame is dropped from the fill rate (β_r) when reporting the results. Finally, the average fill rate ($\bar{\beta}_{rt}$) for retailer r at time t refers to the same fill rate equation, however, it is computed up to time t rather than at the end of the performance review period. For example, if $\tau = 10$ and $t = 96$, the average fill rate at t would be the quotient of the sum of allocated items to the sum of demands for a particular retailer from day 91 to 96. It is evident that the average fill rate is not used for computing the penalty cost of deviating from the target fill rate. The application of the average fill rate is explained in Section 4.4.3. This measure, as part of the state vector, is used to improve allocation decisions for maximizing the supplier's profit.

$$\beta_{r,[t_0,t_0+\tau-1]} := \frac{\sum_{t=t_0}^{t_0+\tau-1} a_{rt}}{\sum_{t=t_0}^{t_0+\tau-1} d_{rt}} \tag{3}$$

Decision Rule and Policy: A decision rule $\mathcal{A}_t(.)$ maps states (ψ_t) to actions (a_t), meaning it defines the action (a_t) to be chosen in state ψ_t at time t. Regarding the allocation decisions problem, *decision rule* is equivalent to the *allocation policy*. Essentially, this means if the supplier is in state ψ_t (i.e., he has $\mathcal{S}$ number of items; demand vector of d_t is realized; and whether a myopic allocation policy or MCTS is used, the vector of $\bar{\beta}_t$ is considered), the decision rule determines the allocation vector (a_t). Additionally, a policy represents a series of these decision rules ($\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_{t-1}$) spanning the planning horizon H. The objective of the allocation problem is to maximize the cumulative profit of the supplier $\sum_{t \in T} \pi_t(\psi_t, a_t)$ throughout the planning horizon by identifying the suitable allocation policy.

4.2. Myopic allocation policies

As highlighted in the Background section, when the capacity of a supplier binds, it is crucial for an effective business setting to employ an allocation policy. Several myopic allocation policies have been proposed by scholars, where allocation decisions are based solely on the demands of the current period. In other words, a myopic allocation policy disregards crucial information such as the remaining days until the end of the performance review period and the current deviation of fill rates from their target values. These pieces of information are essential for an effective allocation decision-making process, yet they are not taken into consideration in a myopic approach.

$$\mathcal{A}_r(d) = \min \left\{ d_r, \ \frac{\mathcal{S}\, d_r}{\sum_{j=1}^{|R|} d_j} \right\} \tag{4}$$

As mentioned earlier, in the absence of strategic reasoning, the performance of well-known myopic allocation policies such as Proportional, Linear, and Uniform is identical. Therefore, we consider the Proportional allocation policy (Eq. (4)) as the baseline for comparing the performance of the MCTS algorithm. Assuming that the capacity of supplier for the current period is $\mathcal{S}$ and vector of demands for all retailers is d, each retailer r with demand of d_r receives $\mathcal{A}_r(d)$ as per the Proportional allocation policy. Note that the indice of time t is dropped for simplicity.

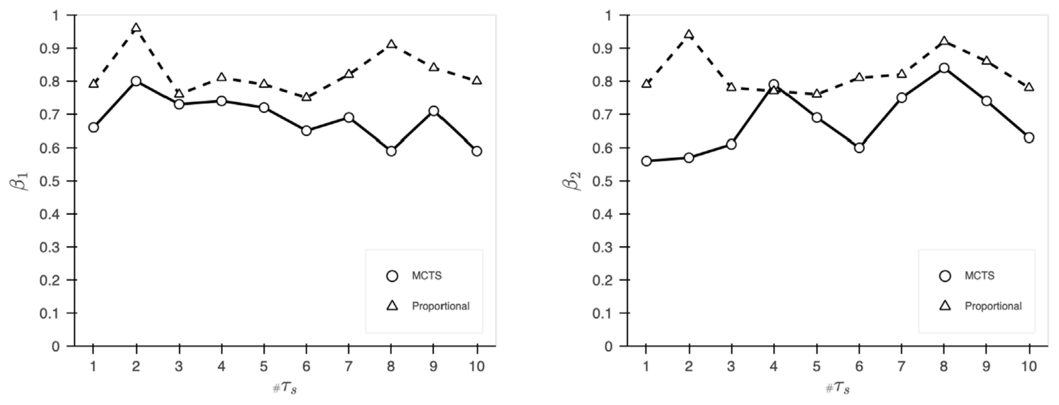
4.3. Performance comparison

To showcase the efficiency of the MCTS original algorithm in comparison to the Proportional allocation policy for managing service level agreements, their performances were compared in the following experiment. A supplier uses a base-stock policy with zero lead time and a capacity of $\mathcal{S} = 10$ units per day and he provides products with the unit profit of $p_s = \$10$ to two retailers. The length of performance period for both retailers is $\tau = 10$ days and the penalty of every percentage of deviation from the target fill rate ($\hat{\beta}_r = 85\%$) is $\lambda_r = \$100$.

After a number of fine-tuning trials, the Exploration (C_e) and Deviation (C_d) parameters have been selected to be $\sqrt{2}$ and 2, respectively. With respect to the demand pattern, note that the MCTS algorithm is not impacted by the type of demand, whether it is stationary or non-stationary. Therefore, for the baseline example, demand of each retailer can be randomly selected from a Uniform distribution $d_r \sim \mathcal{U}(2, 8)$, without loss of generality. If it is desired to generate a non-stationary demand, the following process can be followed: (i) Maintain a constant Coefficient of Variation (CV); (ii) Use either deterministic time intervals or stochastic intervals (randomly selected from a Uniform distribution); (iii) For each time interval i, draw samples from the Normal distribution $\mathcal{N}(\mu_i, \sigma_i)$ where $\mu_i = \mu_{i-1} + \tilde{\epsilon}$ and $\sigma_i = CV \times \mu_i$. Here, $\tilde{\epsilon}$ is randomly selected from $\mathcal{U}(\bar{u}, \underline{u})$, ensuring $\bar{u} > \underline{u}$ and both $\bar{u}$ and $\underline{u}$ are positive integers.

