Erasmus University Rotterdam
Erasmus School of Economics

Bachelor Thesis
Double Degree BSc^2 in Econometrics and Economics

Analysing the Stochastic Lot-Sizing Problem Using a Mixed-Integer Programming Formulation and Incorporating a Receding Horizon

Author:
Tobias Kers
Student Number:
546338tk

Supervisor:
Wilco van den Heuvel
Second Assessor:
Albert Wagelmans

Abstract

The stochastic lot-sizing problem deals with designing an inventory plan that minimises costs under random demand. This thesis studies this problem first by employing a mixed-integer programming formulation developed by Tunc et al. (2018), that uses a piece-wise linear approximation of the cost function, as defined in Rossi, S. A. Tarim, et al. (2014). After, this thesis develops a heuristic in order to incorporate a receding horizon into the model, which means the length of the horizon gradually shrinks. It finds that the mixed-integer programming formulation is computationally efficient, but displays some errors when simulating demand, raising some small doubts about its reliability. Furthermore, the heuristic has limited cost saving potential, as the degree to which it is effective largely depends on the parameter settings.

July 2, 2023

Contents

1 Introduction and literature review

The topic of this thesis is the lot-sizing problem. There are multiple types of costs that are involved in this problem, namely set-up costs, holding costs and direct item costs. The objective of the problem is then to minimise all those costs, while still meeting demand at a satisfactory level. Alternatively, unmet demand can also be penalised through a penalty cost and incorporated into the cost function. The stochastic lot-sizing problem was first discussed by the seminal paper Wagner and Whitin (1958), and since then this topic has been studied mostly in a deterministic setting. However, this research will focus on the stochastic lot-sizing problem, meaning demands are random and not known in advance. This means a certain order-policy must be designed that minimises the expected costs, rather than an exact solution which would be possible in the deterministic case.

Over the years, a large number of different types of order policies for stochastic lot-sizing problems have been developed, of which Bookbinder and Tan (1988) outlines three. All three models differ in terms of how much of the order policy is determined at the start of the cycle and how much is determined during the cycle.

The first of these three is the static uncertainty model. Here, the full set of decision variables, including when to order and how much to order, is determined at the start. This is beneficial if much preparation is needed beforehand and computationally equivalent to the deterministic problem, but is not able to adjust when demand fluctuates heavily from expected demand.

The second model, in a way the opposite of the static uncertainty model, is the dynamic uncertainty model (Bookbinder and Tan 1988). In this case, the order policy is updated every period by taking the information about realised demand into account. Then, the decision whether to place an order and how much to order is made based on the inventory level.

The third and last model strikes a balance between the two and is called the static-dynamic uncertainty model (Bookbinder and Tan 1988). Here, the periods in which to place an order are determined at the start, but the order quantities are not. Rather a base-stock level is determined, which specifies up to which level should be ordered. Then during the cycle, the actual order quantity depends on the inventory at that point. As this strategy specifies a policy at the start, it makes coordination with other supply chain players easier, meaning this approach has attracted a lot of research into which computational methods are most efficient. Bookbinder and Tan (1988) argues that this policy is the most straight-forward to use in many cases.

The period between the placing of two successive orders is called a replenishment cycle, with all cycles together making up the replenishment schedule. A key result here from Özen, Doğrub, and S. A. Tarim (2012) is that once a replenishment schedule has been found, the optimal order quantity in each replenishment period follows a base-stock policy. However, finding these schedules and base-stock levels has proven to be challenging and currently the literature suggests using heuristics for this purpose, depending on the variant of the prob-

lem.

The underlying problem can be approached through tailor-made algorithms, which work well for specific cases, or through mathematical programming models. The latter models are more accessible as they work for a wider variety of cases, but a major challenge concerns how to incorporate the non-linear costs into the model, since the randomness of the demands means the total cost function is not linear. In the literature, several ways have been suggested, such as just using a linear approximation or employing a piece-wise linear function.

When the cost function is linearly approximated, the problem can be solved using a mixed integer programming formulation. Research in this area has developed a lot in recent years, with papers such as Rossi, Kilic, and S. Tarim (2015) demonstrating the value by incorporating several service level constraints. These service level constraints specify a minimum of demand that must be met, often 90%.

However, a lot of these models were not computationally efficient, requiring the contribution by Tunc et al. (2018), where the authors use the dynamic-static uncertainty strategy. They consider two new formulations that included new ways of approaching the cost function. Their first method, the extended formulation, is a mixed integer programming (MIP), already majorly decreased computation times as a result of using a network flow formulation, while still solving to optimality. Secondly, they also developed a dynamic cut generation approach, where constraints are added during solving in order to approximate the non-linear cost function. This approach both allows for more precision and reduces compared to the MIP-formulation.

Building on this model, Tunc (2021) incorporates compression costs into the model, which are incurred when processing times are decreased. Even with this addition, both the MIP-model and dynamic cut generation still achieve a considerably better computational efficiency than other formulations. Other recent papers within the context of the stochastic lot-sizing problem have considered a two-stage process Gruson, Cordeau, and Jans (2021) and multi-item contexts with wait-and-see structure Seyfi et al. (2022), demonstrating the range of research possible.

Recently, research has also been done on more applied contexts of lot-sizing problems. For example, Wang et al. (2022) looked at the aerospace industry, which is characterised by a very complex cost structure. In that paper, the authors use dynamic programming, like Tunc et al. (2018) to create a multi-agent simulation that creates an inventory replenishment strategy.

1.1 Receding Horizon

Some of the literature on stochastic lot-sizing model makes use of either rolling or receding horizons (Jans and Degraeve 2008). In both settings, the model is gradually evaluated at later points in time. In rolling horizon settings, the length of the horizon remains the same, while in a receding horizon the ending period remains the same, but the length of the horizon recedes. Using such a horizon also makes sense from a forecasting point of view, as generally fore-

casts about events more in the future are less accurate (Bookbinder and Tan 1988). However, when information about the distribution of demands is perfect, a rolling horizon makes less sense. As this thesis assumes a known normal distribution for demands, a rolling horizon is not considered.

