European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Production, Manufacturing, Transportation and Logistics

Stochastic dual dynamic programming approach for cash-flow inventory problems with overdraft

Zhen Chen [a,*], Thomas W. Archibald [b]

[a] Business School, Brunel University London, Uxbridge, UB8 3PH, United Kingdom
[b] Business School, University of Edinburgh, Edinburgh, EH8 9JS, United Kingdom

ARTICLE INFO

Keywords:
Cash-flow inventory
SDDP
Overdraft
Lead time

ABSTRACT

We examine a multi-product cash-flow inventory problem that accounts for lead times and uncertain demand. Our analysis includes a specific type of financing — overdrafts — where retailers facing cash constraints can leverage overdrafts to manage unexpected cash shortfalls. To address this issue, we propose a stochastic programming model and solve it using stochastic dual dynamic programming (SDDP). Additional auxiliary variables are introduced in the sub-problems to facilitate the construction of cuts in the presence of lead times. To enhance computational efficiency, we provide two techniques: removing the duplicate added constraints in the model and exploiting dual value similarities of the constraints across sub-problems. Numerical experiments demonstrate that SDDP can solve the problem with small optimality gaps compared to the values obtained from stochastic dynamic programming, and the proposed acceleration strategies significantly reduce computation time with minor impact on solution quality.

1. Introduction

Cash-flow is often regarded as the lifeblood of a business, and small and medium-sized enterprises (SMEs) increasingly factor cash availability into their operational decisions. Unlike larger companies, SMEs face more significant challenges in securing external funding, primarily due to their limited sales volumes and operational scales. Additionally, SMEs typically lack the cash reserves needed to absorb major losses. One report revealed that 38% of failed startups cited running out of cash or an inability to raise additional capital as the main reason for their downfall (CBInsights, 2021). Another study found that nearly three in five UK SMEs have faced cash-flow issues (QuickBooks, 2019).

To address these challenges, many financing services have emerged in the market to help SMEs address cash shortages. These include, but are not limited to, loans, overdrafts, credit cards, leasing options, and government grants. Some Chinese e-commerce platforms also offer order-based loans specifically for online retailers (Chen & Zhang, 2021). A report by the British Business Bank found that 30% of SMEs sought external financing in the past three years (Ipsos, 2024), with bank overdrafts and credit cards being the two most popular services.

Extensive literature already exists on cash-flow management or supply chain financing in the context of inventory problems. However, much of this research has concentrated on single or two stage decision problems (e.g., Cao et al., 2024) or on single-product scenarios (e.g., Chao et al., 2008). These limitations highlight the need for approaches that incorporate more realistic settings and mathematical programming methods. The primary contributions of this study are as follows.

- We address a cash-flow inventory problem involving multiple products, lead times, uncertain demand, and overdraft options, and formulate it as a multi-stage stochastic programming model.
- To solve the model, we apply stochastic dual dynamic programming (SDDP). Given the complexities introduced by lead times, we incorporate auxiliary variables in the sub-problems to facilitate the computation of cuts in SDDP's backward pass.
- We develop two acceleration techniques for SDDP to improve computational efficiency: one eliminates duplicate constraints added during iterations, and the other takes advantage of the similarities in dual values across subproblems.
- We evaluate the optimality gaps of SDDP and the proposed accelerations by comparing them with values obtained via stochastic dynamic programming (SDP), and assess their performance through extensive numerical experiments.

The remainder of this paper is organized as follows. Section 2 reviews the related literature and Section 3 presents the problem description. Section 4 formulates the multi-stage stochastic programming model. Section 5 introduces the SDDP approach and discusses its convergence properties and bounds for our problem, while Section 6

* Corresponding author.
E-mail addresses: zhen.chen@brunel.ac.uk, chen.zhen5526@gmail.com (Z. Chen), t.archibald@ed.ac.uk (T.W. Archibald).

https://doi.org/10.1016/j.ejor.2026.04.003
Received 5 August 2025; Accepted 2 April 2026
Available online 12 April 2026

presents two acceleration techniques for SDDP. The computational study and results are discussed in Section 7. Finally, conclusions and directions for future research are outlined in Section 8.

2. Literature review

We focus our discussion on key studies relevant to our problem, specifically cash-flow inventory issues and SDDP. We review each area as follows.

2.1. Cash-flow inventory problems

Cash-flow and financial services have been widely studied in the production and inventory management literature. For instance, Buzacott and Zhang (2004) highlighted the importance of jointly considering operational and financial decisions using the newsvendor model, and Cao et al. (2024) examined risk-sharing and financing strategies in an inventory management problem. However, many of these studies either overlook the impact of cash constraints on inventory decisions or limit their analysis to single/double stage models.

Regarding multi-stage cash-flow inventory problems, Archibald et al. (2002) examined inventory policies aimed at maximizing survival probability. Chao et al. (2008) characterized the optimal inventory control policy and developed an algorithm to address the self-financing retailer problem. Gong et al. (2014) and Katehakis et al. (2016) extended this work to situations involving short-term financing. Luo and Shang (2019) investigated a single-product cash-flow inventory problem with trade credit, aiming to maximize working capital at the end of the planning horizon. Chen and Rossi (2021) proposed an inventory control policy for addressing cash constraints with fixed ordering costs. Fu et al. (2021) explored inventory-based financing and partially characterized the optimal inventory control policy. Kajjoune et al. (2023) applied dynamic programming to solve a dynamic lot-sizing problem involving short-term financing and external deposits. Chen and Archibald (2024) employed scenario trees and sample average approximation to tackle a survival maximization problem with a joint chance constraint.

Despite the extensive literature on inventory problems with cash constraints, no studies have yet proposed a model that simultaneously considers multiple stages, multiple products, lead times, overdraft options, and uncertain demand. As these complexities are commonly found in practical situations, this paper aims to address this gap and proposes a model that can help retailers manage and optimize inventory under financial constraints. This model can only be useful if it can be solved efficiently. Therefore, this paper develops an efficient solution method based on SDDP, a well-established methodology for multiple stage dynamic optimization problems.

2.2. Stochastic dual dynamic programming

Pereira and Pinto (1991) originally proposed the SDDP method, integrating sampling into the nested decomposition algorithm developed by Birge (1985), which extended the two-stage L-shaped algorithm by Van Slyke and Wets (1969) to a multi-stage setting. Since then, numerous discussions, applications, and extensions of this method have emerged. Philpott and Guan (2008) proved the almost sure convergence of SDDP under mild conditions, while Shapiro (2011) and Lan (2022) further examined its statistical properties and convergence rate, respectively. Additionally, Shapiro et al. (2013) extended SDDP for both risk-neutral and risk-averse applications, as implemented by Soares et al. (2017) in hydrothermal planning. De Matos et al. (2015) provided and compared various computational accelerations to SDDP. Löhndorf and Shapiro (2019) compared the time series approach and Markov chain approximation for the Markovian type dependence structure in SDDP for a power planning problem. Stochastic Dual Dynamic Integer Programming (SDDiP) was developed by Zou et al. (2019) for multi-stage stochastic programming with integer variables. Thevenin et al. (2022) employed SDDP to compare various improvements in a multiechelon lot sizing problem. Bhattacharya et al. (2023) applied concave penalty functions in SDDP. More recently, Kiszka and Wozabal (2025) solved a power flow problem by dual convex semi-definite programming and SDDP, while Hole et al. (2025) applied SDDP to solve a capacity planning problem.

Although the SDDP method is recognized as an effective method for solving multiple stage dynamic optimization problems, it has not yet been applied to inventory management problems with cash-flow constraints. Given the practical significance of this potential application of SDDP, we are motivated to explore the effectiveness of SDDP for cash-flow inventory problems and to develop techniques to accelerate the computation, making the method suitable for the practical setting that we model.

3. Problem description

For clarity, the primary notations used throughout this paper are summarized in Table 1. Additional notations will be introduced as necessary.

A retailer manages a portfolio of N products over a planning horizon of T stages. At the beginning of the planning horizon, the retailer holds an initial inventory of product n, denoted by I_{n0}. The selling price and unit variable ordering cost of product n during stage t are represented by p_{nt} and c_{nt}, respectively. The order quantity for product n at stage t is represented by q_{nt} and is delivered after a lead time of L_n stages. At the beginning of stage t, there will be an outstanding order for q_{nt-L_n} units of product n due to be delivered during the stage. Ordering decisions for each stage are made after observing the initial inventory position but before the demand for that stage is realized. Demand for each product in each stage is uncertain and independent, with D_{nt} denoting the demand for product n at stage t. The retailer should fulfill demand to the greatest extent possible using inventory available at a stage. Any demand that cannot be satisfied results in lost sales. The end-of-stage inventory of product n at stage t is denoted by I_{nt}, while any unmet demand for product n at stage t is recorded as B_{nt}. The inventory flow for product n is governed by the following equation:

$$I_{nt} - B_{nt} = I_{nt-1} + q_{nt-L_n} - D_{nt}. \tag{1}$$

Although Eq. (1) does not define I_{nt} and B_{nt} uniquely, the optimization will ensure that at least one of these variables is equal to 0 to minimize the number of lost sales.

In each stage, the retailer incurs overhead costs, H_t, due to factors such as rent and salaries. To address unexpected cash shortages, the retailer relies on an overdraft facility, paying interest only on the overdrawn balance. The interest rate and overdraft limit are determined based on terms outlined by financial institutions such as Barclays Bank (Barclays, 2025) and HSBC (HSBC, 2025).

The retailer's initial cash balance, after deducting a fixed overdraft application fee, is C_0. Let C_t denote the end-of-stage cash balance at stage t. In each stage, we assume that interest is calculated after the retailer has paid ordering costs and overhead costs. The cash balance after these payments is $C_{t-1} - H_t - \sum_{n\in\mathcal{N}} c_{nt}q_{nt}$, where $\mathcal{N} = \{1, \ldots, N\}$ represents the set of product indices. If the resulting balance is positive, the retailer earns deposit interest at a rate r_0. If the balance is negative, overdraft interest is charged at a rate r_1. The limit of the overdraft facility is denoted by U. The retailer's cash balance can fall into one of three intervals: $(-\infty, -U)$, $[-U, 0)$, or $[0, +\infty)$. To simplify computations and ensure the existence of feasible solutions for any demand realization in the stochastic programming model (i.e., relatively complete recourse, Shapiro, 2011), we assume that the retailer incurs a significantly high interest rate r_2 on any balance exceeding the overdraft limit (i.e. when $C_t < -U$). It is evident that $r_0 < r_1 < r_2$.

