Stochastic dynamic programming heuristic for the (R, s, S) policy parameters computation

Stochastic Dynamic Programming Heuristic for the (R, s, S) Policy Parameters Computation

Andrea Visentin, Steven Prestwich

Insight Centre for Data Analytics, School of Computer Science & IT, University College Cork, Ireland

Roberto Rossi

University of Edinburgh Business School, Edinburgh, UK

S. Armagan Tarim

Cork University Business School, University College Cork, Ireland

Abstract

This paper introduces a new stochastic dynamic program (SDP) based heuristic to compute the (R, s, S) policy parameters for the non-stationary stochastic lot-sizing problem with backlogging of the excessive demand, fixed order and review costs, and linear holding and penalty costs. Our model combines a greedy relaxation of the problem that considers replenishment cycles independent with a modified version of Scarf's (s, S) SDP. A simple model implementation requires a prohibitive computational effort to compute the parameters. However, leveraging the K-convexity property and deploying memoisation techniques strongly reduce the computational effort required. The resulting algorithm is considerably faster than the state-of-the-art, extending its applicability by practitioners. An extensive computational study shows that our approach computes the optimal policy in more than 97% of the analysed instances, with a 0.02% average optimality gap.

Keywords: Inventory, (R,s,S) policy, demand uncertainty, stochastic lot sizing

Email addresses: andrea.visentin@insight-centre.org,s.prestwich@cs.ucc.ie (Andrea Visentin, Steven Prestwich), roberto.rossi@ed.ac.uk (Roberto Rossi), armagan.tarim@ucc.ie (S. Armagan Tarim)

1. Introduction

The computation of solutions for the non-stationary stochastic lot-sizing problem is a well-developed branch of inventory control (Axsäter, 2015). The stochasticity of the demand allows modelling the uncertainty of real-world problems, while its non-stationarity allows considering the seasonality or life cycle of products. Under this setting, the inventory must satisfy a demand represented by a set of stochastic variables of known probability distributions generally considered independent. Various policies have been developed to manage these systems (Silver, 1981). A policy defines when an order has to be placed and its quantity. This work focuses on the computation of (R, s, S) policy parameters for the single-item, single-echelon lot-sizing under ordering, holding and penalty cost, a widely studied and used class of inventory problems.

The (R, s, S) policy is a generalisation of Bookbinder & Tan's dynamic uncertainty — also known as (s, S) policy — and static-dynamic uncertainty — also known as (R, S) policy — strategies (Bookbinder & Tan, 1988). In the (R, s, S) policy, the inventory level is assessed at review intervals R; if it falls under the s level, an order is placed; the order raises the inventory level to S. If the cost of reviewing the inventory is negligible, the policy reviews the inventory in each period behaving as the (s, S) one. If the s level is set equal to S, an order is placed at each review. Figure 1 shows the inventory pattern of the policy. In the non-stationary stochastic problem configuration, the policy parameters change across the time horizon, assuming the $(\mathbf{R}, \mathbf{s}, \mathbf{S})$ form.

The (R, s, S) policy is widely used by practitioners (Silver, 1981). In the case of stochastic non-stationary problems, three sets of parameters have to be jointly optimised to minimise the expected cost. This task has been considered extremely difficult. In a recent work, Visentin et al. (2021) introduced the first algorithm to compute the optimal policy parameters. They apply a branch-and-bound approach to explore the possible replenishment plans while computing the order levels and order-up-to-levels using stochastic dynamic programming

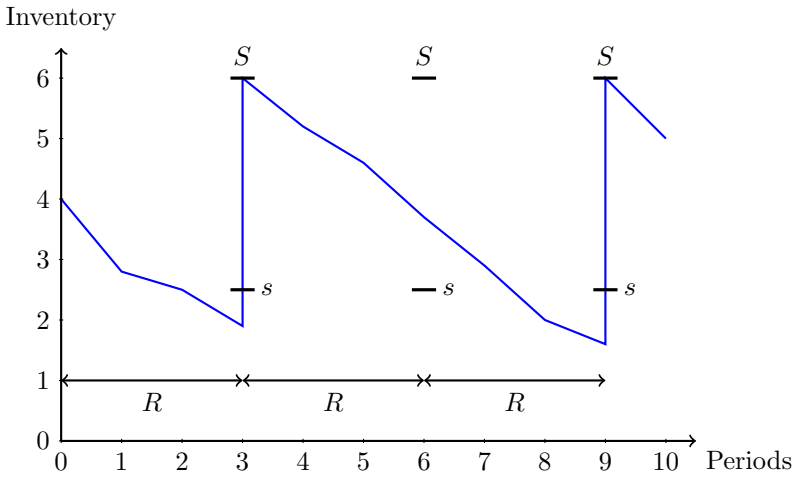


Figure 1: The expected inventory level under the (R, s, S) policy

(SDP). While their method computes the optimal set of parameters, it struggles to scale to big problems, limiting its applicability by practitioners.

In this work, we fill this gap in the literature by

- extending a well-known relaxation to the (R, s, S) policy computation and combining it with an SDP formulation to compute near-optimal policy parameters. The relaxation considers the replenishment cycles independently thus allowing a greedy backward heuristic to compute the replenishment plan;
- introducing computational enhancements that make the model computable in a reasonable time;
- analysing an extensive numerical study that shows that the heuristic computational effort significantly outperforms the optimal method;
- investigating the problem configurations for which the policy computed by the heuristic differs from the optimal one.

The paper is structured as follows. A survey of the literature is presented in section 2. Section 3 provides the description of the problem and of the best-known solution, later used as a comparison. Section 4 introduces the relaxation

used, the greedy approach and the computational enhancements. Section 5 shows a comprehensive numerical study. Finally, Section 7 concludes the paper.

2. Literature review

This section surveys the relevant stochastic lot-sizing literature. In the first part, we position our approach in comparison to other inventory control policies. We then analyse recent practical applications of the (R, s, S) policy.

2.1. Stochastic inventory control policies

An inventory control policy defines when: to assess the inventory, place an order, and the size of the order (Silver, 1981). The problem of computing policy parameters to satisfy a stochastic demand appears in a wide variety of industrial settings, and it has been extensively investigated in the literature. In a recent survey, Vidal (2022) analyse the state of the art in deterministic inventory control models, while Ma et al. (2019) reviews stochastic policy computation approaches. Arrow et al. (1951) is considered the first known work where such random demand is modelled with a cost configuration equivalent to the one considered in this work. They investigate a closed formula to compute the best maximum stock (order-up-to-level) and reorder level as functions of the demand on a continuous time setting. Bookbinder & Tan (1988) propose a broad framework of inventory control strategies: static uncertainty (also known as (R, Q) policy), dynamic uncertainty (also known as (s, S) policy), and static-dynamic uncertainty (also known as (R, S) policy). Bookbinder & Tan classify approaches based on when replenishment decisions are taken: at the beginning of the planning horizon, or after realising a period demand. In the (R, Q) policy, the full replenishment plan is fixed at the beginning. A fixed ordering plan is preferred in industrial settings where rigid production/shipment plans are needed. For these reasons, the computation of this policy under uncertainty has been widely investigated (see e.g. Sox, 1997; Meistering & Stadtler, 2017; Tunc, 2021). Scarf (1959) proves that the (s, S) policy (i.e. Bookbinder &

Tan's dynamic uncertainty strategy) is cost-optimal. In this policy, the decision to place an order and its quantity are taken after observing the demand. This policy is particularly effective in dealing with unexpected demand realisations. However, as shown in (De Kok & Inderfurth, 1997; Tunc et al., 2011), this policy suffers from a high degree of setup-oriented nervousness, meaning that the order timings frequently change, limiting its practical applicability. Recent works involving this policy are (Jiao et al., 2017; Xiang et al., 2018; Azoury & Miyaoka, 2020). Perera & Sethi (2022) presents a survey of discrete (s, S) policy and discusses its relevance in industrial settings. The (R, S) policy (i.e. Bookbinder & Tan's static-dynamic uncertainty strategy) fixes the replenishment moments at the beginning of the planning horizon and decides the size when placing the order. This policy may be appealing as it reduces setup-oriented nervousness, as discussed by Tunc et al. (2011); a known order schedule also allows better deals with the carriers. Relevant studies on this policy are (Tarim & Kingsman, 2004; Rossi et al., 2015; Tunc et al., 2018). One of the limitations of Bookbinder & Tan's classification is that it does not take into account the stock-taking cost commonly present in real-world problems (Fathoni et al., 2019; Christou et al., 2020). When a cost for reviewing the inventory is considered, the (R, s, S) policy features a lower expected total cost compared to other policies. Therefore, the (R, s, S) can be seen as a generalisation of the (s, S) and (R, S) policies.