However, this thesis will in fact consider a rolling horizon, as this approach has been shown to have cost-saving potential (Dural-Selcuk et al. 2020). This paper also further makes the case for dynamic-static uncertainty models, as it shows applying a receding horizon brings the dynamic-static uncertainty models closer to the dynamic uncertainty models. The receding horizon has also shown to be efficient in a setting with multiple products (Sereshti, Adulyasak, and Jans 2021).

Despite its cost-saving potential, the receding horizon remains understudied in the literature, and hence this thesis will study it using the MIP-formulation provided by Tunc et al. (2018).

This thesis is relevant in a multitude of ways. First of all, the receding horizon environment remains understudied, especially in the specific context of stochastic lot-sizing. Furthermore, this thesis will developed a novel heuristic, that has cost-saving potential under some parameter settings, while only causing minor changes to the order policy. Moreover, the heuristic implements changes that are very intuitive, making it more likely to be applied to real-world settings. Lastly, this thesis contributes by affirming the computational efficiency of the MIP-formulation developed in Tunc et al. (2018).

1.2 Thesis outline

The remainder of this thesis is organised as follows. First, I present some economic background to this problem. After, I present the notation and then the model formulation for the stochastic lot-sizing problem, adapted from Tunc et al. (2018), where I use the linear approximation proposed by Rossi, S. A. Tarim, et al. (2014). Afterwards, I adapt the model to a receding horizon environment, by developing a novel heuristic.

In the first part of the results, I show that the MIP-formulation is computationally efficient, and that the derived solutions are the same as Tunc et al. (2018). Next, I assess the reliability of the model by first computing the approximation errors and then the simulation errors. The approximation errors are all small, but the simulation errors are larger, raising doubts about the reliability of the model. Furthermore, the simulation errors are a lot larger than those in Tunc et al. (2018). The results section ends with the receding horizon heuristic. Here I find that the heuristic has cost-saving potential, but only under specific parameter settings. The cost saving-potential is higher under parameter settings where the heuristic is less likely to be triggered. I end this thesis by giving a conclusion and discussing the limitations and possible avenues for future research.

2 Economic Background

Over the years, a lot of economic theory has been developed in the field of inventory management, which includes the stochastic lot-sizing problem, the topic of this thesis. In order to place this research in a wider context, this section will give a brief outline of the economic background.

As there exists a range of business and production processes, there are also different ways of inventory management that have been developed in the literature. The base-stock model, characterised by random demand and continuous replenishment, is covered in this thesis. Still, there are also other models, such as the closely related news-vendor problem, also with random demand but just one replenishment (Qin et al. 2011).

Another classic, and perhaps most famous model, is the economic order quantity model, first developed back in 1913 by Ford Whitman Harris (Erlenkotter 1990). This model assumes demand is constant over time and tries to determine the optimal order quantity, taking into consideration holding costs, production or purchase costs and ordering costs. As this model in its base form boils down to one formula, it is very easy to use (Cárdenas-Barrón, Chung, and Treviño-Garza 2014). There are also many extensions possible to this model, such as considering deterioration of inventory (Yang et al. 2020).

These theoretical models are useful in real-life inventory management to varying degrees. One criticism of the literature articulated by Rumyantsev and Netessine (2007) is that inventory managers care about about a macroscopic view of the entire inventory management, rather than more microscopic views of the inventory management of single products. Furthermore, models do not take a wide range of external factors, such as business cycles, into account, Rumyantsev and Netessine write.

3 Preliminaries

First, I will outline the model used by Tunc et al. (2018). They employ a mixed integer formulation, which I refer to as MIP. In their model they assume N discrete time periods. I will start by explaining the notation, which is adapted from Tunc et al. (2018).

- D_{ij}: Demand in period i until and including j
- L_{ij}: Loss function associated with D_{ij}
- F_{ij}: Cumulative distribution function associated with D_{ij}
- K: fixed cost for every order
- h: holding cost, cost of holding one unit from one period to the next
- p: back-order cost, extra cost for every unit that needs to be back-order when inventory is insufficient to meet the demand

- $T_1, ...T_m$: replenishment periods

As per the static-dynamic uncertainty strategy, the model produces an order policy, that specifies both the replenishment cycles and the replenishment quantities. A replenishment cycle is defined as the interval between two successive replenishment periods, so $[T_n, T_{n+1})$. For simplicity, I assume $T_1 = 1$ and $T_m = N + 1$ such that there are m disjoint replenishment cycles.

I will now start defining the cost function. First, consider the replenishment cycle $[i, j)$ and assume the inventory level at the start of the cycle is y. The total expected cost for that cycle is then given by:

$$K + \sum_{t=i}^{j-1} (h\,\mathbb{E}(y - D_{it})^+ + p\,\mathbb{E}(y - D_{it})^-) \tag{1}$$

Here, x^+ and x^- stand for $max\{0, x\}$ and $max\{0, -x\}$ respectively. The cost function will include a loss function, which has the following properties.

1. if z is a random variable and x is a scalar variable then its loss function is given by $\mathbb{E}[max(z - x), 0]$ Rossi, S. A. Tarim, et al. 2014
2. $\mathbb{E}(y - z)^-$ is the loss function evaluated at y
3. $\mathbb{E}(z) = \mathbb{E}\,z^+ - \mathbb{E}\,z^-$

Following these properties, the cost function transforms into:

$$K + \sum_{t=1}^{j-1} (h * (y - \mathbb{E}\,D_{it}) + (h + p) * L_{it}(y)) \tag{2}$$

3.1 Linear Approximation

Since the aim is to build a mixed integer formulation and the loss function above is non-linear, I will apply linear approximation to the loss function. Keeping in line with Tunc et al. (2018), I will follow the approach outlined in Rossi, S. A. Tarim, et al. (2014).

To explain how the approximation works, I will first define σ_i as the standard deviation for the demand in period i. Then, for the period $[i, j]$ the mean demand is $\mu_{ij} = \mathbb{E}\,D_{ij}$ and then the standard deviation for that period is $\sigma_{ij} = \sqrt{\sum_{k=i}^{j} \sigma_k^2}$, according to the properties of the normal distribution.

Next, I will use the values for the 11-piece lower bound approximation from table 1 of Rossi, S. A. Tarim, et al. 2014. For $k \in [1, 11]$ this table specifies values for the intervals of the the 11 line segments, given by $\mathbb{E}[\omega|\Omega_k]$. For $k \in [2, 11]$ it also denotes values for p_k, which is related to the slope of the loss function. I then adjust the interval values for the mean and standard deviation of the period considered: $\widehat{E_k} = \mu_{ij} + \sigma_{ij} * \mathbb{E}[\omega|\Omega_k]$.

