University of Nottingham

A Heuristic Solution Technique to the Joint Replenishment Problem with Quantity Discounts and Full Truck Loads

By

Mohammed Al-Qaq

Student ID: 4216368

MSc Supply Chain and Operations Management

2013/2014

A Heuristic Solution Technique to the Joint Replenishment Problem with Quantity Discounts and Full Truck Loads

By

Mohammed Al-Qaq

2013/2014

A dissertation presented in part consideration for the degree of MSc supply chain and operations management

Contents

List of Figures

List of Tables

Acknowledgement

This dissertation could not be written to its fullest without Dr. Luc Muyldermans, who served as my supervisor, as well as one who challenged and encouraged me throughout my time spent studying under him. He would have never accepted anything less than my best efforts, and for that, I thank him.

I would also like to thank my parents and sister. They were always supporting me and encouraging me with their best wishes.

Abstract

In this project a stochastic multi-item inventory problem is considered. A wholesaler buys multiple products, with stochastic demand and similar holding and purchase costs, from a single supplier. The supplier offers an all-unit quantity discount whenever a full truckload is replenished. For the delivery of the products trucks with a finite capacity are available. The dispatched trucks arrive at the wholesaler after a constant leadtime and with each truck fixed shipping costs are charged independent on the number of units shipped. Since fixed transportation costs are high coordination of orders and full truckload shipments can benefit from economies of scale and quantity discounts. A new heuristic solution to this problem is proposed. The solution includes a direct grouping strategy and considers the optimal solution from both, the shipping trucks and products perspectives. In implementing the proposed solution an adjusted periodic review system is used. An excellent performance of the proposed solution can be observed when the fixed cost per order is high and the demand of the different products is similar. While the proposed solution presented in this project to the joint replenishment problem under consideration has been shown to be reliable, it nevertheless represents the first step towards the development of more efficient and versatile future solutions to the problem.

CHAPTER 1. Introduction

One of the most valuable and critical resources of any company is its inventory. In many real life situations a reduction of supply chain costs could be obtained by an efficient inventory control and management. In this project, we consider an inventory control environment in which a wholesaler buys multiple products from a single supplier. The supplier offers an all-unit quantity discount whenever a full truckload is replenished. The objective of the wholesaler is two-fold: On the one hand, they aim to minimise their shipping costs, particularly for fast moving products (i.e. highly demanded items). Therefore, if the fixed ordering cost is high and the demand is high, they will aim at high truck utilisation. Ideally, they will aim to have full truckload shipments, not only to minimise the total number of trucks needed but also to exploit the quantity discount schemes on offer and to benefit from economies of scale. On the other hand, due to the stochastic nature of demand, the wholesaler will aim to maximise the flexibility of shipments, so as to minimise the total lost sales if the demand is higher than expected, and to minimise the total inventory holding costs, if the demand is lower than expected. In other words, the wholesaler will seek small and frequent batches, so as to adapt rapidly to the market.

Clearly, these two aims are contradictory, since a full truckload policy (as favoured by the supplier) will reduce flexibility in dealing with stochastic demand, and may increase the inventory holding costs; while replenishing the inventory in small and frequent batches (as favoured by the wholesaler) will not only increase the total fixed cost of ordering and the transportation costs, but may also result in losing the advantages of the quantity discount schemes. Despite this contradiction in goals, however, the ultimate challenge is clearly to find a replenishment policy that minimises the long-run total cost, which consists of the fixed ordering costs and inventory and purchase costs.

The aim of this dissertation is to develop a new heuristic model to solve the joint replenishment problem (JRP) with stochastic characteristics of demand, quantity discounts and a fixed cost charged with each dispatched truck. In light of this, the overall objective is to minimise the total inventory control cost for joint

replenishment environments, in which a fixed cost is charged for each truck used to replenishment the different items. More specifically, the main objectives of the project are:

- To develop a heuristic to form the different items into efficient groups and apply this model using the LINGO optimisation software.

- To develop a replenishment policy where order sizes must create full truck loads to exploit quantity discounts.

- To generate experimental data based on a real life situation, to test the performance of the grouping heuristic and the joint replenishment policy.

- To perform joint replenishment simulation using LINGO on the generated data, so as to evaluate the efficiency of the proposed solution.

- To perform an adjusted EOQ simulation using Microsoft Excel.

- To compare both simulation results using the following set of performance measures: Total inventory cost, Average Inventory Level and the Service Level.

As an outline, Chapter 2 examines a number of inventory control models, notably those for the single item problem, as well as a comprehensive literature review of the joint replenishment problem. Chapter 3, articulate in detail the problem under examination in this project and, consequently, the methodologies adopted to achieve the declared aim of this project, namely to develop and evaluate a heuristic solution to a specific type of JPR. Chapter 4, Presents the set of experimental results obtained. Chapter 5, provides a detailed analysis of the results obtained from the simulation and a general evaluation of these results. Moreover this chapter provides a set of recommendations to improve the solution and presents some suggestions regarding how the proposed solution can be used in more complex environments. Finally, Chapter 6 concludes the main findings of this project and suggests potential areas for further research.

CHAPTER 2. Literature Review

One of the most valuable and critical resources of any company is its inventory. Consequently, managers have long recognised that good inventory control is crucial to corporate success. In any organisation, inventory control managers will aim, on the one hand, to cut costs by reducing on-hand inventory levels while sales managers, on the other hand, will seek to ensure minimum stock-outs. Clearly, these two aims are contradictory and companies thus need to find the right balance between low and high inventory levels by implementing the appropriate inventory control system.

According to Balakrishnan (2012), the aim of most inventory models is to minimise the total inventory cost by determining the appropriate quantities and times to issue an order. This chapter examines a number of inventory control models, notably those for the single item problem (Section 2.1), as well as a comprehensive literature review (Section 2.2.) of the joint replenishment problem.

2.1 Fundamental Inventory Control Models

2.1.1 The EOQ Model – Deterministic Demand

Developed by Ford W. Harris in 1915, the economic order quantity (EOQ) model is generally considered the most fundamental of all inventory models (Nahmias, 2009) and remains widely used in many organisations (Silver, et al., 1998). The model describes the important trade-off between the fixed ordering cost and the holding cost, and forms the basis for many more complex inventory control systems. Balakrishnan (2012) summarised the assumptions behind this model as follows:

1. Demand is known and constant.
2. Lead time is known and constant.
3. Shortages are not permitted.
4. Quantity discounts are not permitted.

5. Two variable costs are relative: The cost of placing an order and the cost of holding inventory items.

Taking these assumptions into account, Figure 1 illustrates the typical inventory usage curve of an EOQ model. In employing this model, a quantity Q is ordered whenever the inventory level reaches zero. As the curve indicates, the inventory level drops at a constant rate over time. This is because of the constant demand assumption. Accordingly, Q is ordered at fixed time intervals, as examined by Balakrishnan (2012).

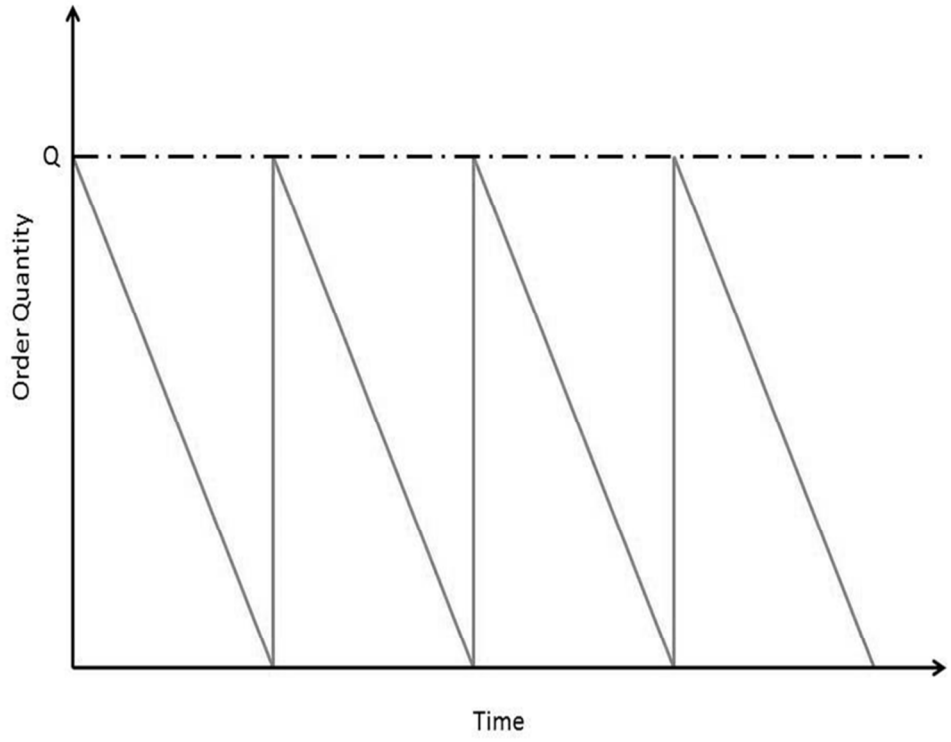


Figure 1: The EOQ inventory usage curve

Similarly to most inventory models, the objective of the EOQ model is to minimise the total cost. Silver, et al. (1998) mentioned five fundamental categories of cost: the basic production or purchase cost, the inventory carrying cost, the cost of insufficient capacity in the short run, the control system cost and the cost of changing workforce size and production rate. For an order quantity system, and in light of the above-mentioned EOQ assumptions, the significant costs are the ordering and the inventory holding costs. All other costs, such as the cost of inventory itself, are constant. Consequently, if the sum of the

ordering cost and inventory carrying cost is minimised, the total cost is also minimised.

In this regard, Figure 2 graphs the total cost as a function of the order quantity (Q), illustrating the fundamental trade-off between the inventory holding cost and the fixed cost per order. It must be noted that a reduction in Q corresponds to decreases in the total number of orders per year and, ultimately, the total ordering cost. Conversely, as the Q value increases, the organisation will have to maintain larger inventory levels and, therefore, the inventory carrying cost increases. Commonly referred to as the 'Optimal Order Quantity' (Q^*), the EOQ is the quantity at which the total cost is minimised. As shown in Figure 2, Q^* is the point at which the ordering cost curve intersects with the carrying cost curve. In other words, Q^* occurs at the point where the holding cost is equal to the ordering cost (Wee, 2013).

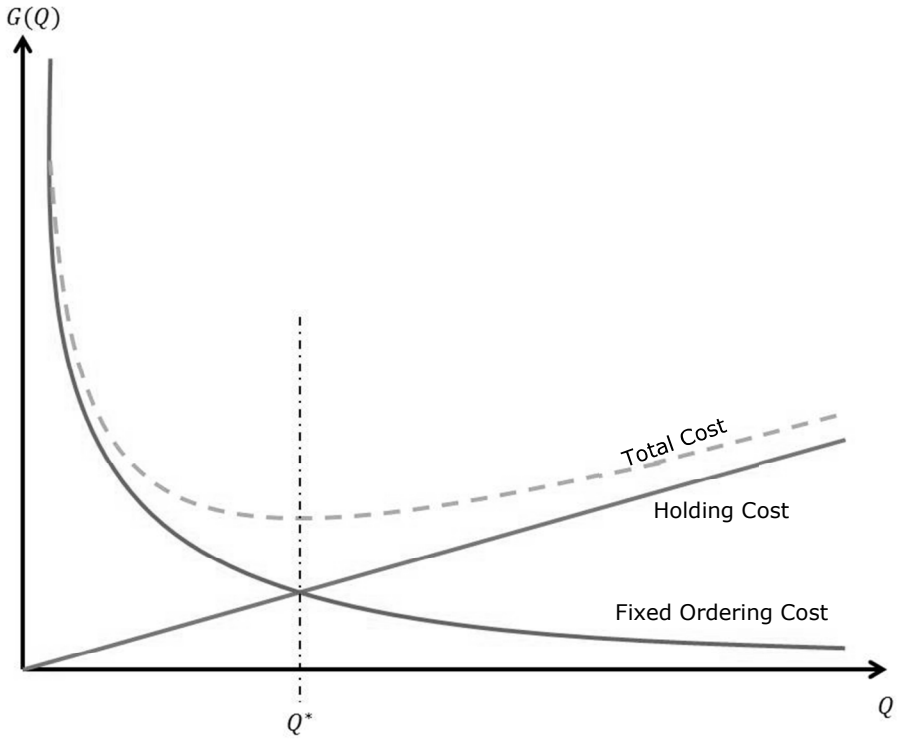


Figure 2: The total cost function curve

2.1.1.1 Determining the EOQ

Since the cost is at a minimum at Q^* and since Q^* occurs at the point where the ordering cost is equal to the holding cost, then according to Nahmias (2009) the total cost function can be expressed as:

Total Cost = Total Holding Cost + Total Purchase Cost + Total Purchase Cost

$$= \frac{Q}{2}h + \frac{\lambda}{Q}K + \lambda c \quad (2.1)$$

Where,
Q is the order quantity,
λ is the annual average demand rate,
K is the fixed cost per order,
h is the holding cost per item per year,
and c is the purchase cost per item.

Assuming a constant demand rate, the purchase cost is usually ignored. Thus, in Figure 2 the total cost function can be expressed as:

$$G(Q) = \frac{Q}{2}h + \frac{\lambda}{Q}K \quad (2.2)$$

Since the ordering cost must equal the inventory carrying cost at Q^*, then:

$$\frac{\lambda}{Q}K = \frac{Q}{2}h \quad (2.3)$$

Equation 2.2 thus becomes linear. Nahmias (2009) shows the EOQ can be determined as follows:

$$EOQ = Q^* = \sqrt{\frac{2K\lambda}{h}} \quad (2.4)$$

2.1.1.2 Inclusion of Lead time

In the previous section, it was assumed that the receipt of inventory is instantaneous. In reality, however, this might take from a few days up to several weeks. The time elapsing between placing an order and receiving items in stock is called the 'lead time' (τ) (Ghiani, et al., 2013). As such, if the assumption of zero lead time is relaxed then determining the value of Q^* is not the only decision that needs to be made. In particular, to allow for τ, it is essential to determine the level of inventory at which an order quantity of Q^* is triggered, an inventory level referred to as the 're-ordering point' (R). Figure 3 illustrates the inventory usage curve when lead times are considered, and shows an order of Q^* units is placed whenever the inventory level reaches R. Thus, the new inventory arrives after τ units of time, by which the inventory level reaches zero (Nahmias, 2009). To determine the value of R, the following equation is used:

$$R = \lambda\tau \quad (2.5)$$

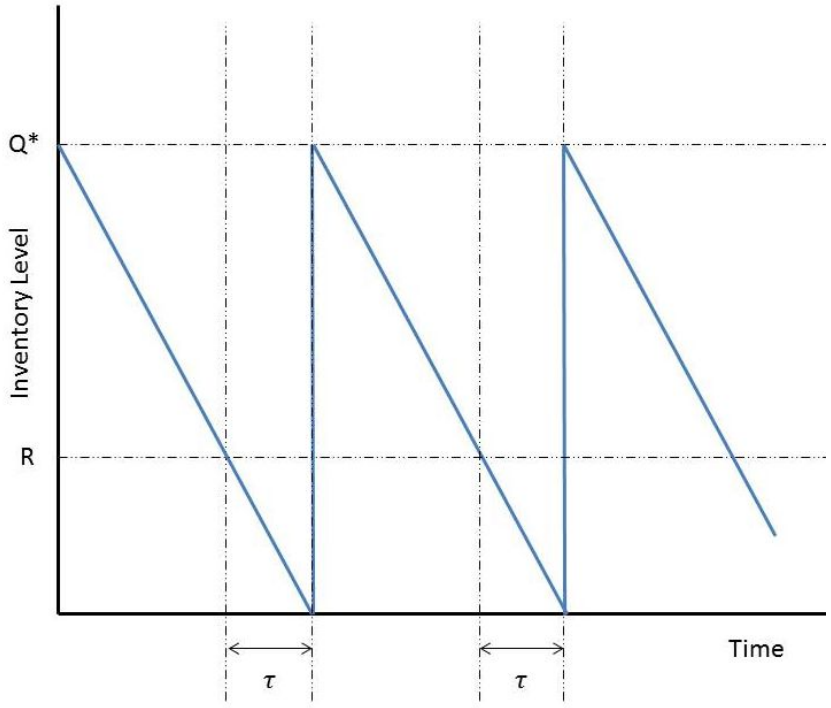


Figure 3: The inventory usage curve with lead time consideration **(Balakrishnan, 2012)**

2.1.1.3 Quantity Discounts

The discussion so far has been based on the assumption that quantity discounts are not possible, i.e. that the purchase cost per item (c) is fixed and is independent of the size of the quantity ordered. In real-life, however, this is not always the case. Indeed, to increase sales revenues and to appropriately adjust inventory levels, many firms offer quantity discounts to their customers (Ghiani, et al., 2013).

The two most common discount schedules are the 'all-units' discount and the 'incremental' discount (Silver, et al., 1998), and their respective discount order cost functions are shown in Figures 4 and 5. The break points shown in these figures represent the order quantities that trigger the quantity discount schemes. Furthermore, the cost per item (c) cam be seen to decrease each time the order quantity reaches a break point. The difference between the two schemes is that in the all-units discount scheme the discount is applied to all units of the order, while in the incremental discount scheme the discount is only applied to the additional units beyond the break point. Silver, et al. (1998) discuss several other discount schemes, while Nahmias (2009) suggests iterative solution techniques for both discount schedules. (This is discussed in further detail in Section 2.2.4).

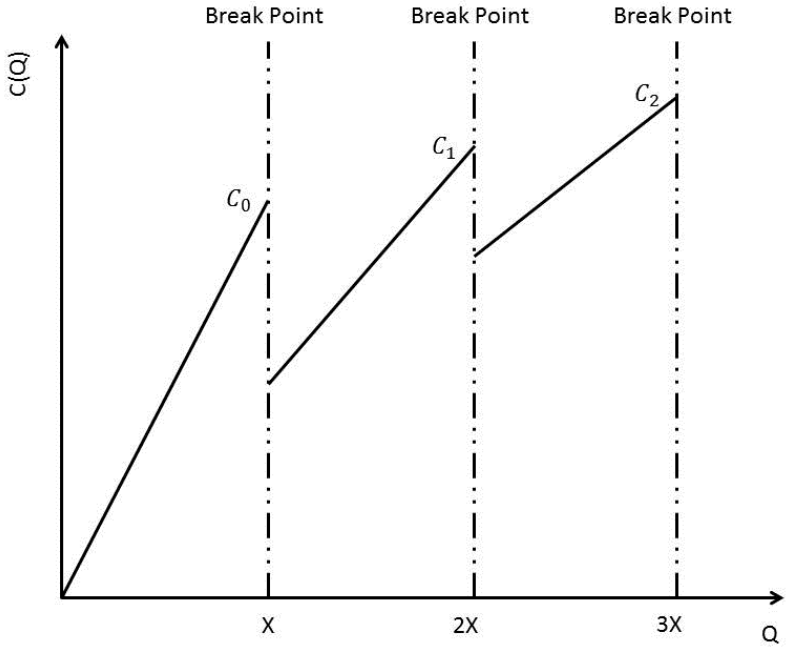


Figure 4: The Incremental Discount Schedule **(Nahmias, 2009)**

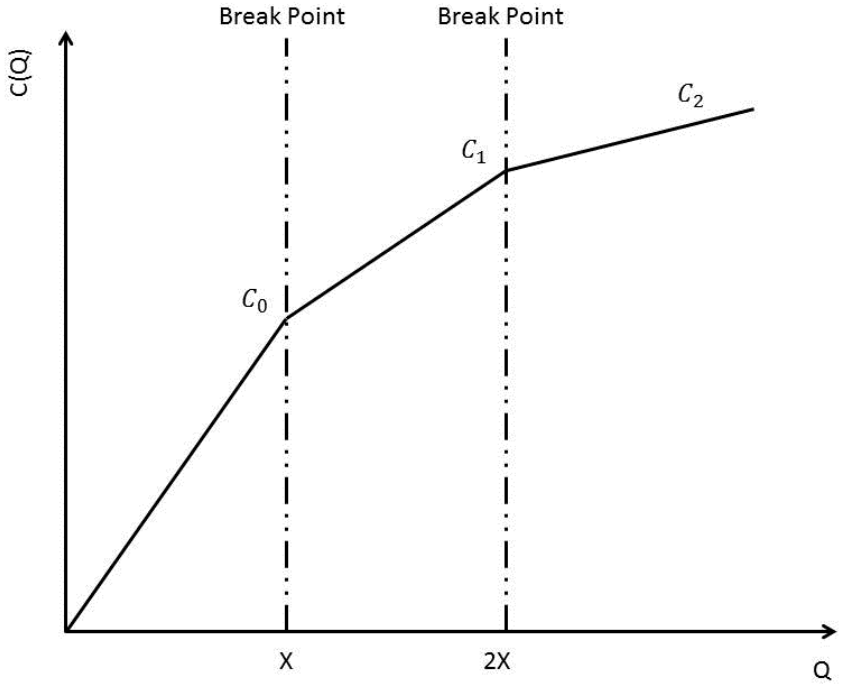


Figure 5: The All-Units Discount Schedule **(Nahmias, 2009)**

2.1.2 The Stochastic Demand Case

While the previous section dealt with the EOQ model for single items with deterministic demand rates, this deterministic demand assumption is relaxed for the present section. Unlike basic control models, stochastic inventory control models are designed to deal with the uncertainty in demand patterns and lead times (Tayur, et al., 1999). In such models, demand is often treated as a random variable that follows a random distribution (Balakrishnan, 2012). According to Silver et al., (1998) a number of inventory control systems deal with the stochastic demand case, though four principal systems remain the most common ones. These are the 'order point, order quantity' (R, Q) system, the 'order-point, order-up-to-level' (R, S) system, the 'periodic review, order-up-to-level' (T, S) system and, finally, the 'Can-Order' (S, c, s) policy. For the purpose of this project, emphasis is placed on the (R, Q) and (R, S) models in the following three sections.

2.1.2.1 The (R, Q) Inventory Model

The (R, Q) model is a continuous review system in which a fixed quantity Q is ordered whenever the inventory position drops to, or below, a predefined re-order point R. In essence, this model is an extension of the EOQ model, but with lead time considerations, as discussed in section 2.1.1.2. However, unlike the basic EOQ, the values of R and Q in this model are treated independently. In an (R, Q) policy, triggering an order of Q items is based on the inventory position and not the available stock on hand. This is because the inventory position takes into account the outstanding orders rather than solely the available items, thus avoiding excessive inventory levels.

Silver, et al. (1998) emphasise that the main advantage of this model is its simplicity and its ability to predict the production requirements for suppliers, thus facilitating production scheduling. The main disadvantage, however, is that an order of Q units may not necessarily raise the inventory position above R, especially in the case of an unexpected increase in demand. In other words, the model is not sufficiently flexible when dealing with unexpected and extreme

fluctuations in the demand pattern. According to Nahmias (2009), the following are the main assumptions behind this policy:

1. A continuous review system.
2. Demand is random and stationary.
3. Lead time is constant and positive.
4. Quantity discounts are not permitted.
5. Shortages are possible and penalty costs are therefore assumed.
6. Two other costs are assumed in the total cost function: The inventory holding cost and the cost per order placed.