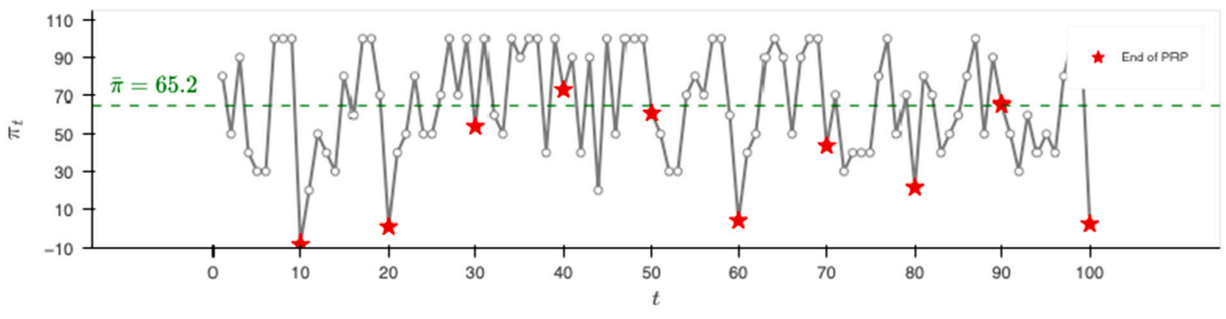
The comparison of the fill rates of retailers, when the MCTS original algorithm and Proportional policy are utilized for allocation decisions is presented in Fig. 1. Note that the horizontal axis ($\#\tau_s$) represents the number of performance review periods. Results indicate that without engaging any augmenting mechanism, MCTS delivers the average fill rates of $\bar{\beta}_1 = 68.8\%$ and $\bar{\beta}_2 = 67.8\%$ whereas using Proportional policy provides $\bar{\beta}_1 = 82.3\%$ and $\bar{\beta}_2 = 82.3\%$ for retailers 1 and 2, respectively. The obvious observation is that the MCTS original algorithm is unable to outperform the Proportional policy in terms of allocation decisions. Note that the Proportional policy is myopic, as it only considers the demands of the current period for allocation decisions. In contrast, the MCTS policy is foresighted, as it takes into account the demands of all remaining periods up to the end of the performance review period when making allocation decisions for the current period. It can be observed that, despite the advantage of MCTS in considering future demands, it does not result in superior performance.

The performance of Proportional policy and MCTS can be also compared from the supplier's perspective. Fig. 2 presents the profit of the supplier over a span of 100 days, with each performance review period consisting of 10 days. The stars in the figure show the end of each performance review period when the penalty of deviation from the target fill rate is applied. It can be observed that the average profit of supplier using the Proportional policy ($\bar{\pi} = \$81.6$) is higher than the MCTS ($\bar{\pi} = \$65.2$). Furthermore, variation of allocation decision-making for the MCTS is greater than the Proportional policy.

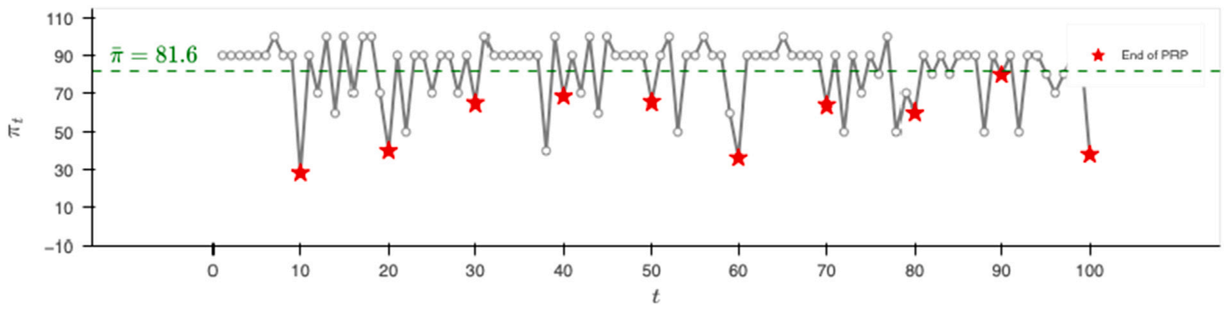


(a) Fill rate of retailer 1 for original MCTS. (b) Fill rate of retailer 2 for original MCTS.

Fig. 1. Fill rate of retailers when original MCTS is used (no augmenting mechanism is engaged).



(a) MCTS original algorithm



(b) Proportional

Fig. 2. Profit of supplier for MCTS original algorithm and Proportional allocation policy.

In order to understand the causes of deficiency in using the MCTS original algorithm for solving supply chain problems, we have designed an experiment. All settings of this experiment are identical to the base settings of this study. The algorithm is however repeated for 100 times with different realization of demand for each retailer.

To capture just the shortages which are driven by the inefficiency of the MCTS algorithm, for each day within the performance review period, the following two cases are reviewed:

Case 1: In any time period (t), if total demand is greater than or equal to the base-stock ($\sum_{i=1}^{|R|} d_{it} \geq \mathcal{S}$), it means that in the best case scenario, the total sum of allocations would be the same as the base-stock level. On the other hand, if any inefficiency is driven by the MCTS algorithm, total allocations would be less than the base-stock ($\sum_{i=1}^{|R|} a_{it} < \mathcal{S}$) which results in a shortfall of $\mathcal{S} - \sum_{i=1}^{|R|} a_{it}$. The extent of deviation from a sensible and feasible allocation at a specific time arises from the imprecision in the original MCTS decision-making process.

Case 2: In this scenario, in a particular time (t), total demand is less than the base-stock ($\sum_{i=1}^{|R|} d_{it} < \mathcal{S}$) which means that the supplier has sufficient stock to satisfy demand of both retailers. Therefore, if MCTS works flawlessly, it is expected that total allocations would be the same as total demand for this case ($\sum_{i=1}^{|R|} a_{it} = \sum_{i=1}^{|R|} d_{it}$), and thus the total shortage should be zero. As a measure of inefficiency of original MCTS algorithm, we record any observed shortage ($\sum_{i=1}^{|R|} d_{it} - \sum_{i=1}^{|R|} a_{it}$) for this scenario.

The result of experiment is presented in Fig. 3. Given the performance review period of $\tau = 10$ days, it can be observed that the total shortages from both cases exponentially decrease from beginning to the end of τ. Note that the recorded shortages are the total shortages from 100 repetitions, and they are only shortages related to the MCTS inefficiencies. The key implication of this experiment is that the precision of the MCTS original algorithm is negatively affected as the distance from the end of the performance review period increases.

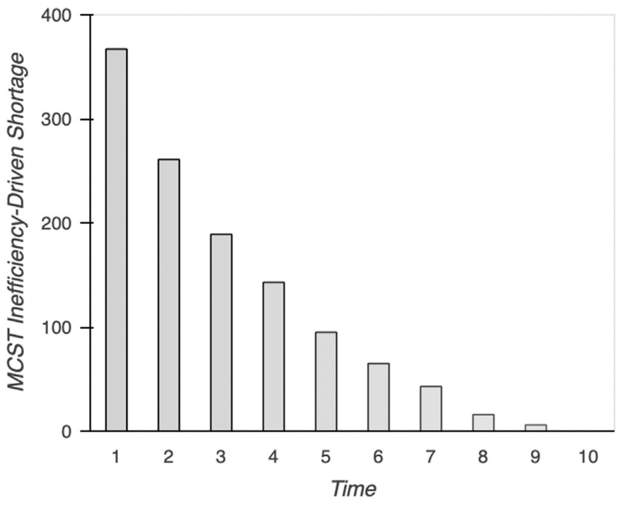


Fig. 3. Shortages for each period (day) within the performance review period.

This occurs due to the design of the MCTS original algorithm, where the randomized selection after expanding each selected node has a more significant impact on higher-level nodes (i.e., those closer to the root) compared to lower-level nodes. For instance, let us consider the allocation problem. On the first day of the Performance Review Period (PRP), one of the feasible allocations is chosen as a new node to expand, and subsequent allocations are randomly selected from the first day to the end of the PRP. This results in nine random selections among the feasible allocations for each level of the decision tree. In contrast, if we examine the eighth day, the number of random selections decreases to two levels.

