Erasmus University Rotterdam

Erasmus School of Economics

Bachelor Thesis Econometrie en Operationele Research

Two approaches for eliminating the nonlinear cost function from the mixed-integer programming formulation of the stochastic lot-sizing problem

Floor Peters (511106)

Supervisor:	W. van den Heuvel
Second assessor:	A.P.M. Wagelmans
Date final version:	2nd July 2023

Abstract

In this study, we present two solution approaches for the stochastic lot-sizing problem adhering to the static-dynamic uncertainty strategy. Both approaches eliminate the nonlinear cost function from the mixed-integer programming formulation, while still achieving optimal solutions with any desired level of accuracy. The first (*static*) approach approximates the nonlinear cost function with a predefined set of linear functions. The second (*dynamic*) approach estimates the cost function on the fly by dynamically incorporating cuts into the formulation. Both approaches are capable of solving variants of the lot-sizing problem that involve penalty costs and service level constraints. The dynamic cut generation approach yields superior computational performance compared to the static approach, especially when considering long planning horizons. We extend this dynamic approach for the more general setting where multiple items are kept in inventory simultaneously to address the multi-item lot-sizing problem. An aggregated service level is imposed for the items combined, which allows the company to flexibly assign service levels to the individual items, thereby reducing costs. As instances for this problem quickly become too complex for the dynamic approach, we propose a heuristic that finds satisfactory solutions within reasonable computational time.

1 Introduction

Optimizing inventory management is crucial for achieving business success, ensuring smooth operations and maximum profitability. Solving the lot-sizing problem coincides with finding the optimal inventory plan, which involves deciding on *when* to reorder and if so, *how much* to order. Ideally, this schedule would satisfy demand in all periods while minimizing total costs. It is a trade-off between avoiding stock-outs and minimizing inventory costs. In the field of the stochastic lot-sizing problem, where some of the parameters such as demand may be uncertain, the static-dynamic uncertainty strategy is known for its practical benefits. With this strategy, orders are placed at predetermined moments in time and the order size is determined upon observing the inventory level. These fixed replenishment periods make it easier for players in the supply chain to collaborate and make consolidated shipping possible. When deciding on the order quantity by looking at the prevailing inventory level, one can take into account the realised demand so far, which was unknown beforehand.

Recently, more research is done on the static-dynamic uncertainty strategy for the stochastic version of the lot-sizing problem, where demand is a random variable. In this paper, we apply two solution approaches proposed by Tunc et al. (2018) to solve the stochastic problem, and compare their computational efficiency. Firstly, we use a static approach that approximates the nonlinear cost function with a piece-wise linear function. Secondly, we apply a dynamic cut generation approach that estimates the cost function on the fly. Both solution methods can handle multiple variants of the problem, of which we look at two. One penalizes unsatisfied demand, and the other imposes a service level constraint. The first research question we aim to answer is: "How efficient are approaches that approximate the non-linear cost function in solving the stochastic lot-sizing problem?"

Afterwards, we extend the two solution approaches to investigate their applicability to the multi-item lot-sizing problem. In practice, it is common to keep multiple items in inventory

simultaneously. Companies often require a minimum aggregated service level for the items combined (e.g. 90%), and a less strict minimum service level for the items individually (e.g. 85%). Consequently, it is possible for certain individual service levels to be below the aggregated service level if others are above it. This way, a company gains flexibility in choosing the individual levels such that the required aggregated level is achieved. As these individual service levels are selected through an optimization process, there is a possibility of reducing the overall costs. This strategy may be of practical relevance when items vary a lot in terms of expected demand, order cost or holding cost. The second research question we address is: "Are the approaches that approximate the non-linear cost function applicable to the multi-item problem as well, and how much of the total costs could be saved by implementing an aggregated service level?"

Our computational experiments show that for the single-item lot-sizing problem, the dynamic cut generation approach is a considerably better solution method than the static approach. Especially when considering long planning horizons, this approach yields superior computational performance. For the multi-item lot-sizing problem, the instances quickly become too large for the two approaches. However, we propose a heuristic that leads to an average cost reduction of almost 2% when imposing the aggregated service level.

In Section 2, we summarize the relevant literature for our research. Section 3 gives an overview of the problem and the notation that we use throughout the paper. The different solution methods that we use to solve the problem are given in Section 4. Section 5 contains a brief description of the data and how it is obtained. In Section 6, we conduct computational experiments with the different methods and compare their efficiency in solving the problem. Finally, in Section 7, we answer our research questions and give suggestions for further research.

2 Literature

A broad variety of heuristic methods are present in the literature, designed to find the optimal inventory plan for the stochastic lot-sizing problem based on the static-dynamic uncertainty strategy. As Tunc et al. (2018) state, these methods can be divided into two categories. On the one hand, there exist methods that cannot be solved by ready-to-use software. They require tailor-made programs which cannot be easily obtained, and the more effective, the more complex these methods are. On the other hand, there exist mixed integer programming (MIP) models, that can be solved by ready-to-use solvers and are easier to use. However, the presence of a nonlinear cost function makes it difficult to solve these models. Rossi et al. (2015) address this problem by using a piece-wise linear approximation of the nonlinear cost function. Their contribution lies in their ability to solve various variants of the problem, characterized by penalty costs for unsatisfied demand and constraints on the service level. Their approach finds near-optimal solutions, but requires a relatively long solving time. Tunc et al. (2014) propose an efficient reformulation of MIP models that were already present in the literature. Their results show that this new model has a superior computational performance compared to methods that already existed. The problem can be solved on large instances within a minute due to their strong linear relaxation. However, their model is not able to solve many variants of the problem.

In the work of Tunc et al. (2018), the methods of Rossi et al. (2015) and Tunc et al. (2014) are combined to benefit from the strengths of both studies. By doing so, it is possible to attain both the superior computation time and the ability to solve multiple variants of the problem. Their model again approximates the nonlinear cost function with a piece-wise linear function. On top of that, they present an extended formulation where no predefined piece-wise linear approximation is needed. Instead, a dynamic cut generation approach is used to estimate the nonlinear cost function on the fly. They claim that this approach finds an optimal solution with any desired level of precision, without increasing the computation time too much. In this paper, we apply both approaches from Tunc et al. (2018) to compare their performances.

However, their approaches are intended to solve the single-item stochastic lot-sizing problem. This can be extended to a more realistic situation, where multiple items are kept in inventory simultaneously. There exist various studies about the multi-item lot-sizing problem, considering different strategies and types of service level constraints. In this study, we employ an α service level constraint for the items individually, which requires that the probability of not reaching a stock-out remains above α during the entire planning horizon. For the multi-item problem, we impose a similar constraint for the items combined, ensuring the maintenance of an aggregated service level. Earlier studies addressed similar variants of the problem.

Bookbinder and Tan (1988) proposed a model for solving the stochastic lot-sizing problem with an α service level constraint, sticking to the static-dynamic strategy. They were among the first researchers to include stochastic demand into their models. However, they did not yet address the multi-item variant of the problem. While many studies, including the work by Tunc et al. (2018), continued to focus on the single-item lot-sizing problem, Gruson et al. (2018) explored the concept of an α service level in a multi-item context. Hence, they imposed a constraint on the service level of the items combined as well. However, in their models, they assumed demand to be deterministic rather than stochastic. In a recent study, Sereshti et al. (2021) combined both concepts by introducing a model that addresses the stochastic lot-sizing problem within a multi-item context, by incorporating a constraint on the aggregated service level. With their static strategy, both the timing and the size of the orders are known at the beginning of the planning horizon, and remain unchanged throughout this period. In our study, however, we use the static-dynamic strategy. This is more cost-efficient as the order size is determined upon observing the current inventory level. This way, one can account for the realised demand so far, which was uncertain before.

To conclude, research on the static-dynamic uncertainty strategy for the multi-item stochastic lot-sizing problem with an aggregated service level is still lacking in the literature. Our study makes a valuable contribution to the existing literature in this field.

3 Problem description

We give a description of the problem and the notation that we use throughout the paper. The main goal is to find a minimum cost inventory plan, where replenishment periods are known at the beginning of the planning horizon and the order size is determined upon observing the

inventory level. We first look at two variants of the single-item problem; the case where unmet demand is back-ordered and hence causes a penalty cost, and the case where restrictions are put on the service level. Our notation strongly relates to that of Tunc et al. (2018), but we explain it here for completeness. Afterwards, we introduce notation for an extended version of the service level variant that considers simultaneous inventory of multiple items.