According to Silver, et al. (1998), following a normal distribution function, the random variable of interest in this model is the demand during lead time (D). The expected mean demand during lead time, $E(D)$, is calculated as follows:

$$E(D) = \mu = \tau\lambda \quad (2.6)$$

Meanwhile, the standard deviation of demand during lead time (σ_τ) is given as:

$$\sigma_\tau = \sqrt{var(D)} \quad (2.7)$$

In contrast to the deterministic inventory control models, the objective of this model is to minimise the 'expected' annual cost rather than the actual total cost. With this in mind, the following equation is used to describe the total cost function (Nahmias, 2009):

Total Expected Cost = Average Annual Expected Cost
+ Average annual Ordering Cost
+ Average Annual Penalty Cost

$$= G(Q,R) = h\left(\frac{Q}{2} + R - \lambda\tau\right) + K\frac{\lambda}{Q} + p\lambda\frac{n(R)}{Q} \quad (2.8)$$

In Equation 2.8, p stands for the stock-out cost identified by managers. Nahmias (2008) described an analytical approach to determine the optimal values for Q

and R, so as to minimise $G(Q,R)$. Based on this approach, Nahmias (2009) shows the optimal solution involves the iterative solving of the following two equations:

$$Q = \sqrt{\frac{2\lambda[K + pn(R)]}{h}} \quad (2.9)$$

$$1 - F(R) = \frac{Qh}{p\lambda} \quad (2.10)$$

This solution involves toggling iteratively between the two equations until two successive values of Q and R are found to be the same. Assuming a normal distribution of demand, the expected number of stock outs incurred in a cycle, $n(R)$, is computed using the following standardised loss function:

$$L(z) = \int_z^{\infty} (\tau - z)\, \phi(\tau)\, d\tau \quad (2.11)$$

Where $\phi(\tau)$ is the standard normal density. Given a normal lead time, with mean μ and standard deviation σ, it follows that:

$$n(R) = \sigma L\left(\frac{R - \mu}{\sigma}\right) = \sigma L(z) \quad (2.12)$$

The value of the standardised variate z is obtained using the normal probability and partial expectations tables provided by Nahmias (2009).

2.1.2.2 Service Levels in (R, Q) Systems

Although the (R,Q) with penalty costs model outlined in the previous section is realistic in describing many real life inventory control environments, it is nevertheless difficult to determine the stockout cost (p). A common alternative to the stockout cost is the service level, generally defined as the probability that a demand or collection of demands is met. Two widely accepted types of services are the Type 1 and the Type 2 services. Type 1 service is defined as the probability of not stocking out during the lead time (α). The values of an (R,Q)

system that is subject to Type 1 service constraint is determined as follows (Nahmias, 2009):

1. Determine R to satisfy $F(R) = \alpha$
2. Set Q = EOQ obtained in Equation 2.4

Meanwhile, the Type 2 service type is defined by the proportion of demands that are filled from stock (β). In general, the Type 2 constraint is formulated as:

$$\frac{n(R)}{Q} = 1 - \beta$$

The term on the left-hand side of the equation represents the average fraction of demand that stocks out every cycle. In a Type 2 service setting, the order quantity to the EOQ provides a valid approximation. Therefore,

$$n(R) = EOQ(1 - \beta)$$

This is also known as the 'fill rate'. This type of service is more complex than Type 1 service. This is because it involves both Q and R. Silver, et al. (1998) explain both types in details and provide the relevant calculations.

2.1.2.3 The (T,S) Inventory Policy

Unlike the (R,Q) model, the (T,S) system is a periodic review model. Also known as the 'replenishment cycle' model, it is widely used in organisations running manual inventory control systems and in firms ordering from a single supplier. Although the term R is commonly used to refer to the review period, T is used in the present project to avoid confusion with the re-order point described in the previous section. The control procedure is that, at every T, a quantity Q is ordered such that the inventory level is raised up to level S, as illustrated in Figure 6.

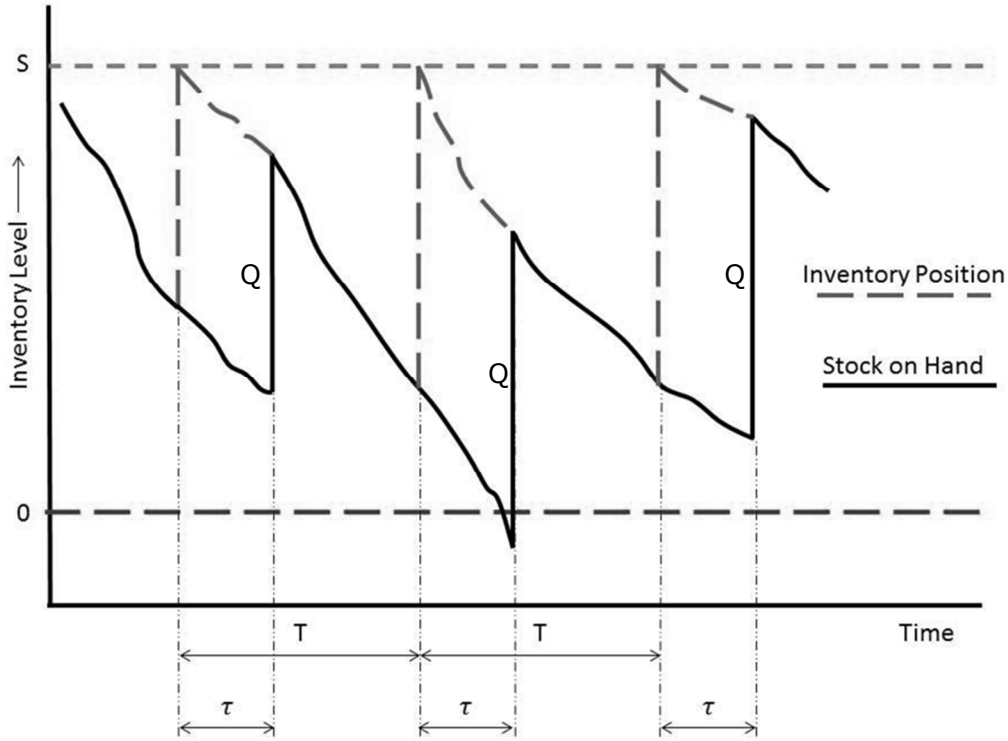


Figure 6: Typical Inventory Usage Curve for (T, S) Systems **(Silver, et al., 1998)**

Due to its periodic nature, this inventory system can provide significant savings through facilitating the coordination of replenishments of related items. Moreover, in this system, the value of S can be adjusted at regular intervals to cope with changing demand patterns over time. A main disadvantage, however, is the high holding cost compared to continuous review inventory control systems. The policy parameters are computed as follows:

The optimal review period (T^*) is essentially Q^* expressed in terms of time supply. The EOQ formula to represent the supply time can be expressed as: (Silver, et al., 1998)

$$T^* = \sqrt{\frac{2K}{\lambda h}} \quad (2.13)$$

Further, according to Silver (1985) and assuming a Type 2 service constraint, the order-up-to level is computed similarly to how the order point was in the previous section:

(R,Q)	(T,S)
R	S
Q	DT
τ	$\tau + T$

2.2 The Multi-Items Inventory Control Problem

While most of the inventory control models described in the previous section are efficient in dealing with the single item inventory replenishment problem, they are less so when it comes to the multi-item inventory control problem.

Commonly referred to as the 'joint replenishment' problem (JRP), the multi-item inventory control problem is frequently encountered in real-life situations; for instance, when a vendor purchases a group of different products from the same supplier, or when different items are waiting to be packaged on the same machine. Indeed, it has been a prominent research topic for years and many solution algorithms have been proposed since the pioneering work by Starr and Miller (1962). The first part of this section provides an introduction to the joint replenishment problem, while the second part presents a number of solutions to the basic deterministic JRP from the literature. The third part discusses two fundamental models for solving the stochastic JRP, while the fourth part considers the JRP with quantity discount. Finally, the fifth part discusses several models for solving a special case of the JRP subject to capacity constraints.

2.2.1 An Introduction to the Joint Replenishment problem (JRP)

2.2.1.1 Definition

There are many types of inventory control models dealing with multiple product systems (Aksoy & Erenguc, 1989) (Goyal & Satir, 1989) (Simpson & Erenguc, 1995) (Silver, et al., 1998) (Khouja & Goyal, 2008) (Glock, 2012). In most cases, the objective of these models is to minimise the total cost while satisfying demand (Aksoy & Erenguc, 1989). Similar to the single-item inventory control models described in Section 2.1, in the multi-item inventory problem the total

cost is composed of the setup cost and the inventory holding cost. In many cases, however, the setup cost structure is split into two components: First, a major ordering (replenishment) cost applies when a family of items is replenished that is independent of the number of different products in the order. Second, minor replenishment costs are associated with every item and will only occur when a particular item is replenished (Goyal, 1973). Additionally, for each item, there is an individual demand rate and inventory holding cost (Andreas, 2006). As mentioned earlier, finding optimal order quantities for multiple products with major and minor costs is a commonly occurring problem called the Joint Replenishment Problem (Schouten, et al., 1994). Goyal (1973) provides further definitions and descriptions of the joint replenishment problem while Silver, et al. (1998) have examined real-life examples on both procurement and manufacturing environments.

2.2.1.2 Advantages and Disadvantages

According to Silver, et al. (1998), in real-life procurement processes the coordinated replenishment of different items is very common for a number of reasons. First, to increase sales, many suppliers offer quantity discounts on the purchase cost when the total number of items in an order is greater than a certain quantity. Further, in many industries, a quantity discount may be realised if a quantity such as a full container on a ship or a full truckload is ordered. In many cases, exploiting such quantity discounts may lead to significant cost savings on the total purchase cost as well as on the transportation cost-per-unit. Second, since the setup cost of placing an order is normally high, a reasonable strategy would thus be to reduce the total number of orders annually by aggregating the different items into a single order. Finally, the coordinated replenishment of items facilitates the scheduling of buyer time, receiving and inspection work load.

However, Donal (1998), Silver, et al., (1998) and Khouja and Goyal (2008) have identified several disadvantages of implementing joint replenishment systems. First, there is a possible increase in the average inventory level. This is because when different items are jointly replenished some will be reordered earlier than

otherwise needed. Second, unlike single item inventory systems, the dependence of one product on other items reduces the flexibility in dealing with unexpected situations such as a sudden increase in demand for a particular item. Third, due to the complex nature of joint replenishment systems, they are usually more costly to implement and run than single alternatives.

2.2.1.3 Strategies for Dealing with the JRP

There are two different types of strategies to deal with the JRP. In their work, Eijs, et al. (1992) classify these into 'direct grouping' strategies and 'indirect grouping' strategies. For consistency, in this project a 'group' is defined as the set of items that are replenished in the same replenishment instance and share the same replenishment cycle. When using indirect grouping strategies, joint replenishment opportunities are scheduled at constant time intervals and the quantity ordered for each item is sufficient to last for an integer multiple of the basic time unit (Andreas, 2006). Further, products that have the same integer multiple are grouped and replenished together.

This is the most common approach to the problem and, clearly, the main challenge it presents is to identify the basic cycle time and the integer multiplier for each product, such that the total cost is minimised. Under direct grouping strategies, different items are economically grouped together where each group has its own base period time (Andreas, 2006). Each group has its own base period time and the number of groups is specified in such a way that the total cost of items in the family is as low as possible. When comparing both appraches as applied to a number of problems, Eijs, et al. (1992) concluded that the indirect grouping methods outperform the direct grouping alternatives in terms of the total cost. This is especially the case where the major ordering cost is high compared to the minor individual costs.

2.2.2 The Basic Joint Replenishment Problem: The Deterministic Case

The assumptions underlying the derivation of the basic joint replenishment inventory control system are similar to those of the economic order quantity discussed earlier in Section 2.1. Typically, the basic joint replenishment problem relies on the following assumptions (Silver, 1998):

- A constant and deterministic demand rate for each item over an infinite time horizon.
- Shortages are not permitted.
- Individual items are replenished at equal time intervals.
- The replenishment lead-time is of zero duration.
- The entire quantity is delivered at the same time.
- There are no quantity discounts.

For the purpose of this dissertation, the following terms are used to describe the JRP:

T Base time period, within zero, one or several items can be replenished
S Major replenishment cost for the family of items
TC Total costs function
i A product index: 1,2,…,n
n Number of items in the family
D_i Demand rate in units-per-unit-time for item i
h_i Inventory holding-cost-per-unit and time unit for item i
s_i Minor replenishment cost for item i
k_i The integer number of T intervals that the replenishment quantity of item i will last
Q_i Order quantity of product i
T_i Time interval between successive replenishments of product i

As mentioned earlier, in solving the JRP the objective, in most cases, is to minimise the total inventory cost-per-unit time. Three costs are assumed in the total costs function: a major fixed ordering cost, a minor ordering cost per item

and the inventory carrying cost. Further, the JRP is expressed in terms of determining the frequency of the basic replenishment cycle and the frequency of replenishing individual items. This ratio is a positive integer used to decide how often a product is replenished (Goyal & Satir, 1989). For example, if this ratio for a particular item is 1 then it is included in every replenishment cycle. If the ratio is 2, then the product is replenished every two replenishment cycles. Sections 2.2.2.1 and 2.2.2.2, respectively, explain the total cost function and highlight the most recognised solution models to the JRP under deterministic demand.

2.2.2.1 The Total Cost Function

As already mentioned, under a joint replenishment inventory control system the time between two successful replenishments for any product (T_i) is a positive integer multiplier (k_i) of T, so that:

$$T_i = k_i T \quad (2.14)$$

The order quantity for product i is given by:

$$Q_i = T_i D_i = T k_i D_i \quad (2.15)$$

The total annual holding cost (C_H) is given by:

$$C_H = \sum_{i=1}^{n} \frac{Q_i h_i}{2} = \frac{T}{2} \sum_{i=1}^{n} k_i D_i h_i \quad (2.16)$$

And the total annual ordering cost (C_o) is given by

$$C_o = \frac{S}{T} + \sum_{i=1}^{n} \frac{s_i}{k_i T} = \left(S + \sum_{i=1}^{n} \frac{s_i}{k_i} \right) / T \quad (2.17)$$

The total cost function under basic joint replenishment system is thus given by:

$$TC\ (T,K) = C_H + C_O = \frac{T}{2}\sum_{i=1}^{n} k_i D_i h_i + \left(S + \sum_{i=1}^{n} \frac{s_i}{k_i}\right)/T \quad (2.18)$$

The time period can be found by deriving equation 2.18, as follows:

$$T(k_1, k_2, , , k_n) = \sqrt{2\left(S + \sum_{i=1}^{n} \frac{s_i}{k_i}\right) \Big/ \sum_{i=1}^{n} k_i D_i h_i} \quad (2.19)$$

If equation 2.19 is substituted back into equation 2.18, then the total cost function, given a set of k values, can be formulated (Khouja & Goyal, 2008) as follows:

$$TC(k_1, k_2, , , k_n) = \left(S + \sum_{i=1}^{n} \frac{s_i}{k_i}\right) \times \sum_{i=1}^{n} k_i D_i h_i \quad (2.20)$$

2.2.2.2 Solutions to the JRP under Constant Demand

According to Khouja and Goyal (2008), a policy defined by the basic replenishment cycle and a set of multipliers for individual products, as in the joint replenishment policy, is known as a 'cyclic policy'. In a special form of the cyclic policy, known as the 'strict cyclic' policy, at least one product has an integer multiplier $k_i = 1$. Under both policies, when solving the JPR the objective is to minimise the total cost function in Equation 2.18. In other words, the objective is to solve the following optimisation problem:

$$\min TC\ (T,K) \quad (2.21)$$

The main difficulty presented by the JRP involves finding the best set of k-values. As such, several authors have proposed heuristic solutions to the minimisation problem in Equation 2.21. For example, in his early work, Brown (1967) proposed a heuristic method based on an iterative technique to find the set of k-values. Further, Shu (1971) developed a specifically designed method for the JRP with two items. Based on the value of the optimal T for a family of

products, Goyal (1973) and (1974a) used upper and lower bounds for T to find best values for the different k_is. All possible combinations of T are then evaluated, and the combination with the lowest cost is chosen.

Although many sub-optimal heuristics are available to solve the JRP, only a few algorithms are able to find an optimal solution. One widely recognised algorithm was developed by Goyal (1974b), who presented a systematic procedure for obtaining the optimum values of T and $k_i s$ to minimise Equation (2.20). The algorithm is based on the enumeration of the total cost function between a lower and an upper bound of T. However, according to Eijs (1993), this method can guarantee optimal solutions only for strict-cyclic policies. Similarly, Andreas (2006) highlighted the difficulty of determining the lower bound of *T,* and pointed out that the method does not guarantee the optimal solution in all situations. Therefore, Goyal (1988) and then Eijs (1993) adjusted the method introduced by Goyal (1974b) to guarantee optimality for basic cyclic policies. The former improved the method by determining a new lower bound.

Moreover, since the JRP is a non-deterministic polynomial-time hard problem (Arkin, et al., 1989) an algorithmic optimal solution may be complex and thus computationally prohibitive for large problems. In addition, the solutions proposed by Goyal (1974b), Goyal (1988) and Eijs (1993) and many others are onerous to implement and use on a regular basis. As such, a number of more efficient alternative heuristic algorithms have been developed to solve the JRP.

For instance, Silver (1976) introduced an efficient and non-iterative heuristic algorithm for solving the JRP that makes use of a convenient graphical aid and has produced excellent results on a number of numerical tests. In addition to the assumptions underlying the basic JRP that were discussed in Section 2.2.2, this model assumes a simple cyclic policy in which different items are jointly replenished every T unit time, with each item i replenished by a quantity sufficient to last $k_i T$ unit time. Through his method, Silver (1976) modified the economic order quantity to handle the replenishment of multiple items. The algorithm ranks the different items in an ascending order of the value of $\frac{s_i}{D_i c_i}$. The

item with the lowest ranking is replenished every T while for all other products, the $k_i s$ are calculated as follows:

$$k_i = \sqrt{\frac{s_i}{D_i c_i} \frac{D_1 c_1}{S + s_i}} \quad (2.26)$$

The different $k_i s$ are subsequently rounded-up to integer multipliers of the first item. This procedure has produced results at, or near, the optimal solution in numerous tests (Silver, et al., 1998). The solution's sub-optimality, however, is due to the adjustment of the k_i values. Goyal and Belton (1979) suggested another method of calculating the different $k_i s$ that was further improved upon by Kaspi and Rosenblatt (1983), who combined it with the iterative approach of finding the $k_i's$ in Goyal (1974a).

Moreover, Kaspi and Rosenblatt (1991) adopted work by Silver (1976) to develop an enumerative heuristic algorithm called the RAND method. Similar in concept to Goyal (1974b), this method depends on computing upper bounds and lower bounds for the optimal replenishment cycle T^*. However, in using the RAND method, the range between both bounds is divided into a number of equally-spaced values of T. For each potential value of T, the improved Silver's (1976) algorithm proposed by Kaspi and Rosenblatt (1983) is used. Goyal and Deshmukh (1993) improved the RAND method by introducing improved tighter bounds. Khouja and Goyal (2008) emphasised that a tighter lower bound is desirable since it reduces the range of T.

Furthermore, Kaspi and Rosenblatt (1991) performed a comparative study to test the performance of the available algorithms, and showed their RAND method outperformed previous non-enumerative algorithms. Olsen (2005) used a direct grouping strategy that slightly outperformed the method adopted by Kaspi and Rosenblatt (1991). Meanwhile, Viswanathan (1996) proposed a new optimal algorithm for the JRP that reduces the computational effort required by Goyal (1974b) and Eijs (1993) by improving the bounds for T in an iterative approach. For their part, Porras and Dekker (2006) showed that it is possible to solve the JRP with minimum order quantities by applying a global optimisation

procedure. Porras and Dekker (2006b) performed a comparative study on the efficiency of optimal algorithms for the JRP, and their study revealed that for large minor set-up costs and moderate major set-up cost, Porras and Dekker (2006) outperformed Goyal (1974b) and Viswanathan (1996). Further comparative studies are examined in Viswanathan (2002).

2.2.3 The Stochastic Case

In the previous section only the case of constant demand has been considered. In many cases in real life, however, the demand is uncertain which adds complexity to the JRP. Although a number of authors have presented several strategies to minimise the total expected cost per unit time, two approaches remain the most commonly used in practice: The Can-Order system, (S, c, s), and the periodic review system (Silver, 1998). The objective function of stochastic joint replenishment models is expressed as the minimisation of the total relevant expected costs. The costs assumed are essentially the same as those assumed in solving the deterministic case. However, unlike the deterministic case, in the stochastic case shortages are permitted and therefore penalty costs are assumed. In the following two sections, both the (S, c, s) systems and the periodic review systems for solving the JRP are presented and discussed.

2.2.3.1 Continues Review Systems

Introduced by Balintfy (1964), the Can-Order (S, c, s) system is a special type of continuous review system for controlling jointly replenished items. In implementing this model, three inventory levels are computed individually for each product in a family. These levels are the order-up-to level (S_i), the can-order level (c_i) and the re-order level (s_i). This system is designed for environments where the main priority is to minimise the setup cost. Can-Order systems are particularly useful in manufacturing environments where several products make use of a single machine. In using this system for a group of items, whenever the inventory position of any item i reaches the 'must order' level, s_i, a replenishment action is triggered to raise the item's inventory position up to its order-up-to level, S_i. At the same time, any other item j within the

same group can also be replenished given that its inventory position is below its can-order level, c_j. If item j is included then its inventory position is also raised up to its order-up-to level, S_j (Goyal & Satir, 1989). The underlying principle behind the can order level, c_j, is to indicate that item j is likely to be replenished soon. Therefore, by including item j with the replenishment instant triggered by item i, savings in future fixed ordering costs may be realised.

Figure 6 shows the behaviour of an item in a (S, c, s) system. The item triggers a replenishment at time t_1. At t_2, a second item in the group triggers a replenishment, but the first item is not included since its inventory position is above the c_i level. At time t_3, a third item triggers a replenishment in which the first item is included as its inventory level is below the c_i level.