Then I calculate the 11 intercept-slope $[\widehat{a_k}, \widehat{b_k}]$ pairs as follows:

$$\widehat{b_k} = \begin{cases} -1, & \text{if } k = 1 \\ -1 + \sum_{j=2}^{k} p_j, & \text{if } k \in [2, 11] \end{cases}$$

$$\widehat{a_k} = \begin{cases} \mu_{ij}, & \text{if } k = 1 \\ \widehat{a_{k-1}} + \widehat{b_{k-1}}\widehat{E_k} - \widehat{E_{k-1}}), & \text{if } k \in [2, 11] \end{cases}$$

These together give the set W_{ij} of intercept-slope pairs to approximate the loss function L_{ij}.

4 Model

4.1 MIP model

I will now start with the formulation of the model. The solution of the problem should be an order strategy, that specifies both the replenishment cycles and quantities. The following decision variable capture this policy:

- x_{ij} indicator variables that is equal to 1 if $[i, j)$ is a replenishment cycle and 0 if it is not
- q_{ij} the expected cumulative order quantity up-to and including period i if $[i, j)$ is a replenishment cycle
- H_{ijt} the approximation of loss function at period $t \in [i, j)$

The x_{ij} variables specify what periods form replenishment cycles, while from the q_{ij} variables the base-stock level can be derived, which is the level of inventory that a replenishment cycle starts with. Lastly, the H_{ijt} variables indicate the value of the loss function linearly approximated using the method specified earlier.

Now circling back to the cost function from equation 2, the inventory at the beginning of each replenishment cycle can be written as $q_{ij} - \mathbb{E}(D_{1i-1}$. Then the cost function for replenishment cycle $[i, j)$ becomes:

$$K + \sum_{t=i}^{j-1} (h(q_{ij} - \mathbb{E}\, D_{1t}) + (h + p) * H_{ijt}) \tag{3}$$

As all the replenishment cycles are disjoint, the sum of the cost function of every cycle gives the total cost function:

$$min \sum_{i=1}^{N} \sum_{j=i+1}^{N+1} \left(K x_{ij} + \sum_{t=i}^{j-1} (h(q_{ij} - \mathbb{E}\, D_{1t} x_{ij}) + (h + p) * H_{ijt}) \right) \tag{4}$$

This assumes that $x_{ij} = 0$ implies both $q_{ij} = 0$ and H_{ijt}, a condition that will be imposed by the constraints that are specified after this. That means, if

$x_{ij} = 0$ the summands are zero and therefore only the costs belonging to the replenishment cycles, where $x_{ij} = 1$, are summed.

For every replenishment cycle the base-stock level is given by $q_{ij} - \mathbb{E}(D_{1i-1})$, which is level to which inventory is filled at the start of every replenishment cycle. Implicitly, the model assumes that the inventory that is carried over from the previous replenishment cycles is never higher than this base-stock level. This could of course happen as demand is random, but the associated costs are assumed to be small and hence ignored, in line with the literature (S. A. Tarim and Kingsman 2004). However, this does mean the solutions are technically not completely optimal.

Next, by means of network flow constraints, it is ensured that the order policy consists of disjoint replenishment cycles.

$$\sum_{i=1}^{t-1} x_{it} = \sum_{t+1}^{N} t \in [2, N] \tag{5}$$

$$\sum_{j=2}^{N} x_{1j} = 1 \tag{6}$$

$$\sum_{i=1}^{N-1} x_{iN} = 1 \tag{7}$$

Constraint (5) ensures that all the replenishment cycles follow each other, while constraints (6) and (7) make sure that the first replenishment cycle starts in the firt period of the horizon and the last one ends in the final period, respectively.

The next two constraints concern the q_{ij} variables. First I make sure $x_{ij} = 0$ implies q_{ij} by imposing the following constraints:

$$q_{ij} \leq M * x_{ij} i \in [1, N], j \in [i + 1, N + 1] \tag{8}$$

Here, M is a sufficiently large value. A potential value could be the cumulative demand over the entire horizon times 10, although a lower bound is likely possible.

Furthermore, as q_{ij} specifies the cumulative order quantity, it must be non-decreasing. This is imposed by the following constraint:

$$\sum_{i=1}^{t-1} q_{it} \leq \sum_{j=t+1}^{N+1} q_{tj}, t \in [2, N] \tag{9}$$

The following constraint concerns the H_{ijt} variables, which are approximations of the loss function. Using the approach described earlier in section 3.1, I obtain intercept pairs $(a, b) \in W_{it}$ associated with loss function L_{it}. Then, as $L_{it}(q_{ij} - \mathbb{E}(D_{1i-1})) \approx max_{(a,b)\in W_{it}}\{ax_{ij} + b(q_{ij} - \mathbb{E}(D_{1i-1}))\}$. I incorporate this as follows:

$$H_{ijt} \geq ax_{ij}+b(q_{ij}-\mathbb{E}(D_{1i-1})), i \in [1, N], j \in [i+1, N+1], t \in [i, j-1], (a, b) \in W_{it} \tag{10}$$

For completeness, below I specify the entire model:

min (4)

subject to (5)-(10

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

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

$x_{ij} \in \{0, 1\}, i \in [1, N], j \in [i+1, N+1]$

4.2 Receding Horizon

Next, I will extend the MIP-model by incorporating a receding horizon. A receding horizon means that the program will be evaluated for different horizons, where the horizon length keeps decreasing, while the ending period remains the same. Generally, the main benefit of using a receding horizon in lot-sizing problems is that information about the realised demands in the previous periods can be used to come up with an adapted order policy. However, this does come at the cost of the possibility of the order policy changing during the horizon.