It could be argued that increasing the number of main iterations in the randomized process of MCTS would alter Fig. 3. This argument holds true regarding the values of shortages resulting from the inefficiency of MCTS. However, the overall shape and trend of the graph would remain the same. It is important to note that the primary objective of this experiment is not to investigate the precise magnitude of shortages in each period. Instead, the experiment emphasizes the diminishing accuracy of allocation decisions made by MCTS as we move further away from the end of the Performance Review Period (PRP). The shape of the graph, however, undergoes changes in two extreme scenarios for any MCTS problem: (i) when decreasing the randomized iterations below a certain level, and (ii) when increasing the randomized iterations to the point where all possible nodes of a decision tree are explored before shifting to the exploitation process. Regarding item (i), it is not a reasonable decision-making process due to the high probability of randomized selection of decisions, leading to unreliable outcomes. As for item (ii), it falls outside the scope of MCTS since exploring all nodes of a decision tree renders the use of MCTS unnecessary. In such cases, applying Dynamic Programming or other Tree Search algorithms would yield highly accurate results.

4.4. Augmenting mechanisms

The results of the conducted experiment, along with the earlier discussion on the inefficiencies of the general MCTS algorithm highlight the need for improvements when utilizing MCTS in designing service level agreements. In light of this, we propose three augmenting mechanisms as follows.

4.4.1. Augmenting mechanism for identifying valid allocations

Without engaging the domain knowledge into the MCTS algorithm, a *feasible action* set is selected for each state whereby all possible actions for a particular state are included. When the algorithm gets to that particular state (ψ), if it has not been expanded, during the expansion process, $|\mathcal{F}_\psi|$ number of new branches are created where $|\mathcal{F}_\psi|$ is the cardinality of the feasible actions for a particular state ψ. Assume that for the case of allocation decisions for two retailers, the tuple (a_1, a_2) represents the feasible allocations when demands are (d_1, d_2) and base-stock level is $\mathcal{S}$. The feasible action set can be developed using Eq. (5).

$$\mathcal{F}_\psi = \{(x_1, x_2) |\, x_1, x_2 \in \mathbb{Z} \wedge 0 \le x_1 \le \min(\mathcal{S}, d_1) \wedge 0 \le x_2 \le \min(\mathcal{S}, d_2) \wedge x_1 + x_2 \le \mathcal{S}\} \tag{5}$$

In the context of MCTS, capacity of computational resources can be generally considered as the collection of three factors (i) speed of hardware, (ii) memory size to accommodate the decision tree in each iteration, and (iii) number of iterations or time of search process. For any MCTS problem, if we consider the capacity of computational resources to be constant, any possible reduction in size of feasible actions for a state leads to improving the accuracy of the search process. Furthermore, guiding the algorithm towards the actions which generate higher rewards (lower costs) using the domain knowledge also improves the algorithm outputs.

With respect to the allocation problem, in each particular state ψ, there are two cases:

Case 1: For a state (ψ), total demand can be greater than or equal to the base-stock ($\sum_{i=1}^{|R|} d_i \ge \mathcal{S}$). In this case, any feasible allocation whereby the total allocations to all retailers is less than the base-stock level leads to a sub-optimal outcome. Therefore, in the augmented version of the algorithm, these allocations are removed from the set of feasible allocations for a particular state. For this case (c_1), the $|\mathcal{F}_\psi|$ is updated to $\mathcal{F}_{\psi,c_1}$ as per Eq. (6):

$$\mathcal{F}_{\psi,c_1} = \{(x_1, x_2) |\, x_1, x_2 \in \mathcal{F}_\psi \wedge x_1 + x_2 = \mathcal{S}\} \tag{6}$$

Case 2: The second possible scenario is realized where the total demand is less than the base-stock ($\sum_{i=1}^{|R|} d_{it} < \mathcal{S}$). For this type of states, any allocations below demands result in sub-optimal profit for the supplier. Therefore, feasible allocations are filtered for Case 2 (c_2) using Eq. (7).

$$\mathcal{F}_{\psi,c_2} = \{(x_1, x_2) |\, x_1 = d_1 \wedge x_2 = d_2\} \tag{7}$$

4.4.2. Augmenting mechanism for the selection process of MCTS

Using the original UCT as the node selection method strikes a balance between the exploitation and exploration processes. Basically, in each *Selection* step, the child with highest UCT is selected. Although UCT is effective as a general purpose selection method, using the specific domain knowledge can improve its performance.

With respect to the allocation problem, in the hierarchy of decision tree, each level of tree represents a day (period) with its associated demands in the decision making process. Each node also represents the potential feasible allocations to address the demand of the hierarchical level that the node belongs to. As presented in Eq. (8), the original $\mathbb{UCT}$ is augmented to $\mathbb{AUCT}$ by including a new term, *Allocation Distance*: $\|\overrightarrow{a_j} - \overrightarrow{d_j}\|_2$.

$$\mathbb{AUCT}_j = \overline{\mathcal{R}}_j + C_e\sqrt{\frac{Log N_j}{n_j}} - C_d\, \|\overrightarrow{a_j} - \overrightarrow{d_j}\|_2 \tag{8}$$

This L^2 norm distance is the element-wise differences between vector of allocations for a particular node $\overrightarrow{a_j}$ and the demand vector $\overrightarrow{d_j}$ of the corresponding period. The term is deducted from the original $\mathbb{UCT}$ since higher distance from the demand vector leads to worsen fill rates. We have also incorporate a *Distance Parameter* (C_d) to ensure that the contribution of distance to the selection process is adjusted. Note that in calculating the reward ($\overline{\mathcal{R}}_j$), the distance between potential allocations and demands is indirectly considered. However, this consideration is only limited to the leaf nodes within the roll-out process when trajectories are created from a particular node to the leaf nodes of the decision tree. Therefore, embedding the distance value to the ($\overline{\mathcal{R}}_j$) ensures that the distances from demands of each level in the hierarchy of decision tree is also considered.

4.4.3. Augmenting mechanism for identifying optimal allocations

Another opportunity to improve the performance of MCTS using the domain knowledge is to monitor the deviation of fill rates for retailers in each period. Particularly, for periods that the total demand of retailers is higher than base-stock level, supplier experiences shortages. Therefore, incorporating an allocation mechanism to consider the history of allocations and measures the aggregate deviation of each retailer from the target fill rate improves the allocation decisions. In other words, the allocation mechanism should be able to allocate more stock in a particular period to the retailer that its deviation from the target fill rate is the largest.