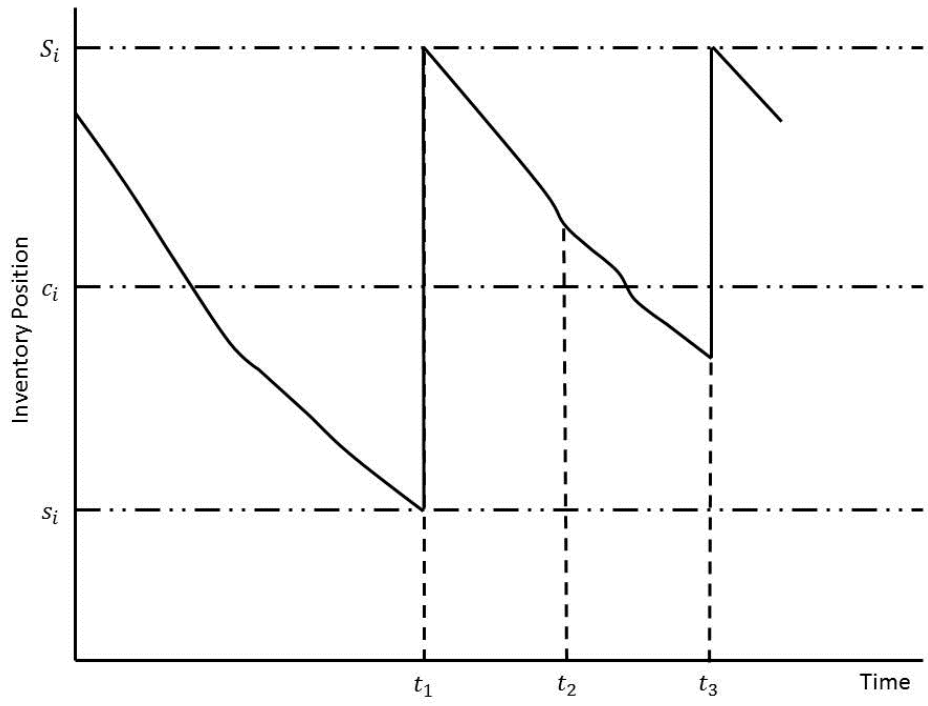


Figure 7: Behaviour of an item under the (S, c, s) System **(Silver, et al., 1998)**

A disadvantage of this model is that finding the control variables for all three levels is a difficult task, especially when the number of different products is high. Therefore, researchers have considered different ways of calculating these levels so as to simplify the implementation of the system. Silver (1965), for example, considered a situation involving two items having identical characteristics, including unit Poisson demand. The simple Joint ordering policy is to order both items up to a level Q whenever either of them drops to a zero level. Ignall

(1969) considered a similar problem where demand is determined by two independent Poisson processes. Assuming zero lead time and Poisson demand, Silver (1973) studied three procedures to determine the total cost function. The zero lead time assumption is relaxed in Silver (1974). The author emphasised that substantial cost savings, especially when the fixed costs are high, can be obtained by implementing the (S, c, s) system. Silver (1981) and Federgruen et al. (1984) also considers (S, c, s) systems with a Poisson demand distribution.
In more recent work, Schultz and Johansen (1999) developed a decomposition algorithm for items coordinated by the (S, c, s,) policy. The algorithm assumes that the time between ordering is Erlang-distributed. The authors emphasised that this algorithm performs well in comparison with others. Meanwhile, Melchiors (2002) presented a new method for calculating the (S, c, s,) policy parameters. The method is based on a compensation approach, where an item placing an order receives compensation from other items benefitting from the order opportunity. Johansen and Melchiors (2003) present a new method to obtain sub-optimal 'can order' policy based on Markov decision theory. This method, however, assumes a periodic review inventory system. Tsai and Huang (2009) used a clustering algorithm to evaluate the correlated demands among different items in (S, c, s,) systems. According to the authors, this method helps to overcome any reduced efficiency when a large number of items are considered.

2.2.3.2 Periodic Review System

Periodic replenishment systems are widely used in solving the JRP (Johansen & Melchiors, 2003). Developed by Atkins & Iyogun (1988) for passion distribution, these systems outperform the (S, c, s) system in many cases (Khouja & Goyal, 2008). In their work, Atkins and Iogyun (1988) propose two replenishment policies: the first is a periodic policy where all items are ordered in quantities that raise their inventory position to a base stock level at every replenishment instance. In essence, this method is an extension of the system discussed earlier in Section 2.1.2.3. The second method is a modified periodic policy where similar items are replenished at each replenishment instance - where other items are ordered at regular replenishment opportunities. In contrast to the first method, this method accounts for sequence-dependent setup times and production

capacity constraints. Both methods are based on the notion of allocating the major replenishment costs in small amounts to the fastest moving products. Silver et al. (1998) provide detailed steps for how to apply the procedures that Atkins and Iogyun (1988) propose. McGee and David (1996) implemented this procedure for a fastener manufacturer using a spreadsheet to find the production quantities and safety stock.

Other periodic review systems include the work of Viswanathan (1997)who presents a policy called the P (s, S) policy, where at every review instant all the items having an inventory position below their order level (s) are included in the replenishment. Backed up by several numerical tests, according to the authors, this method outperforms all earlier periodic review approaches. Nielsen and Larsen (2005) have improved upon the P(s, S) policy by working out an analytical solution procedure, which, they argue, has been shown to be superior to the (S, c, s) system on almost all test problems evaluated in their paper.

2.2.4 The Joint Replenishment Problem with Quantity Discounts

Most of the previously mentioned joint replenishment models assume a constant purchase cost-per-unit, i.e. that the purchase cost per item is independent of the order quantity. However, as mentioned earlier in Section 2.1.1.3, quantity discounts to purchase costs may be offered by suppliers to motivate customers to order larger quantities, whereby the cost per unit is reduced if the quantity purchased is larger than a specified break point. However, ordering in larger quantities to obtain the discounts will not always lead to total cost saving, since additional units increase inventory holding costs. Therefore, it is essential for inventory control managers to determine when to take advantage of such discounts. According to Cha and Moon (2005), the most common discount schemes are the all-units discount schedule and the incremental discount schedule, both discussed in Section 2.1.1.3 below.

Many authors have studied these quantity discount schedules, most notably Silver et al., (1998), Miltenburg (1985) and Miltenburg (1987). Dolan (1987) and Wilson (1993) provide a number of examples of different quantity discount schedules in many real-life situations. Pirkul and Aras (1985) examined the EOQ

model with an all-units discount, and tackled the multi-item problem by finding the EOQ for each item separately while considering the quantity discount scheme. Similarly, Guder et al. (1994) considered the multi-item problem but with an incremental quantity discount scheme based on the independent cycle approach. Guder and Zydiak (1997) studied the multi-item inventory problem with a single resource constraint and quantity discounts. They proposed a heuristic approach for generating non-stationary ordering policies, with order quantities that can vary over time. Hariri et al. (1995) presented a geometric programming approach for determining the inventory policy for the same situation examined by Guder and Zydiak (1997), an approach based on two models formulated as profit maximisation problems rather than the more conventional cost minimisation problem. Benton (1991) studied quantity discount schedules under conditions of multiple items, resource limitation and multiple suppliers, and proposes an efficient heuristic programming procedure for evaluating alternative discount schedules.

While many authors have examined quantity discount schedules under conditions of multiple items using the single item replenishment theories, few have considered the quantity discount schedules under joint replenishment conditions before the work of Chakravarty (1984), who proposed a grouping procedure aimed at replenishing different items at the same instant so as to exploit the group discounts available on the total purchase value of a group, as well as the economies of scale of the fixed cost of ordering. The optimal groups are formed such that the annual (dollar) usage values of the items do not decrease, from the first to the last group. This method, however, is only applicable with an all-unit discount schedule (Cha & Moon, 2005). In contrast to Chakravarty (1984), Chung et al. (1996) present an effective heuristic to solve the JRP with dynamic demand under both incremental discount schedules and all-unit discount schedules. This method uses the variable redefinition technique of Eppen and Martin (1987) to obtain tight lower bounds for reasonable size problems. Moreover, contrary to Miltenburg (1985) and Miltenburg (1987), who used a dynamic re-order point policy, Schouten et al. (1994) adopted a stationary can-order strategy as a basic ordering strategy in their proposed heuristic method, so as to incorporate quantity discount schedules under stochastic demand conditions.

Silver et al. (1998) present a simple solution to the stochastic JRP with quantity discounts, based on the work of Miltenburg (1982), which outperformed the more widely known IMPACT and INFORM systems. The solution works with both periodic review and continuous review systems. When an item reaches its re-order point and triggers group replenishment, the inventory level is examined. Depending on this level, a group replenishment quantity is selected while taking into account the available discounts. Miltenburg and Silver (1989) propose a probabilistic decision rule for similar conditions. Eijs et al., (1992), in their review of the literature on this topic, propose a method for incorporating discounts into the framework of can-order strategies. They found that, for small problems, the optimal strategy within this class can be identified with a semi-Markov decision model. For large sized problems, a one period look ahead heuristic was proposed.

It's worth mentioning that most of the models presented so far deal with stochastic demand characteristics. Cha and Moon (2005), on the other hand, developed an algorithm, based on the adapted RAND method, that deals with the JRP under quantity discounts and constant demand, where the authors considered the all unit discount schedule. Another model dealing with the same case of constant demand was proposed by Silver et al. (1998), who highlighted that the analysis for multiple items with quantity discounts is much more difficult compared to the single item systems. This is because, in a joint replenishment process, it is typically not necessary to have all the items replenished at each replenishment instance. Furthermore, the authors highlight that a solution where the discount is achieved in certain replenishments only is complex to analyse, since T is not the same for each item. Therefore, rather than attempting to explicitly model such complex systems, a compromise is suggested in the form of three possible solutions: The first solution assumes that the total replenishment is always sufficient to archive quantity discounts, the second solution assumes that the best result is achieved at the breakpoint, and the third possibility assumes no quantity discounts. Once evaluated, the solution with the minimum cost is subsequently chosen. It must be noted that numerous further research efforts have been carried out on the topic (Chen & Min, 1994) (Khouja

& Saydam, 2005) (Li & Liu, 2006) (Moon, et al., 2008) (Kamalia, et al., 2011) (Kang & Lee, 2012) (Lee, et al., 2013) (Paul, et al., 2014).

2.2.5 The Full Truck Load JRP Problem

Most of the replenishment policies mentioned in the previous sections charge a fixed cost for a replenishment order, independent of the order quantity. As such, they do not take into account capacity restrictions on the total order volume. According to Kiesmuller (2009), such models, in many cases, result in a truck capacity utilisation of 1% and, in other cases, an order quantity that requires more than a full truckload. Therefore, most of these models are not applicable in situations where a fixed cost is charged for each dispatched truck.

Some studies (Miltenburg, 1985) have considered capacity restrictions in a JRP for continuous review policies and periodic review policies. Assuming that the total volume of the order is known, the author proposes an algorithm for allocating the total order volume to each item within a family of items. Unlike the models presented in previous sections, this model aims on maximising the time between different orders. This method is called the 'Service Point' method, and has been successfully deployed in a number of industrial applications (Silver, et al., 1998). Some studies (Carlson & Miltenburg, 1988) have discussed the method in details, providing an application of this allocation rule where replenishment orders are triggered based on the service level. Other studies (Pantumsinchai, 1992) perform a comparative analysis between the Service Point method, Can-Order policies and Periodic policies. The research emphasises that the Service Point method and the Periodic method performed better than the Can-Order policy.

A further study (Eijs, 1994) provides another approach that takes into account capacity constraints. The author presents a periodic replenishment policy in which an item can be ordered at every review instant. Accordingly, the order sizes are initially determined based on the single item replenishment policies. A Markovian decision model is then used to estimate any additional ordering and holding costs, due to adjustments in order quantity. Subsequently, a decision is taken as to whether to enlarge the initial order quantities, so as to benefit from economies of scale. This decision is based on comparing the expected additional

ordering and holding costs associated with the enlargement of the order quantity and the cost of the extra replenishments when the transportation capacity is not fully utilised. In case of positive expected cost savings, the initial order size is enlarged to a full container-load. According to some researchers (Kiesmuller, 2009) this method can achieve significant cost savings though it "can still lead to small total replenishment volumes resulting in low container utilization".

In another research study on a similar problem (Cachon, 2001), a periodic review policy is presented that avoids low truck utilisation. According to this method, orders are only shipped in full truckloads. Orders with less than a specified truck utilisation are delayed until the next replenishment instant. Unlike other approaches (Eijs, 1994), when implementing this policy, orders can only be reduced in size to allow for full truck utilisation. The allocation of different items into a particular shipment is carried out based on a 'first-come-first-serve' rule.

Another study (Kiesmuller, 2009) combined the methodologies of both (Eijs, 1994) and (Cachon, 2001). The result is a dynamic and periodic joint replenishment control model in which the different order quantities can be enlarged, as well as reduced, so as to match the desired truck utilisation. According to this method, the number of full truck loads needed is initially computed, based on the total inventory position at each review period. Second, the available transportation capacity is then allocated among all products, depending on their demand patterns and cost characteristics. This allocation process is based on the stochastic characteristics of demand, in order to simulate real life situations. The author (Kiesmuller, 2009) conducted a comparative analysis of this proposed method against an uncoordinated periodic replenishment policy (Full Service Policy), in a detailed numerical study, finding that this method outperforms the full service policy when the average time between two successful replenishments is not 'too large' and fixed ordering costs are high. Indeed, the proposed policy is close to optimality in these conditions and reveals large cost savings.

Meanwhile, a further study (Qu, et al., 1999) presents an integrated inventory-transportation system with a modified periodic review inventory policy. The

authors proposed a heuristic decomposition method to minimise the long run total average cost. The decomposition algorithm works by using separate calculations for inventory and routing decisions. Others research (Eijs, 1994) developed a heuristic to decide whether an initial order should be enlarged or not. The heuristic is based on a comparison of the expected saved shipping cost, the expected saved ordering cost and the expected extra holding cost, that would be induced by such an enlargement. Another study (Hoque, 2006) provides an optimal solution technique for the joint replenishment problem, with storage and transport capacities and budget constraints, while others (Moon & Cha, 2006) modified the existing RAND algorithm so as to be applicable to the JRP with resource restriction, developing a genetic algorithm for the JRP with resource restriction in the process.

2.3 Summary

Although many of authors have presented different solutions to the JRP, most of these models are based on indirect grouping strategies and the optimal solution from the products perspective of the problem. In addition, most of these models are complex to implement and requires iterative procedures. In contrast to most of the previously mentioned models, in this project, a new simple solution is proposed based on a novel angle that have so far been unexplored in past efforts, the optimal solution from the trucks perspective. Therefore, the purpose of this project is to solve the joint replenishment problem (JRP) with stochastic characteristics of demand, quantity discounts and a fixed cost charged with each dispatched truck, considering both angles: the optimal solution from the products perspective and the optimal solution from the shipping trucks perspective. In developing this model, similar to the previous efforts, the ultimate aim is to minimise the overall cost in multi-product inventory control environments.

CHAPTER 3. Methodology

This chapter aims to articulate in detail the problem under examination in this project and, consequently, the methodologies adopted to achieve the declared aim of this project, namely to develop and evaluate a heuristic solution to a specific type of JPR. The solution is evaluated by performing eight experiments, in which each is compared against an adjusted EOQ model, so as to evaluate its performance.

In Section 3.1, the aims and objectives of this project are presented. Before the problem is articulated in Section 3.3, Section 3.2 provides a real-life example to illustrate its real life manifestations. Section 3.4 explores the solution to the problem, while Section 3.5 describes the methods used to generate the relevant experimental data. Section 3.6 provides an explanation of the proposed joint replenishment grouping heuristic, and Section 3.7 discusses the methods used to perform the simulation. Finally, Section 3.8 discusses the EOQ model simulation.

3.1 Aims and Objectives

The aim of this dissertation is to develop a new heuristic model to solve the joint replenishment problem (JRP) with stochastic characteristics of demand, quantity discounts and a fixed cost charged with each dispatched truck. In light of this, the overall objective is to minimise the total inventory control cost for joint replenishment environments, in which a fixed cost is charged for each truck used to replenishment the different items. More specifically, the main objectives of the project are:

- To develop a heuristic to form the different items into efficient groups and apply this model using the LINGO optimisation software.

- To develop a replenishment policy where order sizes must create full truck loads to exploit quantity discounts.

- To generate experimental data based on a real life situation, to test the performance of the grouping heuristic and the joint replenishment policy.

- To perform joint replenishment simulation using LINGO on the generated data, so as to evaluate the efficiency of the proposed solution.

- To perform an adjusted EOQ simulation using Microsoft Excel.

- To compare both simulation results using the following set of performance measures: Total inventory cost, Average Inventory Level and the Service Level.

3.2 JRP: A Real-life Example

The problem under study is based on a real world multi-item inventory control problem taking place in Palmer and Harvey (P&H), the UK's number one delivered wholesaler (P&H, 2014). P&H is a major wholesaler for Coca-Cola Enterprise (CCE), as well as a marketer, producer and distributor of Coca-Cola products. P&H stores the products supplied by CCE in three UK warehouses located at Medway, Coventry and Fareham. These products are then distributed to different customers. CCE supplies P&H with a range of products categorised by their packaging size and families. The packaging size ranges from 330ml cans to 2L bottles and the products are classified into three different families: Coke TM family, Core 3 family and Fanta family. Inventory holding costs and purchase costs for the range of available products differ according to packaging size.

CCE delivers its products to P&H in pallets using trucks. Each truck is loaded with up to three containers. CCE charges P&H a fixed cost per order (per truck shipped) which is independent of the number of pallets included within each shipment. The total fixed cost, however, depends on the capacity and number of trucks used. The smallest truck in the CCE fleet carries up to 26 pallets, where each pallet must contain 60 cases of either bottles or cans of soft drink, thus being able to carry up to 1560 (i.e. 60x26) cases. In an attempt to encourage P&H to place larger orders, CCE offers two quantity discount schemes. The first, BULK LEVEL 1, is offered whenever P&H orders a full truckload of soft drinks with

the same packaging size and from the same products family. The second, BULK LEVEL 2, is offered whenever P&H orders a full truckload irrespective of the families or sizes of the products contained. The BULK LEVEL 1 scheme offers an all units discount of 10p per case (i.e. £156 per pallet), while BULK LEVEL 2 offers an all unit discount of 5p per case (i.e. £78 per pallet). Other truck capacities include 52 pallets (two containers) and 78 pallets (three containers).

Despite the complexity of their system, and the fact that the inventory procurement between P&H warehouses and CCE is a typical example of the Joint Replenishment Problem (presented in Section 2), P&H continue to employ conventional inventory control policies that are not specifically designed for their multi-product environment. As such, ignoring specific inventory control needs may lead to losing the opportunity to reduce the overall replenishment costs and, consequently, to improve the overall system efficiency. Therefore, P&H and many other companies facing the same predicament may benefit from implementing an adapted and specifically designed joint replenishment policy that takes into account key factors such as the stochastic nature of demand, the nature of products, discount opportunities, inventory holding costs and transportation costs. In other words, a policy that aims at finding the best replenishment solution so as to minimise the overall costs under such conditions.

3.3 Statement of the Problem and Problem Assumptions

In this project, we consider an inventory control environment in which a wholesaler buys multiple products from a single supplier. To simplify the problem discussed in the previous section, several assumptions have been made.

Contrary to the case of P&H and CCE, it is assumed in the present project that the supplier delivers different items to a single warehouse rather than to multiple warehouses. Moreover, the demand is assumed to be stochastic and normally distributed, and the lead time to be constant. Different items are assumed to have the same purchase cost, and the same inventory carrying cost. In addition, we assume that the system is under a constraint of satisfying a

desired fill rate, the fraction of demand met from the stock (Type 2 Service described in the literature review chapter), in which unsatisfied demand represents lost sales.

Similarly to the real life procurement process between P&H and CCE, the products in this project are assumed to be delivered to the wholesaler using trucks of finite capacity. Different truck types are available and each truck type has a specific fixed cost, whenever used in replenishment, as well as a specific loading capacity. It is also assumed that there is no limit on the number of trucks available for use, and that there are always enough inventories at the supply side of the chain to fulfil any ordered quantity. Finally, the supplier offers an all-unit quantity discount whenever a full truckload is replenished.

The objective of the wholesaler is two-fold: On the one hand, they aim to minimise their shipping costs, particularly for fast moving products (i.e. highly demanded items). Therefore, if the fixed ordering cost is high and the demand is high, they will aim at high truck utilisation. Ideally, they will aim to have full truckload shipments, not only to minimise the total number of trucks needed but also to exploit the quantity discount schemes on offer.

On the other hand, due to the stochastic nature of demand, the wholesaler will aim to maximise the flexibility of shipments, so as to minimise the total lost sales if the demand is higher than expected, and to minimise the total inventory holding costs, if the demand is lower than expected. In other words, the wholesaler will seek small and frequent batches, so as to adapt rapidly to the market.

Clearly, these two aims are contradictory, since a full truckload policy (as favoured by the supplier) will reduce flexibility in dealing with stochastic demand, and may increase the inventory holding costs; while replenishing the inventory in small and frequent batches (as favoured by the wholesaler) will not only increase the total fixed cost of ordering and the transportation costs, but may also result in losing the advantages of the quantity discount schemes.

Despite this contradiction in goals, however, the ultimate challenge is clearly to find a replenishment policy that minimises the long-run average total cost per period, which consists of the fixed ordering costs and inventory and purchase costs, and is calculated as follows:

$$\text{Average Total Cost} = \text{average holding cost} + \text{average ordering cost} + \text{avrage purchase cost} =$$

$$G(T) = \frac{h\lambda}{2} T + \frac{K}{T} + \lambda c \quad (3.1)$$

For the purpose of this project, the following notation is used to formulate the Joint Replenishment Problem and solution:

n	Number of products
i	Product index number
m	Number of trucks used in a replenishment
j	Replenishment index variable (or day index)
T	Replenishment time interval
Q	Order quantity
λ	The average demand (pallets per day)
$*$	Optimality notation
K	Ordering cost
h	Holding cost per unit per day
c	Purchase cost per unit
R	Re-order point
S	Order-up-to level

3.4 Solution Overview

Assuming that the fixed cost per order is high compared to the inventory holding cost, and assuming that a significant quantity discount is achieved whenever a full truckload is shipped, then a reasonable solution is to aim for high truck utilisation. Therefore, in the proposed solution, trucks are dispatched with full loads. Moreover, in solving the JRP under consideration, a periodic review order-up-to level (T, S) system is used. In general, this policy is described as follows.

At each review instant, T, the inventory position - which is the stock on hand plus any outstanding orders - of each item i is reviewed, and an order placed so that the inventory position is raised to the order-up-to level, S_i, of that item.

As per the Literature Review chapter (notably Sections 2.1.2.3 and 2.2.3.2), this policy outperforms the can order (S, c, s) policy in most cases. Moreover, this policy is practical and simple to implement due to the fewer number of policy parameters needed. Additionally, because of the periodic review property, this system is much more preferable in terms of coordinating the replenishments of related items. The coordination afforded by the periodic review system can provide significant savings on the shipping costs. In addition, the (T, S) system offers a regular opportunity to adjust the order-up-to level, a desirable property when demand is stochastic. However, the main disadvantage of the (T, S) system is that the inventory holding costs are, typically, higher than those in continuous review systems.

Typically, (T, S) systems assume no restrictions on the size of the ordered quantities. Therefore, for the purpose of this project, this policy is slightly adjusted in order to take into account the capacity restrictions posed by the full truckload constraint. This is done by allowing the order quantity of each item to deviate from S_i at each review period. Whether the deviation from S_i is an over achievement or an under-achievement depends on the inventory position and the demand rates of different products. The idea is therefore to allow for the coordinated replenishment of different items while maximising truck utilisation by forming efficient groups of products. Each group of products will be replenished in one full truck at equal replenishment instances. This is done by allocating the available truck capacity to different products within one group while aiming to raise their inventory position to different S_is, though allowing for deviation.

The adjusted (T, S) replenishment policy can thus be described as follows: At each review instant, T, the inventory position of each item, i, within group j is reviewed, and an order is placed such that its inventory position is raised to $S_i + d_i^- - d_i^+$ whereby d_i^- and d_i^+ are the over achievement and underachievement variables. As such, the solution policy can be seen from two perspectives: From

the truck perspective, the policy used is a (T,Q) policy in which an order quantity equivalent to the truck capacity is ordered every T. From the perspective of the different products, the policy used is the adjusted (T,S) policy. Both perspectives are illustrated in Figures 8 and 9 below.