Next, I will introduce two algorithms, but I will first introduce some new notation:

- X^k: matrix containing order policy for time horizon $[k, N]$, where X^k_{ij}=1 if replenishment cycle $[i, j)$ is part of the order policy
- X^*: the final result containing all the order cycles after completing algorithm
- Q^k: cumulative order quantities for order policy for time horizon $[k, N]$
- $\widehat{D_{ij}}$: the realised demand in period $[i, j]$

I will incorporate the receding horizon by first developing a naive algorithm that reevaluates the order policy at the start of every new replenishment cycle. First, I will solve the model for the complete horizon. From, this solution, I save the starting and ending index of the first replenishment cycle to matrix X^*. Then I update k by adding the ending index and the algorithm resolves the model, now with time horizon $[k, N+1]$. The algorithm continues this process until period $N+1$ is reached. Like with the MIP model I assume the full cycle is given by $[1, N+1]$.

Important to note here is that the model is reevaluated at the start of every new replenishment cycle, and not at every single period across the time horizon. There are two reasons for this, the first being that current literature on receding horizon also does this (Dural-Selcuk et al. 2020). The second reason is that reevaluating at every period would would require a completely new formulation, as the current network flow formulation requires a replenishment cycle to start

in the first period of the time horizon considered. Ultimately, the matrix X^* contains the final cycles for the order policy.

I will now introduce provide pseudocode for this naive algorithm to illustrate its workings.

Algorithm 1 Naive algorithm

```
k = 1
while k < N + 1 do
    X = X^k
    Q = Q^k
    for p ∈ [0, N + 1 − k] do
        if X_{0p} = 1 then
            X*_{k,k+p} = 1
            k = k + p
            break
        end if
    end for
end while
```

This naive algorithm still has several limitations. The first one is that it is unable to incorporate information about demands in the preceding periods, despite having access to them. This means it is unable to benefit from the main advantage of using a receding horizon. Secondly, the costs from this algorithm are not comparable to that of the original MIP model. This is because the model implicitly assumes that every new replenishment cycle is started with zero inventory. These limitations therefore still require an algorithm that does make use of information about realised demands.

4.2.1 Heuristic

Hence, I will develop a heuristic that both incorporates the receding horizon and also makes potential adjustments to the order policy based on the realised demands. The structure of this algorithm is similar to algorithm 1, but the algorithm also keeps track of the inventory level throughout by generating the actual demands and subtracting them from the starting inventory level, derived from the order-up to level.

The algorithm works by iterating over all replenishment cycles of the total horizon, until it reaches the final period $N+1$. For every period, it first computes the order policy where the first replenishment cycle is given by $[1, p)$. Then, the algorithm updates the inventory by first setting it equal to the base-stock level and then subtracting the realised demand from period $[k, k + p)$, of the original demand horizon. Then the algorithm checks whether the inventory would be large enough to meet expected demand for period p. Here I do not compare the inventory to the expected demand but make an adjustment to account for uncertainty and compare it to $\mathbb{E}\, D_i i + \sigma_{ii} * C$, where $\sigma_i i$ is the standard deviation associated with the demand in period i and C is the value for the confidence

level, with $C \geq 0$. The higher this value, the less likely it is for inventory to be unable to meet the demand from the following period.

If the inventory is larger, the current replenishment cycle is extended by one period to now include the period which demand would likely be able to be met by the current inventory and this replenishment cycle $[k, k+p)$ is added to the final order policy X^*. In this case, k is updated to $k+p+1$. If inventory is not sufficiently large, k is updated to $k = k+p$ and the original cycle $[k, p)$ is added to X^*.

When inventory is large enough to meet that criterion, it intuitively makes sense to postpone an order, as one does not generally order when inventory is large enough to meet demand in the next period.

Lastly, the variable *numberEdits* keeps track of the number of times the order policy is changed and is incremented whenever the inventory is sufficiently large. This also allows for a maximum number of edits to be imposed.

One important note is that at the start of every run of the algorithm the horizon $[1, N+1-k]$ is considered. This means the first cycle in every new solution always start at index 1. The X^* matrix then has the purpose of keeping track of the replenishment cycles over the full horizon.

Algorithm 2 Heuristic incorporating receding horizon

1: $k = 1$
2: **while** $k < N+1$ **do**
3: $X = X^k$
4: $Q = Q^k$
5: **for** $p \in [2, N+1-k]$ **do**
6: **if** $X_{1p} = 1$ **then**
7: $p^* = p$
8: **end if**
9: $inventory = Q_{0p} - \widehat{D_{k,k+p^*-1}}$
10: **if** $inventory < \mathbb{E}\, D_{k+p^*,k+p^*} + C * \sigma_{k+p^*,k+p^*} || numberEdits \geq maxEdits$ **then**
11: $X^*_{k,k+p^*} = 1$
12: $k = k + p^*$
13: **else**
14: $numberEdits = numberEdits + 1$
15: $X^*_{k,k+p^*+1} = 1$
16: $k = k + p^* + 1$
17: **end if**
18: **end for**
19: **end while**

Using this heuristic, I will generate a new order policy, that is potentially different from the original solution, as whenever the condition from line 10 holds, at least one replenishment cycle changes.

The fact that adjustments to the replenishment periods are possible and are

made during the horizon, means that this heuristic is no longer a static-dynamic uncertainty model, but rather a dynamic uncertainty model.

5 Computational Experiment

Following Tunc et al. (2018), I will test the model using two sets of instances, *Set-1* and *Set-2*. The first set of instances, *Set-1* will be used to test the models performance against different set of parameters, while the second set will be used to test the models scalability.

For *Set-1* I will use three different values for most the parameters, which are outlined in table 1. For the holding cost, I will just use the value 1, for both sets of instances. The ρ variable indicates the coefficient of variation, where $\sigma = \mu * \rho$.

For the demands, I assume that they are normally distributed and independent. The means come from either the lumpy or erratic demand pattern. I use the same demands, as Tunc et al. (2018), a link to which can be found in their paper. Tunc et al. (2018) generate 10 random problem instances for every combination of parameters values. However, I only use 9, as the 10^{th} iteration has the same exact values as the 1^{st} iteration of the lumpy pattern with the same horizon length. In total this means that I have 1458 different iterations for *Set-1*.

For *Set-2*, I fix all the parameters to one value, except for the number of periods in the planning horizon. For every horizon length, there are 10 random problem instances. Again, I use the demands that Tunc et al. 2018 generated.