Let W_t^m $(m = 0, 1, 2)$ represent the cash balance allocated to each interval. For instance, assuming an overdraft limit of £20,000: (1) if the cash balance is 500, then $W_t^0 = 500$ and $W_t^1 = W_t^2 = 0$; (2) if the

Table 1
Main notations in the paper.

Indices:	
t	Index of a stage, $t = 1, \ldots, T$.
n	Index of a product, $n = 1, \ldots, N$.
m	Index of a cash interval, $m = 0, \ldots, 2$.
$\mathcal{T}$	Set of stage indices, $\mathcal{T} = \{1, \ldots, T\}$.
$\mathcal{N}$	Set of product indices, $\mathcal{N} = \{1, \ldots, N\}$.
Constants:	
C_0	Initial cash at the beginning of the planning horizon.
I_{n0}	Initial inventory of product n at the beginning of the planning horizon.
q_{nt-i}	Outstanding order for product n at the beginning of stage t due for delivery in stage $t - i + L_n$ $(0 < i \leq L_n)$.
p_{nt}	Selling price of product n at stage t.
c_{nt}	Unit variable ordering cost for product n at stage t.
v_n	Unit salvage value for inventory of product n at the end of the planning horizon.
H_t	Overhead costs at stage t (e.g., wages or rents).
L_n	Ordering lead time for product n.
U	Overdraft limit.
r_m	Interest rate for the kth cash interval where $r_0 < r_1 < r_2$.
Uncertain variable:	
D_{nt}	Demand for product n at stage t.
Decision variables:	
q_{nt}	Order quantity for product n at stage t.
I_{nt}	Inventory level for product n at the end of stage t.
B_{nt}	Quantity of unsatisfied demand for product n in stage t.
C_t	Cash balance at the end of stage t.
W_t^m	Quantity of cash allocated to the mth cash interval at the end of stage t.

cash balance is −2,000, then $W_t^0 = 0$, $W_t^1 = 2000$, and $W_t^2 = 0$; (3) if the cash balance is −25,000, then $W_t^0 = 0$, $W_t^1 = 20,000$, and $W_t^2 = 5000$. The relationship between the cash balance, $C_{t-1} - H_t - \sum_{n\in\mathcal{N}} c_{nt}q_{nt}$, and the variables W_t^0, W_t^1, and W_t^2 is as follows:

$$C_{t-1} - H_t - \sum_{n\in\mathcal{N}} c_{nt}q_{nt} = W_t^0 - W_t^1 - W_t^2. \tag{2}$$

Based on Eq. (2), the cash-flow function for the problem can be written as:

$$\begin{aligned} C_t &= C_{t-1} - H_t - \sum_{n\in\mathcal{N}} c_{nt}q_{nt} - \sum_{m=1}^{2} r_k W_t^m + r_0 W_t^0 + \sum_{n\in\mathcal{N}} p_{nt}(D_{nt} - B_{nt}) \\ &= \sum_{n\in\mathcal{N}} p_{nt}(D_{nt} - B_{nt}) + (1 + r_0)W_t^0 - \sum_{m=1}^{2}(1 + r_k)W_t^m. \end{aligned} \tag{3}$$

In the equation above, $\sum_{m=1}^{2} r_m W_t^m$ and $r_0 W_t^0$ represent the interest paid and earned respectively, while $\sum_{n\in\mathcal{N}} p_{nt}(D_{nt} - B_{nt})$ denotes the total sales revenue earned during stage t. Although Eq. (2) does not define W_t^m uniquely, the optimization will ensure that either $W_t^0 = 0$ or $W_t^1 = W_t^2 = 0$ and either $W_t^2 = 0$ or $W_t^1 = U$ to minimize the interest paid. It should be noted that certain overdraft services provide an interest-free limit. If the cash balance becomes negative but stays above this interest-free threshold, no interest charges are incurred by customers. In this case, an additional decision variable can be introduced to represent the cash balance allocated to the interest-free portion of the overdraft.

Consistent with most cash-flow inventory modeling in the literature (e.g., Chao et al., 2008), inventory holding costs and stock-out penalty costs are not considered in the model. This omission is based on the assumption that holding costs are accounted for implicitly as the opportunity cost of capital, while stock-out penalty costs are captured as lost revenue. However, the stochastic programming models presented in the following sections, designed to address the problem, can also accommodate scenarios that include explicit holding costs or stock-out penalty costs.

Any remaining inventory of product n at the end of the planning horizon has a salvage value of v_n per unit. The final cash balance is given by:

$$C_{T+1} = C_T + \sum_{n\in\mathcal{N}} v_n I_n. \tag{4}$$

The objective is to maximize the expected final cash increment, or equivalently, the profit. Alternatively, this problem can be viewed as minimizing the expected total costs over the planning horizon, treating revenue earnings and deposit interest as cost offsets.

4. Multi-stage stochastic programming model

This section develops the stochastic programming model for the problem. To simplify the presentation, the objectives in these models are expressed as minimization problems. Additional notations are provided in detail in Table 2. In general, at decision stage t $(1 \leq t \leq T + 1)$, the demand dependent decisions for stage $t - 1$ (i.e. I_{nt-1}, B_{nt-1} and C_{t-1}) are calculated and the ordering and cash balance decisions for stage t (i.e. q_{nt} and W_t^k) are taken. Stages 1 and $T + 1$ are treated as special cases.

An important assumption of the SDDP method is that the random data process is stage-wise independent. However, SDDP can also accommodate cross-stage dependencies in certain cases, like an autoregressive model, by incorporating additional state variables into the model (Shapiro, 2011). Fig. 1 illustrates a scenario tree formulation for the demand realizations of two products over three stages. Each node except the first one represents a demand realization for all products (denoted as ω_t) at a given stage, with the demand values displayed within the circles. Specifically, let $d_n^{\omega_t}$ represent the realized demand for product n at stage t. Due to the stage-wise independence, all nodes at stage t are connected to nodes in the same demand realization set at the next stage (denoted as Ω_{t+1}) via branches with identical conditional probabilities (denoted as $\Pr(\omega_{t+1})$), as indicated on the edges linking the nodes.

We use bold letters to represent sets. For instance, $\mathbf{q}_{\mathcal{N}t}$ denotes the set of order quantities q_{nt} for all $n \in \mathcal{N}$ at stage t, and $\mathbf{W}_t$ represents the set $\{W_t^0, W_t^1, W_t^2\}$. The notation $\widehat{X}_t$ is used to denote the value of a decision variable X chosen at stage t. The sub-problems for each decision stage are formulated as the following linear programming (LP) models.

Due to the assumption of non-zero lead time, it is necessary to transfer information about outstanding orders across consecutive stages of the model. We introduce q_{nt}^i to represent the quantity of product n ordered i stages before stage t for $1 \leq i \leq L_n$. Note that $q_{nt}^i = q_{nt-i}$ and

Table 2
Additional notations for the stochastic programming/SDDP model.

ω_t	A demand realization for stage t.
$d_n^{\omega_t}$	The demand for product n at stage t under demand realization ω_t.
Ω_t	The set of all demand realizations at stage t, i.e., $\omega_t \in \Omega_t$.
$\Pr(\omega_t)$	Probability of demand realization ω_t occurring.
$\hat{X}_t$	The value of decision variable X_t (e.g., q_{nt}, I_{nt}, B_{nt}, C_t, W_t^m) chosen in stage t.
$\mathbf{q}_{\mathcal{N}t}$	The set $\{q_{1t}, \ldots, q_{Nt}\}$.
$\mathbf{W}_t$	The set $\{W_t^0, W_t^1, W_t^2\}$.
q_{nt}^i	The outstanding order for product n at stage t placed i stages earlier ($1 \le i \le L_n$).
$\mathbf{q}_{\mathcal{N}t}^{\mathcal{L}}$	The set $\{q_{1t}^1, \ldots, q_{1t}^{L_1}, \ldots, q_{Nt}^1, \ldots, q_{Nt}^{L_N}\}$.
$Q_t(\cdot)$	Optimal total cost from stage t to $T+1$ for the parameter values given in the parentheses.
$\Theta_t^k(\cdot)$	Right hand side of the cut constraint for stage t generated in the kth iteration.
θ_t	Approximation to $\mathbb{E}[Q_t(\cdot)]$ constrained by supporting hyperplane cuts.

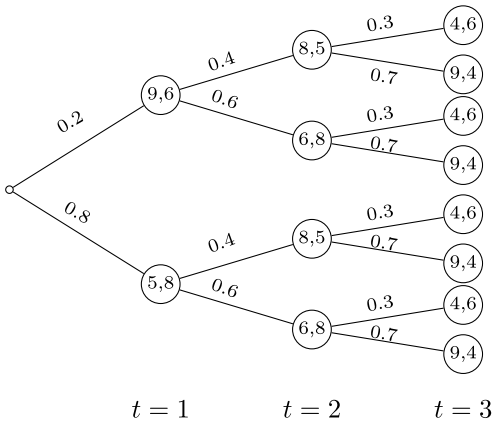

Fig. 1. A scenario tree example.

that $q_{nt}^{L_n}$ units of product n will be delivered during stage t. We use $\mathbf{q}_{\mathcal{N}t}^{\mathcal{L}}$ to represent the set of order quantities for all the outstanding orders at the beginning of stage t.