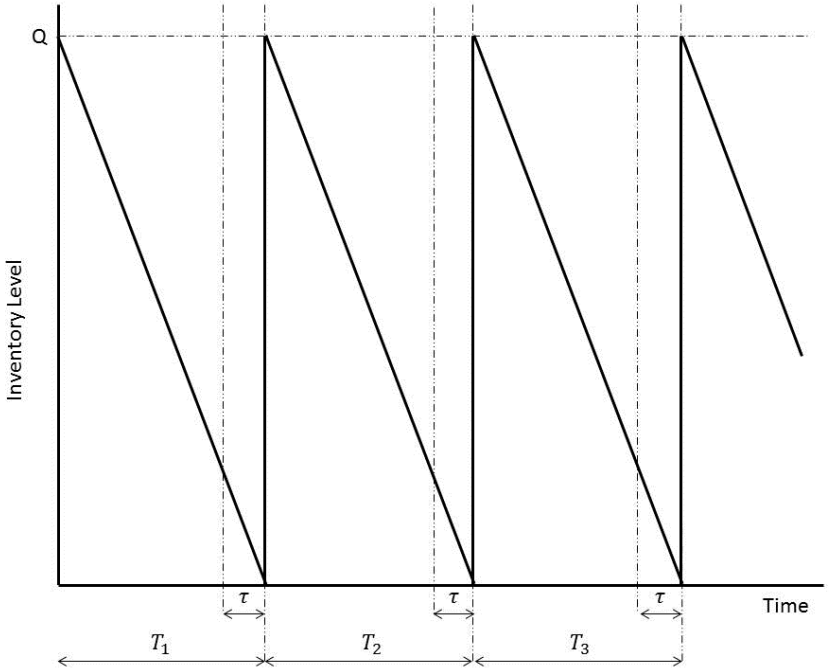


Figure 8: The (T,Q) policy inventory usage curve

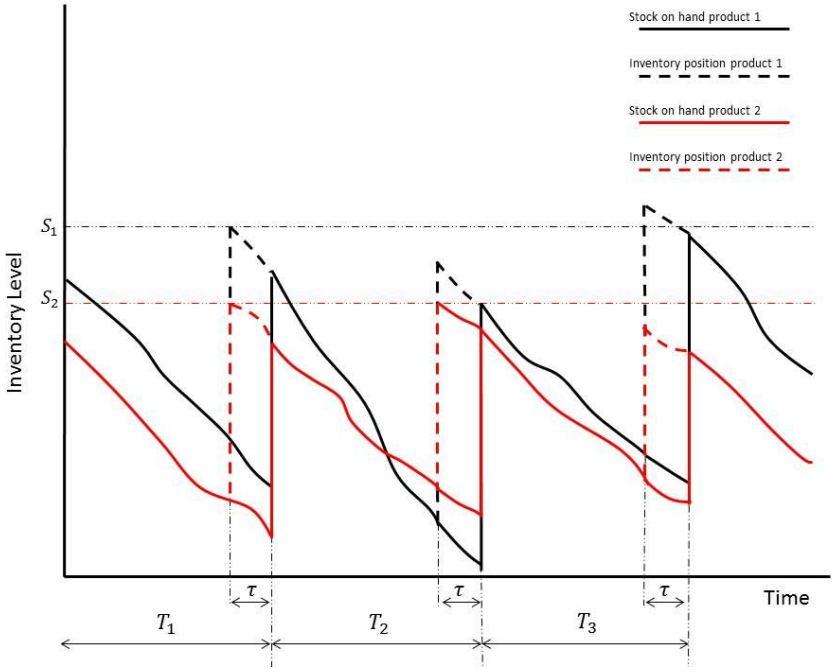


Figure 9: The adjusted (T, S) policy inventory usage curve for two items

3.5 Experimental Data

In this section, the methodology used to generate the set of experimental data is explained. The problem structure is defined by a set of boundaries and limitations to the relevant input data. These limitations are based on P&H's procurement processes with CCE, and include the number of products to consider, the range of fixed ordering costs per truck, the range of inventory holding costs per item per day, the purchase costs per item, the range of truck capacities and the demand nature and boundaries for different items. Basing the project's parameters on those found in P&H is intended to bolster the realistic aspect of the experimental process, so as to help reasonably judge the solution's performance. Second, based on the adopted structure and boundaries, daily demand data is generated so as to be used in the inventory replenishment simulation. In generating the required demand data, two scenarios are represented: A scenario where the average demand for different products is similar (Type 1 demand); and a scenario where the demand is low for most of the products and high for others (Type 2 demand). In both scenarios, we assume a stationary stochastic demand pattern. Finally, the decision regarding which set of experiments is to be carried out is explained.

3.5.1 Problem Structure and Boundaries

Relevant data from P&H's procurement process with CCE have been collected from existing literature on the topic (Shumnij, 2010 & Keerthana, 2013) in order to structure the parameters of the problem under examination in this project. It was found that the average total demand in pallets per day in P&H's case is 20. This total demand rate is adopted in our experiments. Further, it was found that the total number of items replenished at P&H is around 30 products. For this dissertation, we assume 20 products for all the experiments. This reduced number of products is for computational reasons. The more products used, the much longer it takes to run the grouping and simulation experiment using Lingo (Section 3.6). In addition, using 20 products matches the demand rate of 20 products, and hence simplifies the process of generating daily demand for the simulation as the average demand per item is 1. Conversely, using less than 20

products will not sufficiently highlight the importance of the proposed grouping heuristic.

Moreover, it was found that the maximum average demand among all products is 8.11 pallets per day, and the minimum average demand among all products is 0.07 pallets/ day. In our experiment, therefore, an upper limit of 8 pallets/day and a lower limit of 0.1 pallets/day are used to generate random average demand data for the 20 items. In addition, it was observed that the standard deviation of demand during a 22-week period for the different 31 products ranges from 0.01 to 2 pallets a day, while the average standard deviation among all 31 products is 0.2 pallets a day. This reflects a stochastic yet stationary nature of the demand and a stationary stochastic demand is thus assumed for the current problem. Furthermore, in order to avoid extreme deviations from the mean, demand for each product at any day is assumed to follow a normal distribution, and a value of six Sigma is used to generate daily demand data for the 20 products in the simulation experiments.

Furthermore, it was observed that the trucks used for shipping products from CCE to P&H have a capacity of 26, 52 or 78 pallets per truck. In our experiment, however, we consider capacities ranging from 5 pallets per truck to 70 pallets per truck. This is particularly to highlight the importance of the proposed grouping heuristic, as well as to examine the impact of altering the fixed costs of ordering on the grouping process. Indeed, the importance of the grouping heuristic can be further highlighted by alternating the total demand per day figures. However, this is beyond the scope of this project as altering the fixed costs and the capacity of the different trucks is sufficient. The fixed cost per order incurred by CCE is estimated at £50 per order in previous research studies. In this project, we consider a wider range of ordering costs, so as to examine the behaviour of the products grouping heuristic with high and low ordering costs. A range between £5 per order to £100 per order is, therefore, adopted for the purpose of this project. Table 1 summarises the collected data from the example case, while Table 2 summarises the data used for this project.

Table 1: Data from P&H case study

Parameter	Value
Number of Products	31 Products
Average Total Demand	20 Pallets/Day
Maximum Average Demand (Pallets/Day)	8.11 Pallets/Day
Minimum Average Demand (Pallets/Day)	0.07 Pallets/Day
Maximum Standard Deviation of Demand	2
Minimum Standard Deviation of Demand	0.01
Truck Capacity (Pallets)	26 - 78 Pallets
Approximate Fixed Cost per Truck Ordered (Ordering Cost)	£50 /Per Order
Approximate Purchase Cost per Pallet of Soft Drinks (£/Pallet)	£1000/Pallet
Annual Inventory Carrying Charge	20%
Approximate Holding Cost per Pallet per Day (£/Pallet/day)	£0.55 /Pallet

Table 2: Data adopted for this project

Parameter	Value
Number of Products	20 Products
Average Total Demand	20 Pallets/Day
Maximum Average Demand (Pallets/Day)	8 Pallets/Day
Minimum Average Demand (Pallets/Day)	0.1 Pallets/Day
Standard Deviation of Demand	6 Sigma
Truck Capacity	2 - 70 Pallets
Approximate Fixed Cost per Truck Ordered (Ordering Cost)	£5 - £100/Per Order
Approximate Purchase Cost per Pallet of Soft Drinks (£/Pallet)	£1000/Pallet (£800 if discounted)
Annual Inventory Carrying Charge	20%
Approximate Holding Cost per Pallet per Day (£/Pallet/day)	£0.55 /Pallet

3.5.2 Simulation Demand Data

In order to assess the performance of the proposed solution, demand data for one thousand days is randomly generated using statistical methods. The data analysis Add-In in Microsoft Excel is used to model two demand instances. In the first instance, we model similar demand values for each product (Type 1 demand data), whereas in the other we model an instance where the demand rate is very low for most of the products but very high for a small selection (Type 2 demand data). As such, we are able to examine the efficiency and performance of the proposed solution under different conditions.

In order to model similar daily demand data values for the 20 products over a thousand-day period (Type 1 demand), the following steps are taken:

- Step 1: Identify the maximum and minimum values of average demand per product. In our case, the desired minimum is 0.1 pallets/day and the maximum is 8 pallets/day.

- Step2: Decide on the desired mean demand. In our case, since a total demand of 20 pallets/ day is desired and since the number of products is 20, then we expect an average demand of 1 pallets/ day for each product.

- Step 3: Use the random number generation tool in the data analysis tool pack Add-In to generate random average demand values for the 20 products, using a normal distribution with a mean of 1 and a standard deviation of 1.5. A standard deviation of 1.5 is used to increase the probability of having a demand rate close to 1 pallet/day.

- Step 4: Since a normal distribution function is used, there is a possibility of having negative demand rates. Therefore, the ABS function in Excel is used to remove negative values.

- Step 5: Due to the use of the ABS function, and since the generated numbers are random, the sum of average demand rates for the 20 products is not necessarily equal to 20. Therefore, we normalise the

values by multiplying the randomly generated values in the previous step by 20, then dividing them by their sum.

- Step 6: Decide on the desired demand standard deviation values for the 20 products to be used for modelling the thousand-day demand for each product. In our example, we use 6σ to define the standard deviation. This will allow for stochastic, yet stationary, demand rates. The generated average demand data and the standard deviation for each product are shown in Table 3.

- Step 7: Use the random and normal inverse functions in MS Excel to generate demand for the next 1000 days using the average daily demand and standard deviation for each product (as provided in Table 3 below). Figure 10 exemplifies the Excel sheet used to generate simulation Type 1 demand data.

To model an instance where the demand rate is very low for most of the products but very high for the remaining selection (Type 2 demand), the following steps are followed:

- Step 1: Identify the maximum and minimum values of average demand per product. In our case, the desired minimum is 0.1 Pallet/Day and the maximum is 8 Pallets/Day.

- Step 2: In order to generate a model where the demand rate is very low for most of the products but very high for a small selection, a discrete distribution function is used. Again, the random number generation tool in the data analysis tool pack Add-In is used. To compute our discrete function, the desired value of average demand is first decided. As described earlier, our mean demand is 1 Pallet/Day. Second, we decide on the range of values that we anticipate on the discrete function. For low demand products, a range of 0.1 to 0.5 Pallets/Day was used, while for high demand ones a range of 5 to 8 Pallets/Day was used.

- Step 3: The last input for the random generation tool is the probability of each range. Knowing that the total probability is 1, and that the desired average is 1, a probability, P(X), can be allocated to each range. The MS Excel tool is used and the average demand values for the 20 products are computed, as shown in Table 4.

- Step 4: Decide on the desired demand standard deviation values for the 20 products to be used for modelling the one thousand day demand for each product. In our example, we use 6σ to define the standard deviation, thus allowing for stochastic, yet stationary, demand rates.

- Step 7: Use the random and normal inverse functions in MS Excel to generate demand for the next 1000 days using Table 4.

Table 3: Type 1 demand data and standard deviation for 20 products

Product ID	Average Demand per Day (Pallets/ Day)	Standard Deviation (Pallets)
1	1.21	0.20
2	1.78	0.30
3	0.76	0.13
4	0.70	0.12
5	1.36	0.23
6	0.06	0.01
7	0.85	0.14
8	1.46	0.24
9	0.24	0.04
10	1.03	0.17
11	1.57	0.26
12	0.66	0.11
13	0.02	0.00
14	0.77	0.13
15	2.71	0.45
16	0.56	0.09
17	1.25	0.21
18	0.40	0.07
19	0.78	0.13
20	1.84	0.31

Table 4: Type 2 demand data and standard deviation for 20 products

Product ID	Average Demand per Day (Pallets/ Day)	Standard Deviation (Pallets)
1	0.28	0.05
2	0.09	0.02
3	0.38	0.06
4	5.09	0.85
5	4.81	0.80
6	6.60	1.10
7	0.09	0.02
8	0.28	0.05
9	0.47	0.08
10	0.09	0.02
11	0.19	0.03
12	0.09	0.02
13	0.09	0.02
14	0.09	0.02
15	0.19	0.03
16	0.09	0.02
17	0.19	0.03
18	0.19	0.03
19	0.38	0.06
20	0.28	0.05

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Average Daily Demand Data																				
2		Product																			
3	Product No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
4	Average demand	1.210431	1.78133	0.757515	0.697857	1.36124	0.055264	0.848593	1.457873	0.240776	1.028612	1.56872	0.656513	0.020149	0.774823	2.71098	0.562826	1.247419	0.395753	0.784255	1.839069
5	S.D.	0.201739	0.296888	0.126253	0.11631	0.226873	0.009211	0.141432	0.242979	0.040129	0.171435	0.261453	0.109419	0.003358	0.129137	0.45183	0.093804	0.207903	0.065959	0.130709	0.306512
6																					
7		Demand Data for 1000 Days																			
8		Product																			
9	Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
10	1	1.230693	2.265171	0.57407	0.80016	1.075534	0.049006	0.834443	1.138249	0.327459	0.996311	1.7234	0.492336	0.018641	0.482581	3.284334	0.691711	0.98801	0.311927	0.652433	2.325264
11	2	0.984586	1.223361	0.769279	0.668005	1.16078	0.040228	0.781148	1.509958	0.205576	1.068527	1.789013	0.70727	0.020364	0.656398	2.98393	0.532108	0.983632	0.41875	0.847027	2.056326
12	3	1.384154	1.653962	0.812756	0.812445	1.171903	0.054824	0.784245	1.421174	0.213622	1.18878	1.564616	0.656147	0.016788	0.69809	2.26093	0.558694	0.884548	0.365435	0.729573	1.559357
13	4	1.063299	2.099832	0.685753	0.665615	1.018602	0.048283	0.817308	1.345623	0.335373	0.919467	1.085008	0.672336	0.016951	0.940893	2.877204	0.420903	1.499664	0.418995	0.815186	1.252846
14	5	1.09828	2.186584	0.706674	0.615598	1.20805	0.047637	0.866657	1.299263	0.175568	1.295786	1.80318	0.513389	0.022696	0.843658	2.611589	0.24947	0.985238	0.426792	0.853832	1.897329
15	6	1.29825	1.488211	0.48686	0.819479	1.477083	0.046608	0.734309	1.534401	0.253399	0.982516	1.230785	0.429693	0.013563	0.806612	2.611504	0.567772	1.339281	0.434612	0.655915	1.875808
16	7	0.892482	2.064431	0.730189	1.023736	1.073407	0.050424	0.935108	1.547374	0.240654	1.095329	1.414314	0.646696	0.022213	0.914178	3.456819	0.490323	0.921214	0.329922	0.845573	2.801018
17	8	1.053867	1.670942	0.689752	0.79647	1.707324	0.065009	0.905511	1.043898	0.228124	0.957827	1.29018	0.641743	0.019056	0.799397	3.259931	0.66253	1.069117	0.422042	0.880244	1.938527
18	9	0.817384	1.736132	0.741267	0.666493	1.299941	0.054016	0.699033	1.707223	0.229772	0.85533	1.554218	0.67455	0.0212	0.843367	3.180017	0.6504	1.163456	0.388534	0.65153	1.492405
19	10	1.390578	1.806309	0.804952	0.399709	1.427902	0.055729	1.006217	1.432395	0.310702	0.810634	1.475914	0.695201	0.019498	0.659973	2.428178	0.594105	1.385364	0.398006	0.678551	2.077538
20	11	1.266294	1.50605	0.665863	0.747302	1.265698	0.057089	0.903586	1.394787	0.21248	1.063844	1.41409	0.675252	0.018202	0.835062	1.989484	0.698005	1.558227	0.411041	0.606261	1.303111
21	12	1.430514	1.535665	0.753213	0.598167	1.059291	0.057202	0.725522	1.535492	0.258576	1.075973	1.12263	0.692043	0.024024	0.563425	2.571692	0.549195	1.168403	0.409668	0.653132	2.141423
22	13	1.091276	1.539569	0.632491	0.919462	1.132335	0.046375	0.618177	1.468362	0.306316	1.142543	1.345363	0.657726	0.023076	0.998107	2.316537	0.362941	0.859583	0.511876	0.833942	2.480392
23	14	1.158409	1.462531	0.724716	0.608588	1.474895	0.050273	0.704876	0.879483	0.265862	0.803941	1.807084	0.613247	0.019005	0.791601	2.872813	0.341127	1.114072	0.382037	0.647913	2.10213
24	15	1.320408	1.999008	0.599633	0.682893	1.74535	0.059887	0.942608	1.366139	0.279091	1.112118	1.446841	0.465941	0.023583	0.722893	3.063209	0.797378	1.265759	0.472565	0.606888	1.969108
25	16	1.61125	1.737147	0.765094	0.752557	1.155671	0.048828	0.856485	0.975979	0.294677	1.139835	1.695069	0.505718	0.028744	0.587047	2.787056	0.507909	1.23311	0.411236	0.879842	2.310454
26	17	0.971087	2.165543	0.978785	0.637296	1.253419	0.052694	1.161212	1.707778	0.236996	0.795698	2.055512	0.889618	0.023867	0.935021	2.490698	0.509095	1.37316	0.41998	0.852474	1.214862
27	18	1.609833	2.355153	0.79596	0.679704	1.431963	0.053694	0.839908	1.383794	0.233845	1.060027	1.951879	0.693344	0.024543	0.696986	2.563015	0.643343	0.953254	0.404222	0.934586	1.574024
28	19	1.065941	2.020357	0.779708	0.819306	1.689527	0.057609	0.929597	1.708775	0.156363	0.770068	0.986822	0.556128	0.019913	0.723828	4.01587	0.542351	1.207576	0.381168	0.519493	1.506729
29	20	0.98924	1.829254	0.751684	0.890609	1.147361	0.06852	0.690225	1.332563	0.263566	0.644182	1.698941	0.771214	0.018686	0.963687	2.748361	0.626535	1.428182	0.247641	0.819566	1.540886
30	21	0.949158	1.409501	0.82734	0.689581	1.647231	0.049256	0.820609	1.743721	0.308653	1.049776	1.400859	0.533685	0.022035	0.879019	3.230546	0.427357	1.243991	0.437387	0.97026	2.136256
31	22	1.1108	1.474309	0.888247	0.670038	1.787917	0.053522	0.724254	1.188217	0.24382	1.06455	1.461938	0.615117	0.01503	0.854126	3.196937	0.647463	1.225105	0.409623	0.964671	2.019448

Figure 10: Simulation data for 1000 days based on Type 1 demand

3.5.3 Designing the Experiments

Having presented the structure, limitations and demand data for the problem under consideration, the next step is to design a set of experiments to test the model through simulation. Before choosing the various experimental parameters, the following assumptions have been made:

1. In any experiment, three truck capacities must be available: small capacity, medium capacity and large capacity.

2. The capacity of any truck is an integer multiple of that of the smallest truck. This integer is 2 for the medium capacity truck and 3 for the largest truck. This assumption has been made since in real-life situations truck sizes are defined by the number of containers attached to them.

3. The ordering cost does not vary linearly with the truck capacity.

4. Although the fixed cost per order, K, increases with bigger trucks, the ordering cost per item decreases with bigger trucks.

5. If the K_1 is the fixed cost per order in a small truck then this cost is $1.5K_1$ for medium trucks and $1.8K_1$ for the truck with the largest capacity. This assumption has been made to represent real life circumstances, in that it is usually cheaper to send one large truck with a particular capacity than sending two small trucks with the same overall capacity.

To design the experiments the following steps were carried out:

1. Decide on the smallest truck size for the first set of trucks. For the first set of trucks, the minimum capacity in Table 2 is used for Experiment 1.

2. Compute the other two sizes for Set 1. This is done using assumption 2 above.

3. Decide on the biggest truck size for the second set of trucks. Looking at Table 2, one can realise that the biggest possible truck capacity is 70 pallets per replenishment. For the purpose of the analysis, truck sizes between 30 and 70 were eliminated since with the given range of fixed costs, such sizes will always yield a single group solution and, therefore, will not reflect the capability or the efficiency of the proposed grouping heuristic. Accordingly, for the second set the largest truck was allocated a capacity of 30 pallets.

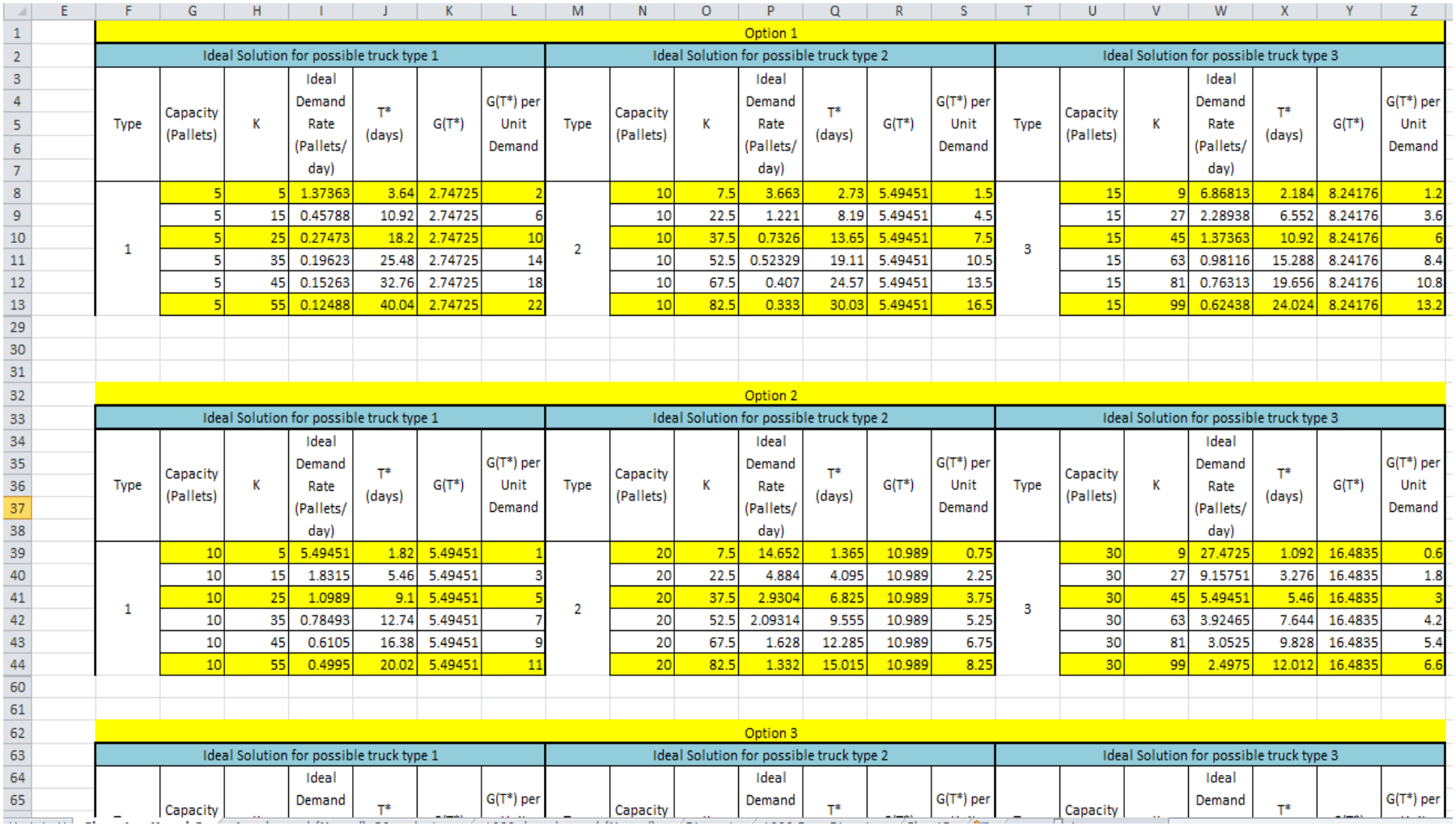
4. Compute the other two sizes for Set 2. This is done using assumption 2 above.