Table 1: Parameters values computational experiment

	Values	
Parameter	Set-1	Set-2
π	Erratic,Lumpy	Erratic
N	20,30,40	50,60,70,80,90,100
K	225,900,2500	225
p	2,5,10	10
ρ	0.1,0.2,0.3	0.3
h	1	1

I carry out all computation using CPLEX on Java. The processor I use is 1.80 GHz Intel Core(TM) i5-8250U CPU with 8GB Ram.

5.1 Simulation

5.1.1 Approximation Error

In order to test the reliability of these policies, I perform two tests.

First, I calculate what the actual cost would have been per the non-linear loss function, based on the solution for each iteration. This differ from the value of the objective cost function, because of the H_{ijt} variables, which is an approximation of the loss function. Therefore, for every $t \in [i, j)$, where $[i, j)$ is a replenishment cycle, I calculate the value $L_{it}(q_{ij} - \mathbb{E}\, D_{1i-1})$ and take the sum. The difference between the costs from the model and the actual costs, that is the approximation error, or A-ERR are then given by equation (11) below. The x_{ij} makes sure only the terms are considered for which $[i, j)$ is a replenishment cycle.

$$\text{A-ERR} = (h + p) \sum_{i=1}^{N} \sum_{j=i+1}^{N+1} \sum_{t=i}^{j-1} x_{ij}(H_{ijt} - L_{it}(q_{ij} - \mathbb{E}\, D_{1i-1})) \tag{11}$$

I calculate the loss function, L_{it} using the formula provided by Rossi, S. A. Tarim, et al. (2014), which is given below.

$$L_{it}(x, \xi) = \sigma_{it} * (\phi(\frac{x - \mu_{it}}{\sigma_{it}}) - (1 - \Phi(\frac{x - \mu_{it}}{\sigma_{it}}))\frac{x - \mu_{it}}{\sigma_{it}}) \tag{12}$$

In this equation, ξ refers to the random variable, and x is the value at which the loss function is evaluated. Furthermore, ϕ refers to the pdf of the standard normal distribution, while Φ refers to the cdf.

5.1.2 Simulation Error

The second simulation is more involved. First, I generate the actual demands for each period, using the fact they are normally distributed. Then for each of the periods t, the algorithm checks whether a new replenishment cycle starts in this period, denoted by $[i, j)$. If so, the inventory level is set to $q_{ij} - \mathbb{E}\, D_{1i-1}$. This is the base-stock level and the starting inventory for every replenishment cycle. Then, for every period t, the inventory is decreased by the realised demand from period t. When inventory is still positive, total costs are increased by the inventory multiplied with the holding cost, as this inventory is carried over to the next period. When inventory is negative, it means products have to be back-ordered, which means inventory times the back-order cost is added to the total cost. If inventory is negative, it is reset back to 0, to make sure the same demand is not back-ordered multiple times were inventory unable to meet demand in consecutive periods. This process repeats until the last period.

Ultimately, S-ERR denotes the difference between the difference between the total costs derived from the MIP-formulation and the simulation.

Algorithm 3 Simulation algorithm

1: **while** $p < numRuns$ **do**
2: $totalCostSim = 0$
3: $inventory = 0$
4: $t = 0$
5: **while** $t < N + 1$ **do**
6: **for** $j \in [t + 1, N + 1]$ **do**
7: **if** $x_{tj} = 1$ **then**
8: $totalCostSim = totalCostSim + K$
9: $inventory = q_{tj} - \mathbb{E}\, D_{1t-1}$
10: **end if**
11: **end for**
12: $inventory = inventory - \widehat{D_{tt}}$
13: **if** $Inventory > 0$ **then**
14: $totalCostSim = totalCostSim + inventory * h$
15: **else**
16: $totalCostSim = totalCostSim + inventory * p$
17: $inventory = 0$
18: **end if**
19: $t = t + 1$
20: **end while**
21: S-ERR=$costMIP - totalCostSim$
22: $p = p + 1$
23: **end while**

To illustrate how the simulation works, I have provided pseudocode above. In line with Tunc et al. (2018) I perform 500 different simulation runs for each iteration. I perform this simulation both for *Set-1* and *Set-2*.

5.2 Receding Horizon Heuristic

In this next section, I will explain how I will test the performance of the receding horizon heuristic. First, for *Set-1*, I will perform 50 different simulations runs for every combination of parameters and demand pattern using the same set of randomly generated instances as I did for the MIP-model. I will keep track of how often the heuristic is applied, meaning an adjustment to the order policy is made, in order to assess to what combination of parameters the heuristic is most effective.

Next to keeping track of the frequency, I will also assess the cost-saving potential of this heuristic. I will do this by simulating costs both under the new policy with the adjustment and under the old policy without the adjustment. Let's say it is true for period i that the inventory is likely large enough to meet demand, see line 10 of algorithm 2. Then for the simulation, I use algorithm 3, for the period $[i, N]$, where N is the last period of the original horizon. However, there is one difference as in this case the simulation does not start with inventory

being equal to zero, but rather with the inventory left after the previous periods. This means I will first add the inventory multiplied with the holding costs.

Under the old policy, the first replenishment cycle will start in period i, while under the new policy the replenishment cycle will start in period $i+1$. This means that for the cost calculation under the new policy, the inventory will be decreased by $\widehat{D_{ii}}$ and based on the inventory left the total costs will be updated. Then moving to the next period, the inventory is set to q_{i+1j}, where $x_{i+1j} = 1$ with $[i+1, j)$ being the first replenishment cycle. Meanwhile, under the old policy, the inventory is immediately set to q_{ij}.

For every adjustment, I will do 500 simulation runs, where I generate all the demands for the horizon $[i, N]$. For every run, I record the total costs under both the old and new policy and then calculate the mean and standard deviation for both and record those. This will allow me to look at the average cost difference per parameter type. During each simulation run, I use the same set of generated demands for the cost calculation under the old and new policy.

For the value for C I use 1.96, derived from the 95% confidence interval of the normal distribution. Furhtermore, I set $maxEdits$ equal to 1, meaning per simulation run I will make a maximum of one adjustment. Because of running time considerations, I will only use the heuristic on *Set-1*.