4.1. First-stage sub-problem

In the first stage, the retailer decides the order quantity q_{n1} for each product before the demands for products are known. W_1^0, W_1^1, W_1^2 are computed after paying the ordering and overhead costs.

$$Q_1 = \min \quad H_1 + \sum_{n\in\mathcal{N}} c_{n1}q_{n1} + \sum_{m=1}^{2} r_m W_1^m - r_0 W_1^0 + \mathbb{E}_{\omega_1}[Q_2(\mathbf{q}_{\mathcal{N}1}, \mathbf{q}_{\mathcal{N}1}^{\mathcal{L}}, \mathbf{W_1}, I_0, \omega_1)] \tag{5}$$

$$\text{s.t.} \quad W_1^1 \le U \tag{6}$$

$$-\sum_{n\in\mathcal{N}} c_{n1}q_{n1} - W_1^0 + \sum_{m=1}^{2} W_1^m = H_1 - C_0 \tag{7}$$

$$q_{n1} \ge 0, W_1^m \ge 0 \qquad \forall n, \forall m. \tag{8}$$

The objective function (5) accounts for overhead costs, ordering costs, interest, and the expected future optimal cost $\mathbb{E}_{\omega_1}[Q_2(\mathbf{q}_{\mathcal{N}1}, \mathbf{q}_{\mathcal{N}1}^{\mathcal{L}}, \mathbf{W_1}, I_0, \omega_1)]$. Constraint (6) ensures that W_1^1 does not exceed the overdraft limit. Constraint (7) establishes the relationship between the cash balance and W_1^m ($m = 0, \ldots, 2$). Finally, constraint (8) enforces the non-negativity of the decision variables. Note that the optimization ensures that the correct values are assigned to the values W_1^m without additional constraints because $r_0 < r_1 < r_2$.

4.2. Sub-problems for stage $t \in \{2, 3, \ldots, T\}$

The formulation of the stage $t \in \{2, \ldots, T\}$ sub-problem is defined as follows:

$$Q_t(\hat{\mathbf{q}}_{\mathcal{N}t-1}, \hat{\mathbf{q}}_{\mathcal{N}t-1}^{\mathcal{L}}, \widehat{\mathbf{W}}_{t-1}, \hat{\mathbf{I}}_{\mathcal{N}t-2}, \omega_{t-1}) =$$

$$\min \quad H_t + \sum_{n\in\mathcal{N}} c_{nt}q_{nt} + \sum_{m=1}^{2} r_m W_t^m - r_0 W_t^0 - \sum_{n\in\mathcal{N}} p_{nt-1}(d_n^{\omega_{t-1}} - B_{nt-1}) + \mathbb{E}_{\omega_t}[Q_{t+1}(\mathbf{q}_{\mathcal{N}t}, \mathbf{q}_{\mathcal{N}t}^{\mathcal{L}}, \mathbf{W_t}, \mathbf{I}_{\mathcal{N}t-1}, \omega_t)] \tag{9}$$

$$\text{s.t.} \quad I_{nt-1} - B_{nt-1} = \hat{I}_{nt-2} + \hat{q}_{nt-1}^{L_n} - d_n^{\omega_{t-1}} \qquad \forall n \tag{10}$$

$$C_{t-1} + \sum_{n\in\mathcal{N}} p_{nt-1}B_{nt-1} = (1+r_0)\widehat{W}_{t-1}^0 - \sum_{m=1}^{2}(1+r_m)\widehat{W}_{t-1}^m + \sum_{n\in\mathcal{N}} p_{nt-1}d_n^{\omega_{t-1}} \tag{11}$$

$$q_{nt}^1 = \hat{q}_{nt-1} \qquad \forall n \tag{12}$$

$$q_{nt}^i = \hat{q}_{nt-1}^{i-1} \qquad \forall i = 2, \ldots, L_n, \forall n \tag{13}$$

$$W_t^1 \le U \tag{14}$$

$$C_{t-1} - \sum_{n\in\mathcal{N}} c_{nt}q_{nt} - W_t^0 + \sum_{m=1}^{2} W_t^m = H_t \tag{15}$$

$$q_{nt} \ge 0, q_{nt}^i \ge 0, W_t^m \ge 0, I_{nt-1} \ge 0, B_{nt-1} \ge 0 \qquad \forall n, \forall m, \forall i. \tag{16}$$

In the objective function (9), the sales revenue from stage $t-1$ is accounted for by the term $\sum_{n\in\mathcal{N}} p_{nt-1}(d_{nt-1}^{\omega} - B_{nt-1})$ and the other terms represent the various costs and interest as in the stage 1 sub-problem. The inventory flow equation and cash-flow equation are defined by (10) and (11), respectively. Constraints (12)–(13) establish the relationship between order quantity variables across consecutive stages. Constraints (14)–(15) define the connection between W_t^m and the cash balance $C_{t-1} - H_t - \sum_{n\in\mathcal{N}} c_{nt}q_{nt}$. Finally, constraint (16) ensures the non-negativity of variables.

4.3. Last stage sub-problem

In the last stage ($T+1$), the sub-problem only needs to compute the inventory level at the end of the planning horizon, and the objective only involves the sales revenue at stage T and the salvage value of the final inventory.

$$Q_{T+1}(\hat{\mathbf{q}}_{\mathcal{N}T}, \hat{\mathbf{I}}_{\mathcal{N}T-1}, \omega_T) = \min \quad -\sum_{n\in\mathcal{N}} \left[p_{nT}(d_n^{\omega_T} - B_{nT}) + v_n I_{nT}\right] \tag{17}$$

$$\text{s.t.} \quad I_{nT} - B_{nT} = \hat{I}_{nT-1} + \hat{q}_{nT}^{L_n} - d_n^{\omega_T} \qquad \forall n \tag{18}$$

$$I_{nT} \ge 0, B_{nT} \ge 0 \qquad \forall n. \tag{19}$$

The objective function (17) minimizes the negative of the total revenue in the last stage. The final stage's inventory level is computed in the constraints.

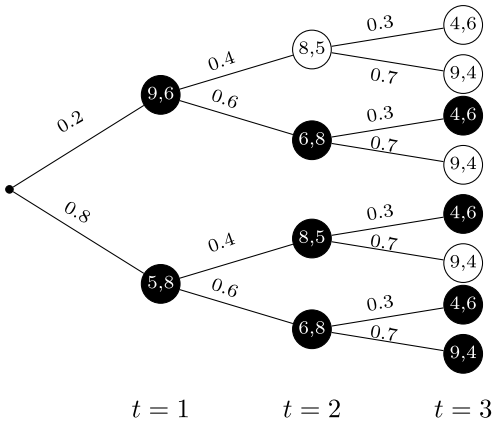

Fig. 2. A sampled scenario tree with sampled nodes shown in black. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. SDDP computation

The core concept of SDDP is to work with a limited number of sampled scenarios from a large scenario tree, rather than exhaustively exploring all scenarios during the forward pass. Supporting hyperplanes are added to approximate the cost-to-go functions in the backward computation, and this process is repeated until certain stopping criteria are met. During each iteration, the forward step generates a feasible solution, while the backward step creates a hyperplane cut for each cost-to-go function. Fig. 2 illustrates an example of a sampled scenario tree derived from the original scenario tree in Fig. 1. The sampled nodes, represented in black, result in 4 sampled scenarios out of the original 8 scenarios.

5.1. Forward pass

In each iteration's forward step, the SDDP algorithm samples a set of scenarios from a scenario tree and solves the LP model at each stage for the sampled scenarios. Sub-problems corresponding to stages 1 through $T+1$ are computed for each sampled scenario. For stages 2 to $T+1$, a sub-problem is initialized with the optimal decision variables for the sub-problem at the preceding stage for a scenario and is solved for the appropriate demand realization for that scenario. In the objective functions, the term $\mathbb{E}[Q_t(\cdot)]$ is replaced by a new decision variable, θ_t, which serves as an outer approximation constructed using cutting plane constraints.

- For stage 1, the objective function is modified as follows:

$$\min \quad H_1 + \sum_{n\in\mathcal{N}} c_{n1}q_{n1} + \sum_{m=1}^{2} r_m W_1^m - r_0 W_1^0 + \theta_2. \tag{20}$$

In addition to the constraints (6)–(8), a lower bound is imposed on the approximate cost-to-go function, i.e.,

$$\theta_2 \geq -MT, \tag{21}$$

where M represents a sufficiently large number, which can correspond to the maximum possible profit in one stage. Starting from iteration 2, additional constraints for θ_2 in the form of supporting hyperplanes are incorporated from the previous backward passes. These constraints take the form

$$\theta_2 \geq \Theta_2^k(\mathbf{q}_{N1}, \mathbf{W}_1) \qquad k \in \{2, 3, \dots, \}, \tag{22}$$

where k denotes the iteration index, and $\Theta_2^k(\mathbf{q}_{N1}, \mathbf{W}_1)$ represents a linear expression of the first-stage decision variables $\mathbf{q}_{N1}$ and $\mathbf{W}_1$.

- For stage 2 to stage T, the objective function is revised to be:

$$\min \quad H_t + \sum_{n\in\mathcal{N}} c_{nt}q_{nt} + \sum_{m=1}^{2} r_m W_t^m - r_0 W_t^0 - \sum_{n\in\mathcal{N}} p_{nt-1}(d_n^{\omega_{t-1}} - B_{nt-1}) + \theta_{t+1}, \tag{23}$$

with additional constraints for θ_{t+1}:

$$\theta_{t+1} \geq -M(T-t+1) \tag{24}$$

$$\theta_{t+1} \geq \Theta_{t+1}^k(\mathbf{q}_{\mathcal{N}t}, \mathbf{q}_{\mathcal{N}t}^{\mathcal{L}}, \mathbf{W}_t, \mathbf{I}_{\mathcal{N}t-1}) \qquad k \in \{2, 3, \dots, \}. \tag{25}$$

- For stage $T+1$, the LP model is (17)–(19) without introducing new constraints or altering the objective function.

Note that as the iteration progresses and additional cutting plane constraints are introduced, the big-M constraints (21) and (24) will become redundant and can be removed.

Details of $\Theta_2^k(\mathbf{q}_{\mathcal{N}1}, \mathbf{W}_1)$ and $\Theta_{t+1}^k(\mathbf{q}_{\mathcal{N}t}, \mathbf{q}_{\mathcal{N}t}^{\mathcal{L}}, \mathbf{W}_t, \mathbf{I}_{\mathcal{N}t-1})$ are provided in the backward pass.