2.2. (R,s,S) applications

The (R, s, S) policy has a vast number of applications in the literature; due to a reduced nervousness compared to the (s, S) and a better cost-performance than the (R, S). These policies have also been studied for different problem configurations. Schneider & Rinks (1991); Schneider et al. (1995) introduce two heuristics to compute (R, s, S) parameters in a two-echelon inventory system with one warehouse and multiple retailers. Strijbosch et al. (2002) propose a technique to simulate an (R, s, S) inventory system in which the parameters remain constant. It can compute fill rates or find parameters values to achieve a prescribed service level. Chen & Lin (2009) adopt a hedge-based (R, s, S) pol-

icy portfolio, with constant parameters in the short term, for a multi-product inventory control problem. In Cabrera et al. (2013) the (R, s, S) policy is used to manage the inventory of multiple warehouses. Göçken et al. (2015) use a simulation optimisation technique to determine the optimal policy for distribution centres in a two-echelon inventory system with lost sales. Johansson et al. (2020) use an (R, s, S) policy for controlling one-warehouse, multiple-retailer inventory systems; their configuration is motivated by a real problem faced by a company selling metal sheet products. In the surveyed papers, the policy parameters are optimised and kept constant across the time horizon; in other words, the R value is a given fixed value reducing the problem to an (s, S) policy. Further works operating in a stationary setting include: Lagodimos et al. (2012), who solves the continuous-time problem with stationary demand; and Christou et al. (2020), who extend Lagodimos et al.'s work to consider the order quantity as a multiple of given batch size. The focus on stationary settings is due to the complexity of jointly optimising the three sets of parameters. However, it is known that it is costly to adopt a stationary policy when the demand is nonstationary (Tunc et al., 2011). This motivated the study by Visentin et al. (2021). Their approach to computing (R, s, S) policy parameters uses a binary search tree to fix the optimal replenishment plan. The tree is travelled using branch-and-bound with tailor-made bounds based on Scarf's SDP that reduces its dimensionality by more than 99%. We will survey this approach in further detail in Section 3.1. However, Visentin et al.'s model requires considerable computational effort to solve big instances, limiting its practical value. To overcome this limitation, in this work we will replace the binary search tree with a greedy heuristic and leverage memoisation to avoid SDP recomputations.

In summary, the (R, s, S) policy has a wide variety of applications due to a number of advantages over other competing policies. The algorithm presented herein aims to boost its adoption by providing a heuristic that computes near-optimal policies using a fraction of the computational effort compared to the state-of-the-art.

3. Problem description

This work considers the single-item, single-stocking location, stochastic inventory control problem over a T-period planning horizon. The (R, s, S) policy defines three aspects of inventory management: the timing of inventory reviews, when an order is placed, and the order size. A review takes place when the inventory level in the warehouse is assessed; these moments are fixed at the beginning of the time horizon. An order can only be placed after a review takes place. The interval between two review moments represents a replenishment cycle.

The demand's stochasticity and non-stationarity of period t are modelled through the random variable d_t. Demands are independent variables with a known probability distribution. Cumulative demand of periods t to the beginning of period j takes the form of $d_{t,j}$ with $j > t$. If the demand in a given period exceeds the on-hand inventory, the excess is backlogged and carried to the next period. In Section 4.4, we extend the model to the lost-sales configuration, where the exceeding demand is lost; a common approach when competitors' products are available. Under these assumptions, the (R, s, S) policy takes the vectorial form form $(\mathbf{R}, \mathbf{s}, \mathbf{S})$, with $\mathbf{R} = (R_1, \ldots, R_T)$; where R_t , s_t and S_t denote respectively the length, the reorder-level and order-up-to-level associated with the t-th inventory review.

Policies are compared based on their expected cost. Stocktaking — the process by which physical inventory is reviewed — has a fixed cost of W. We denote by Q_t the quantity of the order placed in period t. Ordering costs are represented by a fixed value K and a linear cost, but we shall assume that the variable cost is zero without loss of generality. The extension of our solution to the case of a variable production/purchasing cost is straightforward, as this cost can be reduced to a function of the expected closing inventory level at the final period (see Tarim & Kingsman, 2004). At the end of each period, a holding cost h is charged for every unit carried from one period to the next. In case of a stockout, a penalty cost b is charged for each item and period. We denote with I_t the closing inventory level for period t, making I_0 the initial inventory.

We consider the problem of computing the $(\mathbf{R}, \mathbf{s}, \mathbf{S})$ policy parameters that minimize the expected total cost over the planning horizon. The order quantity Q_t is fixed at every review moment before the demand realisation using. The order is placed only if t is a review period and the open inventory is below the order level s_t.

The problem of computing the optimal $(\mathbf{R}, \mathbf{s}, \mathbf{S})$ can be formulated as follow:

$$C_1(I_0) \triangleq \min_{(\mathbf{R},\mathbf{s},\mathbf{S})} f_1(I_0, Q_1, R_1) + E[C_{1+R_1}(I_0 + Q_1 - d_{1,1+R_1})] \tag{1}$$

Where $C_1(I_0)$ is the expected cost of the optimal policy parameters starting at period 1 with the initial inventory I_0. In general, $C_t(I_{t-1})$ represent the expected inventory cost of starting at period t with open inventory I_{t-1}. While $f_t(I_{t-1}, Q_t, R_t)$ is the expected cost of a review cycle starting in period t and ending up in period $t + R_t$; it comprises review, ordering, holding and penalty cost for the review cycle, that is

$$\begin{aligned} f_t(I_{t-1}, Q_t, R_t) \triangleq \quad & K\mathbb{1}\{Q_t > 0\} + W + \sum_{i=1}^{R_t} E[h \max(I_{t-1} - d_{t,t+i} + Q_t, 0) \\ & + b \max(-I_{t-1} - Q_t + d_{t,t+i}, 0)]. \end{aligned} \tag{2}$$

$C_t(I_{t-1})$ values can be computed recursively, when all the policy parameters are known, using the following formula:

$$C_t(I_{t-1}) \triangleq f_t(I_{t-1}, Q_t, R_t) + E[C_{t+R_t}(I_{t-1} + Q_t - d_{t,t+R_t})]) \tag{3}$$

until the base case

$$C_{T+1}(I_T) \triangleq 0 \tag{4}$$

is reached. For a given $(\mathbf{R}, \mathbf{s}, \mathbf{S})$ parameters set, this formulation allows to compute the expected policy cost. However, the number of combinations of parameters is exponential, making this approach unusable for the computation of optimal ones.