6 Results

6.1 MIP Model

In the first part of this results section, I present the results of the MIP model to both assess its computational performance and to compare it to the results of Tunc et al. (2018). I first present the results for *Set-1* in table 2 to assess how the model performs under different parameters settings, while table 3 reports results from Set-2 in order to assess scalability. In both tables, E-Gap refers to the relative optimality gap, while Tunc-Gap refers to the relative in the value of the objective function between my solutions and those of Tunc et al. (2018). Both gaps are denoted in percentages. Lastly, for both tables I present the number of explored nodes and the solution time in seconds. I should note that I did not impose a maximum solving time, unlike Tunc et al. (2018), who imposed a maximum solving time of 30 minutes. Every entries in the table are calculated by taking the average of that solution statistic for one specific parameter.

Table 2: Results MIP Model Set-1

Parameters		E-Gap	Tunc-Gap	Nodes	Time
π					
	Erratic	0.00	0.07	0.01	5.69
	Lumpy	0.00	0.06	0.00	5.25
N					
	20	0.00	0.00	0.00	0.86
	30	0.00	0.19	0.00	4.15
	40	0.00	0.00	0.01	11.78
ρ					
	0.1	0.00	0.04	0.00	5.44
	0.2	0.00	0.07	0.00	5.45
	0.3	0.00	0.09	0.01	5.53
K					
	225	0.00	0.10	0.01	5.32
	900	0.00	0.06	0.00	5.63
	1500	0.00	0.04	0.00	5.48
p					
	2	0.00	0.04	0.00	5.6
	5	0.00	0.06	0.01	5.52
	10	0.00	0.10	0.00	5.3
Average		0.00	0.06	0.00	5.5

Table 2 first shows that both the optimality and the gap with Tunc et al. (2018) are practically zero, showing that solutions I derived are optimal. The average number of nodes explored are also all either zero or 0.01. The average running times only differ between the different horizon lengths, which was to be expected. There are no clear differences in the average solution times, between different values of the other parameters. On the whole, the model shows no difference in performance across different parameter settings.

Table 3 displays a similar pattern, with the optimality gap, the gap with Tunc and the number of explored nodes all being zero for all six horizon lengths, showing the MIP model can still solve to optimality for larger horizon lengths. Compared to *Set-1* the solving times are significantly larger and even go up to an average of 17 minutes for $N = 100$.

For both sets of instances, my running times are nominally higher than those of Tunc et al. (2018), which can be caused a range of issues that includes a different processor. However, my solution times do show a similar patterns as those in Tunc et al. (2018).

Table 3: Results MIP Model Set-2

	MIP Model			
Parameters	E-Gap	Tunc-Gap	Nodes	Time
N				
50	0.00	0.00	0.00	24.82
60	0.00	0.00	0.00	59.17
70	0.00	0.00	0.00	146.65
80	0.00	0.00	0.00	308.75
90	0.00	0.00	0.00	576.53
100	0.00	0.00	0.00	1181.97
Average	0.00	0.00	0.00	382.98

6.2 Simulation

In the following part, I assess the reliability of the MIP formulation by simulating demand, according to the process outlined in section 5.1. I assess the reliability in multiple ways. Firstly, I compute the approximation error, caused by the linear approximation of the loss function. Specifically, A-ERR refers to the absolute difference between the value from the objective function based on the optimal order policy and the actual total costs, as outlined in equation 11. Following on that A-ERR* refers to the relative gap between the two.

Next to assess the reliability, I calculate the simulation error. This is the difference between the value of the objective function under the optimal order policy and the actual cost value of the cost function based on simulated demands. Similar to above, S-ERR refers to the absolute simulation error, while S-ERR* refers to the relative error. I also denote the average standard deviation of the simulation runs.

Both relative gaps are given as percentages. First, table 4 gives the results for *Set-1* while 5 does so for *Set-2*.

To start with A-ERR, it clearly fluctuates for different parameters settings. This is perhaps unsurprising seeing the way of linearly approximating the loss function from Rossi, S. A. Tarim, et al. 2014 is independent of the parameters settings, save for ρ. The A-ERR is higher on average for the erratic demand patterns and also shows a positive correlation for the horizon length, variation coefficient and penalty cost. It also does so for the fixed costs, but the increase is less pronounced. For all parameter settings, the relative gap is very close to zero and does not seem significant anywhere.

The simulation error is much larger than the approximation error, raising some small doubts about the reliability. The relative simulation error is constant across horizon lengths, increases with the variation coefficient, fixed cost and decreases with the penalty cost.

Table 5 shows a different pattern, with both the absolute and relative simulation errors being a lot lower. Furthermore, the absolute error does not even fluctuate too much depending with increasing horizon length. However, the absolute approximation error is positively correlated with the horizon length.

Table 4: Error rates Set-1 Simulation

	Simulation MIP Model				
Parameters	A-ERR	A-ERR*	S-ERR	S-ERR*	stdDev
π					
Erratic	18.84	0.21	-317.49	-2.33	319.45
Lumpy	15.9	0.24	-407.48	-5.29	419.1
N					
20	10.83	0.22	-241.74	-3.72	273.74
30	17.71	0.23	-355.47	-3.77	383.96
40	23.56	0.23	-490.25	-3.93	450.11
ρ					
0.1	8.39	0.14	-237.48	-2.86	190.61
0.2	17.51	0.24	-372.31	-4.01	379.17
0.3	26.2	0.31	-477.67	-4.55	538.03
K					
225	16.11	0.36	-78.16	-2.3	226.66
900	17.68	0.2	-311.58	-4	364.15
2500	18.32	0.12	-697.72	-5.13	517
p					
2	8.09	0.12	-589.18	-6.55	293.36
5	15.92	0.21	-291.4	-2.99	365.19
10	28.09	0.34	-206.88	-1.89	449.26
Average	17.37	0.23	-362.49	-3.81	369.27

Table 5: Error rates Set-2 Simulation

	Simulation MIP Model				
Parameters	A-ERR	A-ERR*	S-ERR	S-ERR*	stdDev
N					
50	92.1	0.83	-11.38	-0.11	642.16
60	113.02	0.85	-40.11	-0.31	654.95
70	125.98	0.84	-37.9	-0.26	721.72
80	147.3	0.84	-31.51	-0.19	775.25
90	162.37	0.83	-66.18	-0.34	815.29
100	182.41	0.85	-31.63	-0.15	848.01
Average	136.38	0.84	-35.86	-0.22	742.9