5.2. Backward pass

In the backward step, using the variable values determined during the forward pass, SDDP solves the LP models at each stage for all demand realizations, generating a set of supporting hyperplanes similar to Benders cuts. Specifically, the cuts for the sub-problem at stage t are derived from the dual problem of the sub-problem at stage $t+1$.

- For stage $T+1$, the objective function of the dual of the sub-problem (17)–(19) is:

$$\max \sum_{n\in\mathcal{N}} \pi_{nk}^{1\omega_T}(\hat{I}_{nT-1} + \hat{q}_{nT}^L - d_n^{\omega_T}) \tag{26}$$

where $\pi_{nk}^{1\omega_T}$ represents the dual variables associated with Constraint (18) at the kth iteration for a given uncertainty realization $\omega_T \in \Omega_T$. Based on the principles of weak duality, removing the hats ($\wedge$) from the variables in (26) and treating them as decision variables gives the following hyperplane cut for stage T:

$$\begin{aligned}\theta_{T+1} &\geq \Theta_{T+1}^k(\mathbf{q}_{\mathcal{N}T}, \mathbf{q}_{\mathcal{N}T}^{\mathcal{L}}, \mathbf{W}_T, \mathbf{I}_{\mathcal{N}T-1}) \\ &= \sum_{\omega_T\in\Omega_T} \Pr(\omega_T)\left(\sum_{n\in\mathcal{N}} \pi_{nk}^{1\omega_T}(I_{nT-1} + q_{nT}^L - d_n^{\omega_T})\right),\end{aligned} \tag{27}$$

where $\Pr(\omega_T)$ is the probability for a demand realization ω_T.

- For stage $t \in 1, \dots, T-1$, where $\pi_{nk}^{1\omega_t}$, $\pi_k^{2\omega t}$, $\pi_{nk}^{3\omega t}$, $\pi_{nk}^{4i\omega t}$, $\pi_k^{5\omega_t}$, $\pi_k^{6\omega_t}$, and $\pi_k^{7\omega_t}$ represent the dual variables corresponding to Constraints (10)–(15), (24) (for $t > 1$), or (21) (for $t = 1$) at the kth iteration for an uncertainty realization $\omega_t \in \Omega_t$, respectively, the hyperplane cut in the first iteration ($k = 1$) can be expressed below, derived in a similar manner to the previous step from the objective function of the dual problem corresponding to (9)–(16) at stage $t+1$.

$$\begin{aligned}\theta_{t+1} &\geq \Theta_{t+1}^k(\mathbf{q}_{\mathcal{N}t}, \mathbf{q}_{\mathcal{N}t}^{\mathcal{L}}, \mathbf{W}_t, \mathbf{I}_{\mathcal{N}t-1}) \\ &= \sum_{\omega_t\in\Omega_t} \Pr(\omega_t)\Bigg(\sum_{n\in\mathcal{N}} \pi_{nk}^{1\omega_t}(I_{nt-1} + q_{nt}^L - d_n^{\omega_t}) \\ &\quad + \pi_k^{2\omega_t}((1+r_0)W_t^0 - \sum_{m=1}^{2}(1+r_m)W_t^m + \sum_{n\in\mathcal{N}} p_{nt}d_n^{\omega_t}) \\ &\quad + \sum_{n\in\mathcal{N}} \pi_{nk}^{3\omega_t} q_{nt} + \sum_{n\in\mathcal{N}}\sum_{i=2}^{L} \pi_{nk}^{4i\omega_t} q_{nt}^{i-1} \\ &\quad + \pi_k^{5\omega_t} U + \pi_k^{6\omega_t} H_{t+1} - \pi_k^{7\omega_t} M(T-t+1) \\ &\quad - \sum_{n\in\mathcal{N}} p_{nt}d_n^{\omega_t} + H_{t+1}\Bigg) \qquad k = 1.\end{aligned} \tag{28}$$

If this is not the first iteration ($k > 1$), the hyperplane cut constraints added for stage $t+1$ in previous iterations also influence the objective values of stage t. Consequently, the hyperplane cut for stage t is updated to:

$$\begin{aligned}\theta_{t+1} \geq \sum_{\omega_t \in \Omega_t} \Pr(\omega_t) \Bigg(& \sum_{n \in \mathcal{N}} \pi_{nk}^{1\omega_t} (I_{nt-1} + q_{nt}^L - d_n^{\omega_t}) \\ & + \pi_k^{2\omega_t} ((1+r_0) W_t^0 - \sum_{m=1}^{2} (1+r_m) W_t^m + \sum_{n \in \mathcal{N}} p_{nt} d_n^{\omega_t}) \\ & + \sum_{n \in \mathcal{N}} \pi_{nk}^{3\omega_t} q_{nt} + \sum_{n \in \mathcal{N}} \sum_{i=2}^{L} \pi_{nk}^{4i\omega_t} q_{nt}^{i-1} \\ & + \pi_k^{5\omega_t} U + \pi_k^{6\omega_t} H_{t+1} - \pi_k^{7\omega_t} M(T-t+1) \\ & - \sum_{n \in \mathcal{N}} p_{nt} d_n^{\omega_t} + H_{t+1} \\ & + \sum_{m=1}^{k-1} \pi_{mk}^{8\omega_t} \Theta_{t+2}^m (\mathbf{q}_{\mathcal{N}t+1}, \mathbf{q}_{\mathcal{N}t+1}^{\mathcal{L}}, \mathbf{W}_{t+1}, \mathbf{I}_{\mathcal{N}t}) \Bigg) \qquad k > 1, \quad (29)\end{aligned}$$

where $\Theta_{t+2}^m(\mathbf{q}_{\mathcal{N}t+1}, \mathbf{q}_{\mathcal{N}t+1}^{\mathcal{L}}, \mathbf{W}_{t+1}, \mathbf{I}_{\mathcal{N}t})$ represents the right-hand side values of the hyperplane cut added for stage $t+1$ during the mth iteration ($1 \leq m \leq k-1$), and $\pi_{mk}^{8\omega_t}$ denotes the corresponding dual variable associated with those cut constraints.

Another approach to constructing the cuts during the backward pass involves introducing local copies of the state variables (in our problem, state variables are $\mathbf{q}_{\mathcal{N}t}^{\mathcal{L}}$, $\mathbf{W_t}$, $\mathbf{I}_{\mathcal{N}t}$ and C_t) as additional constraints. This allows the cuts to be derived in closed-form expressions after solving the sub-problem. While this method simplifies the process of constructing cuts, it increases the number of decision variables and constraints. For further details, interested readers may refer to Ding et al. (2019).

5.3. Convergence analysis and bounds

The almost-sure convergence of SDDP has been established in several previous works, including Philpott and Guan (2008) and Shapiro (2011). We summarize the key ideas and adapt them to our setting in the following proposition, whose validity is unaffected by the overdraft rate, lead times, or cash-flow balances.

Proposition 1. *Under the assumptions of relatively complete recourse, bounded ordering quantities, and finite support of stagewise independent random demands, the SDDP algorithm applied to our problem converges almost surely to the optimal value of the true stochastic program after a finite number of iterations.*

Proof. Since the stochastic program satisfies relatively complete recourse, feasibility holds for every realization of demand. Given a fixed ordering decision q_{nt}, all remaining decision variables in the subproblem are uniquely determined. Together with the nonnegativity of q_{nt} and the assumed upper bound, the feasible set of each subproblem is compact. Because the objective function is continuous on this compact set, the subproblem admits an optimal solution. Applying this argument recursively from the last stage back to the first stage, we construct an optimal decision at each stage given the previous state. Since every stage's subproblem has an optimal solution and all expectations are well-defined due to the finite support of the random demands, the resulting sequence of stagewise decisions constitutes an optimal solution to the entire multi-stage stochastic program.

Because the demand takes values from a finite set and the process is stage-wise independent, the scenario tree contains finitely many nodes. Consequently, each stage's cost-to-go function is the pointwise maximum of finitely many affine functions, and is therefore piecewise linear and convex. Each affine piece of the cost-to-go function corresponds to a supporting hyperplane (cut) generated by a dual extreme point of the next stage's subproblem, and there are only finitely many such extreme points. Thus, at each stage there exists a finite set of cuts that exactly represent the cost-to-go function.

During SDDP, the forward pass independently samples stage-wise realizations. Since each realization has strictly positive sampling probability, the strong law of large numbers implies that every scenario (and hence every required dual extreme point) is sampled infinitely often with probability one. Therefore, every cut needed to represent the true cost-to-go function will almost surely be generated after a finite number of iterations. Once the value function has been fully recovered via all required cuts, the lower bound computed by SDDP equals the true optimal value, implying almost-sure convergence. □

From the above proof, we conclude that SDDP converges to the true optimal value in expectation by producing increasingly tight lower bounds over the course of the iterations. In addition, the SDDP value serves as an lower bound on the true optimal value in expectation; consequently, each run of SDDP may yield a value that fluctuates around the true optimal value, as illustrated in Fig. C.1 in Appendix C through a numerical example.

Regarding the upper bound, the forward pass generates feasible policies for each independently sampled scenario, which allows the computation of a statistical estimate of the expected cost of the policy, serving as an upper bound for the true optimal value of the problem in expectation. Let K denote the number of sampled scenarios, and let v_j represent the sum of the stage-wise objective values under the feasible policy for the jth scenario. Define the sample mean

$$\bar{v} := \frac{1}{K} \sum_{j=1}^{K} v_j,$$

which provides an estimate of the expected objective value of the feasible policy; in expectation, this value is greater than or equal to the true optimal value. The corresponding sample variance is

$$\sigma^2 = \frac{1}{K-1} \sum_{j=1}^{K} (v_j - \bar{v})^2.$$

Let $z_{\alpha/2}$ denote the $(1-\alpha/2)$ quantile of the standard normal distribution. Then, under a confidence level of $100(1-\alpha)\%$, a statistical upper bound for the true optimal value is given by

$$\bar{v} + z_{\alpha/2} \frac{\sigma}{\sqrt{K}}.$$

It is worth noting that, although the stochastic program is formulated to minimize the increment in negative cash balance, our original problem is profit maximization. Consequently, the "lower bound" by the SDDP formulation corresponds to an upper bound in the profit-maximization setting (after multiplying by -1 and adding the initial cash balance C_0), and vice versa. To illustrate the convergence of SDDP, Fig. C.1 in Appendix C reports the upper bound (denoted by UB) obtained from the SDDP value, the lower bound (denoted by LB) as well as their 95% confidence intervals for a numerical example in our problem.