As explained in the Background section, many allocation mechanisms have been proposed by scholars (e.g., Kwon et al., 2008; Abbasi et al., 2022; Kloos and Pibernik, 2020; Chen and Thomas, 2018). However, not all types of allocation policies are suitable to be encapsulated into the MCTS process. In this study, we use the type of allocation mechanism which is based on probabilistic distribution of shortages among the retailers. The allocation policy provides the baseline allocations (a_1^*, a_2^*) considering the demand on a particular time (d_1, d_2) and the history of fill rates. Then, the L^2 norm distance $(\|\overrightarrow{a_j^*} - \overrightarrow{a_j}\|_2)$ between the potential allocation vector $\overrightarrow{a_j}$ and the baseline allocations $\overrightarrow{a_j^*}$ is considered in the final selection of the best node in the decision making process. To identify the adjusted allocations, first we need to estimate a rationing ratio (ρ_r) as a measure of deviation from the target fill rate for each retailer (Eq. (9)).

$$\rho_r = \frac{\bar{\beta}_{rt} - \hat{\beta}_r}{\sum_j |\bar{\beta}_{jt} - \hat{\beta}_j|} \tag{9}$$

In the rationing ratio for retailer r in time t, $\hat{\beta}_r$ is the target fill rate and $\bar{\beta}_{rt}$ represents the average fill rate of retailer r in time t. Basically, this ratio shows how far the fill rate of a particular retailer is from its target fill rate in time t. Then, based on the rationing ratio for each retailer, we can estimate the best proportion (η_r) of shortages for sharing between the retailers (Eq. (10)).

$$\eta_r = \begin{cases} \dfrac{1}{|R|} & \text{if}\, \rho_r = 0,\ \forall r \\ \dfrac{\rho_r}{\sum_j \rho_j} & \text{if}\, (\rho_r \geq 0,\ \forall r) \wedge (\exists r : \rho_r > 0) \\ \dfrac{|1/\rho_r|}{\sum_j |1/\rho_j|} & \text{if}\, (\rho_r \leq 0,\ \forall r) \wedge (\exists r : \rho_r < 0) \\ \dfrac{\rho_r^+}{\sum_j \rho_j^+} & \text{otherwise} \end{cases} \tag{10}$$

To estimate η_r for each retailer, the basic idea is to optimally share products to retailers with the objective of minimizing the gap between the average fill rate and the target fill rate. If the current mean fill rate of both retailers is zero, the same portion of shortages is shared between them. If the mean fill rate of both retailers is above the target, the proportional mechanism for sharing the fill rate is utilized. The same proportional mechanism can be also considered when mean fill rate of both retailers is below the target by considering the proportion of inverse value of ρ_r. Finally, in case that the mean fill rate of one retailer is above and the other is below the target, the proportional mechanism is used with a little bit of tweak of ρ_r to $\rho_r^+ = max(\rho_r, 0)$.

Based on the estimated values η_r for both retailers in time t, adjusted allocations (a_1^*, a_2^*) are estimated using Eq. (11).

$$a_{r,t}^* = d_{r,t} - \eta_r\, [\sum_{j=1}^{|R|} d_{j,t} - S]^+ \tag{11}$$

Once the adjusted allocations (a_1^*, a_2^*) are determined in each level of decision tree hierarchy, the corresponding distance $(\|\overrightarrow{a_j^*} - \overrightarrow{a_j}\|_2)$ is deduction from the exploitation factor for the final node selection process.

4.5. Results after engaging the augmenting mechanisms for the MCTS algorithm

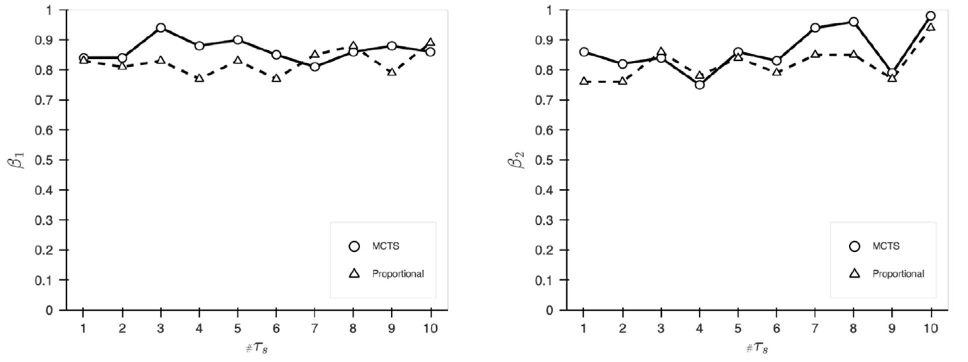
All three proposed augmenting mechanisms have been engaged to improve performance of the MCTS using the service level agreement domain knowledge. The results of simulation process are presented in Fig. 4. The mean fill rate of retailers for the Proportional allocation policy are $\bar{\beta}_1 = 82.5\%$ and $\bar{\beta}_2 = 82.0\%$ whereas the augmented MCTS provides the mean fill rate of $\bar{\beta}_1 = 86.6\%$ and $\bar{\beta}_2 = 86.3\%$ for the same settings over the ten performance review periods.

It can be observed that the average fill rates of both retailers is currently above the threshold of 85% after encapsulating the augmenting mechanisms into the MCTS algorithm. Note that neither Proportional allocation policy nor MCTS original algorithm would be able to meet the target fill rate. This shows 5.0% and 5.2% improvements for retailers 1 and 2, respectively compared to the Proportional allocation policy and 27.3% and 25.9% improvements compared to the MCTS original algorithm. It is also not far from expectation that the supplier's profit is improved as a result of improved fill rates and thus lower penalty costs. Fig. 5 shows the supplier's profit for ten PRPs, each including 10 days. The average profit ($\bar{\pi}$ = \$90.6) resulting from augmented MCTS shows an 11% and 39% improvement in profit compared to the Proportional policy and the original MCTS algorithm, respectively.

So far, we have demonstrated that the proposed augmented MCTS effectively manages allocation decisions under service level agreements. Next, we will further enhance the solution for a specific category of products or services. To understand this category, we need to clarify the scenarios that the base-stock policy can serve as a foundation for inventory management: (i) In the first scenario, changing the base-stock level is challenging due to various constraints, including limitations on infrastructure expansion. For instance, the base-stock policy is used to allocate airport terminal gates to airlines, rent terminal cranes to cargo ships in ports, and allocate berth space capacity to vessels in a container terminal (Jahani et al., 2021). In this category, even when facing non-stationary demand, adjusting the base-stock level is not readily available. For example, on a particular day, the same number of cranes is available for allocation to sea freight service providers. For this category, we have proposed an augmented MCTS solution that can effectively maximize the supplier's profit. (ii) In the second category of products or services, changing the base-stock level over time is feasible. For example, consider scenarios where the base-stock policy is used to manage popular clothing items in retail clothing stores, control the inventory of essential medications in pharmaceutical companies, or manage key ingredients like hamburger patties and buns in fast food chains. For this class of products and services, we propose a solution in the next section on how the supplier can elevate its profit by adjusting the base-stock level. Essentially, for this category, the comprehensive solution, including both components (1. augmented MCTS and 2. changing the base-stock), is necessary to maximize the supplier's profit.