We aim to find the planning for a finite time horizon of N discrete time periods. The demands over these periods are independent random variables, but may have different distributions. The demand over time interval $[i, j]$ is denoted by D_{ij} and has a known cumulative density function F_{ij} and a first-order loss function L_{ij}, which we use to calculate the expected inventory holding and back-order costs. The overall costs can be divided into three components. A fixed setup cost S is charged for each order. For each unit of inventory held into the next time period, there is an associated holding cost h. Furthermore, a back-order cost p is imposed for each unit of back-ordered demand per period. For simplicity, we assume that order lead time is negligible. Without loss of optimality, we assume that purchase price per unit is zero. Since unsatisfied demand is back-ordered, all demand will eventually be ordered. Therefore, the total purchase cost is a constant, which does not affect the optimization.

We denote the replenishment periods by $T_1, \ldots, T_m$. Here, m equals the number of order periods and T_n is the period where the nth order is scheduled. The interval between two consecutive replenishment periods, $[T_n, T_{n+1})$, is what we call a replenishment cycle. For convenience, we assume that the first order is scheduled in period 1 and the last order in period $N + 1$, hence $T_1 = 1$ and $T_{m+1} = N + 1$. This way, the planning horizon $(1, \ldots, N)$ can be divided into m disjoint replenishment cycles. We first solve the problem with a penalty cost p for unmet demand, but later on we adapt our formulation to put restrictions on the service level. We add an α service level constraint, which requires that the probability of not reaching a stock-out remains above α in any given period.

With the loss function L, we can impose penalty costs for failing to satisfy the demand. For a precise definition of the loss function, we refer to the paper of Tunc et al. (2018). Let I be the expected inventory level at the beginning of a replenishment cycle, right after receiving the order. The expected total costs in replenishment cycle $[i, j)$ can then be formulated as follows:

$$S + \sum_{t=i}^{j-1} (h(I - \mathbb{E}[D_{it}]) + (h + p)L_{it}(I)). \tag{1}$$

Besides the fixed setup cost S, we sum over the variable costs of every period t that lies within this replenishment cycle. The loss function L_{it} calculates the number of unsatisfied demand units in period t in cycle $[i, j)$. If this equals zero, only the holding costs h are incurred. If demand is greater than inventory, meaning the loss function is greater than zero, only the penalty costs p are added to the expected total costs. The loss function is a convex, nonlinear function, what makes it challenging to solve a MIP model with (1) in the objective. Therefore, Tunc et al. (2018) approximate the loss function with a piece-wise linear function in the following manner: $L(I) \approx \max_{(a,b)\in W}\{a + bI\}$. We assume that a set W of linear functions is available, consisting

of a finite number of (intercept, slope) pairs.

So far, all notation relates to the single-item lot-sizing problem. We now introduce notation for the multi-item case. The goal is again to find an optimal inventory plan for a finite time horizon of N discrete time periods. Let K denote the set of items, containing the items $k = 1, \ldots, |K|$. Each item has demand over the time interval $[i, j]$, denoted by D_{ij}^k, with known cumulative density function F_{ij}^k and a first-order loss function L_{ij}^k. The costs differ per item, giving setup cost S^k and holding cost h^k for item k. The planning horizon is again divided into disjoint replenishment cycles, but now for each item separately. Hence, the eventual solution consists of $|K|$ separate inventory schedules. Still, these schedules depend on each other through a required aggregated service level.

For the single-item case, the α service level constraint requires that the probability of not reaching a stock-out remains above α in any given period. For the multi-item case, we use the service level constraint for the items individually as well as for the items combined. We require an aggregated service level for the items combined, denoted by α^{agg}, and a minimum individual service level which is less strict, denoted by α^{min}. By letting this individual minimum be below the aggregated level, one is flexible in choosing the individual levels such that the required aggregated service level is achieved.

4 Methodology

Next, we describe the two solution approaches that we use to solve the different variants of the stochastic lot-sizing problem. At the same time, we present the MIP formulation that serves as the basis for these solution approaches. First, we explain the *static* approach of Tunc et al. (2018) for solving the problem with a piece-wise linear approximation of the cost function in Section 4.1. Thereafter, we explain their *dynamic* cut generation approach in Section 4.2, where no predefined approximation of the cost function is needed. Till then, the formulation is designed for the penalty cost variant of the problem, so in Section 4.3 we adjust it to model the α service level constraint. In Section 4.4, to address the multi-item lot-sizing problem, we expand the α service level formulation by introducing an aggregated service level constraint.

4.1 Static piece-wise linear approximation approach

We now present the penalty cost formulation, in order to explain the first solution approach. This approach is static in the sense that it uses an a priori set of linear functions to approximate the nonlinear cost function. Again, our notation is very similar to that of Tunc et al. (2018), but we repeat it for the sake of completeness. The parameters that we use are explained in detail in Section 3 and the decision variables are introduced below.

- x_{ij}: binary variable, equal to 1 if the interval $[i, j)$ is a replenishment cycle, 0 otherwise
- q_{ij}: integer variable, expected cumulative order quantity up to and including period i if $[i, j)$ is a replenishment cycle, 0 otherwise
- H_{ijt}: approximation of the loss function at period t in replenishment cycle $[i, j)$

Once the optimal solution is obtained, the replenishment schedule can be derived from these variables. Here, x determines the replenishment cycles, i.e., the timing of the reorders. q is the expected total quantity ordered so far, starting from period 1. Hence, when subtracting the expected total preceding demand from this, we can calculate the level we order up to for every cycle. Combined, we have an inventory plan for the entire time horizon. Next, we present the objective and the constraints that together form the mathematical formulation.

The objective is to find a cost minimizing replenishment schedule. The expected cost function for a given replenishment cycle $[i, j)$ is given in equation (1) in the previous section. Using the decision variables, the expected inventory level I at the beginning of a replenishment cycle can now be formulated as $q_{ij} - \mathbb{E}[D_{1,i-1}]$. Hence, we subtract the expected demand of all foregoing periods from the expected quantity that has been ordered up to period i. Furthermore, the loss function L_{it} is now replaced by its approximation H_{ijt}. This piece-wise linear approximation is modeled by one of the constraints later on. By making these adjustments, we can model the objective as follows:

$$\min \quad \sum_{i=1}^{N} \sum_{j=i+1}^{N+1} \left(S x_{ij} + \sum_{t=i}^{j-1} (h(q_{ij} - \mathbb{E}[D_{1t}] x_{ij}) + (h+p) H_{ijt}) \right). \tag{2}$$

We sum over all possible replenishment cycles, and costs are incurred only when a cycle belongs to the solution, i.e., $x_{ij} = 1$. Otherwise, the expression inside the double summation will vanish, as then also q_{ij} and H_{ijt} are enforced to zero by the constraints. Hence, (2) equals the expected total costs throughout the planning horizon. To ensure that the planning horizon is divided into disjoint replenishment cycles, we use the three constraints presented below.

$$\sum_{i=1}^{t-1} x_{it} = \sum_{j=t+1}^{N+1} x_{tj} \qquad t \in [2, N] \tag{3}$$

$$\sum_{j=2}^{N+1} x_{1j} = 1 \tag{4}$$

$$\sum_{i=1}^{N} x_{i,N+1} = 1 \tag{5}$$

Constraints (4) and (5) model that $T_1 = 1$ and $T_{m+1} = N + 1$, respectively. Constraint (3) models that any period t occurs with equal frequency as the first and the last period of a replenishment cycle. Together with (4) and (5) we have that this is either once or never. All combined, the planning horizon is divided into non-overlapping replenishment cycles. For q_{ij}, the expected cumulative order quantity up to and including period i, we must have $q_{ij} = 0$ if $[i, j)$ is not a replenishment cycle (i.e. $x_{ij} = 0$). This is enforced by the constraint

$$q_{ij} \leq M x_{ij}, \qquad i \in [1, N],\ j \in [i+1, N+1], \tag{6}$$

where M denotes a sufficiently large number as presented in Section 5. As q_{ij} is cumulative, it should be increasing from one replenishment cycle to the next. We model this by the following

constraint.