Regarding the convergence rate of SDDP, Lan (2022) show that the number of iterations required to obtain a solution with a prescribed accuracy depends only mildly on the number of stages T. This result highlights the scalability of SDDP and its suitability for large-scale multistage stochastic optimization problems.

5.4. Stopping conditions

Pereira and Pinto (1991) proposed a stopping condition that stops the SDDP algorithm when the current lower bound of the minimization problem first lies within the confidence interval of the upper bound. Shapiro (2011) pointed out that this condition may be too optimistic, particularly when the confidence level is high or the variance of the upper bound is large. In practice, it may be more reliable to

terminate the algorithm after a predefi ned number of iterations or a time limit (De Matos et al., 2015; Soares et al., 2017; Thevenin et al., 2022). In our numerical tests, we experimented with stopping criteria based on bounds, time and number of iterations. Detailed steps of the SDDP are shown by Algorithm 1.

Algorithm 1: The SDDP algorithm

Data: Values of problem parameters.
Result: Approximate expected final profit and the first-stage order quantities.
Generate a scenario tree for the uncertain demands in the planning horizon;
iteration index: $k \leftarrow 1$;
while *the chosen stopping condition is not met* **do**
  Sample a number of scenarios;
  forward pass:
  for $t \leftarrow 1$ **to** $T+1$ **do**
    foreach *sampled scenario* **do**
      if $t == 1$ **then**
        Solve the LP1 model: (20)–(22), (6)–(8);
      else if $t == T+1$ **then**
        Solve the LP3 model: (17)–(19);
      else
        Solve the LP2 model: (23)–(25), (10)–(16);
      end
      Record the values of $\hat{\mathbf{q}}_{\mathcal{N}\mathbf{t}}$, $\hat{\mathbf{q}}^{\mathcal{L}}_{\mathcal{N}\mathbf{t}}$, $\widehat{\mathbf{W}}_{\mathbf{t}}$, $\hat{\mathbf{I}}_{\mathcal{N}\mathbf{t}}$ and $\hat{C}_t$;
    end
  end
  backward pass:
  for $t \leftarrow T+1$ **to** 2 **do**
    foreach *sampled scenario* **do**
      foreach *realization* $\omega_t \in \Omega_t$ **do**
        Solve the model LP3 (if $t = T+1$) or LP2 (if $2 \le t \le T$);
      end
      create the cut:
      if $t == T+1$ **then**
        Construct the cut (27) and add it to model LP2 (if $t > 2$) or LP1 (if $t = 2$);
      else
        if $k == 1$ **then**
          Construct the cut (28) and add it to model LP2 (if $t > 2$) or LP1 (if $t = 2$);
        else
          Construct the cut (29) and add it to model LP2 (if $t > 2$) or LP1 (if $t = 2$);
        end
      end
    end
  end
  $k \leftarrow k+1$;
end

6. Computational accelerations

In this section, we introduce two techniques aimed at accelerating the computation of SDDP for our problem. The first does not affect the solution quality of SDDP, while the second is a heuristic approach. Other possible accelerations proposed in the literature (eg., De Matos et al. (2015), Thevenin et al. (2022)) are not discussed, as some are not well-suited to our problem, while others offer only marginal improvements over the original SDDP based on our numerical experiments.

Remark 1. One practical tip for SDDP implementation — though not necessarily to be claimed as an enhancement — is to create a single solver model per stage and dynamically update the right-hand sides of relevant constraints for each demand realization during forward and backward passes, rather than rebuilding the model from scratch each time. This can significantly improve computational efficiency in our tests.

6.1. Remove duplicate cut constraints

A key feature of SDDP is that, during the backward pass, each sampled scenario can generate a new cut at every stage, while all scenarios share the same set of cut constraints from previous iterations. Due to the similarities among sub-problems (as discussed in the next subsection), some cuts derived from one scenario may overlap with those from other scenarios or replicate previously added cuts.

This approach is based on the fact that the optimal value of a linear programming (LP) model is a convex, piecewise-linear function of its right-hand-side (RHS) values (Theorem 5.1, Bertsimas and Tsitsiklis (1997)). Consequently, the subgradients of the LP value function with respect to the RHS — that is, the shadow price (dual values) associated with the constraints, remain constant within each linear segment of the RHS. For two different scenarios, their distinction in the stage-wise LP subproblem is reflected solely through differences in the RHS values of certain constraints. If these constraints yield identical dual values in both scenarios, then the resulting cuts are also identical, as can be seen from the cut expressions (27), (28), and (29). Moreover, because scenarios are sampled independently in the forward pass, the same scenario may be sampled multiple times, which also generates duplicate cuts during the backward pass.

While solvers such as CPLEX or Gurobi do not automatically eliminate duplicate constraints, we observe that detecting and removing such constraints before solving the model can substantially reduce computational time.

6.2. Leverage sub-problem similarities

The idea behind leveraging sub-problem similarities is to avoid solving LPs for every demand realization in the backward pass, instead solving only a few LPs at each stage.

For our problem, the dual values of the LP models largely depend on whether the RHS values of the inventory flow constraints are non-negative and the cash interval in which the retailer's cash balance lies. Specifically, the dual values of the sub-problems at stage t are primarily associated with the following two sets of constraints:

$$I_{nt-1} - B_{nt-1} = \hat{I}_{nt-2} + \hat{q}^{L_n}_{nt-1} - d_n^{\omega_{t-1}} \qquad \forall n, \forall t = 2, \dots, T+1 \tag{30}$$

$$C_{t-1} - \sum_{n \in \mathcal{N}} c_{nt} q_{nt} - W_t^0 + \sum_{m=1}^{2} W_t^m = H_t \qquad \forall t = 2, \dots, T \tag{31}$$

The dual values of the inventory flow Constraint (30) remain identical for many demand realizations, as long as these realizations lead to the same sign (positive or negative) for the RHS of this constraint. Similarly, for Constraint (31), the cash balance $C_{t-1} - \sum_{n \in \mathcal{N}} c_{nt} q_{nt} - H_t$ can fall into one of three intervals: $[0, +\infty)$, $[-U, 0)$, or $(-\infty, -U)$. These three cash balance scenarios influence the values of W_t^0, W_t^1, W_t^2 as well as the interest applied to the retailer's payments in the objective function and result in three distinct sets of dual values.

At each stage, the retailer can encounter 2^N different inventory sign scenarios due to the presence of N products, combined with 3 possible cash balance situations. As a result, there are 3×2^N groups of dual values for all possible realizations of a subproblem. For a given demand realization and given values of $q_{nt}, \forall n$, we can easily calculate the RHS values of all the constraints in a subproblem and identify the relevant group of dual values. From this information, the appropriate cut constraint could be generated without having to solve a subproblem in the backward pass.

However, since the values of q_{nt} for a demand realization are unknown without solving the sub-problem, a heuristic approach is required to reduce computation time. We limit solving the LP model for the values of q_{nt} to once every ℓ demand realizations in the backward pass, where ℓ is a user-defined parameter. We assume that the values of q_{nt} remain unchanged between two consecutive LP computations. Therefore, instead of solving an LP model for each demand realization at a given stage in the backward pass, we only need to solve 3×2^N LP models to obtain the corresponding 3×2^N groups of dual values and one LP model every ℓ demand realizations to update the values of q_{nt}. Algorithm 2 presents the detailed steps of the proposed acceleration approach applied in the backward pass, where the variable $ICashStatus$ is used to store the inventory and cash interval states, as well as the corresponding dual values of the constraints.

This acceleration can significantly reduce computation time, particularly in multi-product problems with a large number of demand realizations. For example, in a two-product problem with 20 demand samples per product, the total number of demand realizations is $20 \times 20 = 400$. By applying this technique with $\ell = 10$, we solve at most $400/10 + 3 \times 2^2 = 52$ LP models per stage per scenario—a substantial reduction compared to solving all 400 LP models.

Remark 2. Since the first acceleration technique does not alter the solutions of the LP models and the generated cuts in SDDP, the standard properties of SDDP — such as the bounds and convergence — remain intact when it is applied. In contrast, the second acceleration technique introduces a heuristic to approximate the dual values of certain LP models. As a result, the lower bound and the convergence of SDDP are no longer ensured after applying this technique, although a statistical upper bound can still be obtained from the forward pass.

Algorithm 2: Detailed steps of acceleration2

```
backward pass:
for t ← T + 1 to 2 do
    foreach sampled scenario do
        index ← 0; ICashStatus ← ∅;
        foreach realization ω_t ∈ Ω_t do
            if index mod l = 0 then
                Solve the model LP3 (if t = T + 1) or LP2 (if
                  2 ≤ t ≤ T);
                Get the values of q_{N_t}, update ICashStatus
                  accordering to (30) and (31), and record the
                  corresponding dual values for this ICashStatus;
            else
                Update ICashStatus accordering to (30) and (31);
                if this ICashStatus has been recorded then
                    Get the corresponding dual values;
                else
                    Solve the model LP3 (if t = T + 1) or LP2 (if
                      2 ≤ t ≤ T);
                    Get the values of q_{N_t}, update ICashStatus
                      accordering to (30) and (31), and record
                      the corresponding dual values for this
                      ICashStatus;
                end
            end
            index ← index + 1;
        end
        create the cut:
        ⋮
    end
end
```

7. Numerical investigation

In this section, we first evaluate the performance of SDDP and the proposed accelerations for our problem by comparing its solutions with the optimal solution obtained using the SDP formulation given in Appendix A. Next, we assess the proposed accelerations to the original SDDP algorithm. Finally, we derive some managerial insights regarding the impact of overdrafts on the retailer's profit.