4.6. Optimal base-stock level

In the previous sections, MCTS has been improved to determine the best allocations under a service level agreement. It should be however noted that the base-stock level has been considered to be fixed for all performance review periods. In reality though for a non-stationary demand, employing a constant base-stock level definitely does not provide the optimal profit for the supplier, and the target service level cannot be constantly met with no overstocking. Therefore, to provide a comprehensive solution for managing a service level agreement, there



(a) Fill rate of retailer 1 for augmented MCTS. (b) Fill rate of retailer 2 for augmented MCTS.

Fig. 4. Fill rate of retailers when original MCTS is improved (all augmenting mechanisms are engaged).

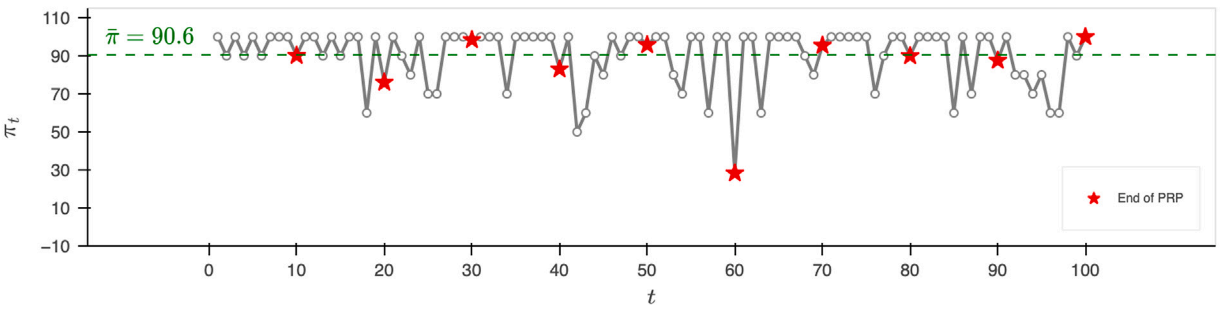


Fig. 5. Profit of supplier for the augmented MCTS algorithm.

is a need for a dynamic and adaptive system to adjust the base-stock level as per the forecasted demand for future periods. It is obvious that using the classic knowledge of formulating the base-stock level for the case of stochastic demand cannot be used for the non-stationary demand.

Note that to identify the optimal base-stock level, the supplier considers the aggregate demand of all retailers. To generate non-stationary aggregate demands where mean and standard deviation of demand change over time, since the length of performance review period (τ) is assumed to be ten days, for the first ten days, aggregate demands are randomly selected from a Uniform distribution: $\sum_{j=1}^{|R|} d_{jt} \sim \mathcal{U}(\bar{u}_0, \underline{u}_0)$ where $\bar{u}$ and $\underline{u}$ are the upper and lower bound of the Uniform distribution, respectively. Then, for the consecutive performance review periods, aggregate demand of retailers are randomly selected from an updated Uniform distribution $\sum_{j=1}^{|R|} d_{jt} \sim \mathcal{U}(\bar{u}_{\tau+1}, \underline{u}_{\tau+1})$ in which upper and lower bounds are also randomly selected: $\bar{u}_{\tau+1} = \bar{u}_\tau + \mathcal{U}(0, \delta^+)$ and $\underline{u}_{\tau+1} = \underline{u}_\tau + \mathcal{U}(0, \delta^-)$. Values of δ^+ and δ^- are selected based on the required changes in the mean and standard deviation of demand in the subsequent periods. Since mean ($\mu = \frac{\bar{u}+\underline{u}}{2}$) and standard deviation ($\sigma = \frac{(\bar{u}-\underline{u})^2}{12}$) are both dependent on the upper and lower bound parameters, randomly changing these parameters in each performance review period ensures that the generated aggregate demand would be non-stationary. Considering the initial settings as $\bar{u}_0 = 15$, $\underline{u}_0 = 14$, $\delta^+ = 1.2$, and $\delta^- = 0.8$, the generated non-stationary aggregate demands for 100 performance review periods are presented in Fig. 6.

In the next step, for each performance review period τ, the aggregate demand of each day should be forecasted based on the history of the aggregate demand up to the starting point of the current performance review period, i.e. $\{0, 1, \dots, \tau-1\}$. With respect to the forecasting process, we have used the Facebook Prophet (FP) V1.1.2. Facebook developed an automated forecasting method called 'Prophet' that strikes a suitable balance between interpretability, forecasting accuracy, and automation. Taylor and Letham (2018), the Facebook team who authored the original Prophet paper, presented evidence that FP models are capable of surpassing widely recognized automated forecasting methods, such as Auto-ARIMA and TBATS. The utilization of FP is widespread and encompasses diverse fields of application, including its implementation in forecasting epidemic patterns for COVID-19 (Wang et al., 2020), and forecasting container freight rates (Saeed et al., 2023).

FP employs a decomposable time series model, like the Generalized Additive Model (GAM), which is comprised of three primary components: trend, seasonality, and holidays/events: $y(t) = g(t) + s(t) + h(t) + \epsilon_t$. The forecasting growth of FP can be set for various types of growths such as logistic, linear, etc. Since, the trend of our non-stationary aggregate demand is obviously linear, the linear growth in considered in the FP settings: $g(t) = (k + a(t)^T\delta)t + (m + a(t)^T\gamma)$. In the linear trend model, k δ, m, γ are growth rate, rate adjustments, offset parameter, adjusting factor based on δ to make the function continuous. With respect to the seasonal effect $s(t)$, considering that P denotes the time period (e.g. $P = 30$ when the data is assumed to show monthly seasonality), a Fourier series with parameter vector of $\beta = [a_1, b_1, \dots, a_n, b_n]$ is employed: $s(t) = \sum_{n=1}^{N}\left(a_n cos(\frac{2\pi nt}{P}) + b_n sin(\frac{2\pi nt}{P})\right)$. To model the last term $h(t)$ as the event effects, assuming that event i occurs in date D_i, and $\kappa_i \in \kappa$ is the effect of event i, it can be formulated as $h(t) = Z(t)\kappa$ where $Z(t) = [\mathbf{1}(t \in D_1), \dots, \mathbf{1}(t \in D_L)]$. The forecasted aggregate demand using FP is presented in Fig. 6.

As mentioned earlier, for the case of a non-stationary aggregate demand, employing MCTS for a fixed value of base-stock level is ineffective. In other words, the base-stock level should be updated based on the forecasted demand of the periods within the next performance review period. An adaptive solution approach to dynamically updating the base-stock level is presented in Fig. 7.