$$\sum_{i=1}^{t-1} q_{it} \leq \sum_{j=t+1}^{N+1} q_{tj} \qquad t \in [2, N] \tag{7}$$

When period t marks the beginning of a replenishment cycle, on both the left and right hand side of (7) we observe only one positive value in the summation, as the others are set to zero by (6). Hence, the constraint forces that the expected cumulative ordered quantity in a replenishment cycle is always larger than it was in the previous cycle. Finally, we have to model the piece-wise linear approximation of the loss function, as given in Section 3. Replacing the expected inventory level I at the beginning of a cycle $[i, j)$ again by $q_{ij} - \mathbb{E}[D_{1,i-1}]$, we get $L_{it}(q_{ij} - \mathbb{E}[D_{1,i-1}]) \approx \max_{(a,b)\in W_{it}}\{a+b(q_{ij}-\mathbb{E}[D_{1,i-1}])\}$. Here, W_{it} is the set of linear functions defining the piece-wise linear approximation of L_{it}. We can set H_{ijt}, the approximated loss function value in period t, equal to this maximum by adding the following constraint.

$$\begin{aligned} H_{ijt} \geq a x_{ij} + b(q_{ij} - \mathbb{E}[D_{1,i-1}] x_{ij}) \qquad & i \in [1, N],\ j \in [i+1, N+1], \\ & t \in [i, j-1],\ (a, b) \in W_{it} \end{aligned} \tag{8}$$

The constraint is active only if $[i, j)$ is a replenishment cycle, because otherwise we have both x_{ij} and q_{ij} equal to zero. By using this constraint, we eliminate the nonlinear cost function from the model. To finalize, we model the domains of the decision variables.

$$q_{ij} \geq 0,\ x_{ij} \in \{0, 1\} \qquad i \in [1, N],\ j \in [i+1, N+1] \tag{9}$$

$$H_{ijt} \geq 0 \qquad i \in [1, N],\ j \in [i+1, N+1],\ t \in [i, j-1] \tag{10}$$

Taken together, the static solution approach comes down to solving the MIP formulation given below, which we refer to as the PM model.

$$\begin{aligned} \min \quad & (2) \\ \text{s.t.} \quad & (3) - (10) \end{aligned}$$

4.2 Dynamic cut generation approach

In this section, we describe the second solution approach. This is the dynamic cut generation approach proposed by Tunc et al. (2018), which does not require any predefined piece-wise linear approximation of the cost function. This approach is build upon an initial MIP formulation, which is identical to the formulation for PM, only that (8) is replaced by

$$H_{ijt} \geq -(q_{ij} - \mathbb{E}[D_{1t}] x_{ij}), \qquad i \in [1, N],\ j \in [i+1, N+1],\ t \in [i, j-1]. \tag{11}$$

The constraint is imposed only for the cycles that belong to the solution. We refer to this extended formulation as the RM model. Tunc et al. (2018) prove that the right hand side of constraint (11) is a lower bound on the exact value of the loss function, L_{it}. Hence, the RM model is a relaxation of the PM model. This model serves as a starting point for the dynamic cut generation approach.

The main idea of this approach is to repeatedly solve the RM model and add constraints to this model if needed. Since the RM model is a relaxation of the PM model, solutions obtained from RM may be infeasible to the original PM model. If that is indeed the case, cuts are added to the RM model to exclude these infeasible solutions. This process continues until an optimal solution for RM is obtained, which is also feasible to the original PM model. The main advantage of this dynamic approach is that there is no need for an a priori piece-wise linear approximation, as the associated constraint (8) is replaced by (11). Moreover, the process of generating the cuts only requires some simple calculations.

We now explain how cuts can be generated from a solution to the RM model. Suppose we have a candidate solution $\{\bar{x}_{ij}, \bar{q}_{ij}, \bar{H}_{ijt}\}$. As (11) provides a lower bound on the real value of the loss function, $\bar{H}_{ijt}$ may be below this exact value. We check feasibility to the original PM model by comparing this approximation of the loss function to the exact value of the loss function evaluated in the current solution. Hence, for every $[i, j)$ and every $t \in [i, j)$, we compute the difference $L_{it}(\bar{q}_{ij} - \mathbb{E}[D_{1,i-1}]) - \bar{H}_{ijt}$. If this difference is no more than some preset small constant ϵ, we mark it as feasible. Otherwise, we generate a cut that excludes $\bar{H}_{ijt}$ from the feasible region. As the loss function is convex, we can do so by adding the tangent line of L_{it} at the current point $\bar{q}_{ij} - \mathbb{E}[D_{1,i-1}]$ as a constraint. The intercept and slope of this tangent line can be calculated as $a = L_{it}(\bar{q}_{ij} - \mathbb{E}[D_{1,i-1}]) - b(\bar{q}_{ij} - \mathbb{E}[D_{1,i-1}])$ and $b = F_{it}(\bar{q}_{ij} - \mathbb{E}[D_{1,i-1}]) - 1$, respectively. For the mathematical proofs and derivations, we refer to Tunc et al. (2018). Finally, the cut is added by including the constraint $H_{ijt} \geq ax_{ij} + b(q_{ij} - \mathbb{E}[D_{1,i-1}]x_{ij})$ in the RM model.

This way, the nonlinear loss function is approximated dynamically rather than beforehand, as was the case with the static solution approach. Whenever we obtain an optimal solution for the RM model with added cuts, which is also feasible for the PM model, we mark this as our final solution and terminate the algorithm. An overview of the dynamic cut generation approach can be found in Algorithm 1.

Algorithm 1 Dynamic Cut Generation

```
Invoke solver on RM model
repeat
    Get candidate solution {x̄, q̄, H̄}
    for i ∈ [1, N] and j ∈ [i + 1, N + 1] such that x̄_ij = 1 do
        for t ∈ [i, j) do
            if L_it(q̄_ij − E[D_1,i−1]) − H̄_ijt ≥ ε then
                b = F_it(q̄_ij − E[D_1,i−1]) − 1
                a = L_it(q̄_ij − E[D_1,i−1]) − b(q̄_ij − E[D_1,i−1])
                add cut H_ijt ≥ ax_ij + b(q_ij − E[D_1,i−1]x_ij) to RM model
until solver cannot find candidate solution; solved to optimality
```

4.3 Formulation with α service level constraint

So far, the formulation incurred a penalty cost p for not satisfying the desired demand. Or in other words: stock-outs were penalized. We can model a similar idea, by imposing constraints on the probability of a stock-out. This is often done by means of an α service level constraint; the probability of not reaching a stock-out must remain above α in each period. Since the

probability of a *non* stock-out decreases over the periods in a replenishment cycle, it suffices to add the constraint only for the last periods of these cycles. We can formulate the inventory level at the end of replenishment cycle $[i, j)$ as: $(q_{ij} - \mathbb{E}[D_{1,i-1}]) - D_{i,j-1}$. Hence, the constraint translates to

$$P((q_{ij} - \mathbb{E}[D_{1,i-1}]) - D_{i,j-1} \geq 0) \geq \alpha,$$

which can be rewritten to a constraint as

$$q_{ij} \geq (\mathbb{E}[D_{1,i-1}] + F^{-1}_{i,j-1}(\alpha))x_{ij}, \quad i \in [1, N],\ j \in [i+1, N+1], \tag{12}$$

where $F^{-1}_{i,j-1}$ is the inverse of the cumulative density function of $D_{i,j-1}$. Similar to before, (12) is active only when $[i, j)$ is part of the solution (i.e. $x_{ij} = 1$). For the derivation of the constraint, we refer to Tunc et al. (2018). When adding (12) to the formulation for the PM and RM models, and removing the penalty cost p from the objective, we can apply both the static and dynamic solution approach to solve this variant of the problem with constraints on the service level.

4.4 Formulation for multi-item lot-sizing

In this section, we modify the α service level formulation of the previous section to solve the multi-item lot-sizing problem. We do so by imposing a service level constraint for the items individually as well as for the items combined. This is often profitable in practice, especially when items vary in terms of demand, order cost or holding cost. One could set the same service level for all items (e.g. 95%), but this may not necessarily be optimal. For example, a lower service level (93%) for items with high holding cost can be compensated by a higher service level (97%) for items with low holding cost. Combined, the company can still achieve the desired aggregated service level (95%).