All numerical tests are conducted on a MacBook with an Apple M1 Pro CPU, 16 GB of RAM, and macOS Monterey operating system. The LP models are solved using Gurobi 10.0.1 and both the SDDP and SDP are coded in C++. The source code and numerical results are publicly available in the GitHub repository: https://github.com/RobinChen121/SDDP_overdraft.

7.1. Optimality gaps of SDDP and the two accelerations

Theoretically, the optimal solution can be obtained using SDP. However, due to the curse of dimensionality, solving even small instances of the problem becomes extremely time-consuming, as SDP requires the computation and storage of values for each state variable at every stage. For example, in a representative problem instance involving a single product across five stages — comparable to the cases examined in the following subsection — the SDP implementation in C++ or Java fails to yield a solution within 8 h, whereas the Python implementation cannot solve even a four-stage instance in under 4 h.

Therefore, we restrict the comparison with the optimal solution using SDP to four-stage single-product problem instances. The parameter values are set as follows: initial cash of 0, initial inventory of 0, selling price of either 10 or 5, unit ordering cost of 1, unit salvage value of 0.5, lead time of 1 stage, outstanding order at the beginning of the planning horizon is 0, and overhead costs of either 50 or 25 at each stage. The overdraft limit is set to 500, the deposit interest rate is 0%, and the overdraft interest rate takes values of 0%, 5%, 10%, 15%, or 20%. The penalty interest rate when the cash balance exceeds the overdraft limit is set to be 200%. The mean demand follows 10 patterns, with data selected from Rossi et al. (2015) and scaled to 4 stages to fit the problem context. These patterns include one stationary pattern (STA), two life cycle patterns (LCY1 and LCY2), two sinusoidal patterns (SIN1 and SIN2), one random pattern (RAND), and four empirical patterns (EMP1, EMP2, EMP3, and EMP4). Table B.1 in Appendix B provides details of the demand data. Considering all combinations of problem parameters, the test set comprises 200 problem instances.

Demand in each stage is assumed to follow a Poisson distribution, and Latin Hypercube Sampling is applied to generate demand samples (McKay et al., 2000). The number of demand samples per stage, the number of sampled scenarios, and the iteration limit for terminating the algorithm are set to 10, 30, and 50, respectively, as the SDDP results are observed to be stable under this configuration. It should be noted that more iterations or number of sampled scenarios did not consistently yield improved optimality gaps in our testing, as illustrated with an example in Appendix C.

In this subsection and the ones that follow, we report the maximum expected profit, which corresponds to the negative of the objective function in the SDDP formulation. We use the mean absolute deviation (MAD) to calculate the optimality gap (reported in the "Gap" column), and also report the mean absolute value of the objective (shown in the "Value" column) for each method. "SDDP accelerate 1" refers to the first acceleration which involves removing duplicated constraints, while "SDDP accelerate 1+2" applies both the first acceleration and the second one that leverages similarities among sub-problems with $\ell = 3$. Each instance is executed 20 times, and the average results are presented in Table 3. The results illustrate the following points.

Table 3
Performance of the SDDP and its accelerations.

	SDP		SDDP			SDDP accelerate 1			SDDP accelerate 1+2			Cases
	Time (s)	Value	Time	Value	Gap	Time	Value	Gap	Time	Value	Gap	
Prices												
5	602.34	67.64	12.01	67.66	0.35	0.94	67.63	0.40	0.43	67.60	0.48	100
10	396.94	273.57	11.32	273.94	0.88	0.90	274.18	0.95	0.43	274.57	1.12	100
Overhead cost												
25	523.76	201.98	11.54	202.21	0.63	0.92	202.4	0.66	0.43	202.61	0.77	100
50	475.52	139.23	11.79	139.39	0.60	0.92	139.42	0.69	0.43	139.55	0.82	100
Overdraft rate												
0%	10.98	181.84	10.31	181.73	0.76	0.77	181.95	0.55	0.36	182.05	0.55	40
5%	1075.20	174.11	11.82	174.32	0.55	0.92	174.36	0.58	0.43	174.49	0.57	40
10%	181.23	168.91	11.96	169.25	0.60	0.94	169.25	0.73	0.44	169.60	1.05	40
15%	1133.84	165.70	12.07	165.89	0.64	0.96	165.87	0.67	0.46	166.33	0.93	40
20%	96.94	162.48	12.17	162.80	0.53	0.98	163.10	0.86	0.47	162.94	0.88	40
Demand pattern												
STA	358.25	144.17	11.70	144.46	0.46	0.92	144.65	0.75	0.24	145.22	2.03	20
STA	358.25	144.17	11.70	144.46	0.46	0.92	144.65	0.75	0.44	144.38	0.56	20
LC1	148.25	110.40	11.93	110.74	0.71	0.95	110.69	0.53	0.46	110.82	0.72	20
LC2	713.00	173.90	11.73	173.61	0.61	0.91	174.31	0.79	0.43	174.51	0.75	20
SIN1	95.89	121.28	11.57	121.49	0.77	0.85	121.31	0.57	0.40	121.63	0.72	20
SIN2	806.32	133.83	11.77	134.14	0.46	0.95	134.27	0.68	0.44	134.06	0.66	20
RAND	1087.66	169.91	11.69	170.12	0.55	0.95	170.25	0.65	0.44	170.75	1.08	20
EMP1	105.26	222.19	11.56	222.58	0.71	0.90	222.40	0.69	0.43	222.84	0.97	20
EMP2	125.87	201.07	11.47	200.95	0.64	0.91	201.39	0.70	0.43	201.72	0.98	20
EMP3	530.77	205.24	11.61	205.34	0.53	0.92	205.43	0.75	0.43	205.56	0.71	20
EMP4	1025.11	224.07	11.62	224.56	0.72	0.92	224.37	0.67	0.42	224.55	0.82	20
Avg.	499.64	170.61	11.66	170.80	0.62	0.92	170.91	0.68	0.43	171.08	0.79	200

- The optimality gaps of SDDP and the two accelerations are very small, with their MADs being 0.62, 0.68, and 0.79, respectively, compared to a mean absolute optimal value of 170.61. Moreover, since the optimality gaps of SDDP and "SDDP accelerate 1" are similar, the results confirm that the first acceleration does not compromise the computational quality of SDDP.
- SDDP and its two accelerations significantly reduce computational time. The average running time of SDP is 499.64 s, and its performance is unstable—highly sensitive to the floating-point precision of state variables; for example, when the overdraft interest rate is 5% or 15%, the average runtime exceeds 1000 s. In contrast, the average computation times for SDDP, "SDDP accelerate 1" and "SDDP accelerate 1+2" are only 11.66 s, 0.92 s, and 0.43 s, respectively.
- The second acceleration, which leverages sub-problem similarities, can further accelerate computation, albeit at the cost of a larger optimality gap (0.79 average MAD compared to 0.62 for SDDP).
- Demand patterns, prices, overhead cost and overdraft interest rate do not significantly affect the performance of SDDP and the accelerations, demonstrating the flexibility and robustness of these methods in solving the problem.

7.2. Comparison of SDDP and its accelerations

We further compare SDDP and its accelerations by evaluating problem instances involving two products with Poisson-distributed demand at each stage. The same 10 demand patterns are considered, but the data is selected and scaled for a 6-stage horizon. Detailed information on the mean demand for Product 1 at each stage is presented in Table B.2 of Appendix B. Product 2 has a mean demand that is half that of Product 1. Both products have a lead time of one stage. The selling prices are 5 for Product 1 and 10 for Product 2, while the unit ordering costs are 1 and 2, respectively. The unit salvage value for each product is set at half its corresponding ordering cost. The overhead cost per stage is 100. All other problem parameters remain consistent with the previous instances: a deposit interest rate of 0%, an overdraft interest rate of 10%, a penalty interest rate of 200%, an overdraft limit of 500, and initial cash, inventory levels and initial outstanding order set to zero.

For the SDDP settings, we use 20 demand samples per stage and 10 sampled scenarios. The iteration limit is set to 100, as the SDDP value slightly improves after 50 iterations across all cases. Each instance is run 10 times, and the average computation time and average MAD between SDDP and its accelerations are reported in Tables 4 and 5, respectively. In the table columns,"SDDP" refers to the original SDDP without any accelerations; "accelerate 1" refers to the first acceleration; and $\ell = 3, \ldots, \ell = 11$ indicates the application of both accelerations, with ℓ set to values from 3 to 11.

Results for SDP are not included in Tables 4 and 5, as computing the optimal value using this method is intractable for the two-product, six-stage problem instances. Table 5 reports the solution quality gap between standard SDDP and the application of the acceleration techniques. Note that the first acceleration has no impact on solution quality. From Tables 4 and 5, we observe the following:

- the first acceleration reduces the computation time of SDDP from an average of 476.57 s to 252.76 s; when both accelerations are applied, the computation time can be further reduced to 59.42 s (when $\ell = 11$);
- as ℓ increases, the computation time tends to decrease; however, the absolute deviations do not necessarily increase and may vary across different instances, as shown in Table 5;
- the deviations do not exhibit a clear correlation with the magnitude of the SDDP values. For instance, under the RAND demand pattern — where the SDDP value is 31.65 due to very low mean demands in some stages — the deviations are not significantly different from those observed under other demand patterns.
- overall, applying the accelerations yields solutions with small deviations while significantly reducing computation time.

7.3. Impact of the overdraft on profit

We perform a sensitivity analysis on the parameters of the overdraft facility, allowing the interest rate to take the values 0%, 5%, 10%,

Table 4
Computational time of SDDP and its enhancements.