3.1. Branch-and-bound approach

Visentin et al. (2021) present the first algorithm for computing the optimal parameters for the $(\mathbf{R}, \mathbf{s}, \mathbf{S})$ problem. Their work is based on the following lemma.

Lemma 1. *If the replenishment cycles ($\boldsymbol{R}$) are fixed, the problem is reduced to a particular version of the ($\boldsymbol{s}, \boldsymbol{S}$) policy computation (Scarf, 1959), and can be solved to optimality using SDP.*

In this case, the problem is formulated as follows:

$$\widehat{C}_1(I_0) = \min_{\mathbf{R}} f_1(I_0, Q_1, R_1) + E[C_{1+R_1}(I_0 + Q_1 - d_{1,1+R_1})], \tag{5}$$

where $\mathbf{s}$ and $\mathbf{S}$ are dependent on $\mathbf{R}$. The proposed baseline computes the optimal replenishment cycles by testing all $\mathbf{R}$ possible combinations and computing the optimal policy cost for each of them. Their best technique, our comparison in the experimental section, uses branch-and-bound to avoid recomputations and prune sub-optimal $\mathbf{R}$ assignment. Optimal s_t and S_t levels can be computed by considering only future periods when R_t is fixed, ignoring the expected opening inventory level; this is not valid for the computation of the $\mathbf{R}$ vector.

4. Heuristic technique

The heuristic introduced in this work aims to compute locally optimal R_t values to produce a near-optimal $(\mathbf{R}, \mathbf{s}, \mathbf{S})$ policy. The main idea is to move the assignment of the decision variable R_t at period t, and not to fix all of them at the beginning of the time horizon, as implied by Equation 5. Similar relaxations have been applied to the computation of the (R, S) policy (see e.g. Tarim, 1996; Rossi et al., 2011; Özen et al., 2012). Under this policy, the relaxation introduces the possibility of placing negative orders and discarding the items at no extra cost. In the (R, s, S) policy, the relaxation also does not consider the possibility that the opening inventory level is higher than the order level.

After applying this relaxation, Equation 3 becomes

$$\widehat{C}_t(I_{t-1}) = \min_{R_t} f_t(I_{t-1}, Q_t, R_t) + E[C_{t+R_t}(I_{t-1} + Q_t - d_{t,t+R_t})]). \quad (6)$$

Solving this recursion leads to different optimal R_t for different opening inventory levels I_{t-1}. For example, if the opening inventory level is slightly higher than s_t, but considerably lower than S_t, an order is not placed and the next review cycle may start earlier. However, in the (R, s, S) policy, the review cycles are fixed at the beginning of the time horizon and not after the demand realisation. This is the reason why we need to know the probability distribution of the opening inventory level to determine the optimal R_t. Our heuristics operates by choosing a locally optimal R_t, assuming that an order is placed in period t and the possibility of placing a negative order. We define these locally optimal replenishment cycles as R_t^a.

Knowing the expected cost of future periods $\widehat{C}_j$ with $j > t$, it is possible to compute the optimal s_t and S_t for that specific replenishment cycle R_t using SDP. The best S_t is the value that minimizes $\widehat{C}_t(S_t)$, since we place an order to reach the point with the lowest future expected cost:

$$S_t = \arg\min_{I_{t-1}} \widehat{C}_t(I_{t-1}). \quad (7)$$

So, assuming that an order is placed, the best replenishment cycle is the one that has the lowest cost after the inventory level is topped up to S_t:

$$R_t^a \triangleq \arg\min_{R_t} \widehat{C}_t(S_t). \quad (8)$$

As mentioned above, the computation of $\widehat{C}_t$ requires the expected costs of future periods $\widehat{C}_j$ with $j > t$, which are dependent on the optimal R_j. We relaxed the cost function by defining C_t^a as the expected cost of using local optimal R_j^a for all periods j after t. Given $C_{T+1}^a(I_T) = 0$, it is possible to compute the relaxed cost function in a backward way using the following approximate SDP functional equation:

$$C_t^a(I_{t-1}) \triangleq f_t(I_{t-1}, Q_t, R_t^a) + E[C_{t+R_t^a}^a(I_{t-1} + Q_t - d_{t,t+R_t^a})]. \quad (9)$$

This formula computes a near-optimal replenishment schedule $\mathbf{R}^a$, and the set of order and order-up-to levels optimal for that given schedule. Due to the relaxation, $\mathbf{R}^a$ can differ from the optimal $\mathbf{R}$; however, as the experimental section shows, this event is rare.

The resulting approximate SDP formulation is more complex than the (s, S) one, making the computational effort required to solve it prohibitive. This is mainly due to the computation of the expected cycle cost (Equation 2). This computation involves three variables in each period: current inventory, order size and length of the replenishment cycle. Associated computational effort can be considerably reduced by leveraging the K-convexity property (Scarf, 1959) within the (s, S) SDP formulation. The deployment of search reduction and memoisation techniques further improve the performances, and it has a crucial impact on the applicability of this model. In the next subsections, we present the pseudocode for the solution and how these enhancements affect it.

4.1. Pseudocode

Algorithm 1 shows the procedure to compute the heuristics backwards. Lines 1-2 contain the boundary condition. Line 3 goes through all the periods in backwards order. Line 5 searches through all the possible replenishment cycles, line 6 through all the inventory levels and line 7 through all the possible order quantities. Lines 12-13 save the current value of R_t^a according to Equation 8, while line 14 updates the relative expected costs, Equation 9.

For clarity and for the sake of the enhancements, we separate the computation of the immediate cost. Let $\zeta_{t,t+j}$ be a value of the random variable $d_{t,t+j}$ and $P(\zeta_{t,t+j})$ be the probability of assuming that value. Algorithm 2 computes the immediate cost, Equation 2.

4.2. K-convexity

We can exploit the property of K-convexity presented in Scarf (1959) in solving the SDP. This approach is widely used to optimise traditional (s, S) SDP computation.

Algorithm 1 RsS-SDP()

for i from $min\_inventory$ to $max\_inventory$ **do**
    $C^a_{T+1}(i) = 0$
end for
for t from T down to 1 **do**
    $best\_review\_cost \leftarrow \infty$
    for r from 1 to $T - t + 1$ **do**
        for i from $min\_inventory$ to $max\_inventory$ **do**
            $C_{cycle}(i) \leftarrow \infty$
            for q from 0 to max_order **do**
                $expected\_cost \leftarrow f_t(i, q, r) + E[C^a_{t+r}(i + q - d_{t,t+r})]$
                if $expected\_cost < C_{cycle}(i)$ **then**
                    $C_{cycle}(i) \leftarrow expected\_cost$
                end if
            end for
        end for
        if $min(C_{cycle}) < best\_review\_cost$ **then**
            $R^a_t \leftarrow r$
            $C^a_t \leftarrow C_{cycle}$
            $best\_review\_cost \leftarrow min(C_{cycle})$
        end if
    end for
end for

Algorithm 2 $f_t(i, q, r)$

```
cost ← W
if q > 0 then
    cost ← cost + K
end if
for j from 1 to r do
    for each ζ_{t,t+j} value of d_{t,t+j} do
        close_inv ← i + q − ζ_{t,t+j}
        if close_inv ≥ 0 then
            cost ← cost + h close_inv P(ζ_{t,t+j})
        else
            cost ← cost − b close_inv P(ζ_{t,t+j})
        end if
    end for
end forreturn cost
```

The property is defined as

Definition 1. *Let $K \geq 0$, then function $f(x)$ is K-convex if:*

$$K + f(a + x) - f(x) - a\left(\frac{f(x) - f(x - b)}{b}\right) \geq 0$$

for all positive a, b and x.