6.3 Receding Horizon Heuristic

The final part of the results section is dedicated to the receding horizon heuristic. First, I report the percentage of times for a specific parameter that the heuristic is triggered, which I call the average hit rate. As there can only be 1 adjustment per simulation run, the maximum hit rate is 1.00. Then for all simulation runs under both the old policy and the new policy with the adjustment, I report the average standard deviation. I also report the average difference between the total costs under both order policies in both absolute and relative terms, denoted by DIFF and DIFF* respectively. Following the equations below, a negative value for DIFF and DIFF* means that the heuristic caused a cost decrease.

$$\text{DIFF} = Average(CostsOldPolicy - CostsNewPolicy) \tag{13}$$

$$\text{DIFF}^* = Average(\text{DIFF}/CostsOldPolicy) \tag{14}$$

Table 6 below first shows stark differences in the hit rate, with the average being 13.72%. This hit rate depends partly on the value used for the confidence level used, which in this case is 1.96. The hit rate is much larger for the lumpy pattern than for the erratic pattern and also positively correlated with the coefficient of variation, the penalty costs and the horizon length, while it is negatively correlated with the fixed cost. The emerging patterns seems to be that the more uncertainty in the model and the more total number of replenishment cycles, the higher the chance of the heuristic being triggered. The latter part makes sense as with more replenishment cycles, there are more opportunities for the heuristic to be triggered.

Next, the standard deviations for both the old and the new policy do not show stark differences. Again, it increases with the coefficient of variation and the horizon length.

Lastly, by looking at the DIFF column, it becomes clear that the heuristic does not always cause a cost decrease and rather causes a cost increase in some cases. This cost increase can also be a result of the base-stock level being higher than the starting inventory, which the model does not account for. However, the sign of DIFF and its magnitude are both heavily dependent on the parameters. First, it causes a big cost decrease for the erratic pattern, while causing a small decrease for the lumpy pattern. Also, the cost decrease is bigger for the lowest coefficient of variation. Lastly, there are also strong patterns visible for the fixed cost and the penalty cost. Small fixed costs and penalty cost are associated with a cost increase, while the larger fixed costs and penalty costs demonstrate a cost decrease, meaning the heuristic performs well.

Table 6: Simulation results heuristic

		Heuristic			
Parameters	Avg Hitrate	stdDev_old	stdDev_new	DIFF	DIFF*
π					
Erratic	2.77	286.34	276.72	-131.12	-3.68
Lumpy	24.66	218.68	216.97	9.07	0.39
N					
20	8.39	115.32	111.22	-12.99	-0.44
30	16.21	216.77	215.28	-1.7	0.19
40	16.56	289.89	287.18	-4.39	0.00
ρ					
0.1	5.28	105.09	104.49	-24.99	-2.06
0.2	14.26	194.04	192.84	-2.5	0.08
0.3	21.62	272.6	268.78	-2.66	0.37
K					
225	32.15	190.85	189.51	8.56	0.32
900	8.26	347.51	343.3	-36.51	-1
2500	0.74	322.1	295.63	-194.16	-3.03
p					
2	7.44	141.86	140.63	27.82	1.66
5	13.54	207.59	206.04	1.12	-0.02
10	20.16	268.57	265.1	-20.79	-0.61
Average	13.72	225.51	223	-5.09	-0.02

Another pattern that becomes clear is that the lower the average hit rate, the higher the cost saving realised by the heuristic. This holds true for all parameters. The largest relative differences are also found when the average hit rate is lowest, namely for the erratic demand pattern and $K = 2500$. On average, the heuristic does cause a cost decrease.

7 Conclusion

In this thesis, I have investigated the stochastic lot-sizing problem, first using a MIP-formulation developed by Tunc et al. (2018). This MIP-formulation is characterised by a linear approximation of the loss function, which is done using a piece-wise linear approximation outlined in Rossi, S. A. Tarim, et al. (2014). Afterwards, I have extended the MIP model by implementing a receding horizon.

The MIP-model was shown to be computationally efficient, with reasonable running times, similar to Tunc et al. (2018). Furthermore, the gap between the values of the objective function derived in this thesis and in Tunc (2021) is negligible. This holds true both for *Set-1*, that tests a range of parameters and *Set-2*, that tests scalability.

To assess the reliability of the model, I have also performed several simulation runs, to determine both the approximation and the simulation error. For both *Set-1* and *Set-2* the approximation error is very small. The simulation error, however, is considerably large for *Set-1* and also much larger than in Tunc et al. (2018). This raises some question about the reliability of the model, although the simulation errors are very small for *Set-2*. Furthermore, the average simulation error for *Set-1* is 3.83%, which one could argue is still manageable.

Next, I have developed a heuristic that solves the model within a receding horizon environment. At the start of every replenishment cycle, it checks whether inventory is sufficiently large to meet demand the next period and if so, updates the replenishment schedule. The possibility of updates to the replenishment schedule means it is no longer a static-dynamic uncertainty model.

The cost-saving potential of the heuristic is highly dependent on the parameter settings. It works very well for high penalty costs, high fixed costs and the erratic demand pattern, but does not perform well for low penalty costs, low fixed costs and the lumpy demand pattern, even causing a cost increase at times. What is true across all parameter settings is that the lower the chance of the heuristic being triggered, the higher the cost saving potential.

As applying the heuristic also comes at the cost of having less uncertainty over the order policy, since the timing of the placement of orders could change during the time horizon. Still, the heuristic can be set to incorporate a maximum of one adjustment to the original order policy. This means that for specific parameter settings, but heuristic is useful to save costs.

The first limitation of my thesis is that the solutions derived using the MIP-formulation are technically not optimal, due to the assumption that the base-stock level at the start of ever replenishment cycle is always higher than the inventory left after the end of the preceding replenishment cycle. Secondly, the current parameter I have used for the confidence level I have chosen independent of the model parameters. The heuristic could work better if this parameter is chosen based on the ratio between the holding and penalty cost.

Another avenue for future research could be looking into allowing multiple edits per iteration, as I have currently just allowed for one. This would come at the cost of having less certainty over the replenishment schedule, but would potentially save more costs. Furthermore, the heuristic could also be extended

to variations of the MIP-formulations, such as those that employ service level constraints instead of penalty costs.