In order to model the individual and aggregated service levels, we introduce the following parameters and decision variables. Besides the minimum aggregated service level (α^{agg}), we require a minimum service level for the items individually (α^{min}). By letting this individual minimum be below the aggregated minimum, one is flexible in choosing the individual levels such that the minimum aggregated service level is achieved. The actual aggregated service level is calculated as a weighted average of the individual levels. Here, w^k denotes the weight of item k, which is proportional to the item's expected demand over the planning horizon and therefore calculated as

$$w^k = \frac{D^k_{1N}}{\sum_{p=1}^{|K|} D^p_{1N}}, \quad k \in [1, |K|].$$

The actual individual service levels are a choice now rather than a fixed value. These levels can take values between the minimum required service level α^{min} and 1. To prevent non-linear functions of the decision variables in the constraints, we model this in a similar way as Sereshti et al. (2021) do. Let L be a set of service levels α^l, with $l \in [1, |L|]$. For example, this set contains $|L| = 10$ service levels equally distributed between α^{min} and 0.9999, hence $L = \{\alpha^{min}, \ldots, 0.9999\}$. We take 0.9999 as the maximum possible service level, as a service level of 1 would require an infinite amount of inventory. The decision to be made for each item

k, is which service level $\alpha^l \in L$ to select. We therefore introduce the following decision variable.

- y^{kl}: binary variable, equal to 1 if service level α^l is selected for item k, 0 otherwise

By choosing the individual service levels from the set L, we approximate the situation where the service levels can take any value between α^{min} and 1. When service level α^l is selected for item k, it implies that the probability of not reaching a stock-out remains above α^l for this item.

4.4.1 Multi-item static solution approach

We now present the PM model for the static solution approach, adjusted for the multi-item lot-sizing problem. The earlier decision variables get an additional index k, giving x_{ij}^k, q_{ij}^k and H_{ijt}^k (e.g., $x_{ij}^k = 1$ if $[i, j)$ is a replenishment cycle for item k). Then, the expected total costs can be calculated in a similar way as before, but now by also summing over all items:

$$\min \quad \sum_{k=1}^{|K|} \sum_{i=1}^{N} \sum_{j=i+1}^{N+1} \left(S^k x_{ij}^k + \sum_{t=i}^{j-1} (h^k (q_{ij}^k - \mathbb{E}[D_{1t}^k] x_{ij}^k) + h^k H_{ijt}^k) \right). \tag{13}$$

The following constraints ensure that for each item k, the planning horizon is divided into disjoint replenishment cycles.

$$\sum_{i=1}^{t-1} x_{it}^k = \sum_{j=t+1}^{N+1} x_{tj}^k \quad t \in [2, N],\ k \in [1, |K|] \tag{14}$$

$$\sum_{j=2}^{N+1} x_{1j}^k = 1 \quad k \in [1, |K|] \tag{15}$$

$$\sum_{i=1}^{N} x_{i,N+1}^k = 1 \quad k \in [1, |K|] \tag{16}$$

In the constraints below, (17) ensures that the expected cumulative order quantity of item k can only be positive if the replenishment cycle belongs to the solution for this item. M^k represents a sufficiently large number, depending on the demand distribution of item k. The expected cumulative order quantity of item k is increasing from one replenishment cycle to the next by (18).

$$q_{ij}^k \leq M^k x_{ij}^k \quad i \in [1, N],\ j \in [i+1, N+1],\ k \in [1, |K|] \tag{17}$$

$$\sum_{i=1}^{t-1} q_{it}^k \leq \sum_{j=t+1}^{N+1} q_{tj}^k \quad t \in [2, N],\ k \in [1, |K|] \tag{18}$$

The loss function value of item k in period t is approximated by a set of linear functions W_{it}^k, which depends on the distribution of D_{it}^k, in (19). As previously, this eliminates the non-linearity.

$$H_{ijt}^k \geq a x_{ij}^k + b(q_{ij}^k - \mathbb{E}[D_{1,i-1}^k] x_{ij}^k) \quad \begin{array}{l} i \in [1, N],\ j \in [i+1, N+1], \\ t \in [i, j-1],\ k \in [1, |K|],\ (a, b) \in W_{it}^k \end{array} \tag{19}$$

The individual and aggregated service levels are included in the model by the following constraints. Firstly, we model the individual service levels, in a similar way as for the single-item

case in Section 4.3. The constraint (20) ensures that, if $[i, j)$ is a replenishment cycle for item k, the expected cumulative order quantity satisfies the selected service level for item k.

$$q_{ij}^k \geq (\mathbb{E}[D_{1,i-1}^k] + F_{i,j-1}^{k\,-1}(\alpha^l))y^{kl} - F_{1N}^{k\,-1}(\alpha^l)(1 - x_{ij}^k) \quad \begin{array}{l} i \in [1, N],\ j \in [i+1, N+1], \\ k \in [1, |K|],\ l \in [1, |L|] \end{array} \tag{20}$$

Here, $F_{i,j-1}^{k\,-1}$ denotes the inverse of the cumulative density function of $D_{i,j-1}^k$. If $[i, j)$ is indeed a replenishment cycle for item k and α^l is the selected service level (i.e., $x_{ij}^k = 1$ and $y^{kl} = 1$), the constraint ensures that the service level is met. If the service level is not selected for item k or the cycle does not belong to the solution, the constraint will be inactive. $F_{1N}^{k\,-1}(\alpha^l)$ is a sufficiently large number such that the right-hand side is non-positive in the latter case. We take the α^lth percentile of the distribution of D_{1N}^k, as the first term of the right-hand side will never exceed that value. For each item, we must select exactly one service level from the set L. This is modeled by the following constraint.

$$\sum_{l=1}^{|L|} y^{kl} = 1 \quad k \in [1, |K|] \tag{21}$$

Lastly, we need to ensure that the required aggregated service level is preserved. The weighted average of all selected individual service levels must be above the aggregated service level:

$$\sum_{k=1}^{|K|} \sum_{l=1}^{|L|} w^k \alpha^l y^{kl} \geq \alpha^{agg}. \tag{22}$$

To finalize, (23), (24) and (25) model the domains of the decision variables.

$$q_{ij}^k \geq 0,\ x_{ij}^k \in \{0, 1\} \quad i \in [1, N],\ j \in [i+1, N+1],\ k \in [1, |K|] \tag{23}$$

$$H_{ijt}^k \geq 0 \quad i \in [1, N],\ j \in [i+1, N+1],\ t \in [i, j-1],\ k \in [1, |K|] \tag{24}$$

$$y^{kl} \in \{0, 1\} \quad k \in [1, |K|],\ l \in [1, |L|] \tag{25}$$

This completes the MIP formulation for the PM model that we use with the static solution approach for solving the multi-item lot-sizing problem with an aggregated service level:

$$\begin{array}{ll} \min & (13) \\ \text{s.t.} & (14) - (25). \end{array}$$

The main advantage of an aggregated service level is that costs can be reduced by allowing for different service levels per item instead of setting a fixed level for all items. As long as the aggregated service level is met, the individual levels can be chosen such that costs are minimized.

4.4.2 Multi-item dynamic solution approach

For the RM model used in the dynamic solution approach, we make the same adjustments as for the single-item case in Section 4.2. Hence, we replace (19) by the following constraint:

$$H^k_{ijt} \geq -(q^k_{ij} - \mathbb{E}[D^k_{1t}]x^k_{ij}), \quad i \in [1, N],\ j \in [i+1, N+1],\ t \in [i, j-1],\ k \in [1, |K|]. \tag{26}$$

This RM model serves as a starting point for the dynamic cut generation approach, as given in Section 4.2. Again, no predefined set of linear functions is needed for this approach. The only difference is that the cuts are added for all items now. The dynamic cut generation approach for the multi-item problem is summarized in Algorithm 3 in Appendix A.

4.4.3 Multi-item heuristic

When considering long planning horizons or a large value for the number of items, the instances become too complex for the two solution approaches. For that reason, we propose a heuristic approach that finds satisfactory solutions within reasonable computational time. This heuristic mainly relies on the following insight: solving the single-item lot-sizing problem for each item independently requires minimal computational time. Thus, we only (repeatedly) solve the single-item problem. Moreover, increasing the number of service levels in the set L significantly increases the run time of the earlier solution approaches, but it also results in higher cost savings. As the former does not happen with this heuristic, we can take a larger value for $|L|$, potentially leading to higher cost savings. The intuition behind the heuristic is described below and the steps are summarized in Algorithm 2. In all steps, increasing/decreasing the service level of an item implies taking the service level from the set L that is one level higher/lower than the item's current level (e.g., going from α^l to α^{l+1}).