Scarf (1959) shows that considering s_t^* and S_t^* the optimal reorder level and order up-to level for period t:

$$C_t(I_{t-1}) = \begin{cases} f(I_{t-1}, 0) + E[C_{t+1}(I_{t-1} - d_t)] & s_t^* \leq I_{t-1} \leq S_t^* \\ f(I_{t-1}, 0) + E[C_{t+1}(S_t^* - d_t)] + K & 0 \leq I_{t-1} < s_t^*. \end{cases} \tag{10}$$

This is done by computing the $C_t(y)$ for different values of y starting from an upper bound of S_t. The value y is then decremented, and the lowest value of C_t is remembered. When the cost is greater than $C_t + K$ the search terminates. S_t is the inventory level in which the cost assumes the minimum value, s_t is the one

in which we stop the search. This approach greatly speeds up the computation of the SDP.

Similarly to the computation of the (s, S) policy, we can use the K-convexity property for the (R, s, S). Considering the Equation 9, for a fixed R_t the problem is reduced to an (s, S) one with the next $R_t - 1$ periods in which an order can not be placed.

Algorithm 3 shows the pseudocode of the enhanced SDP, clarifying the improvement's reason. For a fixed review cycle length R_t, there is no need to search for the best order quantity Q_t. When the order level s_t is determined, the lower inventory levels assume the same expected cost.

4.3. Cycle Cost Memoisation

The calculation of the cycle cost is particularly time demanding. There is a summation of expected costs over multiple periods. However, it is possible to identify situations in which the same computations occur multiple times. Let $l_t(I_t, R_t)$ be the function that computes the holding and penalty expected cost of starting at the end of period t with closing inventory I_t and with the next review moment in R_t periods. This new function is defined as:

$$l_t(I_t, R_t) \triangleq \sum_{i=1}^{R_t} E[h \max(I_t - d_{t+1,t+i}, 0) + b \max(-I_t + d_{t+1,t+i}, 0)] \tag{11}$$

considering $d_{i,j} = 0$ when $i = j$. Equation 2 can be rewritten as:

$$f_t(I_{t-1}, Q_t, R_t) = K\mathbb{1}\{Q_t > 0\} + W + l_t(I_{t-1} - d_{t,t+i} + Q_t, R_t). \tag{12}$$

The $l_t(I_t, R_t)$ function can be computed in a recursive way

$$l_t(I_t, R_t) = h \max(I_t, 0) + b \max(-I_t, 0) + E[l_{t+1}(I_t - d_{t+1}, R_t - 1)], \tag{13}$$

this can be considered as the functional equation of an SDP, where the holding/penalty cost of period t is the immediate cost. There are two boundary conditions:

$$l_{T+1}(I_T + 1, R_t) = 0 \tag{14}$$

$$l_t(I_t, 0) = 0. \tag{15}$$

Algorithm 3 RsS-SDP-KConv()

for i from $min\_inventory$ to $max\_inventory$ **do**
  $C^a_{T+1}(i) = 0$
end for
for t from T down to 1 **do**
  $best\_review\_cost \leftarrow \infty$
  for r from 1 to $T - t + 1$ **do**
    $best\_cycle\_cost \leftarrow \infty$
    for i from $max\_inventory$ down to $min\_inventory$ **do**
      $C_{cycle}(i) \leftarrow f_t(i, 0, r) + E[C^a_{t+1}(I_{t-1} + Q_t - d_t)]$
      if $C_{cycle}(i) < best\_cycle\_cost$ **then**
        $best\_cycle\_cost \leftarrow C_{temp}(i)$
        $S^{cycle}_t \leftarrow i$
      end if
      if $C^{cycle}_t(i) > best\_cycle\_cost + K$ **then**
        $s^{cycle}_t \leftarrow i$
        break for
      end if
    end for
    if $best\_cycle\_cost < best\_review\_cost$ **then**
      $R^a_t = r$
      $best\_review\_cost \leftarrow best\_cost\_cycle$
      for i from $min\_inventory$ to s^{cycle}_t **do**
        $C^a_t(i) \leftarrow C^{cycle}_t(s_t)$
      end for
      $C^a_t \leftarrow C_{cycle}$
    end if
  end for
end for

The states are represented by the tuple (t, I_t, R_t) and are computed in a forward manner. To avoid recomputations, we store the computed tuples in a dictionary with constant access time.

4.4. Unit cost and lost sales extensions

Similarly to (Visentin et al., 2021), unit ordering cost can be easily modelled as a function of the expected closing inventory or included in the immediate cost function. In the case of a stockout, the lost sales model is more common than a delay of the demand (Verhoef & Sloot, 2006), especially in a retail setting. Lost sales models have been underrepresented in the inventory control literature (Bijvank & Vis, 2012); however, many recent works are considering mixed lost-sales and backorder configurations (ElHafsi et al., 2021). The model presented herein can be adapted to include partially lost sales. Dos Santos & Oliveira (2019) defines as β the percentage of unmet demand that is backlogged; the remaining is lost. The functional equation 9 becomes

$$C_t^a(I_{t-1}) = \min_{0 \leq Q_t \leq M\gamma_t} (f_t(I_{t-1}, Q_t) + E[C_{t+1}^a(\max(I_{t-1} + Q_t - d_t, \beta(I_{t-1} + Q_t - d_t))]). \tag{16}$$

5. Experimental Results

This section conducts an extensive computational study of the heuristic presented in this paper. We aim to evaluate the quality of the policies computed by the heuristic and the computational effort required. In Section 5.1, we assess the computational effort required to compute a policy and the quality of the policy itself under an increasing time horizon. An analysis of the heuristics behaviour under different demand patterns and cost parameters is presented in Section 5.2. Finally, we analyse an example in which the algorithm computes a near-optimal replenishment plan.

For the experiments, we use as a comparison the branch-and-bound (BnB) technique presented in (Visentin et al., 2021). This is the only (R, s, S) solver

for this problem configuration available in the literature. We use the same solver to compute the optimality gap. The solvers are:

- **BnB-Guided**, the fastest branch-and-bound approach presented in (Visentin et al., 2021). It pre-computes an initial replenishment plan using (Rossi et al., 2015) to improve the computational performances.
- **SDP**, the basic implementation of the SDP heuristic model presented in Algorithm 1. We include this to appreciate the impact of the optimisation techniques deployed.
- **SDP-Opt**, the heuristic implementation deployed using the K-convexity property (Algorithm 3) and the immediate cost memoisation.

All experiments are executed on an Intel(R) Xeon E5640 Processor (2.66GHz) with 12 Gb RAM. For the sake of reproducibility, we made the implementation of all the techniques and the data generators available[1].

Since our approach is a heuristic, we use the optimality gap as a measure to compute the computed policy's quality. The optimality gap is the estimated extra cost of using the policy instead of the cost-optimal one for a particular problem. It is defined as

$$\text{Optimality gap} \triangleq \frac{\text{Policy cost} - \text{Optimal cost}}{\text{Optimal cost}}. \tag{17}$$

Better policy parameters exhibit a lower optimality gap. It can be used to estimate the inventory cost of deploying a non-optimal system.

5.1. Scalability

We used the same testbed presented in (Visentin et al., 2021). A fixed holding cost per unit $h = 1$. The other cost factors are sampled from uniform random variables: fixed ordering cost $K \in [80, 320]$, fixed review cost $W \in [80, 320]$ and linear penalty cost $b \in [4, 16]$. The demand is modelled as a series of Poisson random variables. A uniform random variable draws the average demands

[1] https://github.com/andvise/inventory-control

per period with a range of 30 to 70. We generate 100 different instances. We replicate the experiments for increasing values of the number of periods.