5. Develop an instance in which K is low for both sets of trucks. Knowing that the smallest possible K value is 5, we use this value for the smallest trucks in both sets. Compute K values for the other truck in both sets using assumption 4 above.

6. Develop an instance in which K is high for both sets of trucks. Knowing that the largest possible K value is 100, we use this value for the largest trucks in both sets. Compute K values for the other trucks in both sets using assumption 4 above. (Note that 99 was used instead of 100 so as to avoid decimal places when computing K values for the other trucks.)

7. Perform an initial analysis to see whether these options will help in evaluating the performance. For this purpose, a spreadsheet was developed to compute the ideal demand rate for each truck and, consequently, to estimate the possible answers when the model is run. The formulas used in this spreadsheet are explained in Section 3.7.

8. Design the required experiments. Tables 5 to 9 show the main input parameters for the chosen eight experiments.



Option 1

Ideal Solution for possible truck type 1							Ideal Solution for possible truck type 2							Ideal Solution for possible truck type 3						
Type	Capacity (Pallets)	K	Ideal Demand Rate (Pallets/ day)	T* (days)	G(T*)	G(T*) per Unit Demand	Type	Capacity (Pallets)	K	Ideal Demand Rate (Pallets/ day)	T* (days)	G(T*)	G(T*) per Unit Demand	Type	Capacity (Pallets)	K	Ideal Demand Rate (Pallets/ day)	T* (days)	G(T*)	G(T*) per Unit Demand
1	5	5	1.37363	3.64	2.74725	2	2	10	7.5	3.663	2.73	5.49451	1.5	3	15	9	6.86813	2.184	8.24176	1.2
	5	15	0.45788	10.92	2.74725	6		10	22.5	1.221	8.19	5.49451	4.5		15	27	2.28938	6.552	8.24176	3.6
	5	25	0.27473	18.2	2.74725	10		10	37.5	0.7326	13.65	5.49451	7.5		15	45	1.37363	10.92	8.24176	6
	5	35	0.19623	25.48	2.74725	14		10	52.5	0.52329	19.11	5.49451	10.5		15	63	0.98116	15.288	8.24176	8.4
	5	45	0.15263	32.76	2.74725	18		10	67.5	0.407	24.57	5.49451	13.5		15	81	0.76313	19.656	8.24176	10.8
	5	55	0.12488	40.04	2.74725	22		10	82.5	0.333	30.03	5.49451	16.5		15	99	0.62438	24.024	8.24176	13.2

Option 2

Ideal Solution for possible truck type 1							Ideal Solution for possible truck type 2							Ideal Solution for possible truck type 3						
Type	Capacity (Pallets)	K	Ideal Demand Rate (Pallets/ day)	T* (days)	G(T*)	G(T*) per Unit Demand	Type	Capacity (Pallets)	K	Ideal Demand Rate (Pallets/ day)	T* (days)	G(T*)	G(T*) per Unit Demand	Type	Capacity (Pallets)	K	Ideal Demand Rate (Pallets/ day)	T* (days)	G(T*)	G(T*) per Unit Demand
1	10	5	5.49451	1.82	5.49451	1	2	20	7.5	14.652	1.365	10.989	0.75	3	30	9	27.4725	1.092	16.4835	0.6
	10	15	1.8315	5.46	5.49451	3		20	22.5	4.884	4.095	10.989	2.25		30	27	9.15751	3.276	16.4835	1.8
	10	25	1.0989	9.1	5.49451	5		20	37.5	2.9304	6.825	10.989	3.75		30	45	5.49451	5.46	16.4835	3
	10	35	0.78493	12.74	5.49451	7		20	52.5	2.09314	9.555	10.989	5.25		30	63	3.92465	7.644	16.4835	4.2
	10	45	0.6105	16.38	5.49451	9		20	67.5	1.628	12.285	10.989	6.75		30	81	3.0525	9.828	16.4835	5.4
	10	55	0.4995	20.02	5.49451	11		20	82.5	1.332	15.015	10.989	8.25		30	99	2.4975	12.012	16.4835	6.6

Option 3

Ideal Solution for possible truck type 1							Ideal Solution for possible truck type 2							Ideal Solution for possible truck type 3						
	Capacity		Ideal Demand	T*		G(T*) per		Capacity		Ideal Demand	T*		G(T*) per		Capacity		Ideal Demand	T*		G(T*) per

Figure 11: Initial analysis for the design of the experiments

Table 5: Fixed parameters for all experiments

Parameter	Value
Number of Products	20 Products
Average Total Demand	20 Pallets/Day
Purchase Cost per Pallet (£/Pallet)	£1000 /Pallet
Discounted Purchase Cost per Pallet (£/Pallet)	£800/Pallet
Annual Inventory Carrying Charge	20%
Approximate Holding Cost per Pallet per Day (£/Pallet/Day)	£0.55 /Pallet

Table 6: Experiment 1 (Type 1 Demand – Small Trucks – Low Ordering Cost)

Truck Type	Capacity (Pallets)	K (£/ Order)	Demand Type
1	5	5	1
2	10	7.5	
3	15	9	

Table 7: Experiment 2 (Type 1 Demand – Small Truck – High Ordering Cost)

Truck Type	Capacity (Pallets)	K (£/ Order)	Demand Type
1	5	55	1
2	10	82.5	
3	15	99	

Table 8: Experiment 3 (Type 1 Demand – Large Trucks – Low Ordering Cost)

Truck Type	Capacity (Pallets)	K (£/ Order)	Demand Type
1	10	5	1
2	20	7.5	
3	30	9	

Table 9: Experiment 4 (Type 1 Demand – Large Trucks – High Ordering Cost)

Truck Type	Capacity (Pallets)	K (£/ Order)	Demand Type
1	10	55	1
2	20	82.5	
3	30	99	

Table 10: Experiment 5 (Type 2 Demand – Small Trucks – Low Ordering Cost)

Truck Type	Capacity (Pallets)	K (£/ Order)	Demand Type
1	5	5	2
2	10	7.5	
3	15	9	

Table 11: Experiment 6 (Type 2 Demand – Small Truck – High Ordering Cost)

Truck Type	Capacity (Pallets)	K (£/ Order)	Demand Type
1	5	55	2
2	10	82.5	
3	15	99	

Table 12: Experiment 7 (Type 2 Demand – Large Trucks – Low Ordering Cost)

Truck Type	Capacity (Pallets)	K (£/ Order)	Demand Type
1	10	5	2
2	20	7.5	
3	30	9	

Table 13: Experiment 8 (Type 1 Demand – Large Trucks – High Ordering Cost)

Truck Type	Capacity (Pallets)	K (£/ Order)	Demand Type
1	10	55	2
2	20	82.5	
3	30	99	

3.6 The Grouping Heuristic

To successfully implement the joint replenishment policy described in Section 3.4, the first step is to place the different items into efficient groups. Each group is thus replenished in a single truck at every replenishment period. As indicated earlier, in order to achieve quantity discounts and to benefit from economies of scale, the proposed solution requires full truckload replenishments. Therefore, at each review period, the aggregate order quantities of all products in the same group must be equivalent to a full truckload. Bearing this in mind, in addition to the multi-item nature of the problem, and the various truck types available, an optimal solution to the grouping problem may be extremely difficult to find. This is attributable to two factors: First, due to the complex nature of the problem, and the significant number of different products to be replenished, a large number of grouping options may arise. As such, this makes the optimal solution computationally prohibitive. Second, as indicated earlier, the solution policy described in Section 3.4 can be viewed from two perspectives. From the truck perspective of the solution, the policy used is a (T,Q) policy in which an order quantity equivalent to the truck capacity is ordered every T. From the perspective of the different products, the policy used is the adjusted (T,S) policy described in Section 3.4 and, therefore, the capacity of the allocated trucks, is not necessarily equal to the order quantity such the inventory level is raised to $\sum S_i$. Due to the multiple objective nature of the problem, and knowing that each policy yields different ideal total λ, total Q and T values, it is extremely difficult to find a solution that satisfies both perspectives. For this reason, we propose a simple grouping heuristic based on the capacities of the available trucks and the real demand rates of the different products. The following three sections describe this grouping model in details.

3.6.1 Model Formulation

3.6.1.1 The optimal solution for a group of products

In Section 3.3, three types of costs were assumed. These are the average total holding cost, the average total ordering cost and the average total purchase

cost. Accordingly, for a given group of products, the total cost function (Equation 3.1) can be rewritten as:

$$G(T) = \frac{1}{2}T\sum_{i=1}^{n}\lambda_i h_i + \frac{K}{T} + \sum_{i=1}^{n}\lambda_i c_i \quad (3.2)$$

For a constant $\sum_{i=1}^{n}\lambda_i$, the annual purchase cost is no longer relevant in the decision making process and is therefore ignored. Further, since it was assumed (in Section 3.3) that all products have the same purchase and holding costs, Equation 3.2 can be rewritten as:

$$G(T) = \frac{T}{2}\sum_{i=1}^{n}\lambda_i h_i + \frac{K}{T} \quad (3.3)$$

According to Silver et al. (1998) for any group of products the value of T such that $G(T)$ is at a minimum is given by:

$$T^* = \sqrt{\frac{2K}{h\lambda_{total}}} \quad (3.4)$$

Where λ_{total} is the sum of the demand rates for a family of products included in a replenishment cycle, expressed as:

$$\lambda_{total} = \sum_{i=1}^{n}\lambda_i \quad (3.5)$$

It follows that the optimal order quantity Q^* for a family of products with a demand rate of λ_{total} can be expressed as:

$$Q_j^* = \lambda_{total} \times T^* \quad (3.6)$$

Note that Equation 3.6 computes the optimal solution from the perspective of the family of products. In other words, Q^* is calculated without taking into consideration the capacities of the different truck types available. Accordingly, grouping the different products based on this solution can lead to high average total ordering costs, especially when the fixed cost per order is high. Nonetheless, many authors continue to use this solution by finding Q^* and T^* for the entire range of products. All the items are then replenished every T^* in different trucks. The number of trucks needed is computed as follows:

$$\frac{Q^*}{Capacity\ of\ Truck}$$

In many cases, this method yields high transportation costs, since the number of trucks obtained is a rounded number, and the cost per truck shipped is not considered. For example, a truck where only 20% of its capacity is utilised in the P&H/CCE case will lead to an opportunity loss due to the lost quantity discount, and thus to a high transportation cost per item.

3.6.1.2 The truck optimal solution

The optimal solution from a truck perspective can be described by a (T, Q) policy, where Q is equal to its full truckload capacity (v). Therefore, for any replenishment j (each replenishment j is associated with one truck) to obtain the ideal solution, the first step is to set:

$$Q_j^* = Truck\ Capacity = v_j \quad (3.7)$$

Hence, for any replenishment j, substituting Equation 3.7 in Equation 3.6 and rearranging, the optimal replenishment cycle for j (i.e. the optimal replenishment cycle of truck j) is given as:

$$T_j^* = \frac{v_j}{\lambda_j^*} \quad (3.8)$$

Further, from Equation 3.4, for a full truckload T_j^* is also obtained using:

$$T_j^* = \sqrt{\frac{2K}{h\lambda_j^*}} \quad (3.9)$$

Note that λ_j^* is the ideal demand from the truck perspective, and that the value of λ_j^* is independent of the λ_{total} obtained in Equation 3.5. To find λ_j^*, Equation 3.8 is set equal to Equation 3.9, as follows:

$$\frac{v_j}{\lambda_j^*} = \sqrt{\frac{2K}{h\lambda_j^*}} \quad (3.10)$$

By rearranging Equation 3.10, the ideal demand rate from the truck perspective of the problem is obtained as follows:

$$\lambda_j^* = \frac{h(v_j)^2}{2K} \quad (3.11)$$

Clearly, λ_j^* is independent of the λ_{total} obtained in Equation 3.5, and only depends on the capacity of the truck used, v_j, the holding cost, h, and the fixed cost per order, *K*, while λ_{total} depends on the actual demand of the different products. Therefore:

$$\lambda_{\text{total}} \neq \lambda_{\text{j}}^* \quad (3.12)$$

However, using a set of optimal replenishments might not guarantee the least total cost for the overall replenishment process. This is because a grouping strategy based on this solution is typically pursued by dividing λ_{total} by λ_{j}^* in order to obtain the total number of replenishments needed, and the value of $\lambda_{\text{total}}/\lambda_{\text{j}}^*$ is rounded to the nearest integer. In some cases, this leads to excess inventory levels, while in other cases to very low service levels.

3.6.1.3 The grouping heuristic for single truck type

To avoid unnecessary replenishments, low service levels, poorly utilised trucks and quantity discount opportunity losses, a reasonable method for grouping the products would fully utilise the truck capacity while satisfying the total demand. Thus, the proposed grouping heuristic is based on a full truckload policy and the value of λ_{total}. Aiming at minimising the total average cost, the principle underlying the proposed grouping heuristic is to find the best grouping solution by allowing for deviations from the λ_{j}^{*} calculated using Equation 3.11 to match the λ_{total} for a group of products, while using an order quantity that is equal to a full truckload, v, or multiple full truckloads. Given these two restrictions, the relationship between the cost of using a single replenishment, and of using more than one replenishment of the same type to satisfy the same demand rate, is obtained as follows:

We know that for a replenishment j, using a single truck,

$$Q_j^* = v = \lambda_{total} \times T_j \quad (3.13)$$

Therefore, for the same λ_{total}, for any number of trucks, m, of the same replenishment type, j, the optimal order quantity is:

$$Q_j^* = m \times v = m \times \lambda_{total} \times T_j \quad (3.14)$$

For a single truck, the total cost is obtained by using:

$$G(T_j) = \frac{h\lambda_{total}}{2} T_j + \frac{K}{T_j} \quad (3.15)$$

Hence, the average total cost for any number of trucks, m, is:

$$G(T_j) = \frac{mh\lambda_{total}}{2} T_j + \frac{mK}{T_j} \quad (3.16)$$

Therefore, by dividing Equation 3.15 by Equation 3.16 we obtain:

$$\frac{G(T_j)}{G(T_j)_{m\geq 1}} = \frac{h\lambda_{total}}{2}T_j + \frac{K}{T_j} \Big/ \frac{mh\lambda_{total}}{2}T_j + \frac{mK}{T_j} \quad (3.17)$$

Based on Equation 3.14, Equation 3.17 can be rewritten as:

$$\frac{G(v)_{m=1}}{G(mv)_{m\geq 1}} = \frac{G(v)_{m=1}}{G(mv)_{m\geq 1}} = \left(\frac{hv}{2} + \frac{K\lambda_{total}}{v}\right) \Big/ \left(\frac{mhv}{2} + \frac{mK\lambda_{total}}{mv}\right) = \frac{1}{m} \quad (3.18)$$

Consequently, the relationship between the cost of using a single truck and the cost of using multiple trucks of the same type for the same demand rate is given as:

$$G(v)_{m=1} = \left(\frac{1}{m}\right) G(mv)_{m\geq 1} \quad (3.19)$$

Therefore, for any number of trucks ($\mathrm{m} > 1$), we obtain:

$$G(mv)_{m=1} \; is \; always < G(mv)_{m>1} \quad (3.20)$$

In theory, as indicated by Equation 3.20, for any given λ_{total}, $\mathrm{Q}^*_{\mathrm{Truck}}$, K and h values, it is always cheaper to replenish products using one truck. In real life, however, as λ_{total} increases, the value of T decreases exponentially as indicated by equation 3.14. Therefore, as $\lambda_{\mathrm{total}} \to \infty$, we also observe that $\mathrm{T} \to 0$.

In reality, however, a $\mathrm{T} \approx 0$ is neither practical nor possible. Therefore in our solution, we assume 1 day to be the minimum value of any T_j. Hence, for any replenishment j (i.e. for any truck) the maximum λ_{total} that any truck can withstand is $T_j = 1$. Accordingly, for any truck, we obtain:

$$Max\ \lambda_j = \frac{Q_j^*}{Minimum\ T_j} = \frac{Q_j^*}{1} = Q_j^* \quad (3.21)$$

Therefore, only if $\lambda_{\text{total}} > \text{Max}\,\lambda_{\text{j}}$ is a second truck used to replenish the remaining λ, and so on.

3.6.1.4 The proposed grouping heuristic for the current problem

Since it was assumed, in Section 3.3, that several truck types with different capacities are available, and that all the truck types have a capacity of an integer multiple, m, of the smallest truck capacity, v, then for any given λ_{total}, the average total cost ratio between using a single truck of capacity v and a single truck with capacity mv, is given as:

$$\frac{G(T_j)_{j=1}}{G(T_j)_{j\geq 1}} = \left(\frac{h\lambda_{total}}{2}\,T + \frac{K_j}{T}\right) \Big/ \left(m_{j\geq 1}\left(\frac{h\lambda_{total}}{2}\,T + \frac{K_j}{T}\right)\right)$$

$$= \frac{G(Q_j)_{j=1}}{G(Q_j)_{j\geq 1}} = \left(\frac{hv}{2} + \frac{K_j\lambda_{total}}{v}\right) \Big/ \left(\frac{m_{j\geq 1}hv}{2} + \frac{K_j\lambda_{total}}{m_{j\geq 1}v}\right)$$

$$= \frac{\text{m}_{\text{j=1}}}{\text{m}_{\text{j}\geq 1}} + \left(\frac{K_{j=1}m_{j\geq 1}}{m_{j=1}K_{j\geq 1}}\right) \quad (3.23)$$

Hence:

$$G(Q_j)_{j=1} = \left(\frac{m_{j=1}}{m_{j\geq 1}} + \left(\frac{K_{j=1}m_{j\geq 1}}{m_{j=1}K_{j\geq 1}}\right)\right) G(Q_j)_{j\geq 1} \quad (3.24)$$

If $m_{j=1} = 1$ then:

$$G(Q_j)_{j=1} = \left(\frac{1}{m_{j\geq 1}} + \left(\frac{K_{j=1}m_{j\geq 1}}{K_{j\geq 1}}\right)\right) G(Q_j)_{j\geq 1} \quad (3.24)$$

Given Equation 3.23, and assuming an infinite number of trucks in which each truck is a possible replenishment, the grouping solution in the previous section (Section 3.6.1.3) is adapted as follows:

1. Since the values of m_j, h, v and K_j for any truck type are constant, then for any truck type the total cost equation is linear with changes in λ_{total}. Figure 12 illustrates an example in which three truck types are considered. Using the intersection of the three lines, three ranges can be identified: if λ_{total} is within Range 1, then truck Type 1 is chosen, if λ_{total} is within Range 2, then truck Type 2 is chosen, and so on.

2. In the previous step the first replenishment is identified. Therefore, update the value of λ_{total} using:

$$\lambda_{\text{total}} = \lambda_{\text{total}} - \lambda_{\text{j}}$$

   Where λ_{j} is the total demand rate of the previously identified group

3. Go back to Step 1 until λ_{total} is 0

Note that if λ_{total} is greater than $3v$, then truck Type Three is selected, and we proceed immediately to Step 2. Moreover, though we assume that as mv increases the value of K_j also increases, the ordering cost per item decreases with increases in mv. This explains the different slopes of the cost function lines in Figure 12. Furthermore, knowing the different groups of products minimising the cost in Equation 3.15, we can allocate the different λ_is to the available trucks using the Bin Packing approach. In a typical bin packing problem, objects of different volumes must be bin packed into a finite number of bins (containers in our case), in a way that minimises the number of bins used. For this purpose, a LINGO optimisation model was developed to solve the set of linear equations and to perform the Bin Packing process.

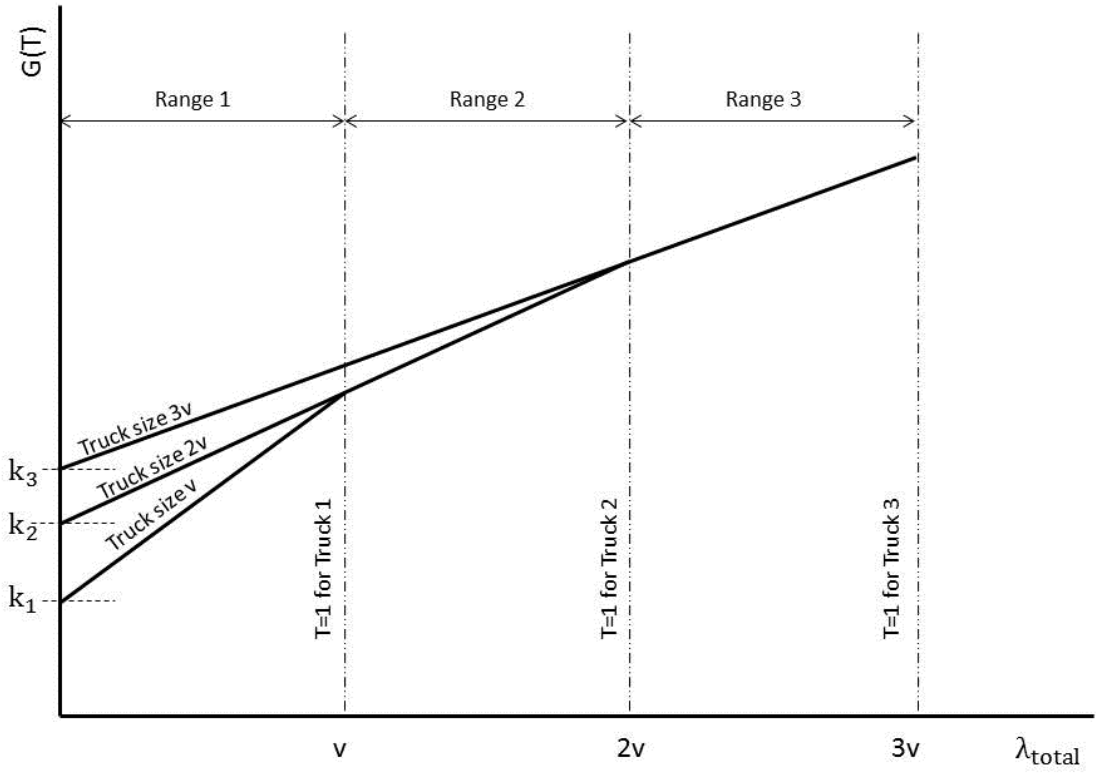


Figure 12: Finding the best groups using G(T) vs Average demand rate graph

Note that for the purpose of this project, a G(T) vs Demand Rate graph is produced for each set of trucks used in the experiments. This is in particular to evaluate the performance of the Lingo grouping model. The Excel sheet used to develop these graphs is included in the electronic appendix.