The initial step (1) is to assign the required aggregated service level to all items. Afterwards, we repeatedly switch the service levels of items to other levels from the set L, with the aim of reducing the overall costs. As explained in more detail in Section 5, we assume that the items differ only in terms of holding and setup costs. Therefore, when solving the single-item problem for all items with a service level equal to α^{agg}, we expect that more costly items will attain higher objectives, while less costly items will yield lower objectives. The main idea behind relaxing the individual service levels is to assign low service levels to costly items while assigning higher levels to less costly items. This is accomplished in the next step (2) by repeatedly searching for the lowest objectives and increasing the service levels of the corresponding items, while also decreasing the service levels of the items with the highest objectives. The number of items that switch service levels each iteration depends on the value of α^{agg} and is determined by two sub steps (2A and 2B). Algorithm 2 explains this in detail for an aggregated service level of 0.91, but Appendix B provides an explanation for other values of α^{agg}. We keep switching service levels until the overall objective increases two iterations in a row, and stick with the solution obtained before this. The next step (3) ensures that the required aggregated service level is met, if this was not yet the case. We iterate over the items in ascending order based on costs and increase their service levels until the aggregated service level is achieved. As a final step (4), we try to further reduce the overall costs once more. We iterate over the items in descending order based on costs and decrease their service levels if this still satisfies the required aggregated service level.

Algorithm 2 Heuristic for multi-item problem

Step 1: Solve the single-item lot-sizing problem with an α service level constraint for all items independently, by applying the dynamic solution approach as described in Section 4.2. In this initial step, give all items the required aggregated service level α^{agg}. Store the objective values for all items individually and sum them to obtain the overall objective for the items combined.

repeat

Step 2A: *Increase* the service levels of the two items with the lowest objectives and *decrease* the service level of the item with the highest objective.

Step 2B: *Decrease* the service level of the item with the second-highest objective, but only if then the actual aggregated service level is at least α^{agg}.

Solve the single-item problem for the items that have switched service levels, store their new objectives and update the overall objective.

until the overall objective increased two iterations in a row

Step 3: If the required aggregated service level is not met after step 2, iterate over the items in ascending order based on costs. In each iteration, *increase* the service level of the current item and solve the single-item problem. Store the item's new objective and update the overall objective. Stop once the aggregated service level is met.

Step 4: Iterate over the items in descending order based on costs. In each iteration, *decrease* the service level of the current item if this still satisfies the aggregated service level. If so, solve the single-item problem, store the item's new objective and update the overall objective. Stop once all items have been examined.

5 Data

In this section, we outline the data that we use to solve the three variants of the stochastic lot-sizing problem and explain how it is obtained. The main goal of our study is to compare the computational efficiency of the two solution approaches in solving the stochastic lot-sizing problem. In order to do so, we generate a wide range of instances, such that the performance of the approaches can be tested in various scenarios. For the single-item problems, we use the exact same data as Tunc et al. (2018), but describe it here for the sake of completeness. Afterwards, we explain how the data is generated for the multi-item problem.

5.1 Single-item data

For the single-item problems (i.e. with penalty costs and α service level constraints), two different sets of instances are generated. The first set, referred to as set-A, is utilized to measure the performance of the solution approaches for different values of the parameters. This enables us to evaluate the approaches' flexibility in handling different scenarios. The second set, set-B, is designed to see how well the approaches perform on longer planning horizons. We maintain a constant holding cost $h = 1$ in all our experiments for the single-item problems.

For set-A, we look at three different lengths for the planning horizon $N = \{20, 30, 40\}$ and three setup costs $S = \{225, 900, 2500\}$. For the penalty cost variant and the α service level variant, we consider the values $p = \{2, 5, 10\}$ and $\alpha = \{0.90, 0.95, 0.99\}$, respectively. For M to be sufficiently large, we look at the distribution of the demand over the entire planning horizon, and take the critical value that corresponds to the $p/(h + p)$ percentile for the penalty

cost variant. For α equal to 0.90, 0.95 and 0.99, we take the percentiles 0.85, 0.90 and 0.99, respectively. We assume that, in every period, demand follows a normal distribution with a fixed coefficient of variation equal to $\rho = \{0.1, 0.2, 0.3\}$. Expected demands for all periods in the planning horizon are generated according to two demand patterns $\pi = \{\text{Erratic}, \text{Lumpy}\}$. For the Erratic case, expected demands are drawn from a uniform distribution over the interval $[0, 200]$. With the Lumpy pattern, we have a 20% chance of generating an expected demand from a uniform distribution on the interval $[0, 420]$ and a 80% chance of generating an expected demand from a uniform distribution on the interval $[0, 20]$. As a result, the Lumpy pattern represents a scenario where demand is more volatile over time. For each possible combination of horizon length and demand pattern, 10 random instances of expected demands are generated by Tunc et al. (2018). However, their data file contains an error for the 10th instance, so we only use the first 9 instances of expected demands. This gives a total of 2916 instances in set-A.

With set-B, our focus is more on the scalability of the solution approaches. To this end, we investigate longer planning horizons with $N = \{50, 60, 70, 80, 90, 100\}$. We fix the other parameters to a single value. We take $S = 225$ and for the penalty cost variant and the α service level variant, $p = 10$ and $\alpha = 0.99$, respectively. Moreover, we assume $\rho = 0.3$ and $\pi = \text{Erratic}$. For each planning horizon length, we generate 10 instances of expected demands, resulting in a total of 120 instances in set-B.

For the static solution approach, we need the set W_{it} of linear functions for approximating the loss function L_{it}. We use the 11-piece lower bound approximation for a standard normal variable as given by Rossi et al. (2014), from which the approximation for a normal variable can be easily derived. Hence, their set W can be used to calculate the set W_{it} for all periods $t \in [i, j)$, by making use of the distribution of D_{it}. For the dynamic solution approach, we must specify the parameter ϵ that is used in the dynamic cut generation. Tunc et al. (2018) propose variant-specific values, which guarantee that the approximation error in the expected total costs is negligible. This corresponds to $\epsilon = 1/N(h + p)$ for the penalty cost variant and $\epsilon = 1/Nh$ for the α service level variant.

5.2 Multi-item data

For the multi-item problem, we generate two additional sets of instances. With both sets, we solve the problem with and without the aggregated service level constraint. In the latter case, each item must independently satisfy the required aggregated service level. By comparing the resulting costs, we can analyze the benefits of relaxing the individual service levels while requiring a minimum aggregated service level, rather than imposing a fixed service level for all items. Set-C is created for an initial test of how well the solution approaches perform in the multi-item setting and to conduct a sensitivity analysis. The other set, set-D, is used to investigate the scalability of the solution approaches in the multi-item setting. It therefore contains varying values for the number of items and the length of the planning horizon.

In set-C, we take horizon length N and number of items $|K|$ equal to 10 and 5, respectively. For these 5 items, we generate 10 instances of expected demands according to demand pattern $\pi = \text{Erratic}$. We consider three values for the coefficient of variation $\rho = \{0.1, 0.2, 0.3\}$.

We take $\alpha^{min} = 0.80$, and look at different values for the aggregated service level: $\alpha^{agg} = \{0.91, 0.93, 0.95\}$. After conducting some preliminary tests, we found that taking $|L|$ larger than 10 led to a significant increase in computational time while yielding only negligible improvements in the solutions. Hence, we look at $|L| = \{8, 10\}$. Here, $|L| = 8$ means that the set contains 8 service levels equally distributed between α^{min} and 0.9999. With the sensitivity analysis, our main goal is to measure the effect of different degrees of variation between the items' holding and setup costs on the cost savings achieved when relaxing the individual service levels. For instance, when the items are very similar in terms of these costs, we expect the savings to be minimal. When there is significant variation among the items, on the other hand, we expect that relaxing the individual service levels can result in increased cost savings. To examine this, we create five options for combinations of holding and setup costs for the 5 items, as given in Table 1. The last column contains the standard deviation of the holding cost, serving as a measure of the variation between the items. The standard deviation of the setup cost is $100 * \sigma_{hc}$. Hence, the cost options are arranged in decreasing order based on their degree of variation. With the last option, there is no variation at all between the items, so we expect no cost reduction. This gives a total of 900 instances in set-C.