Demand pattern	Time (s)						
	SDDP	SDDP accelerate 1	SDDP accelerate 1+2				
			$\ell = 3$	$\ell = 5$	$\ell = 7$	$\ell = 9$	$\ell = 11$
STA	493.23	270.74	120.74	86.59	73.09	61.32	56.05
LC1	483.53	232.44	118.51	86.00	71.19	62.89	55.67
LC2	496.75	280.33	125.5	87.71	72.75	62.13	54.70
SIN1	495.79	281.96	126.23	86.33	70.10	59.61	52.01
SIN2	484.88	262.81	119.81	87.19	70.25	60.15	54.49
RAND	486.19	256.34	128.99	94.79	82.08	71.28	66.31
EMP1	417.52	195.97	137.36	110.76	102.62	99.27	94.20
EMP2	470.37	255.02	122.87	89.17	74.31	63.99	57.07
EMP3	459.51	217.05	111.81	80.40	67.57	59.31	53.76
EMP4	477.90	274.90	119.54	83.86	67.20	56.30	49.93
Avg.	476.57	252.76	123.14	89.28	75.12	65.63	59.42

Table 5
Computational gaps of the enhancements.

Demand pattern	SDDP value	SDDP accelerate 1+2 MAD				
		$\ell = 3$	$\ell = 5$	$\ell = 7$	$\ell = 9$	$\ell = 11$
STA	477.16	1.68	4.10	0.61	0.02	0.64
LC1	523.76	3.19	4.82	1.00	0.19	0.41
LC2	554.74	3.55	5.90	2.00	0.50	1.21
SIN1	635.08	2.69	5.84	0.14	0.46	0.04
SIN2	360.81	3.87	4.55	1.93	1.32	1.34
RAND	31.65	1.90	5.71	4.38	7.41	6.42
EMP1	3011.15	4.54	7.30	2.15	4.17	4.46
EMP2	2318.55	3.70	8.80	2.30	2.40	2.79
EMP3	1348.03	6.42	8.95	4.83	3.31	2.98
EMP4	2139.32	4.00	8.19	1.21	2.81	1.57
Avg.	1140.03	3.55	6.42	2.06	2.26	2.19

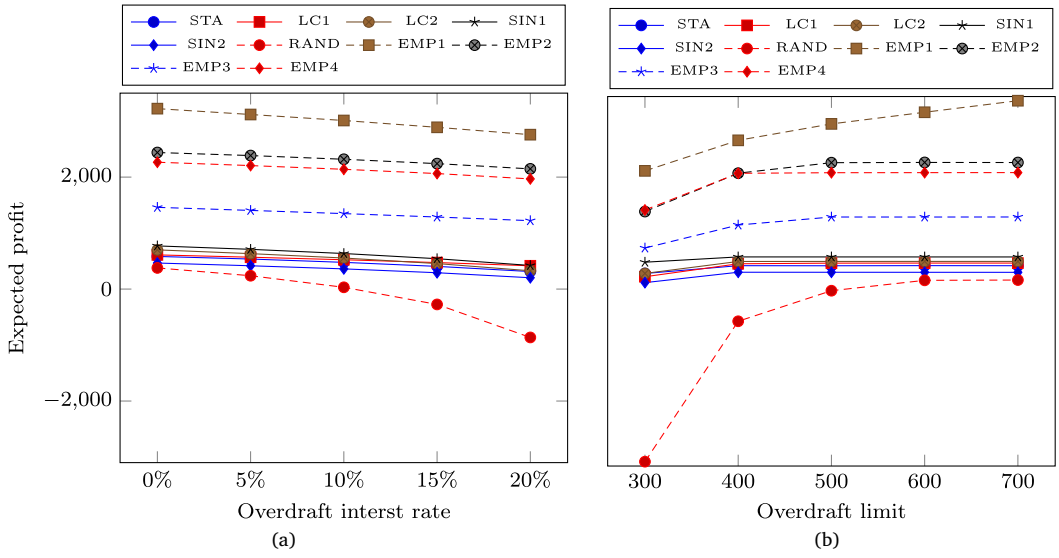

Fig. 3. Average total profit variations of different overdraft terms.

15% and 20% and the limit to take the values 300, 400, 500, 600 and 700. We report the retailer's average expected total profit for each demand pattern in Fig. 3 after running each instance 10 times. All other parameters of the problem instance and method are as in the previous experiment. The first acceleration is applied to solve the problems as it does not affect the solution quantity of SDDP.

In Fig. 3, the retailer's expected total profit demonstrates a negative correlation with the overdraft interest rate and a positive correlation with the overdraft limit, aligning with expectations. The expected profit is relatively sensitive to changes in the interest rate. In contrast, the model's responsiveness to changes in the overdraft limit diminishes once the limit exceeds the retailer's operational requirements. This is evident as the expected profit plateaus across various demand patterns when the overdraft limit rises from 500 to 600 or 700. In some cases, the expected profit is lower than the negative overdraft limit value, as our model imposes a 200% penalty interest rate for negative cash balances. In real-world applications, such stringent overdraft terms would generally be deemed unacceptable for the retailer.

8. Conclusions

In this paper, we employ the SDDP method to address a cash-flow inventory problem involving multiple products, lead times, and a specific type of financing service. To facilitate the computation of cuts, we introduce auxiliary variables into the sub-problems. Additionally, we propose two acceleration techniques that remove the duplicate constraints added in the model and leverage the similarity of dual values across sub-problems, which contribute to reducing computational time while maintaining relatively small solution gaps.

Future research could explore using the Stochastic Dual Dynamic Integer Programming (SDDiP) approach (Zou et al., 2019) to tackle cash-flow problems with integer decision variables. Furthermore, the framework could be extended to incorporate other types of financing services.

CRediT authorship contribution statement

Zhen Chen: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Thomas W. Archibald:** Writing – review & editing.

Declaration of competing interest

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

Acknowledgments

Dr. Zhen Chen is supported by Brunel University's Mid and Early Career Award (Project No. 12334134) and the National Natural Science Foundation of China (Project Nos. 72101213 and 72571017). The authors also thank the three anonymous referees for their constructive comments and detailed reviews that improved this paper.

Appendix A. The stochastic dynamic programming formulation

Denote $V_t(I_{1t-1},\dots,I_{Nt-1},q^1_{1t},\dots,q^{L_1}_{1t},\dots,q^1_{Nt},\dots,q^{L_N}_{Nt},C_{t-1})$ as the maximum expected terminal cash balance at stage t given that the initial inventory for each product: $I_{1t-1},\dots,I_{Nt-1}$, the ordering quantity for each product placed 1 to L_n stages ago: $q^1_{1t},\dots,q^{L_1}_{1t},\dots,q^1_{Nt},\dots,q^{L_N}_{Nt}$ and initial cash balance C_{t-1}. The optimality equation is:

$$
\begin{aligned}
&V_t(I_{1t-1},\dots,I_{Nt-1},q^1_{1t},\dots,q^{L_1}_{1t},\dots,q^1_{Nt},\dots,q^{L_N}_{Nt},C_{t-1})\\
&=\max_{\substack{q_{1t}\geq 0\\ \dots\\ q_{Nt}\geq 0}} V_{t+1}\big((I_{1t-1}+q^{L_1}_{1t}-D_{1t})^+,\dots,(I_{Nt-1}+q^{L_N}_{Nt}-D_{Nt})^+,\\
&\quad q_{1t},q^1_{1t},\dots,q^{L_1-1}_{1t},\dots,q_{Nt},q^1_{Nt},\dots,q^{L_N-1}_{Nt},C_t\big),
\end{aligned}
$$

where

$$
C_t = C_{t-1}-H_t-\sum_{n\in\mathcal{N}} c_{nt}q_{nt}-\sum_{m=1}^{2} r_k W^m_t+r_0W^0_t+\sum_{n\in\mathcal{N}} p_{nt}\min\{I_{nt-1}+q^{L_n}_{nt},D_{nt}\},
$$

and

$$
W^0_t=\begin{cases} C_{t-1}-H_t-\sum_{n\in\mathcal{N}} c_{nt}q_{nt} & \text{if } C_{t-1}-H_t-\sum_{n\in\mathcal{N}} c_{nt}q_{nt}\geq 0\\ 0 & \text{otherwise,}\end{cases}
$$

$$
W^1_t=\begin{cases} \max\{U,H_t+\sum_{n\in\mathcal{N}} c_{nt}q_{nt}-C_{t-1}\} & \text{if } C_{t-1}-H_t-\sum_{n\in\mathcal{N}} c_{nt}q_{nt}<0\\ 0 & \text{otherwise,}\end{cases}
$$

$$
W^2_t=\begin{cases} H_t+\sum_{n\in\mathcal{N}} c_{nt}q_{nt}-C_{t-1}-U & \text{if } C_{t-1}-H_t-\sum_{n\in\mathcal{N}} c_{nt}q_{nt}\leq -U\\ 0 & \text{otherwise.}\end{cases}
$$

The boundary condition is:

$$
V_{T+1}(I_{1T},\dots,I_{NT},q^1_{1T+1},\dots,q^{L_1}_{1T+1},\dots,q^1_{NT+1},\dots,q^{L_N}_{NT+1},C_T)=C_T+\sum_{n\in\mathcal{N}} v_nI_{nT}.
$$

Table B.1
Detailed mean demand data for the tests of SDDP optimality gaps.

Demand pattern	Stage			
	1	2	3	4
STA	15	15	15	15
LC1	25	20	10	7
LC2	7	12	17	23
SIN1	24	15	5	20
SIN2	5	11	22	10
RAND	31	14	25	12
EMP1	40	23	8	30
EMP2	7	30	15	12
EMP3	17	6	31	22
EMP4	9	17	35	10

Table B.2
Detailed mean demand data for the first product.

Demand pattern	Stage					
	1	2	3	4	5	6
STA	30	30	30	30	30	30
LC1	50	46	38	28	23	18
LC2	14	18	23	33	42	49
SIN1	47	30	13	30	47	54
SIN2	21	24	39	30	24	18
RAND	63	10	4	33	67	14
EMP1	15	140	147	74	109	88
EMP2	14	71	49	152	78	33
EMP3	13	35	79	43	44	59
EMP4	15	56	19	84	136	67

Appendix B. Detailed demand data for numerical investigation

See Tables B.1 and B.2.