3.6.2 The Experimental setup

Executing the procedure as described in the previous section manually is clearly time-consuming. This is especially the case because of the large number of possible linear equations to be computed in the case of more than three truck types. Moreover, an allocation of the products to the different selected replenishments is still needed. Therefore, a LINGO model was developed to solve our minimisation problem by solving the sets of linear equations. Further, the model will allocate the different products to the selected replenishments using the bin packing model. We formulate the model as follows:

- **Step 1: Decide on decision variables**

$x_{i,j}$ A binary number. If it is equal to zero then item i is not included in replenishment j. Otherwise, if $x_{i,j} = 1$ then item i is included in replenishment j.

y_j A binary number. If $y_j = 0$ then replenishment j is used. Otherwise if $y_j = 1$ then replenishment j is not used.

d_j^- Under-load decision variable to allow for deviation from the ideal demand for replenishment j.

d_j^+ Over-load decision variable to allow for deviation from the ideal demand for replenishment j.

- **Step 2: Formulate the objective function**

The ultimate goal of this grouping heuristic is to minimise the total cost of replenishing the n items while using full truckloads and satisfying λ_{total}. Accordingly, the objective function is expressed as follows:

$$\sum_{j=1}^{m}\left(\frac{1}{2}\times h\times Q_j\times y_j+\frac{K_j}{Q_j}\times\sum_{i=1}^{n}\lambda_{\text{i}}x_{i,j}\right)$$

Where:

h Is the inventory holding cost per pallet per day

Q_j Is the order quantity and, therefore, the truck capacity of replenishment j

K_j Is the fixed cost per order

λ_{i} Is the average daily demand for any product i

- **Step 3: Define the constraints**

1. Every item should be allocated to one replenishment only, such that:

$$\sum_{j=1}^{m} x_{i,j} = 1 \quad \forall\, item\ i$$

2. Logical condition: If item i is loaded in replenishment j, then replenishment j is used, such that:

$$x_{i,j} \leq y_j \quad \forall\, item\ i\ \&\ replenishment\ j$$

3. Capacity constraint: For any used replenishment j, the sum of the demand rate included in replenishment j plus a deviation must equal to λ_j^*.

$$\sum_{i=1}^{n} \lambda_i x_{i,j} + d_j^- - d_j^+ = \lambda_j^* y_j \quad \forall\ replenishment\ j$$

4. Special condition: For any replenishment j, $T_j \geq 1$.

$$\sum_{i=1}^{n} \lambda_i\, x_{i,j} \leq Q_j \quad \forall\ replenishment\ j$$

5. $x_{i,j}$ and y_j are binary numbers.

$$x_{i,j} = 0\ or\ 1 \quad \forall\, product\ i\ and\ replenishment\ j$$
$$y_j = 0\ or\ 1 \quad \forall\ replenishment\ j$$

6. The deviation variables are greater than, or equal to, zero.

$$d_j^- \geq 0 \qquad \forall\ replenishment\ j$$
$$d_j^+ \geq 0 \qquad \forall\ replenishment\ j$$

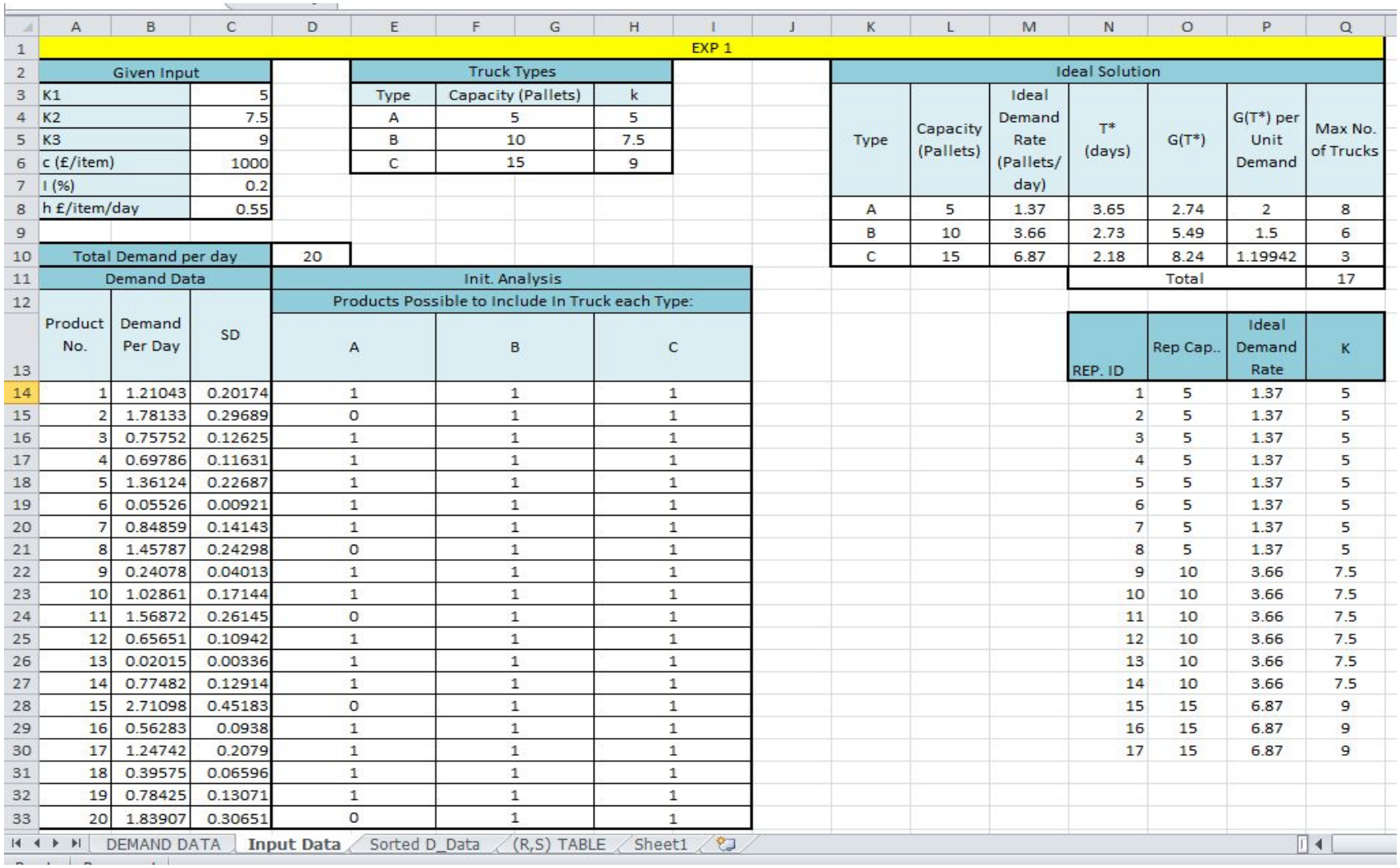
- **Step 4: Develop the LINGO code using the above formulas**

```
! GROUPING_EXPERIMENT_1;

! Initiate the model;

Model:

! Initiate the sets;

Sets:
! Create a set of 20 products;

PRODUCT/1..20/:

! For each product define the average demand rate;

DEMAND;

! Create a set of x replenishments. Note that the number of replenishments needed is
computed in an external initial preparation analysis excel sheet;

REP/1..17/:
y,   ! is a binary decision variable. If y=1 include rep. j. if y=o do not include rep j;
Q,  ! is the truck capacity used in replenishment j;
IDEAL_DEMAND, ! is the ideal demand rate of replenishment j;
D1, ! is the overachievement decision variable;
D2, ! is the underachievement decision variable for replenishment j;
K; ! is the fixed ordering cost of replenishment j;

! define the  set of product i and replenishment j;
PXR(PRODUCT,REP):
X; ! is a binary decision variable. If x=0 then product I is not included in replenishment
j. if x=1 then product I is included in replenishment j;
endsets

data:
DEMAND=@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','AVG_DEMAND');
Q=@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','Q');
IDEAL_DEMAND=@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','IDEAL_DEMAND');
H=@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','HOLDING_COST');
K=@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','FIXED_COST');
enddata

!Objective function;
MIN = @SUM (REP(J): (0.5*H*Q(J)*Y(J))+(K/Q(J))*( @SUM (PRODUCT(I): DEMAND(I)*X(I,J))));
!constraints;
!every item should be allocated to one replenishment only;
@FOR ( PRODUCT(I): @SUM (REP(J): X(I,J)) =1);
!x is an intejer = 0 or 1 (i.e. x is binary;
@FOR (PXR (I,J): @BIN (X(I,J)));
!y is binary;
@FOR (REP(J): @BIN (Y(J)));
!logical condition;
@FOR(PXR(I,J): X(I,J) <= Y(J));
!loading/capacity constraint;
@FOR(REP(J): (D1(J)- D2(J)+ @SUM (PRODUCT (I): DEMAND(I)*X(I,J)))= IDEAL_DEMAND(J)*Y(J));
! lower bound for T (i.e. T>= 1);
@for (REP(J): (@SUM (PRODUCT (I): DEMAND(I)*X(I,J))) <= Q(J));

end
```

Figure 13: The grouping model for Experiment 1

The LINGO IP grouping model for Experiment 1 is illustrated in Figure 13 above. The grouping models for the other experiments are included in the appendix. It is worth mentioning that the LINGO software tool is a simple tool for deploying the power of linear and nonlinear optimization in order to formulate large and solve problems concisely, as well as analyse their solutions. Optimization (whether for linear or non-linear problems) can help identify, for instance, the lowest costs, while maximising resources.

Moreover, Note that in the lingo model the data section is used to import data relating to variables such as the real demand, trucks ideal demand, ideal order quantities (i.e. replenishment capacity), holding costs and the fixed ordering cost for each replenishment from an MS Excel spreadsheet (shown in Figure 14). The sheet is divided into several parts: Part 1, (the given data), includes the set of fixed data for the experimental instance. This includes the fixed ordering cost for the three truck types, the purchase cost per product and the holding cost per pallet per day. Part 2, (the truck types), presents the specific capacity and fixed ordering cost for each truck type. Part 3, (the ideal solution), is where the optimal solution from the truck perspective is computed. In this part, for each truck, the ideal demand rate, the ideal replenishment cycle, the total cost of the ideal solution, the cost per unit demand satisfied and the total number of trucks needed from each truck type are all calculated. Part 4, (the demand data), demonstrates the average daily demand and the standard deviation of demand for the products under consideration. Furthermore, in this part, initial analysis to reduce the number of replenishments used in the LINGO models is performed. The main rationale behind this is to reduce the total computational time. In this part, for any truck type, we determine whether a product i is included in replenishment j for all the truck types. If the average demand rate for product i is greater than the ideal demand rate of truck j, then the product is not considered when the maximum number of replenishments is computed for truck j. Part 5, finally, lists all the possible replenishments for this particular experimental instance.



EXP 1

Given Input	
K1	5
K2	7.5
K3	9
c (£/item)	1000
I (%)	0.2
h £/item/day	0.55

Truck Types		
Type	Capacity (Pallets)	k
A	5	5
B	10	7.5
C	15	9

Total Demand per day	20

Demand Data			Init. Analysis		
			Products Possible to Include In Truck each Type:		
Product No.	Demand Per Day	SD	A	B	C
1	1.21043	0.20174	1	1	1
2	1.78133	0.29689	0	1	1
3	0.75752	0.12625	1	1	1
4	0.69786	0.11631	1	1	1
5	1.36124	0.22687	1	1	1
6	0.05526	0.00921	1	1	1
7	0.84859	0.14143	1	1	1
8	1.45787	0.24298	0	1	1
9	0.24078	0.04013	1	1	1
10	1.02861	0.17144	1	1	1
11	1.56872	0.26145	0	1	1
12	0.65651	0.10942	1	1	1
13	0.02015	0.00336	1	1	1
14	0.77482	0.12914	1	1	1
15	2.71098	0.45183	0	1	1
16	0.56283	0.0938	1	1	1
17	1.24742	0.2079	1	1	1
18	0.39575	0.06596	1	1	1
19	0.78425	0.13071	1	1	1
20	1.83907	0.30651	0	1	1

Ideal Solution						
Type	Capacity (Pallets)	Ideal Demand Rate (Pallets/ day)	T* (days)	G(T*)	G(T*) per Unit Demand	Max No. of Trucks
A	5	1.37	3.65	2.74	2	8
B	10	3.66	2.73	5.49	1.5	6
C	15	6.87	2.18	8.24	1.19942	3
				Total		17

REP. ID	Rep Cap..	Ideal Demand Rate	K
1	5	1.37	5
2	5	1.37	5
3	5	1.37	5
4	5	1.37	5
5	5	1.37	5
6	5	1.37	5
7	5	1.37	5
8	5	1.37	5
9	10	3.66	7.5
10	10	3.66	7.5
11	10	3.66	7.5
12	10	3.66	7.5
13	10	3.66	7.5
14	10	3.66	7.5
15	15	6.87	9
16	15	6.87	9
17	15	6.87	9

DEMAND DATA | Input Data | Sorted D_Data | (R,S) TABLE | Sheet1

Figure 14: Grouping initial analysis for Experiment 1

3.7 The Joint Replenishment Model

In Section 3.6, the method of forming different groups of products was explained. The next step is to implement the inventory control policy presented in Section 3.3 through the simulation model. Section 3.7.1 explains the formulation of the policy parameters. Section 3.7.2 sets out the method used to obtain the order quantities, while Section 3.7.3 explains the simulation model used to test the solution. Finally, Section 3.7.4 illustrates how the performance of the performed simulation is assessed.

3.7.1 Formulating T and S

Having arrived at the grouping of products in Section 3.6, the next step is to calculate the review period for each group of products (T_j) and to calculate the order-up-to level for each product (S_i).Typically, in implementing a (T, S) policy, for any group of products with the same holding cost the value of T_j is computed as:

$$T_j^* = \sqrt{\frac{2K}{h\lambda_j}} \quad (3.25)$$

Where,

T_j Is the review period (in days) of the group for replenishment j

λ_j Is the total average demand for a group of products replenished at T_j

In our solution, the optimal replenishment cycle T_j^* is not used. This is because calculating T_j using Equation 3.25 does not take into account the capacity of the truck used in replenishment j. Instead, T_j is calculated using the demand rate of the allocated group and the capacity of the truck as follows:

$$T_j = \frac{Truck\ capacity\ of\ replenishment\ j}{\lambda_j} = \frac{Q_j}{\lambda_j} \quad (3.26)$$

Moreover, note that Equation 3.26 does not necessarily lead to an integer value of T_j. Therefore, in the proposed solution, T_j is rounded to the nearest integer

day. Assuming a Type 2 service level with a target Service Level of 99.5%, S_i can be calculated as follows:

$$S_i = EDD(T_j + \tau) + SS \quad (3.27)$$

Where the expected demand during τ and T_j ($EDD(T_j + \tau)$) can be expressed as:

$$EDD(T_j + \tau) = \lambda_j \times (T_j + \tau)$$

The safety stock (SS) is calculated as:

$$SS = k_i \times \sigma_{(T_j+\tau)}$$

The standard deviation of τ plus T_j ($\sigma_{(T_j+\tau)}$) is obtained as follows:

$$\sigma_{T_j+l} = \sigma\sqrt{(R + L)}$$

Note that σ is the standard deviation of demand, and that k_i is the safety stock factor. As such, the value of k_i can be estimated using the loss function $G(K)$ and the corresponding k value in the normal distribution tables as provided by Silver (1998). To determine the value of the Safety Factor k_i, we use an approximation approach presented in Silver & Bischak (2011)

$$G(k) = \frac{R(1 - P)}{\sqrt{R + l} \times CV}$$

Where,

$$CV = \frac{\sigma}{\mu}$$

In the above equation, P is the fill rate and μ is the mean of the demand.

3.7.2 Determination of the Order Quantities

After forming the different groups of products and after calculating the (T, S) policy parameters for each group and for each product, the next step is to obtain the order quantity for each product. As already stated, the adjusted (T, S) used here is described as follows: At each review instant, T_j, the inventory position of each item i in group j, is reviewed and an order is placed such that its inventory position is raised to $S_i + d_i^- - d_i^+$. In other words, at each review instant, the order quantity is obtained such that the following equation is satisfied:

$$Q_{i,j} + d_{i,j}^- - d_{i,j}^+ + I_{i,j} = S_i^* \quad (3.28)$$

Where,

$Q_{i,j}$ Is the order quantity of item i on day j

$d_{i,j}^-$ Order quantity underachievement variable of item i on day j

$d_{i,j}^+$ Order quantity overachievement variable of item i on day j

$I_{i,j}$ Is the inventory position of item i on day j

S_i Is the order up to level of item i

Further, since the total order quantity must reach a full truckload, the following equation must also be satisfied:

$$\sum_{i=1}^{n} Q_{i,j} = Q_j \ \forall \, day\, j \quad (3.29)$$

Clearly, for the solution to remain valid, and to avoid unbalanced ordered quantities for different items, in computing the order quantities one should aim to minimise the deviation from the ideal order quantity ($S_i^* - I_{i,j}$). To this purpose, the following minimisation problem is solved:

$$min \sum_{i=1}^{n} \left(d_{i,j}^- + d_{i,j}^+\right) \quad \forall \, day\, j \quad (3.30)$$

In computing the required $Q_{i,j}'s$, the minimisation problem in Equation 3.30 is solved such that Equation 3.28, and the capacity constraint posed by Equation 3.29 are both satisfied. In addition, the order quantity and the deviation variables must also be positive integer values.

3.7.3 The (T, S) Simulation

In Sections 3.6, 3.71 and 3.72, the methods deployed to address the following three questions were explained:

1. How are the different products grouped?
2. When do we place an order?
3. How much do we order for each product?

The next step is, consequently, to deploy these methods to perform the adjusted (T, S) policy simulation. As per Section 3.5, the period of interest is 1000 days and, therefore, an automated simulation model is needed to compute the order quantities for each product at each review period. To this purpose, a minimisation model using LINGO is developed, and is described in detail in the present section. Section 3.7.3.1 explains the MS Excel spreadsheets used to organise the input parameters for the model. Section 3.7.3.2 lists the notations used in this model. Section 3.7.3.3 formulates the model used for finding Q using the formulas in section 3.7.2. Section 3.7.3.4 explains the rationale behind performing the joint replenishment simulation over the 1000 day period. Section 3.7.3.5 presents the decision logic codes used in LINGO. Finally, Section 3.7.3.6 describes the methods used to compute the performance measures.

3.7.3.1 T and S Calculations in Excel

Before implementing the modified (T,S) policy, the value of T_j for each group and the value of S_i for each product, are all calculated using Equation 3.25 and 3.26, respectively. For this purpose, a spreadsheet is generated for each experiment. Figure 15 shows the spreadsheet used to calculate the parameters of the (T,S) policy for the same experimental example used to demonstrate the grouping heuristic (i.e. Experiment 1). The Excel sheets for all the experiments under investigation are provided in the Appendix. The list of formulas used in the Excel

sheets is provided in Table 14. Note that the groups in Figure 15 were formed according to the results of the grouping example in the previous section.

Table 14: Excel formulas to compute T & S

Parameter	Formula	Cell
Group	Name the Group	B3
Product ID	From the grouping heuristics	C3
Average daily demand (Pallets/ Day)	=VLOOKUP(C3,'Input Data'!A$14:C$33,2)	D3
Average Group Daily Demand (Pallet/Day)	=SUM(D3:D7)	E3
Truck Type	From the grouping heuristics	F3
Truck Capacity (Pallets)	=VLOOKUP(F3,'Input Data'!E4:H6,4)	G3
Kj (£/Order)	=VLOOKUP(F3,'Input Data'!E4:H6,4)	H3
h (£/Pallet/Day)	='Input Data'!C8	I3
Tj (Day)	=G3/E3	J3
Tj to the nearest integer	=ROUND(J3,0)	K3
Exp Demand Lead time & R	=(K$3+3)*D3	L3
SD of average demand	=VLOOKUP(C3,'Input Data'!A$14:C$33,3)	M3
SD Lead time & R	=SQRT(3+K$3)*M3	N3
G(k)	=(K$3*(1-0.995))/((SQRT(3+K$3))*(M3/D3))	O3
k (From tables)	From distribution tables	P3
SS	=P3*N3	Q3
Si	=ROUND((Q3+L3),2)	R3

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		review period & order-up-to level calculations																
2		Group	Product No.	Average daily demand	Average Group Daily Demand	Truck Type	K	Capacity	h	R (group)	R to the nearest integer	Exp Demand Leatime & R	SD λi	SD Leadtime & R	G(k)	k (from tables)	SS	Si
3		A	1	1.210431	5.00006	B	7.5	10	0.55	1.999976	2	3.631294	0.201739	0.349421	0.034641	1.43	0.499673	4.13
4			5	1.36124								4.083721	0.226873	0.392956	0.034641	1.43	0.561927	4.65
5			6	0.055264								0.165793	0.009211	0.015953	0.034641	1.43	0.022813	0.19
6			11	1.56872								4.70616	0.261453	0.45285	0.034641	1.43	0.647576	5.35
7			13	0.020149								0.060448	0.003358	0.005817	0.034641	1.43	0.008318	0.07
8			19	0.784255								2.352764	0.130709	0.226395	0.034641	1.43	0.323745	2.68
9		B	2	1.78133	14.99994	C	9	15		1.000004	1	3.562661	0.296888	0.419864	0.0173205	1.73	0.726364	4.29
10			3	0.757515								1.51503	0.126253	0.178548	0.0173205	1.73	0.308888	1.82
11			4	0.697857								1.395714	0.11631	0.164487	0.0173205	1.73	0.284562	1.68
12			7	0.848593								1.697187	0.141432	0.200015	0.0173205	1.73	0.346027	2.04
13			8	1.457873								2.915746	0.242979	0.343624	0.0173205	1.73	0.594469	3.51
14			9	0.240776								0.481552	0.040129	0.056751	0.0173205	1.73	0.09818	0.58
15			10	1.028612								2.057224	0.171435	0.242446	0.0173205	1.73	0.419432	2.48
16			12	0.656513								1.313026	0.109419	0.154742	0.0173205	1.73	0.267703	1.58
17			14	0.774823								1.549645	0.129137	0.182627	0.0173205	1.73	0.315946	1.87
18			15	2.71098								5.42196	0.45183	0.638984	0.0173205	1.73	1.105443	6.53
19			16	0.562826								1.125653	0.093804	0.132659	0.0173205	1.73	0.229501	1.36
20			17	1.247419								2.494838	0.207903	0.294019	0.0173205	1.73	0.508654	3
21			18	0.395753								0.791505	0.065959	0.09328	0.0173205	1.73	0.161374	0.95
22			20	1.839069								3.678138	0.306512	0.433473	0.0173205	1.73	0.749908	4.43
23																		

Figure 15: T & S calculations for Experimetn 1

3.7.3.2 The Simulation Notation

The first step in developing the simulation model is to define the notation to be used. For the purposes of this project the following notation is adopted:

i	Index to denote product
j	Index to denote day
L	Lead time
R	The review period in days
S	The order up to level (in pallets) for product i
Q	The order quantity for product i at day j
INV	The inventory position of product i at day j
$INVE$	The inventory position of product i at the end of day j
SOH	The stock on hand of product i at day j
$SOHE$	The stock on hand of product i at the end of day j
D	The demand rate of product i at day j
$D1$	The overachievement decision variable
$D2$	The underachievement decision variable
REC	Received order quantities of product i at the end of day j
DEL	Quantity of item i delivered to the end customer at day j to satisfy demand
$NDEL$	Unsatisfied demand of product i at day j
$SLEVEL$	Service Level of product i at day j

3.7.3.3 The Main Model

This section summarises and reformulates the minimisation model presented in Section 3.7.3 using the simulation notation, as follows:

1. Decision Variables: These are the variable to be determined so as to optimise the objective function. In our case, the decision variables are:

   - $Q_{i,j}$
   - $D1_{i,j}$
   - $D2_{i,j}$

2. Data: These are the given variables that quantify the relationships represented in the objective function and the constraints. In our problem these are:

   - S_i
   - CAP
   - $D_{i,j}$
   - R
   - L

   Note that the values of S_i for each product, the review time for each group and the value of CAP for the group of products are all obtained using the spreadsheet presented in Section 3.7.2. The demand data for each product at each day is obtained from the demand data Excel sheet.