Table 1: Five options for holding and setup costs for the 5 items

Option	holding costs	setup costs	σ_{hc}
1	[1, 1, 5.5, 10, 10]	[100, 100, 550, 1000, 1000]	4.02
2	[1, 3, 5, 7, 9]	[100, 300, 500, 700, 900]	2.83
3	[3, 3, 5.5, 8, 8]	[300, 300, 550, 800, 800]	2.24
4	[3, 4, 5, 6, 7]	[300, 400, 500, 600, 700]	1.41
5	[5.5, 5.5, 5.5, 5.5, 5.5]	[550, 550, 550, 550, 550]	0

With set-D, we consider three different lengths for the planning horizon $N = \{10, 15, 20\}$ and three values for the number of items $|K| = \{10, 15, 20\}$. For each item, we have $\pi =$ Erratic and $\rho = 0.3$. We generate 10 instances of expected demands for each possible combination of horizon length and number of items. We take $\alpha^{min} = 0.80$ and $\alpha^{agg} = \{0.91, 0.95\}$. For the number of service levels in the set L we have $|L| = 8$ for the dynamic approach, but $|L| = 20$ for the heuristic. For the holding cost and setup cost of item k, we take $h^k = k$ and $S^k = 100k$, where $k = 1, \ldots, |K|$. In total, there are 180 instances in set-D.

In both sets, we must take a sufficiently large number for M^k. Since the selected service level for item k is unknown in advance, we take the 0.9999 percentile of the distribution of D^k_{1N}. With the heuristic, we repeatedly solve the single-item problem, hence we take M as before. For the static solution approach, we apply the exact same method as for the single-item case to derive the set W^k_{it}, but do it for all items now. For the dynamic solution approach, we take $\epsilon = 1/N|K|$, as this gives negligible differences between the outcomes of the two solution approaches.

6 Computational experiments

We conduct computational experiments with all sets of instances to analyze the performance of both solution approaches when solving the different variants of the stochastic lot-sizing problem.

We run all experiments on a 2.40 GHz Intel Core i5-1135G7 CPU with 8 GB RAM with CPLEX V22.1.0 as a MIP solver. We use default solver settings and impose a time limit of half an hour.

6.1 Flexibility of the solution approaches

We first look at the single-item variants of the problem: the penalty cost and α service level variant. To evaluate the flexibility of both solution approaches, we apply them to solve the instances of set-A. Table 2 contains the computational statistics for the α service level variant. The statistics for the penalty cost variant can be found in Appendix C. Here, E-GAP denotes the relative optimality gap in percentages, TIME the run time in seconds and NODES the number of nodes explored at termination. The default setting in CPLEX for the tolerance gap is 0.01%. The statistics are averaged over all instances that have the corresponding parameter value.

We observe that both approaches solve all instances to optimality within reasonable run times, and that the optimal solution is found on average at the root node. Still, the α service level variant reveals clear differences between the two solution approaches. For all parameter values, the dynamic cut generation approach is on average three times faster than the static solution approach. Hence, for the α service level variant, the dynamic approach outperforms the static approach significantly. Both solution approaches are hardly affected by the varying parameter settings of the instances in set-A. Only increasing the length of the planning horizon shows substantial differences in the run times. In general, both solution approaches exhibit flexibility in handling different combinations of the parameter values.

Table 2: Performance statistics of both solution approaches for the α service level variant

		static			*dynamic*		
	Parameters	E-GAP	TIME	NODES	E-GAP	TIME	NODES
π							
	Erratic	0.00	0.62	0.0	0.00	0.18	0.0
	Lumpy	0.00	0.74	0.1	0.00	0.20	1.1
N							
	20	0.00	0.17	0.0	0.00	0.06	0.2
	30	0.00	0.57	0.0	0.00	0.19	0.5
	40	0.00	1.29	0.2	0.00	0.33	0.9
S							
	225	0.00	0.74	0.2	0.00	0.22	1.2
	900	0.00	0.67	0.0	0.00	0.19	0.4
	2500	0.00	0.63	0.0	0.00	0.17	0.0
ρ							
	0.1	0.00	0.65	0.0	0.00	0.18	0.1
	0.2	0.00	0.68	0.0	0.00	0.19	0.4
	0.3	0.00	0.71	0.2	0.00	0.21	1.1
α							
	0.90	0.00	0.64	0.0	0.00	0.18	0.2
	0.95	0.00	0.63	0.0	0.00	0.18	0.4
	0.99	0.00	0.77	0.1	0.00	0.21	1.0

6.2 Scalability of the solution approaches

As we observed with set-A, the length of the planning horizon significantly affects the computational performance of both solution approaches. We now compare the approaches on the instances of set-B, which contain longer planning horizons. The performance statistics for the penalty cost variant can be found in Appendix C. Table 3 shows the average run time per horizon length for the α service level variant. We again observe that the dynamic approach is much faster than the static approach. This advantage becomes even bigger when considering longer planning horizons: for $N = 100$, the dynamic approach is on average 20 times faster, where for $N = 50$ this is only 5 times. So, especially for instances with long planning horizons, the dynamic cut generation approach is a considerably better solution method.

Table 3: Run times for both solution approaches for the α service level variant

	N					
	50	60	70	80	90	100
static	3.30	6.79	13.40	28.33	71.01	163.48
dynamic	0.63	1.14	1.65	2.69	4.48	7.93

In conclusion, the dynamic approach yields superior computational performance for the single-item problems, given that the difference in resulting costs between the approaches is negligible.

6.3 Sensitivity analysis in the multi-item setting

Next, we analyze the performance of both solution approaches in the multi-item setting with an aggregated service level. We first only solve the instances in set-C that have $|L|$ equal to 8, to compare the approaches. For those 450 instances, the dynamic approach solved the problem to optimality within on average 13.5 seconds, where the static approach needed on average 57.7 seconds. Once again, the dynamic approach is much faster than the static approach. For that reason, we conduct the remaining experiments only with the dynamic approach.

In addition, we observed two remarkable outcome statistics. Firstly, we saw that for the fifth cost option, the cost savings achieved when relaxing the individual service levels were on average negative. However, this strategy should always result in a solution that is at least as good as when fixing all individual service levels to α^{agg}. This was not the case since α^{agg} was not included in the set L. Secondly, we noticed that the cost savings were considerably lower for α^{agg} equal to 0.95 compared tho the other values. This can be attributed to the fact that the service levels in the set L are equally distributed between α^{min} and 0.9999. When this set contains 8 service levels, only two of them are above 0.95, namely 0.97 and 0.9999. As a result, some of the items have to take these relative high service levels to achieve the aggregated service level, leading to minimal cost savings.

We therefore make the following adjustments regarding the set L. Half of the service levels are equally distributed between α^{min} and α^{agg} (excluding α^{agg}), and the remaining half between α^{agg} and 0.9999 (including α^{agg}). For $|L| = 8$ and $\alpha^{agg} = 0.95$, this results in the set $L = \{0.80, 0.84, 0.88, 0.91, 0.95, 0.97, 0.98, 0.9999\}$. After making these adjustments, we apply

the dynamic solution approach to solve all 900 instances in set-C. The performance statistics of this approach are given in Table 4, separately for $|L| = 8$ and $|L| = 10$. $\Delta cost$ is the cost reduction in percentages achieved when imposing an aggregated service level and is therefore calculated as $\Delta cost = \frac{obj - obj'}{obj} * 100\%$, where obj' is the objective value obtained when solving the problem with an aggregated service level whilst relaxing the individual levels, and obj is the value obtained when fixing all individual levels to the required aggregated service level.

At this time, we observe zero cost reduction for cost option five, which is in line with our expectations. Moreover, we see a rise in cost savings as the variation between the items' costs increases. Thus, the strategy of imposing an aggregated service level is valuable especially when there is significant variation among the items. For $|L| = 10$, we notice slightly greater cost savings than for $|L| = 8$, but this goes in hand with longer run times: on average 34.9 seconds compared to 14.5 seconds. This is precisely the trade-off that needs to be made when employing this modeling approach; more service levels in the set L yields greater cost savings, but at the same time, it results in longer run times. With the original set L, the average cost reduction for α^{agg} equal to 0.95 was 0.40% for $|L| = 8$, hence it doubled after revising this set. Lastly, we observe greater cost savings when the coefficient of variation increases. In situations where customer behaviour is uncertain, it may be beneficial to implement an aggregated service level.