Figure 2 shows the logarithm of the average computational time over the 100 instances in comparison with the fastest technique available in the literature. The simple implementation of the heuristic can barely solve tiny instances before the time limit, making it useless for every practical use. Figure 2 shows that the reduction of computational effort provided by K-convexity and memoisation is considerable. The guided BnB slightly outperforms the optimised SDP for small instances up to 8 periods, and then the gap between the two strongly increases, making it able to solve instances more than twice as big in the same amount of time. The K-convexity performance improvement is more significant than the memoisation one. Moreover, it generally avoids the computation of all the DP states associated with a negative inventory (Algorithm 3, line 13). Memoisation offers a significant speed-up in computational times, which is greater in bigger instances. For bigger instances, the physical memory needed grows to require the usage of memory swap and a slow down in performances. In this testbed, the heuristic always computes the optimal replenishment plan.

5.2. Instance type analysis

These experiments aim to analyse the performances of the heuristic under different instance parameters. We want to analyse which cost parameters are affecting the computational performances and the optimality gap of the heuristic. We use a modified version of the instances used in (Visentin et al., 2021, Section 6.2); this is a commonly used testbed originally proposed by (Berry, 1972) and widely used in the literature (Özen et al., 2012; Dural-Selcuk et al., 2019; Xiang et al., 2018). The algorithm proposed herein computes the optimal policy parameters for all the instances used therein. Our extension aims to find problem settings where the heuristic under-performs the optimal approach. We do it by examining a wider range of review and ordering costs, and by adopting a normally distributed demand with varying standard deviation, expressed as a proportion of its mean value (i.e. via a coefficient of variation).

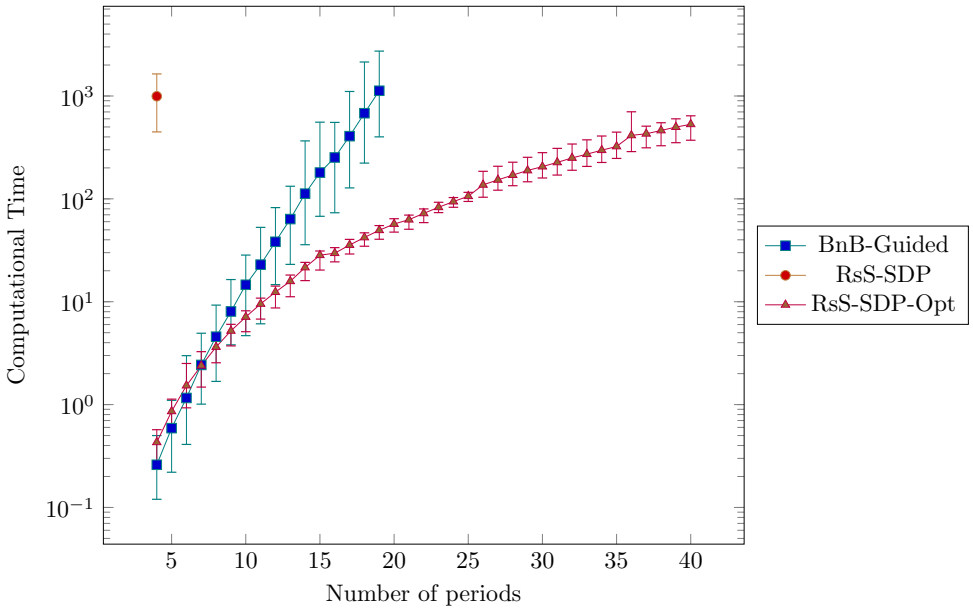


Figure 2: Caption: Computational time of the (R, s, S) SDP over the number of periods, time limit 1 hour.

We use two different planning horizons: 10 and 20 periods. For the cost parameters, we use all the possible combinations of review and ordering cost values $K, W \in \{20, 40, 80, 160, 320\}$, holding and penalty cost fixed respectively at $h = 1$ and $b = 10$. We consider Poisson demand and normally distributed one with a coefficient of variation $\rho \in \{0.1, 0.2, 0.3\}$, where $\rho = \sigma/\mu$, where σ denotes the standard deviation, and μ the expected value of the demand. In the literature, the coefficient of variation used is generally not higher than 0.3; because a higher value would lead to a non-negligible probability of observing negative demand values, which would violate the assumptions our model is built upon.

We consider six different demand patterns: stationary (STA), positive trend (INC), negative trend (DEC), two life-cycle trends (LCY1, LCY2) and an erratic one (RAND); more details on these patterns can be found in (Visentin et al., 2021). The combinations of the parameters mentioned above lead to the creation of 1200 instances.

Table 1 and Table 2 show the results for the 10 and 20-period instances. Regarding the policies' quality, we consider the average optimality gap, the percentage of computed policies that differs for the optimal and their optimality gap. We also compare the time required to compute the policies and the average number of reviews. Figure 3 shows the boxplots of the optimality gap for the instances in which the two algorithms compute different policies.

The cost factors suggest that the algorithm does not compute the optimal policy in situations with a high ordering cost and a low review cost. Due to the relaxation, the approximate SDP computes the parameters of the cycles based only on the state values, considering the uncertainty of the future periods, but ignoring the one related to the period opening inventory. When the review cost is low, the BnB uses more review periods compared to the SDP to counteract this uncertainty. Up to 7.33% and 9.33% (for the 10 and 20 instances) of the policy computed differ from the optimal one; however, their gap averages less than 1%. Figure 3 shows that even in the worse case, the optimality gap does not exceed 2% and that even in the bigger instances with high uncertainty, the majority is under 1%. The average optimality gap across all the instances with the lowest review cost is 0.09% and 0.1%. A higher ordering cost leads to longer intervals between orders, so a higher uncertainty on the opening inventory level of a period. This leads to a maximum of 6.67% and 7.5% of near-optimal policies. While for these particular settings, the percentage of non-optimal policy is relatively high; their optimality gap is low.

The direct correlation between the demand uncertainty and the optimality gap is evident. When realisations strongly differ from the expected demand, it is more likely that the opening inventory level is higher than the order level in a review moment. Our approach selects the review plan by considering that an order takes place in those periods anyway. If the demand is low compared to its expected value, it can also happen that the inventory level is higher than the order-up-to-level. In these cases, our approach relaxes the problem by placing a negative order and setting the inventory to S, so the policy is not optimal. In essence, SDP performance is worse for increasing ρ.

The pattern analysis provides interesting insights. The approach performs better with the increasing (INC) pattern regardless of the other instance parameters; it always computes the optimal policy. We have the worse performance in the decreasing (DEC) and random (ERR) ones. This is in line with Özen et al. (2012) that considers a similar problem relaxation. If we have increasing demands, the base stock levels likely increase as well to satisfy higher demands. If the base stock levels increase monotonically, the relaxation generally computes the optimal policy. The random pattern suffers similar drawbacks due to randomly generated decreasing patterns. We observe the biggest gap between the number of reviews with 0.05 and 0.7 fewer reviews on average when the demand is random.

On average, the optimality gap between the two approaches is only 0.01% and 0.02%, with 2.53% and 2.67% of the policy computed that are near-optimal for the 10 and 20-period instances respectively. This proves the quality of the heuristic in computing policies.

Our approach is 3.4 and 194 times faster, respectively, on the 10 and 20 periods regarding computational time. Moreover, the cost parameters do not affect the SDP performances, while they affect the BnB pruning efficacy. For example, low-review cost 20-period instances take more than 10 times more effort than high-review ones. Uncertainty on the forecast affects the performances of both approaches since it makes the computation of a state's expected cost more expensive. However, the SDP manages to reduce this impact using memoisation. The SDP approximately double its computational effort for $\rho = 0.3$ compared to $\rho = 0.1$, while the increment for the BnB approach is higher than 20.