3. Objective Function: This function represents how the decision variables affect the value to be optimised (minimised in our case).

$$min \sum_{i=1}^{n} D1_{i,j} + D2_{i,j} \quad \forall day\ j$$

Note that this is essentially the same minimisation problem shown in Equation 3.30.

4. Constraints: To represent how the decision variables use resources, which are available in limited quantities and to articulate any special conditions. In our problem, the objective function is subject to:

    - A capacity constraint, in which the sum of the order quantities must equal the capacity of the truck:

$$\sum_{i=1}^{n} Q_{i,j} = CAP \quad \forall\, day\ j$$

    - A resource constraint, in which the order quantity for each item at each day, plus the minimum deviation obtained using the objective function, plus the current inventory position, must all equal the order up to level.

$$Q_{i,j} + D1_{i,j} - D2_{i,j} + INV_{i,j} = S_i \quad \forall\, product\ i\ \&\ day\ j$$

    - A condition, according to which the deviation variables are positive:

$$D1_{i,j} \geq 0 \quad \forall\, product\ i\ \&\ day\ j$$

$$D2_{i,j} \geq 0 \quad \forall\, product\ i\ \&\ day\ j$$

    - A condition, according to which the ordered quantities must be a positive integer number or zero:

$$Q_{i,j} \geq 0 \quad \forall\, product\ i\ \&\ day\ j$$

$$Q_{i,j}\ is\ an\ integer\ number$$

3.7.3.4 The Sequence of Decisions

To apply the model presented in the previous section at each replenishment day, the following sequence of decisions is used:

Step 1. Define the size of the replenishment period.

$$MAXDAY = 1000$$

Step 2. Initialise data for the first day, such that:

$$\begin{cases} j = 1 \\ INV_{i.1} = SOH_{i,1} = INVE_{i,1} = SOHE_{i,1} = S_i \\ D_{i,1} = 0 \\ Q_{i,1} = 0 \end{cases}$$

Step 3. Set j = j + 1

Step 4. Perform modulo calculations to check for incoming goods:

If $j \; mod \; R + L \equiv 0$

Then, $REC_{i,j} = Q_{i,j-l}$

Else, $REC_{i,j} = 0$

Step 5. Update Stock on Hand:

$$SOH_{i,j} = REC_{i,j} + SOHE_{i,j-1}$$

Step 6. If $SOH_{i,j} \geq D_{i,j}$, then

$$\begin{cases} DEL_{i,j} = D_{i,j} \\ NDEL_{i,j} = 0 \\ SOHE_{i,j} = SOH_{i,j} - DEL_{i,j} \end{cases}$$

Else if $SOH_{i,j} < D_{i,j}$, then

$$\begin{cases} DEL_{i,j} = SOH_{i,j} \\ NDEL_{i,j} = D_{i,j} - DEL_{i,j} \\ SOHE_{i,j} = 0 \end{cases}$$

Step 7. Update inventory position:

$$INV_{i,j} = INVE_{i,j-1}$$

Step 8. Perform modulo calculations to check if it's a review day:

If $j \bmod R \equiv 0$ Then solve the optimisation model in section 3.7.3.3 to determine $Q_{i,j}$,

Else

$Q_{i,j} = 0$

Step 9. Update inventory position at the end of the day:

$$INVE_{i,j} = INV_{i,j} - D_{i,j} + Q_{i,j}$$

Step 10. Loop:

If $j < MAXDAY$ go to step 3

Else

If $j \geq MAXDAY$

End

3.7.3.5 The LINGO Simulation Model Explained

With the high number of products in the JRP, solving the set of linear equations to find Q over a long time period can be a difficult and time-consuming task. As a result, the simulation of the proposed joint replenishment model is programmed into LINGO as an integer programming problem. The use of integer programming is due to the fact that full pallets are needed for each product. This integer programing problem is applied over the required replenishment period, using the sequence of decisions provided in Section 3.7.3.4.

Integer programming is a mathematical programming technique used to solve linear problems that require integer solutions, and LINGO uses the well-known branch-and-bound (B&B) algorithm to search for the optimal solution. The LINGO simulation model used consists of the following sections:

1. The Sets Section

In LINGO, a set is simply a group of related objects. Each member of the set may have one or more characteristics associated with it. Figure 16 shows an example of the defined sets for the first group in Experiment 1.

```
SETS:

! Generate a set of 6 products;
PRODUCT/1..6/:

! An order-up-to level, S, is assigned to each product, I;
S;

! Generate a set of 1000 days;
DAY/1..1000/:

! Modulo operation is used to determine if day J is a review day or not and to
determine if a previously Ordered batch will be received. The reminder of division
of day, J, by the review period of the group, R (Modulo calculations);
MOD,

!The reminder of division of day, J, by the review period of the group, R, plus the
lead time, L;
MODRL;

! Product i at day j set;
PXD(PRODUCT,DAY):

! Product i at day j the following parameters are defined;

INV,    ! Inventory position of product I at day J ;
INVE,   ! Inventory Position of product I at the end of day J ;
SOH,    ! Stock on Hand of product I at the end of day J;
SOHE,   ! Stock on Hand of product I at the end of day J;
Q,      ! Order quantity of product I at the end of day J;
D,! Demand of product I at day J;
D1,     ! Over Achievement;
D2,     ! Under Achievement;
REC,    ! Item received at day J;
DEL,    ! Satisfied demand of product I at day J;
NDEL,   ! Un satisfied demand of product I at day J;
SLEVEl;! Service level;

ENDSETS
```

Figure 16: The Sets Section in the Lingo simulation

2. The Data Section

The data section is used for inputting set members and data values. This section also allows for data isolation from the rest of the model, which is very useful to facilitate the model's maintenance and scaling. In LINGO, the model starts with DATA and ends with ENDDATA. Figure 17 illustrates the data section used in the LINGO programme for Group 1 in Experiment 1.

```
DATA:

! Import Order-up-to level data for each product from the Excel sheet -
(T,S) Table;
S =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','S_LEVEL_GA');

! Given mean of demand and the standard deviation of demand,
  use excel to generate random demand for the needed number of days.
  Then use @OLE function to import demand data;
D = @OLE('\Users\Cripps Hire
Laptop\Desktop\EXP1.xlsx','REAL_DEMAND_GA');

Cap= 10; ! The capacity of the truck allocated to this group;
L = 1;   ! Lead time;
R = 2;   ! Group review time calculated in the Excel sheet;

!Export to excel;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','SL') = SLEVEL;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','SOH') = SOH;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','Q_OUTPUT') = Q;

ENDDATA
```

Figure 17: The Data Section in the Lingo simulation

Note that @OLE is a lingo function that allows for data importation from MS Excel. As shown in Figure 14, the S and D data are imported from the external Excel sheet for Experiment 1, using the @OLE function. The use of this function is very useful as it automates the process of inputting data, which saves time and effort and helps prevent errors. Furthermore, this function is also used to export the results to the Excel sheet. In particular, it is used to export the data of the SLEVEL, SOH and Q, which can subsequently be used to obtain the performance measures and thus to conduct the necessary analyses.

3. The Main Model (The Objective Function)

This section represents the core of the model, as this is where the values of Q are obtained using the model described in Section 3.7.3.3. Figure 18 presents the LINGO code used to programme the main model at each review period.

```
! The following sub model is used to allocate the available truck
capacity to the different products in the group. The model will minimise
the total deviation from the pre-identified order-up-to level;

SUBMODEL FIND_Q:

! At each day minimise the sum of the deviation from S as a percentage
of S of each product;

@FOR (DAY (J)| J #EQ# DAYN: MIN = @SUM ( PRODUCT(I):
(D1(I,J)+D2(I,J))));

! Subject to the following constraints:

  1- The order quantity is a positive integer value;
@for ( pxd (i,J)| J #EQ# DAYN: @gin ( Q(i,J)));

! 2- At day J the, the sum of the ordered quantities must fill the
truck;
@for ( day (J)| J #EQ# DAYN: @sum (product(i): q(i,j)) = cap);

! 3- define the order quantity function;

@for ( day (j)| J #EQ# DAYN: @for (product (i): INV (I,J) + d1(i,j) -
d2(i,j) + Q(i,j) = s(i)));

! The deviation variables are positive;

@for ( day (j)| J #EQ# DAYN: @for (product (i): D1(I,J) >= 0));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D2(I,J) >= 0));

ENDSUBMODEL
```

Figure 18: The Objective Function in the Lingo simulation

4. The Secondary Model

This model is only used when the current day is not a review day, so as to set a value of zero to the order quantities. This simple model is shown in Figure 19.

```
! This simple model is used if day j is not a review day to set Q(i,j)
values to 0;
SUBMODEL NO_Q:

@FOR (DAY (J)| J #EQ# DAYN: @FOR (PRODUCT (I): Q(I,J) = 0));

ENDSUBMODEL
```

Figure 19: The Secondary Model in the Lingo simulation (Q=0)

5. The Calculations Section

This section is for performing computations on raw input data. In LINGO, a CALC section begins with the keyword CALC, and ends with the keyword ENDCALC. Each expression must be in the form of an assignment statement, in which a single variable appears on the left-hand side of an expression, followed by an equality sign and an arbitrary mathematical expression on the right-hand side. In our model, the calculation section is used to automatically apply the main model on the replenishment period using the sequence of decisions described in Section 3.7.3.4. Further, this section is used to update all the variables that depend on Q each day. Figure 20 illustrates the initiation of this section. Note that a WHILE loop is used to perform the same calculations for the required number of days and that we initiate the loop by setting Day to Day 1. Figure 21 illustrates the rest of the model. Note the use of the @IFC function to distinguish between the calculations of Day 1 from those of the other day.

```
CALC:

! define the size of the replenishment period;
MXDAY = @SIZE (DAY);

! Set day to day 1;
DAYN = 1;

! Initiate a while loop to apply the sub models (when needed);
@WHILE ( DAYN #LE# MXDAY:
```

Figure 20: The Calculations Section (start)

```
! First we distinguish between day 1 and all the other days;
@IFC ( DAYN #EQ# 1:

! For day 1, calculate the reminder of division of day, J, by the review period of the
group, R;
@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!for day 1, calculate he reminder of division of day, J, by the review period of the
group, R, plus the lead time, L;
@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

! Initiate the data for the first day;
! The next 4 functions will set SOH(i,j) = SOHE(I, J) = INV (I, J) = INVE(I, J) = S(I);
@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOHE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INV (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INVE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: REC (i,j) = 0));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SLEVEL (i,j) = 0));

! the second branch of the first @IFC function applies to days greater than 1;
! Here a set of calculations will be performed for any particular day greater than 1;
@ELSE@IFC (DAYN #GT# 1:

@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

@FOR (PRODUCT (I): @FOR (DAY (J) | J #EQ# DAYN: @IFC (DAYN #GE# L+1: REC(I,J) = Q(I, J-L);

@ELSE@IFC (DAYN #LT# L+1: REC (I,J) = 0;););));

@for ( product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = REC(I,J) +SOHE(i, j-1)));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): @IFC ( SOH(I,J) #GE# D(I,J): DEL(I,J) =
D(I,J); NDEL(I,J) = 0; SOHE (I,J) = SOH(I,J) - DEL (I,J);

@ELSE@IFC ( SOH(I,J) #LT# D(I,J): DEL(I,J) = SOH (I,J); NDEL(I,J) = D(I,J) - DEL(I,J);
SOHE (I,J) = 0;););));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): INV (I,J) = INVE (I,J-1)));

););

@for ( Day (j) | J #EQ# DAYN: @IFC ( MOD(J) #EQ# 0:

@SOLVE (FIND_Q);

@ELSE@IFC ( MOD(J) #GT# 0:

@SOLVE (NO_Q););););

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): INVE (I,J) = INV(I,J) -
D(I,J)+ Q(I,J)));

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): SLEVEL (I,J) = DEL (I,J)/
D (I,J) ));

DAYN= DAYN + 1;);

ENDCALC

END
```

Figure 21: The Calculations Section (end)

3.7.3.6 Results

As shown in Figure 17, the @OLE function is used to export the daily performance measures of each product within the group to an external Excel sheet. Figure 22 shows the daily performance Excel sheet for group A in Experiment 1.

	A	B	C	D	E
1	Poduct ID.	Day	SOH	Q	SL
2	1	1	4.26	0	0
3	1	2	4.26	10	1
4	1	3	13.18447	0	1
5	1	4	13.13546	0	1
6	1	5	11.41206	0	1
7	1	6	11.39342	0	1
8	1	7	10.74098	0	1
9	1	8	9.756399	0	1
10	1	9	8.595619	0	1
11	1	10	8.555392	0	1
12	1	11	6.766378	0	1
13	1	12	6.746014	0	1
14	1	13	5.898988	0	1
15	1	14	4.514834	0	1
16	1	15	3.342931	0	1
17	1	16	3.288107	1	1
18	1	17	2.723491	0	1
19	1	18	2.706704	2	1
20	1	19	3.97713	0	1
21	1	20	2.913831	2	1
22	1	21	3.895229	0	1
23	1	22	3.846946	1	1
24	1	23	3.761938	0	1

(R,S) TABLE | LINGO_OUTPUT GA | LINGO_O

Figure 22: Example of a Daily Performance Excel Sheet for the simulation model.

The data in this sheet is then used to compute the overall performance, as shown in Figure 23. First the performance of each group is calculated individually, before the overall performance is calculated. The procedure is carried out as follows:

1. Calculate the average SOH over the 1000 day-period for all products in the same group. The AVG function in Excel is used for this purpose.

2. Calculate the Average service level over the 1000-day period for all products in the same group. The AVG function in Excel is used for this purpose.

3. For each group, compute the sum of the ordered quantities:

$$\sum_{i=1}^{n} \sum_{day=1}^{1000} Q_{i,j}$$

4. Calculate the total purchase cost for each group:

$$\sum_{i=1}^{n} \sum_{day=1}^{1000} Q_{i,j} \times c_{discount}$$

(The $c_{discount}$ is the purchase cost minus any discounts.)

5. Calculate the total inventory holding cost for each group:

$$\sum_{i=1}^{n} \sum_{day=1}^{1000} h \times SOH_{i,j}$$

6. Calculate the total ordering cost by counting the number of orders placed and then multiplying it by K_j.

7. The overall result is then calculated by adding all the different parameters for the different groups. Note that the average overall SOH and average overall service level account need to be taken for the number of products in each group when calculating these parameters.

	A	B
1	Results Group A	
2	Average SOH (pallets/day)	6.29
3	Average Service Level %	99.78%
4	Purchase Cost (£)	£ 4,000,000.00
5	Total Holding Cost (£)	£ 20,719.94
6	Total Ordering Cost (£)	£ 3,750.00
7	Total Cost (£)	£ 4,024,469.94
8		
9	Results Group B	
10	Average SOH (pallets/day)	6.21
11	Average Service Level %	99.82%
12	Purchase Cost (£)	£ 12,000,000.00
13	Total Holding Cost (£)	£ 47,751.14
14	Total Ordering Cost (£)	£ 4,500.00
15	Total Cost (£)	£ 12,052,251.14
16		
17	Overall Results	
18	Average SOH (pallets/day)	6.23
19	Average Service Level %	99.81%
20	Purchase Cost (£)	£ 16,000,000.00
21	Total Holding Cost (£)	£ 68,471.08
22	Total Ordering Cost (£)	£ 8,250.00
23	Total Cost (£)	£ 16,076,721.08
24		

Figure 23: Overall Performance Sheet for the joint replenishment simulation in Experiment 1

3.8 The Adjusted EOQ model

As discussed earlier in the Literature Review, the EOQ model is one of the oldest production and replenishment scheduling models. It describes the fundamental trade-off between fixed ordering costs and holding costs. Despite its simplicity, the EOQ model forms the foundation for many inventory management models. A well-known extension to the EOQ theory is the 're-order point - order quantity', (R, Q), policy. The assumptions behind this policy are essentially the same as those underlying the EOQ model. In implementing this model, a constant lead time and stochastic characteristics of demand are assumed. As such, in order to evaluate the performance of the proposed solution to the JRP, the performance of an adjusted version of the (R, Q) policy is compared with the proposed solution in terms of the performance measures identified earlier. This section explains the procedures used to perform an (R, Q) simulation on the problem under consideration. Sections 3.8.1 and 3.8.2 formulate the policy parameters, which are the re-order point R and the order quantity Q for each product, while Section 3.8.4 discusses the simulation procedures and experimental setup.

3.8.1 Finding the Economic Order Quantity

The EOQ model is a single item replenishment model. As such, in using this model to solve a multi-item inventory problem, each item is considered, and thus replenished, separately. The key principle underlying the (R, Q) policy is that of placing an order of quantity Q whenever the inventory position drops below a pre-defined re-order point R. This is illustrated in Figure 24 below, where τ is the lead time, $I(t)$ is the inventory level at time t, and Q is the order quantity. To optimise the solution in an (R, Q) policy, the optimal order quantity, Q^*, is typically calculated using the following EOQ formula:

$$EOQ = Q^* = \sqrt{\frac{2K\lambda}{h}} \quad (3.31)$$

In the Adjusted (R, Q) policy, however, to make use of quantity discount schemes, Q is adjusted to the nearest full truckload for every replenishment.

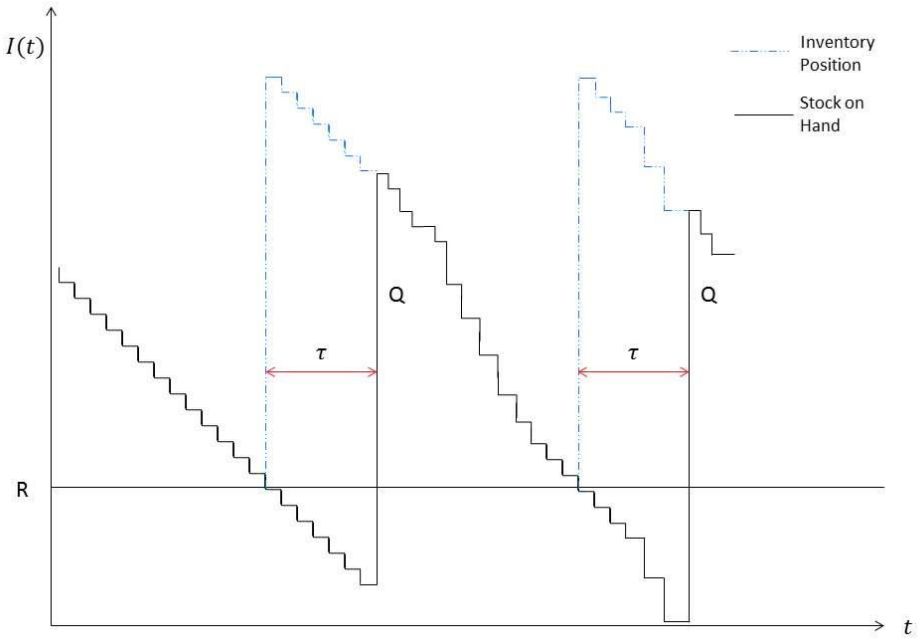


Figure 24: The (R,Q) Policy

3.8.2 Finding the Re-Order Point

To determine when to place an order, assuming a Type 2 Service Level, the re-order point, R, is calculated as follows:

$$R = EDDLT + SS \quad (3.32)$$

In the above, $EDDLT$ is the expected demand during lead time, expressed as:

$$EDDLT = \lambda\tau$$

The SS, the safety stock, is expressed as:

$$k \times \sqrt{\lambda} \times \sigma$$

Here σ_τ is the standard deviation of demand, and k is the safety stock factor. The value of k is obtained using the standardised loss function:

$$L(k) = Q \times \frac{1 - target\ Service\ level}{\sigma_\tau}$$

Note that for this present study, a target service level – which is the fraction of demand satisfied immediately from stock on hand - of 99.5% is adopted. The standard deviation during lead time is calculated as follows:

$$\sigma_{\tau} = \lambda\tau \times \sigma$$

After finding the value of $L(k)$, the specific value of k can be found in the standard normal distribution tables provided in Silver et al. (1998).

3.8.3 The adjusted EOQ simulation

To be able to compare the proposed joint replenishment model with the adjusted EOQ model, an Excel model is developed to perform the (R, Q) simulation. This section discusses the Excel spreadsheet used for the adjusted EOQ simulation over the 1000-day period. Section 3.8.3.1 explains the sequence of decisions used in the simulation mode, while Section 3.8.3.2 explains the logic behind the Excel formulae and the experimental setup in MS Excel. Finally, Section 3.8.3.3 explains how the different performance measures are computed. These performance measures will be used as the basis of comparison between the proposed solution and the adjusted EOQ model.

3.8.3.1 Sequence of Decisions

Using the same notion adopted in the joint replenishment simulation, the EOQ sequence of decisions for each product is as follows:

1. Update the starting inventory position using:

$$INV_j = INVE_j$$

2. Check for any incoming goods and update REC_j using:

$$REC_j = Q_{j-1}$$

3. Update stock on hand at the beginning of the day using:

$$SOH_j = SOHE_j + REC_j$$

4. Decide whether to place an order or not, as follows:

   If

   $$SOH_j > R$$

   Then,

   $$Q_j = 0$$

   Else If

   $$SOH_j \leq R$$

   Then set,

   $$Q_j = Q^*$$

5. Update Q_j to the nearest full truckload.

6. Satisfy demand using the stock on hand and update the other variables as follows:

   If $SOH_{i,j} \geq D_{i,j}$, then

   $$\begin{cases} DEL_j = D_j \\ NDEL_j = 0 \\ SOHE_j = SOH_j - DEL_j \end{cases}$$

   Else if $SOH_j < D_j$, then

   $$\begin{cases} DEL_j = SOH_j \\ NDEL_j = D_j - DEL_j \\ SOHE_j = 0 \end{cases}$$

7. Compute the service level such that:

$$SLEVEL_j = \frac{Total\ DEL_j}{Total\ D_j}$$

This sequence is performed for the entire period. However, it must be pointed out that this sequence of decisions can only be used for a lead time of 1 day.