Appendix C. An example of the influence of iteration limit and number of sampled scenarios on SDDP

In this numerical example, the lead time is one stage, and the initial cash, initial outstanding orders, and initial inventory are set to zero. The selling price is 10, the unit variable ordering cost is 1, the unit salvage value is 0.5, and the overhead cost per stage is 50. The overdraft limit is 500, the deposit interest rate is 0%, the overdraft interest rate is 10%, and the penalty interest rate is 200%. Demand follows a Poisson distribution with a mean of 15 at each of the four stages. The optimal value of the problem, obtained using SDP, is 167.38. Each stage uses 10 demand samples, and the instance is run 20 times. The average values obtained by SDDP under different iteration limits (with the number of sampled scenarios fixed at 30) and under different numbers of sampled scenarios (with the iteration limit fixed at 50) are shown in Figs. C.1 and C.2, respectively.

The results indicate that the SDDP values, which also serve as upper bounds (UBs) of the true problem, converge rapidly during the initial iterations and then stabilize, lying within the confidence interval (CI) of the lower bounds (LBs) after approximately 20 iterations. Recall that the SDDP value provides an upper bound on the true optimal value in expectation; accordingly, we also report a confidence interval for this bound. Moreover, there is no clear evidence that increasing the iteration limit — i.e., extending the computational time of SDDP — consistently yields values closer to the optimal solution.

On the other hand, the number of sampled scenarios has little influence on the SDDP values. Even using a single sampled scenario can yield solutions close to the optimal value as shown by Fig. C.2.

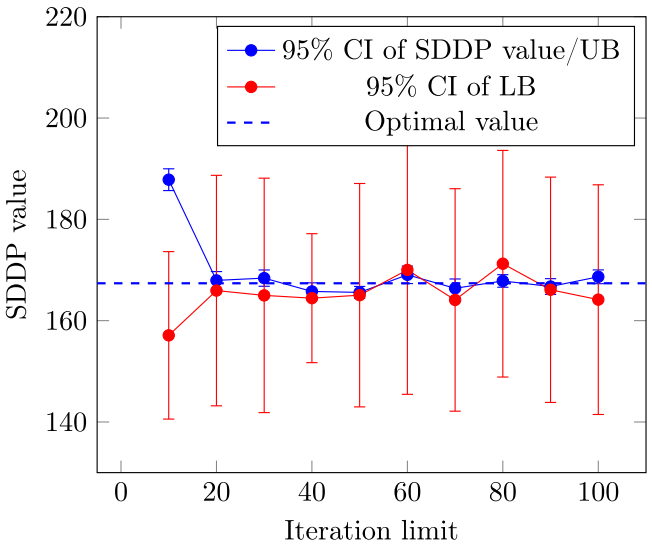

Fig. C.1. Influence of iteration limit on SDDP.

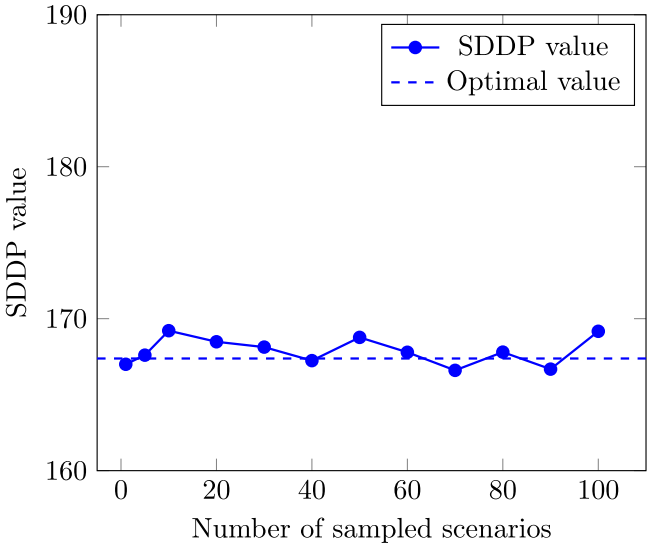

Fig. C.2. Influence of the number of sampled scenarios on SDDP.

References

Archibald, T. W., Thomas, L. C., Betts, J. M., & Johnston, R. B. (2002). Should start-up companies be cautious? Inventory policies which maximise survival probabilities. *Management Science*, *48*(9), 1161–1174.

Barclays (2025). Business overdrafts–A flexible overdraft that moves with your business demands. https://www.barclays.co.uk/current-accounts/bank-account/overdrafts/. (Accessed 31 Jul 2025).

Bertsimas, D., & Tsitsiklis, J. N. (1997). Global dependence on the right-hand side vector. In *Introduction to linear optimization*: *Vol. 6*, (p. 213). Athena scientific Belmont, MA, chapter 5.

Bhattacharya, A., Kharoufeh, J. P., & Zeng, B. (2023). A nonconvex regularization scheme for the stochastic dual dynamic programming algorithm. *INFORMS Journal on Computing*, *35*(5), 1161–1178.

Birge, J. R. (1985). Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research*, *33*(5), 989–1007.

Buzacott, J. A., & Zhang, R. Q. (2004). Inventory management with asset-based financing. *Management Science*, *50*(9), 1274–1292.

Cao, B., Zhong, Y., & Zhou, Y.-W. (2024). The role of completely joint liability in financing multiple capital-constrained firms: Risk sharing, inventory and financial strategies. *European Journal of Operational Research*, *313*(3), 1072–1087.

CBInsights (2021). The top 12 reasons startups fail. https://www.cbinsights.com/research/report/startup-failure-reasons-top/. (Accessed 31 October 2024).

Chao, X., Chen, J., & Wang, S. (2008). Dynamic inventory management with cash flow constraints. *Naval Research Logistics*, *55*, 758–768.

Chen, Z., & Archibald, T. W. (2024). Maximizing the survival probability in a cash flow inventory problem with a joint service level constraint. *International Journal of Production Economics*, *270*, Article 109191.

Chen, Z., & Rossi, R. (2021). A dynamic ordering policy for a stochastic inventory problem with cash constraints. *Omega*, *102*, Article 102378.

Chen, Z., & Zhang, R.-q. (2021). A multi-period multi-product stochastic inventory problem with order-based loan. *International Journal of Production Research*, 1–14.

De Matos, V. L., Philpott, A. B., & Finardi, E. C. (2015). Improving the performance of stochastic dual dynamic programming. *Journal of Computational and Applied Mathematics*, *290*, 196–208.

Ding, L., Ahmed, S., & Shapiro, A. (2019). A Python package for multi-stage stochastic programming. *Optimization Online*, 1–42.

Fu, K., Gong, X., Hsu, V. N., & Xue, J. (2021). Dynamic inventory management with inventory-based financing. *Production & Operations Management*, *30*(5), 1313–1330.

Gong, X., Chao, X., & Simchi-Levi, D. (2014). Dynamic inventory control with limited capital and short-term financing. *Naval Research Logistics*, *61*(3), 184–201.

Hole, J., Philpott, A., & Dowson, O. (2025). Capacity planning of renewable energy systems using stochastic dual dynamic programming. *European Journal of Operational Research*, *322*(2), 573–588.

HSBC (2025). Business overdraft. https://www.hsbc.co.uk/current-accounts/products/overdrafts/. (Accessed 31 Jul 2025).

Ipsos (2024). SME finance survey. https://www.british-business-bank.co.uk/sites/g/files/sovrnj166/files/2024-03/ipsos-sme-finance-survey-2023.pdf. (Accessed 31 October 2024).

Kajjoune, O., Aouam, T., Zouadi, T., & Ranjan, R. P. (2023). Dynamic lot-sizing in a two-stage supply chain with liquidity constraints and financing options. *International Journal of Production Economics*, *258*, Article 108799.

Katehakis, M. N., Melamed, B., & Shi, J. (2016). Cash-flow based dynamic inventory management. *Production & Operations Management*, *25*(9), 1558–1575.

Kiszka, A., & Wozabal, D. (2025). Stochastic dual dynamic programming for optimal power flow problems under uncertainty. *European Journal of Operational Research*, *321*(3), 814–836.

Lan, G. (2022). Complexity of stochastic dual dynamic programming. *Mathematical Programming*, *191*(2), 717–754.

Löhndorf, N., & Shapiro, A. (2019). Modeling time-dependent randomness in stochastic dual dynamic programming. *European Journal of Operational Research*, *273*(2), 650–661.

Luo, W., & Shang, K. H. (2019). Managing inventory for firms with trade credit and deficit penalty. *Operations Research*, *67*(2), 468–478.

McKay, M. D., Beckman, R. J., & Conover, W. J. (2000). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, *42*(1), 55–61.

Pereira, M. V. F., & Pinto, L. M. V. G. (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, *52*, 359–375.

Philpott, A. B., & Guan, Z. (2008). On the convergence of stochastic dual dynamic programming and related methods. *Operations Research Letters*, *36*(4), 450–455.

QuickBooks (2019). SMEs' most common issue is cashflow. https://www.uktech.news/accountancy/smes-most-common-issue-is-cashflow-20190402. (Accessed 31 October 2024).

Rossi, R., Kilic, O. A., & Tarim, S. A. (2015). Piecewise linear approximations for the static–dynamic uncertainty strategy in stochastic lot-sizing. *Omega*, *50*, 126–140.

Shapiro, A. (2011). Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, *209*(1), 63–72.

Shapiro, A., Tekaya, W., da Costa, J. P., & Soares, M. P. (2013). Risk neutral and risk averse stochastic dual dynamic programming method. *European Journal of Operational Research*, *224*(2), 375–391.

Soares, M. P., Street, A., & Valladão, D. M. (2017). On the solution variability reduction of stochastic dual dynamic programming applied to energy planning. *European Journal of Operational Research*, *258*(2), 743–760.

Thevenin, S., Adulyasak, Y., & Cordeau, J.-F. (2022). Stochastic dual dynamic programming for multiechelon lot sizing with component substitution. *INFORMS Journal on Computing*, *34*(6), 3151–3169.

Van Slyke, R. M., & Wets, R. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, *17*(4), 638–663.

Zou, J., Ahmed, S., & Sun, X. A. (2019). Stochastic dual dynamic integer programming. *Mathematical Programming*, *175*, 461–502.