5.2.1. Non-optimality of the relaxation

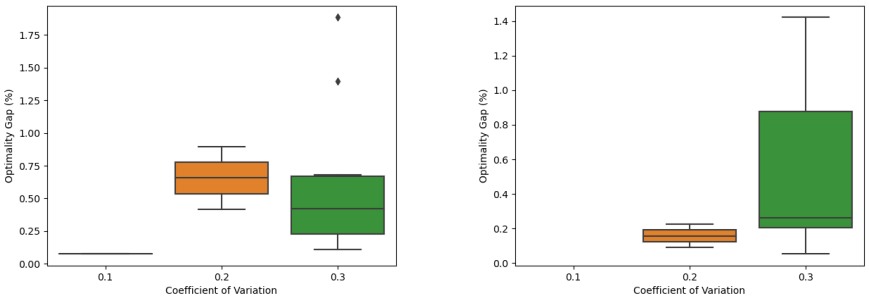
This section analyses a single instance to understand the differences between the computed policies better. This example shows a situation in which the heuristic computes a non-optimal policy. When computing the solution, it considers only the expected demand for future periods. On the other hand, the BnB approach presented in (Visentin et al., 2021) tests all the possible re-

		Optimality			Time (min)		Nr Reviews		Expected cost error	
		Optimality Gap	% Non-Optimal	Non-Optimal OG	SDP	BnB	SDP	BnB	SDP	BnB
K values	20	0.0 (0.0, 0.0)	0.0	0.0 (0.0, 0.0)	0.22	0.61	5.18	5.19	0.17	0.17
	40	0.0 (0.0, 0.01)	0.0	0.0 (0.0, 0.0)	0.22	0.7	4.52	4.53	0.17	0.16
	80	0.01 (0.0, 0.01)	1.67	0.24 (0.02, 0.47)	0.23	0.74	3.7	3.72	0.15	0.14
	160	0.01 (0.0, 0.02)	3.33	0.3 (0.1, 0.51)	0.24	0.87	3.09	3.12	0.13	0.12
	320	0.05 (0.02, 0.09)	6.67	0.8 (0.46, 1.14)	0.25	1.0	2.38	2.45	0.09	0.09
W values	20	0.06 (0.03, 0.1)	9.17	0.65 (0.37, 0.92)	0.22	1.04	5.25	5.36	0.16	0.16
	40	0.01 (0.0, 0.02)	1.67	0.45 (0.08, 0.83)	0.22	1.02	4.51	4.53	0.16	0.16
	80	0.0 (0.0, 0.0)	0.0	0.0 (0.0, 0.0)	0.23	0.83	3.71	3.71	0.15	0.15
	160	0.0 (0.0, 0.01)	0.83	0.07 (0.03, 0.1)	0.25	0.62	3.06	3.06	0.13	0.13
	320	0.0 (0.0, 0.0)	0.0	0.0 (0.0, 0.0)	0.25	0.4	2.34	2.34	0.11	0.11
Poisson	p	0.0 (0.0, 0.0)	0.0	0.0 (0.0, 0.0)	0.17	0.47	3.65	3.65	0.12	0.12
σ values	0.1	0.0 (0.0, 0.0)	0.67	0.07 (0.03, 0.11)	0.16	0.24	3.68	3.68	0.08	0.08
	0.2	0.01 (0.0, 0.02)	1.33	0.66 (0.26, 1.05)	0.24	0.69	3.81	3.81	0.16	0.16
	0.3	0.05 (0.02, 0.08)	7.33	0.61 (0.33, 0.89)	0.36	1.74	3.95	4.05	0.2	0.2
Pattern	STA	0.02 (0.01, 0.03)	2.0	0.37 (0.0, 0.84)	0.13	0.64	3.82	3.85	0.12	0.12
	INC	0.0 (0.0, 0.0)	0.0	0.0 (0.0, 0.0)	0.31	0.82	4.01	4.01	0.13	0.13
	DEC	0.04 (0.0, 0.08)	5.0	0.84 (0.35, 1.32)	0.27	1.13	3.37	3.41	0.12	0.12
	LCY1	0.0 (0.0, 0.0)	0.0	0.0 (0.0, 0.0)	0.2	0.76	3.95	3.96	0.17	0.17
	LCY2	0.01 (0.0, 0.02)	2.0	0.33 (0.17, 0.48)	0.25	0.78	3.78	3.81	0.19	0.19
	ERR	0.03 (0.0, 0.05)	5.0	0.51 (0.13, 0.88)	0.24	0.57	3.71	3.76	0.11	0.11
Average		0.02 (0.01, 0.02)	2.33	0.58 (0.35, 0.81)	0.23	0.78	3.77	3.8	0.14	0.14

Table 1: Optimality gap and pruning percentage for the techniques for instances of 10 periods. Between brackets, the 90% confidence intervals of the optimality gaps.

		Optimality			Time (min)		Nr Reviews		Expected cost error	
		Optimality Gap	% Non-Optimal	Non-Optimal OG	SDP	BnB	SDP	BnB	SDP	BnB
K values	20	0.0 (0.0, 0.0)	0.0	0.0 (0.0, 0.0)	1.83	197.65	10.33	10.33	0.04	0.04
	40	0.0 (0.0, 0.01)	0.83	0.26 (0.0, 0.0)	1.75	242.99	9.06	9.09	0.04	0.04
	80	0.0 (0.0, 0.01)	1.67	0.14 (0.0, 0.28)	1.78	318.49	7.36	7.37	0.05	0.05
	160	0.01 (0.0, 0.03)	3.33	0.41 (0.1, 0.71)	1.87	418.64	6.07	6.11	0.05	0.05
	320	0.05 (0.02, 0.08)	7.5	0.62 (0.35, 0.89)	1.76	576.3	4.63	4.79	0.04	0.04
W values	20	0.06 (0.03, 0.09)	10.83	0.55 (0.34, 0.76)	1.68	657.18	10.42	10.63	0.04	0.04
	40	0.01 (0.0, 0.01)	2.5	0.2 (0.13, 0.26)	1.85	537.93	9.07	9.1	0.04	0.04
	80	0.0 (0.0, 0.0)	0.0	0.0 (0.0, 0.0)	1.75	340.27	7.32	7.32	0.05	0.05
	160	0.0 (0.0, 0.0)	0.0	0.0 (0.0, 0.0)	1.86	161.08	6.04	6.04	0.05	0.05
	320	0.0 (0.0, 0.0)	0.0	0.0 (0.0, 0.0)	1.85	57.62	4.59	4.59	0.04	0.04
Poisson	p	0.0 (0.0, 0.0)	0.0	0.0 (0.0, 0.0)	1.34	130.59	7.37	7.37	0.04	0.04
σ values	0.1	0.0 (0.0, 0.0)	0.0	0.0 (0.0, 0.0)	1.27	46.83	7.37	7.37	0.03	0.03
	0.2	0.0 (0.0, 0.0)	1.33	0.16 (0.04, 0.27)	1.89	279.82	7.51	7.51	0.05	0.05
	0.3	0.05 (0.02, 0.08)	9.33	0.53 (0.34, 0.73)	2.69	946.02	7.71	7.9	0.06	0.06
Pattern	STA	0.01 (0.0, 0.03)	2.0	0.6 (0.04, 1.16)	0.81	286.97	7.48	7.55	0.02	0.02
	INC	0.0 (0.0, 0.0)	0.0	0.0 (0.0, 0.0)	2.18	177.57	7.68	7.68	0.04	0.04
	DEC	0.02 (0.0, 0.04)	3.0	0.55 (0.0, 1.27)	2.09	784.24	7.05	7.11	0.08	0.07
	LCY1	0.01 (0.0, 0.02)	3.0	0.35 (0.08, 0.61)	1.47	251.47	7.88	7.92	0.04	0.04
	LCY2	0.01 (0.0, 0.03)	4.0	0.32 (0.11, 0.53)	1.98	285.64	7.35	7.4	0.05	0.04
	ERR	0.03 (0.0, 0.05)	4.0	0.64 (0.22, 1.06)	2.26	318.99	7.5	7.57	0.04	0.04
Average		0.01 (0.01, 0.02)	2.67	0.48 (0.31, 0.66)	1.8	350.81	7.49	7.54	0.04	0.04

Table 2: Optimality gap and pruning percentage for the techniques for instances of 20 periods. Between brackets, the 90% confidence intervals of the optimality gaps.