Assume that the system is at the end of performance review period $n\tau$. The first step is to estimate the aggregate demand ($\dot{D}_{(n,n+1)\tau}$) for each day of the next performance review period based on the history of demand ($D_{(0,n)\tau}$). For the forecasting process, Facebook Prophet algorithm has been used. Note that for the current problem, we have assumed the

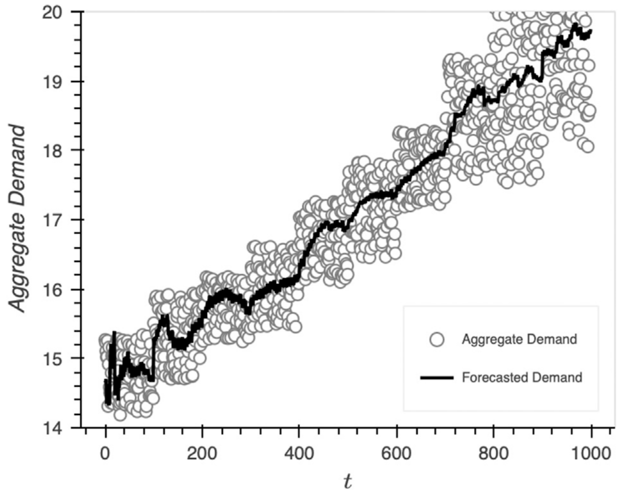


Fig. 6. Forecasted aggregate demand using the Facebook Prophet.

history of aggregate demands starts from day zero. However, after a reasonable amount of time passes, to maximize prediction accuracy, it is not necessary to consider very old periods.

After obtaining the forecasted aggregate demand, the second step would be to estimate the probability distribution function of aggregate demand using Kernel Density Estimation (KDE). KDE is a way to estimate the Probability Density Function (PDF) of a random variable in a non-parametric way. We used the Gaussian KDE which includes automatic bandwidth determination. In the third step, a number of random demands were generated and then used to get the Cumulative Density Function (CDF) of aggregate demand based on the estimated KDE.

In the fourth step, we computed the optimal base-stock level ($S^*_{(n,n+1)\tau}$) for the period stating from $n\tau$ and ending at $(n+1)\tau$ using the cumulative distribution function (CDF) of the aggregate demand obtained in the previous steps. To estimate the base-stock level, there are two approaches: (i) using the target service level ($\hat{\beta}_s$) to estimate $S^*_{(n,n+1)\tau}$, and (ii) using the critical ratio of holding and shortage costs. In this study, we employ the first method. Finally, the theoretical long term fill rate of supplier (β_s) can be estimated using the estimated base-stock level and the KDE. Details of the process are presented in Algorithm 2. Note that the entire process is repeated at the beginning of each performance review period to update the optimal base-stock level ($S^*_{(n,n+1)\tau}$) for the upcoming periods.

Algorithm 2: Estimating optimal base-stock level and supplier's fill rate based on the forecasted aggregate demand using the Facebook Prophet.

```
Inputs      : Sets as listed in Notations (section 3) and settings (section 4.5)
Parameters: $\hat{\beta}_s$: Target fill rate of supplier.
            $\dot{D}_{(n,n+1)\tau}$: Forecasted demand for period between $n\tau$ and $(n+1)\tau$.
            $D_{(0,n)\tau}$: Actual demand for periods between the first PRP and $n\tau$.
Outputs     : $\beta_s$: Fill rate of supplier.
            $S^*_{(n,n+1)\tau}$: Optimal base-stock level for the period between $n\tau$ and $(n+1)\tau$.
Initialization
$\dot{D}_{(n,n+1)\tau} \leftarrow$ Initialize a vector of actual aggregate demands from the first period up to $n\tau$.

$\dot{D}_{(n,n+1)\tau} = Facebook\_Prophet(D_{(0,n)\tau})$
$KDE = GaussianKDE(\dot{D}_{(n,n+1)\tau})$
$\dot{d} = genRandom[min(\dot{D}_{(n,n+1)\tau}), max(\dot{D}_{(n,n+1)\tau})]$
for $x$ in $\dot{d}$ do
    $CDF \xleftarrow{+} \int_{-\infty}^{x} KDE(x)\,dx$
$S^*_{(n,n+1)\tau} \leftarrow Interpol(\hat{\beta}_s,\ CDF,\ \dot{d})$
$\beta_s \leftarrow \int_{-\infty}^{S^*_{(n,n+1)\tau}} KDE(x)\,dx$

Return $S^*_{(n,n+1)\tau}$, $\beta_s$
```

The proposed solution approach for an adaptive base-stock level is employed for ten performance review periods. In each period, the optimal base-stock level (S^*) is estimated, and then MCTS is employed for the allocation decisions. In other words, the decision regarding the level of base-stock is dynamic, and the supplier selects the best base-stock level for each performance review period at the beginning of the performance review period, based on the method explained in Algorithm 2. This automated and augmented process is now fully capable of managing the non-stationary demand and meeting the fill rate with a reasonable margin. The results of the analysis are provided in Fig. 8.

As illustrated in Fig. 8a, the proposed algorithm updates the base-stock level when the forecasted aggregate demand, aiming to meet the target service level, signals that the previous base-stock level is no longer suitable. It can be also observed that the average fill rate of retailers 1 and 2 ($\bar{\beta}_1 = 88.8\%$, $\bar{\beta}_2 = 85.0\%$) confirms that the encapsulating the adaptive optimal base-stock level and the augmented MCTS mechanism into a seamless algorithm effectively manages the allocation decisions even for the case of non-stationary demand (Fig. 8b).

5. Discussions and implications

Monte Carlo Tree Search is a highly influential algorithm due to its versatility and effectiveness in tackling complex decision-making problems. It plays a crucial role in domains that require strategic planning, particularly in game theory and artificial intelligence. MCTS shines in games with a vast search space like Go, where it was a key component of AlphaGo, the first AI to defeat a world champion.

Beyond games, MCTS has widespread applications in real-world problems such as logistics, planning, and recommendation systems. By balancing the exploration of new strategies and the exploitation of known good strategies, MCTS provides an effective approach to sequential decision-making under uncertainty. It achieves this without needing domain-specific knowledge, making it a versatile and robust algorithm. Basically, the importance of MCTS lies in its wide applicability, its ability to handle complex problems with large search spaces and uncertainty, and its scalability with computational resources.

While MCTS is known for its robust performance in the absence of domain-specific knowledge, incorporating such knowledge can significantly boost its efficiency. MCTS operates by exploring the search space, but when equipped with domain knowledge, it can make more informed decisions about which branches of the search tree to explore. This can lead to a more focused search, allowing the algorithm to reach higher quality solutions more quickly.

Domain knowledge can be used to guide the selection or simulation stages of the MCTS, helping to improve both the speed and quality of the search. For example, in game-playing applications, strategies or heuristics known to be effective can be used to bias the search towards more promising moves. This targeted approach reduces unnecessary exploration and computational expenditure, increasing the algorithm's overall performance. Therefore, the integration of domain knowledge with MCTS can significantly enhance its efficiency, making it a powerful tool for complex decision-making tasks.