Table 4: Performance statistics of the dynamic approach in the multi-item setting

		$\|L\| = 8$		$\|L\| = 10$	
Parameters		TIME	$\Delta cost(\%)$	TIME	$\Delta cost(\%)$
option					
	1	8.5	2.65	16.1	2.91
	2	11.1	1.52	22.9	1.68
	3	15.2	0.70	37.6	0.76
	4	17.2	0.21	43.6	0.30
	5	20.4	0.00	54.3	0.00
α^{agg}					
	0.91	16.9	1.23	44.1	1.29
	0.93	14.5	1.02	35.0	1.18
	0.95	12.0	0.81	25.6	0.93
ρ					
	0.1	15.2	0.68	38.4	0.75
	0.2	13.9	1.07	33.4	1.19
	0.3	14.2	1.30	32.9	1.45
overall		14.5		34.9	

Table 5 exhibits an example of the selected service levels for the items across all five cost options. Here, we considered $|L| = 8$, $\rho = 0.1$ and $\alpha^{agg} = 0.91$. The more variable the cost option, the more variation in the selected service levels. Conversely, less variable options result in less variation, and option 5 even selects the same service level for all items. This is not surprising, as the items are identical in terms of cost under that option.

Table 5: Example of the selected individual service levels per cost option with $\alpha^{agg} = 0.91$

	item				
option	1	2	3	4	5
1	0.97	0.97	0.94	0.88	0.80
2	0.97	0.97	0.94	0.88	0.80
3	0.97	0.94	0.94	0.86	0.86
4	0.94	0.94	0.94	0.88	0.86
5	0.91	0.91	0.91	0.91	0.91

6.4 Scalability in the multi-item setting

To assess its scalability, we apply the dynamic approach to solve the instances of set-D. However, already with $N = 15$ and $|K| = 10$, it takes one hour to only reach a GAP of 45%. So, when the length of the planning horizon or the number of items increases, CPLEX is no longer capable of finding a (nearly) optimal solution within reasonable time. Therefore, we apply the heuristic described in Section 4.4.3. Since this heuristic tends to assign the highest possible service level to some of the items, we introduce an additional change to the set L: the maximum attainable individual service level is set to 0.99 instead of 0.9999. Assigning a service level of 0.9999 to a few items already results in a substantial increase in costs. Moreover, the dynamic approach does not even select this highest possible service level for the least expensive item for any of the cost options in set-C (also visible in Table 5). Hence, we exclude this service level from L.

The performance statistics for the heuristic are presented in Table 6. The run times have significantly improved compared to the dynamic approach: the instances are solved within 3.6 seconds on average. Additionally, for all 120 instances, the heuristic led to a better solution compared to the scenario where all items take the required aggregated service level. On average, the heuristic resulted in a cost reduction of 1.8%. Increasing the length of the planning horizon results in a significant rise in the run times, while increasing the number of items has less impact. Therefore, this heuristic can be particularly valuable for companies dealing with the inventory holding of many items. Imposing an aggregated service level of 0.91 yields greater cost savings than a level of 0.95. This outcome is not surprising, as relaxing the individual service levels provides even more flexibility when $\alpha^{agg} = 0.91$.

When considering $|K| = 10$, the holding costs exhibit a similar standard deviation as with cost option 2 in set-C. As shown in Table 4, this cost option leads to an average cost reduction of approximately 1.6%. When applying the heuristic with 10 items, we achieve similar cost savings, namely 1.7%. With 15 or 20 items, the standard deviation becomes even higher, so we may expect greater cost savings. For example, cost option 1 for set-C saves almost 3%. However, with 15 or 20 items, the savings remain below 2%. When looking at the selected individual service levels for an instance with 20 items and $\alpha^{agg} = 0.91$, wee see that many items share the same service level: [0.99, 0.99, 0.99, 0.99, 0.96, 0.96, 0.91, 0.91, 0.91, 0.91, 0.91, 0.91, 0.91, 0.91, 0.91, 0.86, 0.87, 0.80, 0.83, 0.80]. Since all items differ in terms of costs, we would expect more variation in the selected service levels. Hence, the heuristic may be further improved when ensuring more variation in the eventual individual service levels. Nevertheless, the heuristic already demonstrates its effectiveness by finding solutions that yield an average cost reduction

of almost 2%. For some instances, this corresponds to saving 10,000 on the total costs.

Table 6: Performance statistics for the heuristic for the multi-item problem

	N			$\|K\|$			α^{agg}		
	10	15	20	10	15	20	0.91	0.95	*overall*
$\Delta cost(\%)$	1.70	1.82	1.88	1.72	1.79	1.90	2.15	1.46	1.80
TIME	1.1	2.2	7.5	2.6	3.6	4.6	3.7	3.5	3.6

7 Conclusion

The primary goal of this research was to evaluate and compare the performances of the static and dynamic solution approaches in solving the stochastic lot-sizing problem, adhering to the static-dynamic uncertainty strategy. Firstly, we considered the single-item problem in order to answer our first research question: "How efficient are approaches that approximate the non-linear cost function in solving the stochastic lot-sizing problem?". Our computational experiments have shown that both solution approaches are capable of solving multiple variants of the single-item lot-sizing problem on real-size instances within reasonable time. They exhibit flexibility in handling various combinations of the parameter values. However, especially when examining the scalability of both approaches, it became evident that the dynamic approach outperforms the static approach significantly. As the length of the planning horizon increases, the difference in computational time between the two solution approaches becomes even more pronounced.

Given that the dynamic approach yields superior computational performance, it may be worthwile to consider applying this solution approach to other variants of the lot-sizing problem as well. In today's world, preserving the environment has become more crucial than ever. Therefore, the carbon footprint cannot be ignored when determining the optimal inventory plan. Ries et al. (2017) mention that operating large warehouses causes considerable consumption of energy due to lightning, cooling, heating and material handling machines. 13% of the carbon footprint in the supply chain sector is due to managing inventory centers. Our formulations could be extended to integrate this footprint, for example by using the concept of cap-and-trade: a limit is put on the amount of carbon a company may release. Consequently, the objective becomes to minimize both the costs and carbon emissions. Further research could investigate the applicability of the dynamic solution approach to this variant of the problem.

Secondly, to address a more realistic situation, we evaluated the dynamic solution approach in the multi-item setting by incorporating an aggregated service level constraint into the formulation. By doing so, we intended to answer our second research question: "Are the approaches that approximate the non-linear cost function applicable to the multi-item problem as well, and how much of the total costs could be saved by implementing an aggregated service level?". For small instances, the dynamic approach remained successful in finding the optimal solution. The computational experiments have highlighted the advantages of imposing an aggregated service level while relaxing the individual levels, especially when there is considerable variation among items in terms of costs. The most variable instances resulted in a cost reduction of almost 3%. However, as the instances became larger and more complex, the dynamic approach failed to find

the optimal solution. At this point, our heuristic approach proved its effectiveness by achieving cost reductions of almost 2%. Still, this approach could be further improved when ensuring more variation in the individual service levels of the items. Also, in this study, we assumed that the items only vary in terms of costs. However, in practice, items also exhibit dissimilarities in their demand distributions. As our heuristic is specifically designed for items that follow similar demand patterns, further research could focus on scenarios where demand varies between items.

References

[1] James H. Bookbinder and Jin-Yan Tan. "Strategies for the Probabilistic Lot-Sizing Problem with Service-Level Constraints". In: *Management Science* 34.9 (1988), pp. 1096–1108. DOI: 10.1287/mnsc.34.9.1096.

[2] Matthieu Gruson, Jean-François Cordeau and Raf Jans. "The impact of service level constraints in deterministic lot sizing with backlogging". In: *Omega* 79 (2018), pp. 91–103. ISSN: 0305-0483. DOI: https://doi.org/10.1016/j.omega.2017.08.003. URL: https://www.sciencedirect.com/science/article/pii/S0305048316305679.

[3] Jörg M. Ries, Eric H. Grosse and Johannes Fichtinger. "Environmental impact of warehousing: a scenario analysis for the United States". In: *International Journal of Production Research* 55.21 (2017), pp. 6485–6499. DOI: 10.1080/00207543.2016.1211342. URL: https://doi.org/10.1080/00207543.2016.1211342.