(a) (b)

Figure 3: Optimality Gap for the instances in which the heuristics fail to compute the optimal policy. **(a)** for the 10-period instances and **(b)** for the 20-period

plenishment combinations of the previous periods during the search process. Not considering the previous demands means ignoring the possibility of having such a low demand that at a period t, the opening inventory level I_t is higher than s_t, which will strongly affect future decisions. This difference worsens the heuristic's performance under high uncertainty and decreasing patterns (DEC and ERR). In these instances, the high demand with high uncertainty at the beginning of the time horizon makes unexpected high inventory levels at a replenishment moment more likely. In this situation, the BnB solution adds more review moments (especially when the cost associated W is low) to assess the inventory level and react to the uncertainty.

For example, considering the instance of Table 1 with $K = 320$, $W = 20$, $\rho = 0.3$ and decreasing demand pattern. Table 3 shows the two policies. The BnB approach considers the higher uncertainty at the beginning of the time horizon; it also reviews the inventory level at period 5. While this review adds an extra cost in an almost deterministic system, it allows a better reaction to unexpected demand. At the end of the time horizon, the uncertainty on the inventory level is lower, and the two policies are identical from period 6 on when a lower demand leads to lower absolute variations of the realised demand. The BnB policy has an expected cost of 1706, the SDP of 1737, a difference of 31

Period		1	2	3	4	5	6	7	8	9	10	Policy cost
	γ_t	1	0	0	1	0	0	0	1	0	0	
RsS-SDP	S_t	282	-	-	242	-	-	-	53	-	-	1737
	s_t	206	-	-	170	-	-	-	25	-	-	
	γ_t	1	0	0	1	1	0	0	1	0	0	
RsS-BnB	S_t	302	-	-	242	186	-	-	53	-	-	1706
	s_t	212	-	-	45	111	-	-	25	-	-	

Table 3: (R, s, S) policy parameters for the $K = 320$, $W = 20$, $\sigma = 0.4$, DEC pattern instance.

that leads to an optimality gap of 1.8%. Figure 4 shows the average simulated cost of the two policies over 100000 demand simulations. We can see that the optimal policy has increased review costs that lead to lower holding, ordering and penalty costs.

6. Discussion

The experimental analysis shows that the proposed algorithm scale better than (Visentin et al., 2021), with limited degradation of the quality of the computed policy only for high demand uncertainty. The basic formulation requires a prohibitive computational effort. The two enhancements based on K-convexity (Scarf, 1959) and memoisation strongly improve the computational performance, making it able to solve instances twice as big as the state-of-the-art and making it two orders of magnitude faster in 20-period instances. This allows practitioners to use such a policy in a wider range of real-world situations. We then investigated the SDP performance under different types of instances. We measured the computational effort to compute the policy and how much the relaxation affects its quality. The heuristic's computational effort is less affected by the instance configuration. The proposed algorithm rarely computes a non-optimal policy when there is less uncertainty on demand and high review, low fixed ordering cost instances. Even in the rare worse instances, the added cost of using the heuristic never exceeds 2%. For Poisson distributed demand, the

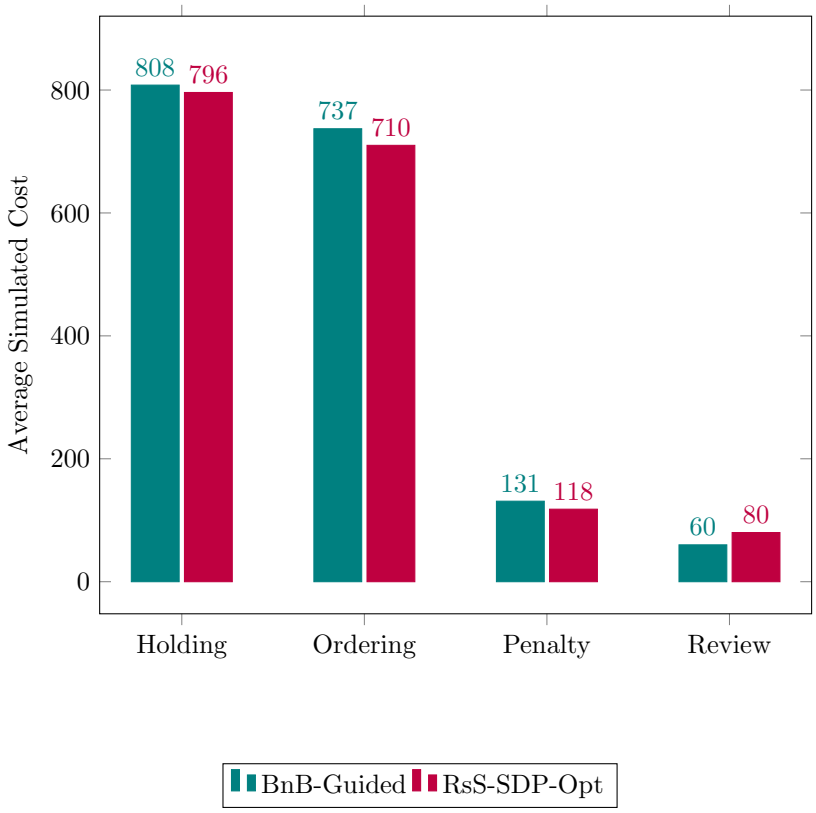


Figure 4: Caption: Comparison of the average cost of a policy over 100 000 demand simulations.

SDP always computes the optimal policy. Regarding the demand pattern, the heuristic produces sub-optimal replenishment plans when the demand decrease over time. The average optimality gap is 0.04% and 0.05% with 95.6% and 94.13% of computed policies identical to the optimal respectively for the 10 and the 20-period instances; almost all the non-optimal policies are related to high uncertainty of the demand ($\rho = 0.3$). These differences are caused by reduced review moments in the SDP computed policies. Overall, the computational benefits of heuristics make it the better choice for practitioners. If the tackled problem is of limited size, the review cost is low compared to the ordering cost, and the uncertainty of the demand prediction is high, the optimal algorithm is still to be preferred.

7. Conclusions

This paper presented a heuristic for the non-stationary stochastic lot-sizing problem with ordering, review, holding and penalty cost, a well-known and widely used inventory control problem. Computing (R, s, S) policy parameters is computationally hard due to the three sets of parameters that must be jointly optimised. We presented the first pure SDP formulation for such a problem. The algorithm introduced solves to optimality a relaxation of the original problem, in which review cycles are considered independently, and items can be returned/discarded at no additional cost. A similar relaxation has been previously used in (R, S) policy computation works. The extensive numerical study proved the reduction of the computational effort needed to compute a policy. The heuristic computes the optimal policy in most cases.