This study makes three theoretical contributions to allocation decisions involving a service level agreement between a supplier with capacity constraints and multiple retailers. We introduce augmenting mechanisms designed to enhance the original MCTS algorithm in the

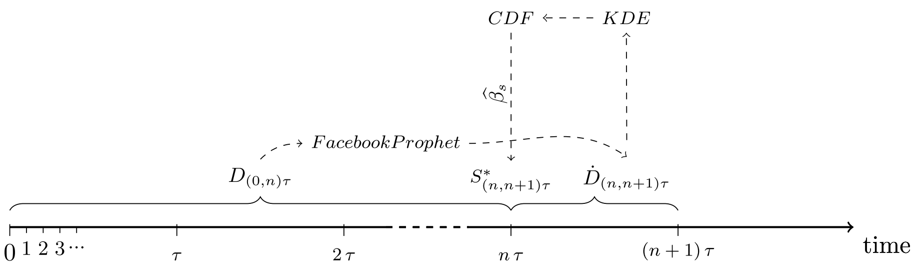
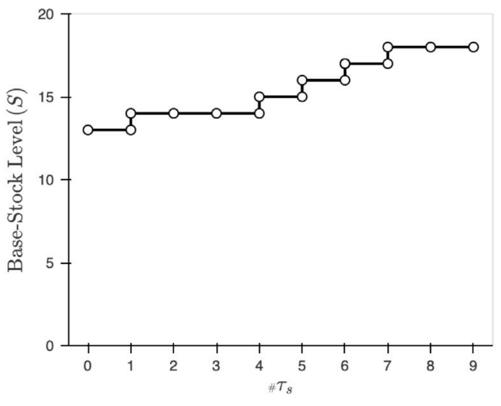
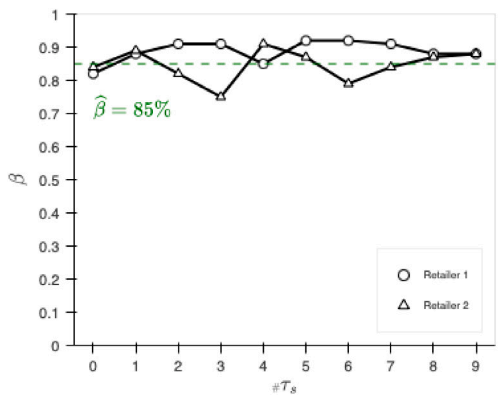


Fig. 7. Schematic overview of solution approach for an adaptive model whereby the base-stock level is dynamically updated.



(a) Adaptive base-stock level for the case of non-stationary aggregate demand.



(b) Fill rate of retailers for augmented MCTS using the adaptive base-stock level.

Fig. 8. Implementing adaptive base-stock level for managing a service level agreement using the MCTS.

following ways: (i) improving precision of the MCTS search process by reducing the solution space through identifying valid allocations; (ii) improving the MCTS selection process whereby allocations with lower distance from the realized demands are prioritized; (iii) improving allocation decisions by prioritizing the demands when the current deviation from the target service level is the largest.

The first proposed enhancement serves as a crucial mechanism to handle problems of large scale. Undeniably, one of the primary applications of MCTS lies in scenarios where the solution space is so vast that conventional search algorithms struggle to efficiently pinpoint optimal or near-optimal solution. However, when computational resources are limited, or when a solution must be found within a strict time frame (as in games or emergency operations), any strategy to reduce the solution space size leads to an improvement in the quality of the identified solution. Indeed, in supply chain management, problems often exhibit immense complexity and scale, highlighting the critical importance of minimizing the solution space for MCTS to deliver superior outcomes. When it comes to allocation decisions, complexity of the problem can increase exponentially as the number of retailers and/or the number of SKUs grows. Therefore, the proposed mechanism to screen sub-optimal allocations becomes critical. This approach allows MCTS to function effectively even when handling large solution spaces in the supply chain domain, ensuring the delivery of near-optimal or optimal results despite the increased scale and complexity.

Beyond the initial screening process aimed at limiting the solution space, this study also proposes improvements to the selection process that traditionally employed in MCTS. In allocation decisions, it is not rational to incur shortages when stock is available. This observation informs the second proposed augmenting mechanism. Under this mechanism, any potential allocation that does not fully utilize available stock is given lower priority in the selection process.

The third mechanism has been specifically designed to accommodate the unique requirements of service level agreements, particularly the potential for penalty costs when the target fill rate is not met. A myopic allocation policy might deem the distribution of stocks proportional to demand magnitudes as an optimal solution. However, this approach does not necessarily minimize the total cost, which includes potential penalties for not meeting service levels. Therefore, in each stage of allocation decisions, deviation of retailers' fill rate from the target service level should be measured and then the allocations are adjusted such that the retailer with higher deviation from the target fill rate receives more stock in the current period. The proposed dynamic and adaptive mechanism of adjusting the allocations based on the distance from the contracted service level provides an opportunity to minimize the distance between the proposed solution and offline optimal allocations.

The final contribution of this study involves the design of an adaptive base-stock algorithm. When combined with the enhanced MCTS formulated in this work, it provides a powerful mechanism that can effectively manage service level agreements, even in the face of non-stationary demands. In the proposed solution, demands of the next performance review period are forecasted using the Facebook Prophet, and then the optimal base-stock level is computed based on the estimated KDE and its corresponding CDF.

From a practical standpoint, the proposed solution approaches in this study carry significant managerial implications. As the solution space for allocation decisions under service level agreements inevitably expands due to the proliferation of allocation problem aspects, these problems can quickly fall prey to the curse of dimensionality. In such cases, conventional exact methods would fail to manage the scale of the problem effectively. The proposed solution, which can provide near-optimal allocations, is further enhanced by the development of adaptive base-stock levels. This combination enables the augmented MCTS to effectively cope with non-stationary demands.

6. Conclusion

In conclusion, the Monte Carlo Tree Search algorithm has demonstrated its versatility and effectiveness in tackling complex decision-making problems, making it a valuable tool in supply chain analytics. The proposed enhancements in this study, including mechanisms to reduce solution space, prioritize allocations, and consider deviations from service level agreements, offer practical solutions for allocation decision problems, in particular for the service level agreements.

Despite the various theoretical and practical implications, the findings of this study can be further improved when considering its limitations. For example, future studies could focus on developing an updated algorithm that accommodates nonidentical penalty costs for retailers based on their specific service level agreement. Additionally, improvements can be made to the augmented MCTS algorithm to handle cases where the lengths of performance review periods vary among retailers. Another area for improvement lies in addressing scenarios where the length of the performance review period is either too small or too large. Customizing the algorithm to handle these special cases and effectively manage the increased solution space is an important avenue for future research.

CRediT authorship contribution statement

Masih Fadaki: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Atie Asadikia:** Writing – review & editing, Writing – original draft, Formal analysis, Data curation, Conceptualization.

Data availability

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

References

Abbasi, B., Fadaki, M., Hosseinifard, Z., Jahani, H., Thomas, D.J., 2022. Allocation policies to fulfil heterogeneous service requirements under resource pooling. Decis. Sci. 53 (2), 277–319.

Abramson, B., 1986. Thesis Proposal: The Expected-Outcome Model of Two-Player Games (Ph.D. thesis). Columbia University.

Axsäter, S.a., 2015. Inventory Control, third ed. Springer International Publishing: Imprint: Springer, Cham, 2015. ed..

Barletta, C., Garn, W., Turner, C., Fallah, S., 2023. Hybrid fleet capacitated vehicle routing problem with flexible Monte–Carlo tree search. Int. J. Syst. Sci. Oper. Logist. 10 (1), 2102265.