References

Bookbinder, James H. and Jin-Yan Tan (1988). "Strategies for the probabilistic lot-sizing problem with service-level constraints". In: *Management Science* 34.9, pp. 1096–1108. DOI: 10.1287/mnsc.34.9.1096.

Cárdenas-Barrón, Leopoldo Eduardo, Kun-Jen Chung, and Gerardo Treviño-Garza (2014). "Celebrating a century of the economic order quantity model in honor of Ford Whitman Harris". In: *International Journal of Production Economics* 155. Celebrating a century of the economic order quantity model, pp. 1–7. ISSN: 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2014.07.002. URL: https://www.sciencedirect.com/science/article/pii/S0925527314002102.

Dural-Selcuk, Gozdem et al. (2020). "The benefit of receding horizon control: Near-optimal policies for stochastic inventory control". In: *Omega* 97, p. 102091. ISSN: 0305-0483. DOI: https://doi.org/10.1016/j.omega.2019.07.007. URL: https://www.sciencedirect.com/science/article/pii/S0305048319302440.

Erlenkotter, Donald (1990). "Ford Whitman Harris and the Economic Order Quantity Model". In: *Operations Research* 38.6, pp. 937–946. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/170961 (visited on 07/02/2023).

Gruson, Matthieu, Jean-François Cordeau, and Raf Jans (2021). "Benders decomposition for a stochastic three-level lot sizing and replenishment problem with a distribution structure". In: *European Journal of Operational Research* 291.1, pp. 206–217. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2020.09.019. URL: https://www.sciencedirect.com/science/article/pii/S0377221720308055.

Jans, Raf and Zeger Degraeve (2008). "Modeling industrial lot sizing problems: a review". In: *International Journal of Production Research* 46.6, pp. 1619–1643. DOI: 10.1080/00207540600902262. eprint: https://doi.org/10.1080/00207540600902262. URL: https://doi.org/10.1080/00207540600902262.

Özen, Ulas, Mustafa Doğrub, and S. Armagan Tarim (June 2012). "Static-dynamic uncertainty strategy for a single-item stochastic inventory control problem". In: *Omega* 40. DOI: 10.1016/j.omega.2011.08.002.

Qin, Yan et al. (2011). "The newsvendor problem: Review and directions for future research". In: *European Journal of Operational Research* 213.2, pp. 361–374. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2010.11.024. URL: https://www.sciencedirect.com/science/article/pii/S0377221710008040.

Rossi, Roberto, Onur A. Kilic, and S. Armagan Tarim (Jan. 2015). "Piecewise linear approximations for the static-dynamic uncertainty strategy in stochastic lot-sizing". English. In: *Omega: The International Journal of Manage-*

ment Science 50, pp. 126–140. ISSN: 0305-0483. DOI: 10.1016/j.omega.2014.08.003.

Rossi, Roberto, S. Armagan Tarim, et al. (Mar. 2014). "Piecewise linear lower and upper bounds for the standard normal first order loss function". In: *Applied Mathematics and Computation* 231, pp. 489–502. DOI: 10.1016/j.amc.2014.01.019. URL: https://doi.org/10.10162%20Fj.amc.2014.01.019.

Rumyantsev, Sergey and Serguei Netessine (Oct. 2007). "What Can Be Learned from Classical Inventory Models? A Cross-Industry Exploratory Investigation". In: *Manufacturing Service Operations Management* 9, pp. 409–429. DOI: 10.1287/msom.1070.0166.

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

Seyfi, Seyed Amin et al. (2022). "Capacitated Stochastic Lot-sizing and Production Planning Problem Under Demand Uncertainty". In: *IFAC-PapersOnLine* 55.10. 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022, pp. 2731–2736. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2022.10.130. URL: https://www.sciencedirect.com/science/article/pii/S2405896322021425.

Tarim, S. Armagan and Brian G. Kingsman (2004). "The stochastic dynamic production/inventory lot-sizing problem with service-level constraints". In: *International Journal of Production Economics* 88.1, pp. 105–119. URL: https://EconPapers.repec.org/RePEc:eee:proeco:v:88:y:2004:i:1:p:105-119.

Tunc, Huseyin (2021). "A mixed integer programming formulation for the stochastic lot sizing problem with controllable processing times". In: *Computers Operations Research* 132, p. 105302. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2021.105302. URL: https://www.sciencedirect.com/science/article/pii/S0305054821000940.

Tunc, Huseyin et al. (2018). "An extended mixed-integer programming formulation and dynamic cut generation approach for the stochastic lot-sizing problem". In: *INFORMS Journal on Computing* 30.3, pp. 492–506. DOI: 10.1287/ijoc.2017.0792.

Wagner, Harvey M. and Thomson M. Whitin (1958). "Dynamic Version of the Economic Lot Size Model". In: *Management Science* 50.12, pp. 1770–1774. ISSN: 00251909, 15265501. URL: http://www.jstor.org/stable/30046142 (visited on 06/18/2023).

Wang, Hao et al. (2022). "Dynamic inventory replenishment strategy for aerospace manufacturing supply chain: combining reinforcement learning and multi-agent simulation". In: *International Journal of Production Research* 60.13, pp. 4117–4136. DOI: 10.1080/00207543.2021.2020927. eprint: https://doi.org/10.1080/00207543.2021.2020927. URL: https://doi.org/10.1080/00207543.2021.2020927.

Yang, Ya et al. (2020). "Deterioration control decision support for perishable inventory management". In: *Decision Support Systems* 134, p. 113308. ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2020.113308. URL: https://www.sciencedirect.com/science/article/pii/S0167923620300634.

A Comments on code

All the programming has been done on Java, using CPLEX.

In the accompanying zip file there are three files. The first is *thesisMain*. This is the code I used to run both the MIP model and the simulation for set 1. There are 6 different demand matrices, either lumpy or erratic, and either 20, 30 or 40 horizon length. I ran the code for all six. The parameters for the model are specified in the text and I used 500 simulation runs.

The second file theisSetTwo is analagous to *thesisMain* but instead computes the MIP model and simulation runs for *Set-2*. Here all relevant demands are put into one matrix, so there is just the need for a single run.

The last file is called *thesisExtensionHeuristic* and is used to test the heuristic. Again, one needs to run this for 6 different demand files.