In future studies, we plan to extend such a method's applicability by considering more complex supply chains such as multiple items, multiple echelons, and different cost structures. We plan to further enhance the current formulation to improve the non-optimal computed policies, similarly to what (Rossi et al., 2011) did with state space augmentation.

Acknowledgments

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 12/RC/2289-P2, which is co-funded under the European Regional Development Fund.

References

Arrow, K. J., Harris, T., & Marschak, J. (1951). Optimal inventory policy. *Econometrica: Journal of the Econometric Society*, (pp. 250–272).

Axsäter, S. (2015). *Inventory control*. International Series in Operations Research & Management Science (3rd ed.). Basel, Switzerland: Springer International Publishing.

Azoury, K. S., & Miyaoka, J. (2020). Optimal and simple approximate solutions to a production-inventory system with stochastic and deterministic demand. *European Journal of Operational Research*, *286*, 178–189.

Berry, W. L. (1972). Lot sizing procedures for requirements planning systems: A framework for analysis. *Production and Inventory Managements*, *13*, 19–34.

Bijvank, M., & Vis, I. F. (2012). Inventory control for point-of-use locations in hospitals. *Journal of the Operational Research Society*, *63*, 497–510.

Bookbinder, J. H., & Tan, J.-Y. (1988). Strategies for the probabilistic lot-sizing problem with service-level constraints. *Management Science*, *34*, 1096–1108.

Cabrera, G., Miranda, P. A., Cabrera, E., Soto, R., Crawford, B., Rubio, J. M., & Paredes, F. (2013). Solving a novel inventory location model with stochastic constraints and inventory control policy. *Mathematical Problems in Engineering*, *2013*.

Chen, Y. M., & Lin, C.-T. (2009). A coordinated approach to hedge the risks in stochastic inventory-routing problem. *Computers & Industrial Engineering*, *56*, 1095–1112.

Christou, I., Skouri, K., & Lagodimos, A. (2020). Fast evaluation of a periodic review inventory policy. *Computers & Industrial Engineering*, (p. 106389).

De Kok, T., & Inderfurth, K. (1997). Nervousness in inventory management: comparison of basic control rules. *European Journal of Operational Research*, *103*, 55–82.

Dos Santos, F. S. P., & Oliveira, F. (2019). An enhanced l-shaped method for optimizing periodic-review inventory control problems modeled via two-stage stochastic programming. *European Journal of Operational Research*, *275*, 677–693.

Dural-Selcuk, G., Rossi, R., Kilic, O. A., & Tarim, S. A. (2019). The benefit of receding horizon control: Near-optimal policies for stochastic inventory control. *Omega*, (p. 102091).

ElHafsi, M., Fang, J., & Hamouda, E. (2021). Optimal production and inventory control of multi-class mixed backorder and lost sales demand class models. *European Journal of Operational Research*, *291*, 147–161.

Fathoni, F. A., Ridwan, A. Y., & Santosa, B. (2019). Development of inventory control application for pharmaceutical product using abc-ved cycle counting method to increase inventory record accuracy. In *2018 International Conference on Industrial Enterprise and System Engineering (ICoIESE 2018)*. Atlantis Press.

Göçken, M., Boru, A., Dosdoğru, A. T., & Geyik, F. (2015). (r, s, s) inventory control policy and supplier selection in a two-echelon supply chain: An optimization via simulation approach. In *2015 Winter Simulation Conference (WSC)* (pp. 2057–2067). IEEE.

Jiao, W., Zhang, J.-L., & Yan, H. (2017). The stochastic lot-sizing problem with quantity discounts. *Computers & Operations Research*, *80*, 1–10.

Johansson, L., Sonntag, D. R., Marklund, J., & Kiesmüller, G. P. (2020). Controlling distribution inventory systems with shipment consolidation and com-

pound poisson demand. *European Journal of Operational Research*, *280*, 90–101.

Lagodimos, A., Christou, I., & Skouri, K. (2012). Computing globally optimal (s, s, t) inventory policies. *Omega*, *40*, 660–671.

Ma, X., Rossi, R., & Archibald, T. (2019). Stochastic inventory control: A literature review. *IFAC-PapersOnLine*, *52*, 1490–1495.

Meistering, M., & Stadtler, H. (2017). Stabilized-cycle strategy for capacitated lot sizing with multiple products: Fill-rate constraints in rolling schedules. *Production and Operations Management*, *26*, 2247–2265.

Özen, U., Doğru, M. K., & Tarim, S. A. (2012). Static-dynamic uncertainty strategy for a single-item stochastic inventory control problem. *Omega*, *40*, 348–357.

Perera, S. C., & Sethi, S. P. (2022). A survey of stochastic inventory models with fixed costs: Optimality of (s, s) and (s, s)-type policies—discrete-time case. *Production and Operations Management*, .

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.

Rossi, R., Tarim, S. A., Hnich, B., & Prestwich, S. (2011). A state space augmentation algorithm for the replenishment cycle inventory policy. *International Journal of Production Economics*, *133*, 377–384.

Scarf, H. (1959). The optimality of (s, s) policies in the dynamic inventory problem, .

Schneider, H., & Rinks, D. B. (1991). Empirical study of a new procedure for allocating safety stock in a wholesale inventory system. *International Journal of Production Economics*, *24*, 181–189.

Schneider, H., RNKS, D. B., & Kelle, P. (1995). Power approximations for a two-echelon inventory system using service levels. *Production and Operations Management*, *4*, 381–400.

Silver, E. A. (1981). Operations research in inventory management: A review and critique. *Operations Research*, *29*, 628–645.

Sox, C. R. (1997). Dynamic lot sizing with random demand and non-stationary costs. *Operations Research Letters*, *20*, 155–164.

Strijbosch, L. W. G., Moors, J. J. A. et al. (2002). *Simulating an (R, s, S) inventory system*. Tilburg University.

Tarim, S. (1996). *Dynamic lotsizing models for stochastic demand in single and multi-echelon inventory systems*. Ph.D. thesis PhD thesis, Lancaster University.

Tarim, S. A., & Kingsman, B. G. (2004). The stochastic dynamic production/inventory lot-sizing problem with service-level constraints. *International Journal of Production Economics*, *88*, 105–119.

Tunc, H. (2021). A mixed integer programming formulation for the stochastic lot sizing problem with controllable processing times. *Computers & Operations Research*, *132*, 105302.

Tunc, H., Kilic, O. A., Tarim, S. A., & Eksioglu, B. (2011). The cost of using stationary inventory policies when demand is non-stationary. *Omega*, *39*, 410–415.

Tunc, H., Kilic, O. A., Tarim, S. A., & Rossi, R. (2018). An extended mixed-integer programming formulation and dynamic cut generation approach for the stochastic lot-sizing problem. *INFORMS Journal on Computing*, *30*, 492–506.

Verhoef, P. C., & Sloot, L. M. (2006). Out-of-stock: reactions, antecedents, management solutions, and a future perspective. In *Retailing in the 21st Century* (pp. 239–253). Springer.

Vidal, G. H. (2022). Deterministic and stochastic inventory models in production systems: a review of the literature. *Process Integration and Optimization for Sustainability*, (pp. 1–22).

Visentin, A., Prestwich, S., Rossi, R., & Tarim, S. A. (2021). Computing optimal (r, s, s) policy parameters by a hybrid of branch-and-bound and stochastic dynamic programming. *European Journal of Operational Research*, .

Xiang, M., Rossi, R., Martin-Barragan, B., & Tarim, S. A. (2018). Computing non-stationary (s, s) policies using mixed integer linear programming. *European Journal of Operational Research*, *271*, 490–500.