Bianco, P., Lewis, G.A., Merson, P., 2008. Service Level Agreements in Service-Oriented Architecture Environments. Carnegie Mellon University, Software Engineering Institute, pp. 2–5.

Brügmann, B., 1992. Monte Carlo Go. Technical Report, Technical Report, Physics Department, Syracuse University, Syracuse, NY.

Bughin, J., Seong, J., Manyika, J., Chui, M., Joshi, R., 2018. Notes from the AI frontier: Modeling the impact of AI on the world economy. URL: https://www.mckinsey.com/featured-insights/artificial-intelligence/notes-from-the-ai-frontier-modeling-the-impact-of-ai-on-the-world-economy.

Carvalho, T.P., Soares, F.A., Vita, R., Francisco, R.d.P., Basto, J.P., Alcalá, S.G., 2019. A systematic literature review of machine learning methods applied to predictive maintenance. Comput. Ind. Eng. 137, 106024.

Chen, C.-M., Thomas, D.J., 2018. Inventory allocation in the presence of service-level agreements. Prod. Oper. Manage. 27 (3), 553–577.

Coulom, R., 2006. Efficient selectivity and backup operators in Monte-Carlo tree search. In: Computers and Games: 5th International Conference, CG 2006, Turin, Italy, May 29-31, 2006. Revised Papers 5. Springer, pp. 72–83.

Edelkamp, S., Gath, M., Greulich, C., Humann, M., Herzog, O., Lawo, M., 2016. Monte-Carlo tree search for logistics. In: Commercial Transport: Proceedings of the 2nd Interdiciplinary Conference on Production Logistics and Traffic 2015. Springer, pp. 427–440.

Ellram, L.M., Tate, W.L., Billington, C., 2004. Understanding and managing the services supply chain. J. Supply Chain Manage. 40 (3), 17–32.

Fadaki, M., Abareshi, A., Far, S.M., Lee, P.T.-W., 2022a. Multi-period vaccine allocation model in a pandemic: A case study of COVID-19 in Australia. Transp. Res. E 161, 102689.

Fadaki, M., Abbasi, B., Chhetri, P., 2022b. Quantum game approach for capacity allocation decisions under strategic reasoning. Comput. Manag. Sci. 19 (3), 491–512.

Goutham, M., Stockar, S., 2023. Recomputing solutions to perturbed multi-commodity pickup and delivery vehicle routing problems using Monte Carlo tree search. arXiv preprint arXiv:2304.11444.

Islam, S., Amin, S.H., Wardley, L.J., 2021. Machine learning and optimization models for supplier selection and order allocation planning. Int. J. Prod. Econom. 242, 108315.

Jahani, H., Abbasi, B., Hosseinifard, Z., Fadaki, M., Minas, J.P., 2021. Disruption risk management in service-level agreements. Int. J. Prod. Res. 59 (1), 226–244.

Katok, E., Thomas, D., Davis, A., 2008. Inventory service-level agreements as coordination mechanisms: The effect of review periods. Manuf. Serv. Oper. Manage. 10 (4), 609–624.

Kaynov, I., van Knippenberg, M., Menkovski, V., van Breemen, A., van Jaarsveld, W., 2024. Deep reinforcement learning for one-warehouse multi-retailer inventory management. Int. J. Prod. Econom. 267, 109088.

Kloos, K., Pibernik, R., 2020. Allocation planning under service-level contracts. European J. Oper. Res. 280 (1), 203–218.

Kocsis, L., Szepesvári, C., 2006. Bandit based monte-carlo planning. In: Machine Learning: ECML 2006: 17th European Conference on Machine Learning Berlin, Germany, September 18-22, 2006 Proceedings 17. Springer, pp. 282–293.

Kwon, I.-H., Kim, C.O., Jun, J., Lee, J.H., 2008. Case-based myopic reinforcement learning for satisfying target service level in supply chain. Expert Syst. Appl. 35 (1–2), 389–397.

Li, K., Deng, Q., Zhang, L., Fan, Q., Gong, G., Ding, S., 2021. An effective MCTS-based algorithm for minimizing makespan in dynamic flexible job shop scheduling problem. Comput. Ind. Eng. 155, 107211.

Oblicore, 2007. 2007 Service-Level Management Survey: Results, Trends and Analysis. White Paper, Oblicore Inc., URL: https://www.pmi.it/app/uploads/2018/02/000172-7cFlZQ.pdf.

Pournader, M., Ghaderi, H., Hassanzadegan, A., Fahimnia, B., 2021. Artificial intelligence applications in supply chain management. Int. J. Prod. Econ. 241, 108250.

Powell, W.B., 2011. Approximate Dynamic Programming: Solving the Curses of Dimensionality, Second Ed., second ed. In: Wiley Series in Probability and Statistics, John Wiley & Sons, Hoboken, N.J.

Preil, D., Krapp, M., 2022. Artificial intelligence-based inventory management: a Monte Carlo tree search approach. Ann. Oper. Res. 1–25.

Puterman, M.L., 2005. Markov decision processes discrete stochastic dynamic programming. In: Markov Decision Processes Discrete Stochastic Dynamic Programming. In: Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics Section, John Wiley & Sons, New York.

Qiu, H., Wang, S., Yin, Y., Wang, D., Wang, Y., 2022. A deep reinforcement learning-based approach for the home delivery and installation routing problem. Int. J. Prod. Econ. 244, 108362.

Saeed, N., Nguyen, S., Cullinane, K., Gekara, V., Chhetri, P., 2023. Forecasting container freight rates using the prophet forecasting method. Transp. Policy 133, 86–107.

Saqlain, M., Ali, S., Lee, J., 2022. A Monte-Carlo tree search algorithm for the flexible job-shop scheduling in manufacturing systems. Flexible Serv. Manuf. J. 1–24.

Senington, R., 2022. The multiple uses of Monte-Carlo tree search. In: 10th Swedish Production Symposium (SPS2022), Skövde, April 26–29 2022. IOS Press, pp. 713–724.

Senington, R., Schmidt, B., Syberfeldt, A., 2021. Monte Carlo tree search for online decision making in smart industrial production. Comput. Ind. 128, 103433.

Silver, E.A., Pyke, D.F., Peterson, R., et al., 1998. Inventory Management and Production Planning and Scheduling, vol. 3, Wiley, New York.

Taylor, S.J., Letham, B., 2018. Forecasting at scale. Amer. Statist. 72 (1), 37–45.

Thomas, D.J., 2005. Measuring item fill-rate performance in a finite horizon. Manuf. Serv. Oper. Manage. 7 (1), 74–80. http://dx.doi.org/10.1287/msom.1040.0064, URL: http://pubsonline.informs.org/doi/abs/10.1287/msom.1040.0064.

Wang, Y., Minner, S., 2024. Deep reinforcement learning for demand fulfillment in online retail. Int. J. Prod. Econom. 269, 109133.

Wang, P., Zheng, X., Li, J., Zhu, B., 2020. Prediction of epidemic trends in COVID-19 with logistic model and machine learning technics. Chaos Solitons Fractals 139, 110058.