[4] Roberto Rossi, Onur A. Kilic and S. Armagan Tarim. "Piecewise linear approximations for the static-dynamic uncertainty strategy in stochastic lot-sizing". In: *OMEGA-INTERNATIONAL JOURNAL OF MANAGEMENT SCIENCE* 50 (Jan. 2015), pp. 126–140. ISSN: 0305-0483. DOI: 10.1016/j.omega.2014.08.003.

[5] Roberto Rossi, S. Armagan Tarim, Steven Prestwich and Brahim Hnich. "Piecewise linear lower and upper bounds for the standard normal first order loss function". In: *Applied Mathematics and Computation* 231 (2014), pp. 489–502. ISSN: 0096-3003. DOI: https://doi.org/10.1016/j.amc.2014.01.019. URL: https://www.sciencedirect.com/science/article/pii/S0096300314000563.

[6] Narges Sereshti, Yossiri Adulyasak and Raf Jans. "The value of aggregate service levels in stochastic lot sizing problems". In: *Omega* 102 (2021), p. 102335. ISSN: 0305-0483. DOI: https://doi.org/10.1016/j.omega.2020.102335. URL: https://www.sciencedirect.com/science/article/pii/S0305048320306897.

[7] Huseyin Tunc, Onur A. Kilic, S. Armagan Tarim and Burak Eksioglu. "A reformulation for the stochastic lot sizing problem with service-level constraints". In: *OPERATIONS RESEARCH LETTERS* 42.2 (Mar. 2014), pp. 161–165. ISSN: 0167-6377. DOI: 10.1016/j.orl.2014.01.010.

[8] Huseyin Tunc, Onur A. Kilic, S. Armagan Tarim and Roberto Rossi. "An Extended Mixed-Integer Programming Formulation and Dynamic Cut Generation Approach for the Stochastic Lot-Sizing Problem". In: *INFORMS JOURNAL ON COMPUTING* 30.3 (2018), pp. 492–506. ISSN: 1091-9856. DOI: 10.1287/ijoc.2017.0792.

A Multi-item dynamic cut generation algorithm

Algorithm 3 Dynamic Cut Generation for multi-item problem

Invoke solver on RM model
repeat
  Get candidate solution $\{\bar{x}, \bar{q}, \bar{H}\}$
  for $k \in [1, |K|]$ **do**
    for $i \in [1, N]$ and $j \in [i+1, N+1]$ such that $\bar{x}_{ij}^k = 1$ **do**
      for $t \in [i, j)$ **do**
        if $L_{it}^k(\bar{q}_{ij}^{\,k} - \mathbb{E}[D_{1,i-1}^k]) - \bar{H}_{ijt}^k \geq \epsilon$ **then**
          $b = F_{it}^k(\bar{q}_{ij}^{\,k} - \mathbb{E}[D_{1,i-1}^k]) - 1$
          $a = L_{it}^k(\bar{q}_{ij}^{\,k} - \mathbb{E}[D_{1,i-1}^k]) - b(\bar{q}_{ij}^{\,k} - \mathbb{E}[D_{1,i-1}^k])$
          add cut $H_{ijt}^k \geq a x_{ij}^k + b(q_{ij}^k - \mathbb{E}[D_{1,i-1}^k] x_{ij}^k)$ to RM model
until solver cannot find candidate solution; solved to optimality

B Heuristic for multi-item problem: step 2A

With the algorithm of Section 4.4.3, we increase the service levels of the *two* items with the lowest objectives in each iteration (step 2A). This algorithm is designed for the case where $\alpha^{agg} = 0.91$. When $\alpha^{agg} = 0.95$, we increase the service levels of the *four* items with the lowest objectives in each iteration. All other steps remain unchanged: we decrease one service level in step 2A and a second one in step 2B if this satisfies the aggregated service level. We conducted some preliminary tests to find the optimal number of switches to be made in step 2A, but the intuition behind it is explained below.

With the heuristic approach, the set L contains 20 service levels, of which half is equally distributed between α^{min} and α^{agg} and the other half between α^{agg} and 0.99. As a result, the interval between the service levels is not the same throughout the whole set. As shown in Figure 1, for $\alpha^{agg} = 0.91$, the intervals are almost identical. Therefore, we try to increase and decrease two service levels in each iteration. For $\alpha^{agg} = 0.95$, the second interval is much smaller than the first interval. Ideally, after completing steps 2A and 2B, we should be relatively close to the required aggregated service level. However, if we would increase only two service levels in each iteration while still decreasing at least one, the actual aggregated service level will quickly fall below α^{agg}. For that reason, we increase four service levels rather than two.

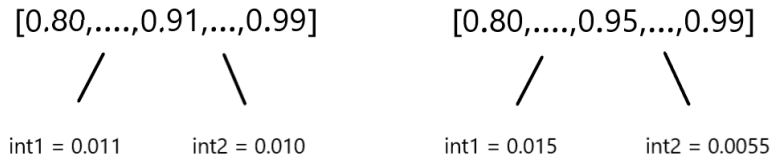

Figure 1: Intervals between the service levels in the set L

C Performance statistics for the penalty cost variant

Table 7: Performance statistics of both solution approaches for the penalty cost variant for set-A

		static			*dynamic*		
	Parameters	E-GAP	TIME	NODES	E-GAP	TIME	NODES
π							
	Erratic	0.01	1.56	25.0	0.01	1.11	286.1
	Lumpy	0.01	1.22	2.0	0.01	1.05	218.0
N							
	20	0.01	0.30	0.1	0.01	0.21	116.8
	30	0.01	0.94	5.4	0.01	0.77	246.6
	40	0.01	2.93	35.0	0.01	2.27	392.8
S							
	225	0.01	2.15	40.5	0.01	1.51	517.1
	900	0.01	1.06	0.1	0.01	1.01	177.4
	2500	0.00	0.95	0.0	0.01	0.73	61.6
ρ							
	0.1	0.01	1.33	9.7	0.01	0.90	200.8
	0.2	0.01	1.39	15.3	0.01	1.07	243.0
	0.3	0.01	1.45	15.6	0.01	1.28	312.4
p							
	2	0.00	1.27	0.1	0.01	0.86	134.3
	5	0.01	1.27	3.3	0.01	1.06	240.8
	10	0.01	1.63	37.1	0.01	1.32	381.0

Table 7 contains the performance statistics of both solution approaches for the penalty cost variant for set-A. All instances are solved to optimality within reasonable run times. For each of the specific parameter values, the dynamic approach is faster than the static approach, but the differences are not massive. With both approaches, we see that longer planning horizons increase the run times significantly. In addition, we see substantial differences in run times for the different values of the setup cost. A low setup cost gives longer run times, as the trade-off between ordering often or rarely becomes more complicated. For this variant of the problem, the differences between the two solution approaches are not as obvious as they are for the α service level variant. Moreover, the run times are more strongly affected by the value of the setup cost for this variant.

Table 8: Run times for both solution approaches for the penalty cost variant for set-B

	N					
	50	60	70	80	90	100
static	99.27	218.54	413.15	744.37	-	-
dynamic	10.42	21.20	47.39	73.32	115.78	200.54

Table 8 shows the average run time per horizon length for the penalty cost variant for set-B. The static solution approach was not able to solve all instances with $N = 90$ and $N = 100$ to optimality within half an hour, so we put a dash there. For all other horizon lengths, we see that the dynamic approach is on average approximately 10 times faster than the static approach.

The dynamic approach even solved the instances with the longest planning horizon to optimality within on average 4 minutes. Particularly when considering long planning horizons, the dynamic solution approach outperforms the static approach in solving the lot-sizing problem with penalty costs for unmet demand.

D Programming code

A zip file is attached to this paper, which contains the programming code used to carry out all experiments in this study. The problems are solved with CPLEX in java. Included are also the text files that are necessary as input for the programs (stored within the project but outside the scr file). Separate java files are designed to solve the different variants of the problem, using the different solution approaches. For example, AlphaPMmodel.java solves the α service level variant of the problem, by applying the static solution approach. Additional java files are available to compute all performance statistics from the output of the solution approaches. A detailed overview of the executed runs can be found in the text file 'Runs.txt'.