3.8.3.2 The Experimental Setup in Excel

Having presented how R and Q are calculated, and established the sequence of decisions needed to perform the adjusted EOQ simulation, the next step is to develop an Excel sheet to perform the simulation for each product in every experiment. Figure 25 illustrates the MS Excel sheet devised for this purpose for Product 1 in Experiment 1. The spreadsheet is divided into the following sections:

1. **Given data**

In this section of the spreadsheet, the input data necessary to perform the simulation are entered, including the purchase price, the discounted purchase price, the holding cost, lead time and the target service level. These values are fixed across all experiments, in both the EOQ and the proposed joint replenishment model.

2. **Available Truck Types**

In this section, the relevant truck types and their data are entered for all products and experiments. For any item, the decision of which truck type to use is reached by calculating the optimal order quantity, Q^*, using the different K values for the various truck types. The K value that yields the minimum deviation from Q_i^* for any truck type is chosen. The Q^* value is then adjusted to the full capacity of the chosen truck. (In the example shown in Figure 25 the chosen truck type is highlighted in green.)

3. **R and Q Calculations**

In this section, the value of R and Q^* are calculated using Equation 3.31 and Equation 3.32, respectively. The average daily demand, λ, and the standard

deviation of λ is entered using an HLOOKUP function from the demand data sheet. Furthermore, the safety stock factor (k) value is obtained from the standard normal distribution tables. The Excel formulae used in this section are presented in Table 15.

Table 15: R & Q formulas in Excel

Parameter	Excel Formula	Cell
Average demand per day λ_i	=HLOOKUP(B1,'Demand Data'!B1:U3,2)	E3
SD λ	=HLOOKUP(B1,'Demand Data'!B1:U3,3)	F3
SD τ	=F3*SQRT(P13)	G3
Qi (Pallets)	=SQRT((2*P18*E3)/P12)	H3
L(k)	=(H3*(1-P14))/G3	I3
K	Find from tables	J3
ED (Pallets)	=E3*P13	K3
SS (Pallets)	=J3*SQRT(P13)*F3	L3
R (Days)	=K3+L3	M3

4. **The Simulation**

The simulation involves the conducting of the sequence of decisions described in Section 3.8.3.1, aimed at determining when to place an order. Once the R, Q and all the other relevant data are determined, this part of the spreadsheet is dedicated to perform the simulation of the replenishment process in a period of a thousand days. The daily demand data is inserted into the spreadsheet using an HLOOKUP function. This function will look for the daily demand values for any product and any experiment in the demand data sheet produced in Section 3.5. After defining the initial values of stock on hand, SOH, the inventory position, INV, stock on hand at the end of the day, SOHE and the inventory position at the end of the day, INVE, for the first day, the simulation is carried out using the formulas presented in Table 16.

Table 16:Excel formulas for the adjusted EOQ simulation

Parameter	Excel Formula	Cell
Demand	=HLOOKUP(B$1,'Demand Data'!B$9:U$1009,1+A9)	B9
INV	=I8	C9
SOH	=J8	D9
Ordered	=IF(C9 >=A$3,0,C$3)	E9
Received	=E8	F9
Delivered	=IF(D9>=B9,B9,D9)	G9
Not Delivered	=B9-G9	H9
INVE	=C9+E9+F9-B9	I9
SOHE	=D9-G9+F9	J9
Service level	=SUM(G$8:G9)/SUM(B$8:B9)	K9

5. **Simulation Results**

This section of the spreadsheet calculates the set of performance measures for each product. These include the average inventory level, the average service level and the total inventory costs. The average SOH for the thousand-day-period is calculated as follows:

$$\sum_{day=1}^{1000} SOH \Big/ 1000$$

The average service level is calculated as:

$$\sum_{day=1}^{1000} SL \Big/ 1000$$

The overall replenishment cost for the full period is calculated as follows:

Total Cost = Total Inventory Holding Cost

+ Total Ordering Cost

+ Total Purchase Cost

Where,

$$\text{Total Inventory Holding Cost} = \text{h} \sum \text{Q}$$

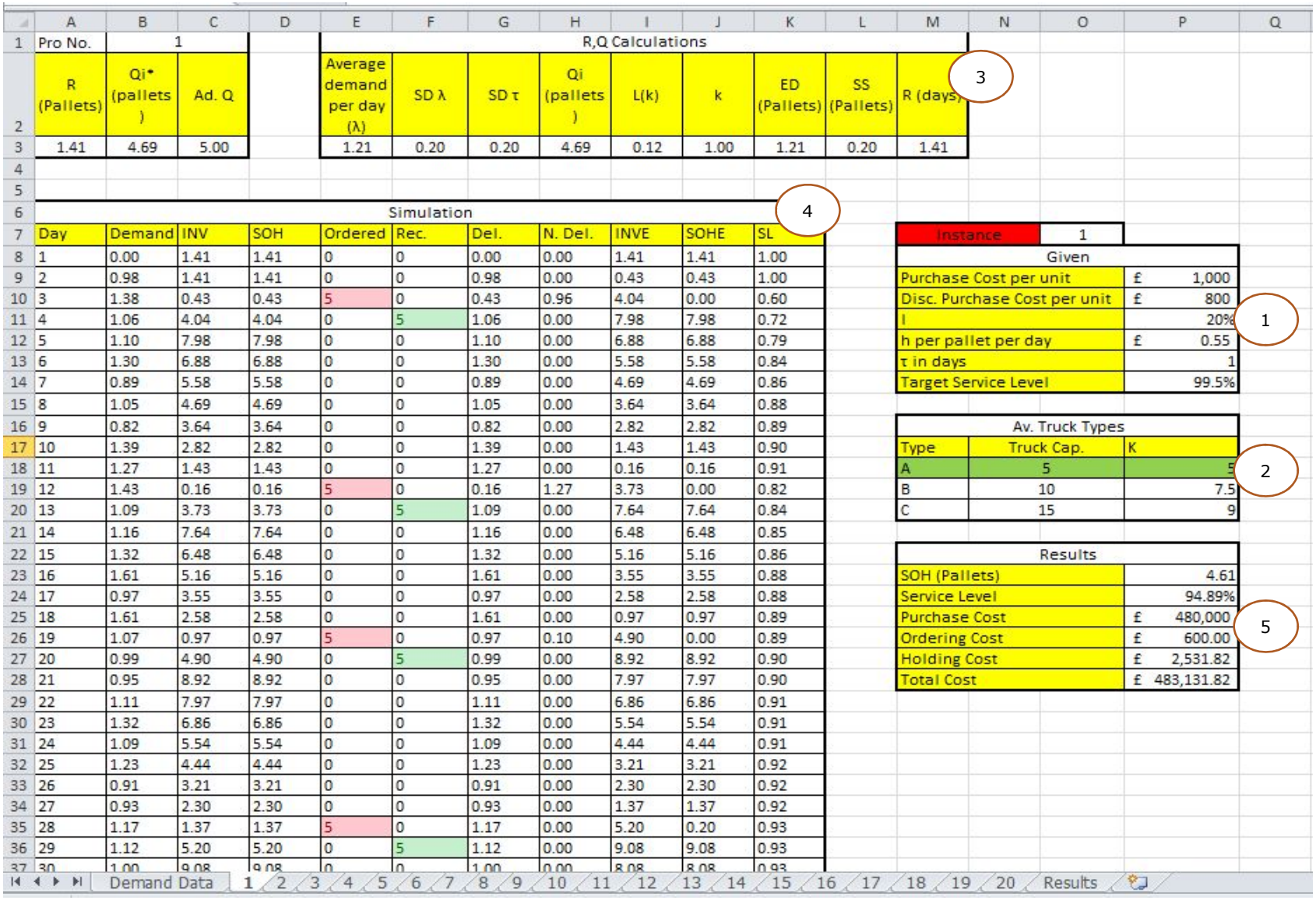
$$\text{Total Ordering Cost} = \text{number of orders} \times \text{K}$$

$$\text{Total Purchse Cost} = \text{Discounted purchase price per pallet} \times \sum \text{Q}$$

Table 17 below summarises the Excel formulae used in determining the performance parameters:

Table 17:EOQ simulation performance parametres in Excel

Parameter	Excel Formula	Cell
Average SOH	=AVERAGE(D8:D1007)	P23
Average Service Level	=AVERAGE(K8:K1007)	Q23
Purchase cost	=SUM(E8:E1007)*P10	R23
Ordering cost	=COUNTIF(E8:E1007,C3)*P18	S23
Holding cost	=SUM(D8:D1007)*P12	T23
Total cost	=SUM(P25:P27)	U23



Pro No.	1			R,Q Calculations									
R (Pallets)	Qi* (pallets)	Ad. Q		Average demand per day (λ)	SD λ	SD τ	Qi (pallets)	L(k)	k	ED (Pallets)	SS (Pallets)	R (days)	
1.41	4.69	5.00		1.21	0.20	0.20	4.69	0.12	1.00	1.21	0.20	1.41	

Simulation

Day	Demand	INV	SOH	Ordered	Rec.	Del.	N. Del.	INVE	SOHE	SL
1	0.00	1.41	1.41	0	0	0.00	0.00	1.41	1.41	1.00
2	0.98	1.41	1.41	0	0	0.98	0.00	0.43	0.43	1.00
3	1.38	0.43	0.43	5	0	0.43	0.96	4.04	0.00	0.60
4	1.06	4.04	4.04	0	5	1.06	0.00	7.98	7.98	0.72
5	1.10	7.98	7.98	0	0	1.10	0.00	6.88	6.88	0.79
6	1.30	6.88	6.88	0	0	1.30	0.00	5.58	5.58	0.84
7	0.89	5.58	5.58	0	0	0.89	0.00	4.69	4.69	0.86
8	1.05	4.69	4.69	0	0	1.05	0.00	3.64	3.64	0.88
9	0.82	3.64	3.64	0	0	0.82	0.00	2.82	2.82	0.89
10	1.39	2.82	2.82	0	0	1.39	0.00	1.43	1.43	0.90
11	1.27	1.43	1.43	0	0	1.27	0.00	0.16	0.16	0.91
12	1.43	0.16	0.16	5	0	0.16	1.27	3.73	0.00	0.82
13	1.09	3.73	3.73	0	5	1.09	0.00	7.64	7.64	0.84
14	1.16	7.64	7.64	0	0	1.16	0.00	6.48	6.48	0.85
15	1.32	6.48	6.48	0	0	1.32	0.00	5.16	5.16	0.86
16	1.61	5.16	5.16	0	0	1.61	0.00	3.55	3.55	0.88
17	0.97	3.55	3.55	0	0	0.97	0.00	2.58	2.58	0.88
18	1.61	2.58	2.58	0	0	1.61	0.00	0.97	0.97	0.89
19	1.07	0.97	0.97	5	0	0.97	0.10	4.90	0.00	0.89
20	0.99	4.90	4.90	0	5	0.99	0.00	8.92	8.92	0.90
21	0.95	8.92	8.92	0	0	0.95	0.00	7.97	7.97	0.90
22	1.11	7.97	7.97	0	0	1.11	0.00	6.86	6.86	0.91
23	1.32	6.86	6.86	0	0	1.32	0.00	5.54	5.54	0.91
24	1.09	5.54	5.54	0	0	1.09	0.00	4.44	4.44	0.91
25	1.23	4.44	4.44	0	0	1.23	0.00	3.21	3.21	0.92
26	0.91	3.21	3.21	0	0	0.91	0.00	2.30	2.30	0.92
27	0.93	2.30	2.30	0	0	0.93	0.00	1.37	1.37	0.92
28	1.17	1.37	1.37	5	0	1.17	0.00	5.20	0.20	0.93
29	1.12	5.20	5.20	0	5	1.12	0.00	9.08	9.08	0.93
30	1.00	9.08	9.08	0	0	1.00	0.00	8.08	8.08	0.93

Instance	1
Given	
Purchase Cost per unit	£ 1,000
Disc. Purchase Cost per unit	£ 800
I	20%
h per pallet per day	£ 0.55
τ in days	1
Target Service Level	99.5%

Av. Truck Types		
Type	Truck Cap.	K
A	5	5
B	10	7.5
C	15	9

Results	
SOH (Pallets)	4.61
Service Level	94.89%
Purchase Cost	£ 480,000
Ordering Cost	£ 600.00
Holding Cost	£ 2,531.82
Total Cost	£ 483,131.82

Figure 25: R& Q calculations and the EOQ simulation (Experiment 1 – product 1)

3.8.3.3 The Overall Simulation Results

In terms of the overall EOQ simulation results, a spreadsheet is generated for each experiment. Figure 26 below presents the Overall Results Spreadsheet for Experiment 1.

	A	B	
1	Results		
2	Average SOH (pallets/day)		3.84
3	Average Service Level		99.51%
4	Purchase Cost	£	16,020,000.00
5	Total Holding Cost	£	20,005.00
6	Total Ordering Cost	£	42,237.64
7	Total Cost	£	16,082,242.64
8			

Figure 26: Overall Results Spreadsheet - Experiment 1

The average SOH for entire simulation is calculated as:

$$\sum_{n=1}^{20}\sum_{day=1}^{1000} SOH_{i,day} \Big/ 20$$

The average service level is calculated as:

$$\sum_{n=1}^{20}\sum_{day=1}^{1000} SLEVEL_{i,day} \Big/ 20$$

The Total Purchase Cost is calculated as follows:

$$\text{Discounted purchase price per pallet} \times \sum_{n=1}^{20}\sum_{day=1}^{1000} Q_{i,day}$$

The Total Holding Cost is calculated as follows:

$$\sum_{n}^{20} \sum_{day=1}^{1000} h \times SOH_{i,day}$$

The Total Ordering Cost is calculated as follows:

$$\sum_{n}^{20} \sum_{day=1}^{1000} number\ of\ orders\ \times K_j$$

The total cost is given by the sum of the three previous equations. Table 18 presents the formulae used in MS Excel to compute the performance measures shown in Figure 26.

Table 18: Performance Measures Formulas in Excel

Parameter	Excel Formula	Cell
Average SOH	=('1'!P23+'2'!P23+'3'!P23+'4'!P23+'5'!P23+'6'!P23+'7'!P23+'8'!P23+'9'!P23+'10'!P23+'11'!P23+'12'!P23+'13'!P23+'14'!P23+'15'!P23+'16'!P23+'17'!P23+'18'!P23+'19'!P23+'20'!P23)/20	B2
Average Service Level.	=('1'!P24+'2'!P24+'3'!P24+'4'!P24+'5'!P24+'6'!P24+'7'!P24+'8'!P24+'9'!P24+'10'!P24+'11'!P24+'12'!P24+'13'!P24+'14'!P24+'15'!P24+'16'!P24+'17'!P24+'18'!P24+'19'!P24+'20'!P24)/20	B4
Purchase Cost	=('1'!P25+'2'!P25+'3'!P25+'4'!P25+'5'!P25+'6'!P25+'7'!P25+'8'!P25+'9'!P25+'10'!P25+'11'!P25+'12'!P25+'13'!P25+'14'!P25+'15'!P25+'16'!P25+'17'!P25+'18'!P25+'19'!P25+'20'!P25)+16000	B5
Ordering Cost	=('1'!P26+'2'!P26+'3'!P26+'4'!P26+'5'!P26+'6'!P26+'7'!P26+'8'!P26+'9'!P26+'10'!P26+'11'!P26+'12'!P26+'13'!P26+'14'!P26+'15'!P26+'16'!P26+'17'!P26+'18'!P26+'19'!P26+'20'!P26)	B6
Holding Cost	=('1'!P27+'2'!P27+'3'!P27+'4'!P27+'5'!P27+'6'!P27+'7'!P27+'8'!P27+'9'!P27+'10'!P27+'11'!P27+'12'!P27+'13'!P27+'14'!P27+'15'!P27+'16'!P27+'17'!P27+'18'!P27+'19'!P27+'20'!P27)	B7
Total Cost	=SUM(B4:B6)	B8

CHAPTER 4. Results

The fourth chapter presents the results obtained from the simulation of the joint replenishment model and the adjusted EOQ model. First, the performance measures are explained. Second the simulation results for each experiment are presented.

4.1 Performance Measures

In this section the three performance measures for the analysis are briefly reviewed:

1. **Average Inventory Level**. The average inventory level plays a vital role in determining the overall cost of the inventory control process. With low average inventory level the service rate is low. On the other hand, with high average inventory levels the inventory holding cost is high.

2. **Service Level**. The service level defines the satisfaction of the customers. A low service level due to low inventory levels will impact the company on the long run due to lost sales and unsatisfied customers.

3. **Total Inventory Cost**. This is one of the most reliable and important performance measures. Managers aim at keeping this value as low as possible through the implementation of efficient inventory control systems that minimise the average inventory level while maximising the overall service level and customer satisfaction. The total inventory cost is composed of the total purchase cost, the total inventory holding cost and the total ordering cost.

4.2 Results

4.2.1 Experiment 1: Similar Demand – Low Ordering Cost - Small Truck Capacity

1. **Joint Replenishment Grouping Results**

GROUP A						
Product ID	Average Daily Demand	Group Demand	Truck Capacity	K	Original T	Rounded T
1	1.21	5	10	7.5	2	2
5	1.36					
6	0.06					
11	1.57					
13	0.02					
19	0.78					

GROUP B						
Product ID	Average Daily Demand	Group Demand	Truck Capacity	K	Original T	Rounded T
2	1.78	15	15	9	1	1
3	0.76					
4	0.70					
7	0.85					
8	1.46					
9	0.24					
10	1.03					
12	0.66					
14	0.77					
15	2.71					
16	0.56					
17	1.25					

18	0.40					
20	1.84					

2. <u>Joint Replenishment Simulation Results</u>

Performance Measure	Result
Average Inventory Level (Pallets/ day)	6.23
Average Service Level	99.81%
Total Purchase Cost	£16,000,000.00
Total Holding Cost	£68,471.08
Total Ordering Cost	£8,250.00
Total Cost	£16,076,721.08

3. <u>Adjusted EOQ Simulation Results</u>

Performance Measure	Result
Average Inventory Level (Pallets/ day)	3.84
Average Service Level	99.51%
Total Purchase Cost	£16,020,000.00
Total Holding Cost	£ 20,005.00
Total Ordering Cost	£ 42,237.64
Total Cost	£16,082,242.64

4.2.2 Experiment 2: Similar Demand – High Ordering Cost - Small Truck Capacity

1. **Joint Replenishment Grouping Results**

GROUP A						
Product ID	Average Daily Demand	Group Demand	Truck Capacity	K	Original T	Rounded T
2	1.78	5.32	15	99	2.82	3
3	0.76					
9	0.24					
13	0.02					
14	0.77					
16	0.56					
18	0.40					
19	0.78					

GROUP B						
Product ID	Average Daily Demand	Group Demand	Truck Capacity	K	Original T	Rounded T
1	1.21	14.68	15	99	1.02	1
4	0.70					
5	1.36					
6	0.06					
7	0.85					
8	1.46					
10	1.03					
11	1.57					
12	0.66					
15	2.71					
17	1.25					

20	1.84					

2. <u>Joint Replenishment Simulation</u>

Performance Measure	Result
Average Inventory Level (Pallets/ day)	14.13
Average Service Level	96.92%
Total Purchase Cost	£ 15,996,000.00
Total Holding Cost	£ 155,282.72
Total Ordering Cost	£ 131,967.00
Total Cost	£16,283,249.72

3. <u>Adjusted EOQ Simulation Results</u>

Performance Measure	Result
Average Inventory Level (Pallets/ day)	7.98
Average Service Level	99.55%
Total Purchase Cost	£ 16,073,000.00
Total Holding Cost	£ 87,641.42
Total Ordering Cost	£ 133,732.50
Total Cost	£ 16,294,373.92

4.2.3 Experiment 3: Similar Demand – Low Ordering Cost - Large Truck Capacity

1. **Joint Replenishment Grouping Results**

GROUP A						
Product ID	Average Daily Demand	Group Demand	Truck Capacity	K	Original T	Rounded T
1	1.21	20	20	7.5	1	1
2	1.78					
3	0.76					
4	0.70					
5	1.36					
6	0.06					
7	0.85					
8	1.46					
9	0.24					
10	1.03					
11	1.57					
12	0.66					
13	0.02					
14	0.77					
15	2.71					
16	0.56					
17	1.25					
18	0.40					
19	0.78					
20	1.84					

2. Joint Replenishment Simulation

Performance Measure	Result
Average Inventory Level (Pallets/ day)	5.82
Average Service Level	99.73%
Total Purchase Cost	£16,000,000.00
Total Holding Cost	£63,987.54
Total Ordering Cost	£7,500.00
Total Cost	£16,071,487.54

3. Adjusted EOQ Simulation

Performance Measure	Result
Average Inventory Level (Pallets/ day)	6.26
Average Service Level	99.55%
Total Purchase Cost	£16,054,000.00
Total Holding Cost	£10,030.00
Total Ordering Cost	£68,831.31
Total Cost	£16,132,861.31

4.2.4 Experiment 4: Similar Demand – High Ordering Cost - Large Truck Capacity

1. **Joint Replenishment Grouping Results**

GROUP A						
Product ID	Average Daily Demand	Group Demand	Truck Capacity	K	Original T	Rounded T
1	1.21	20	20 (originally 30)	82.5	1	1
2	1.78					
3	0.76					
4	0.70					
5	1.36					
6	0.06					
7	0.85					
8	1.46					
9	0.24					
10	1.03					
11	1.57					
12	0.66					
13	0.02					
14	0.77					
15	2.71					
16	0.56					
17	1.25					
18	0.40					
19	0.78					
20	1.84					

2. <u>Joint Replenishment Simulation</u>

Performance Measure	Result
Average Inventory Level (Pallets/ day)	6.29
Average Service Level	99.78%
Total Purchase Cost	£ 16,000,000.00
Total Holding Cost	£69,107.18
Total Ordering Cost	£82,500.00
Total Cost	£16,151,607.18

3. <u>Adjusted EOQ Simulation</u>

Performance Measure	Result
Average Inventory Level (Pallets/ day)	8.70
Average Service Level	99.46%
Total Purchase Cost	£ 16,094,000.00
Total Holding Cost	£ 95,552.48
Total Ordering Cost	£ 88,759.00
Total Cost	£16,278,311.48

4.2.5 Experiment 5: Type 2 Demand – Low Ordering Cost - Small Truck Capacity

1. **Joint Replenishment Grouping Results**

GROUP A						
Product ID	Average Daily Demand	Group Demand	Truck Capacity	K	Original T	Rounded T
2	0.09	5	5	5	1	1
5	4.81					
7	0.09					

GROUP A						
Product ID	Average Daily Demand	Group Demand	Truck Capacity	K	Original T	Rounded T
1	0.28	15	15	9	1	1
3	0.38					
4	5.09					
6	6.60					
8	0.28					
9	0.47					
10	0.09					
11	0.19					
12	0.09					
13	0.09					
14	0.09					
15	0.19					
16	0.09					
17	0.19					
18	0.19					
19	0.38					
20	0.28					

2. Joint Replenishment Simulation Results

Performance Measure	Result
Average Inventory Level (Pallets/ day)	16.34
Average Service Level	99.61%
Total Purchase Cost	£ 16,000,000.00
Total Holding Cost	£ 151,456.61
Total Ordering Cost	£ 14,000.00
Total Cost	£ 16,165,456.61

3. Adjusted EOQ Simulation

Performance Measure	Result
Average Inventory Level (Pallets/ day)	4.30
Average Service Level	99.55%
Total Purchase Cost	£ 15,972,000.00
Total Holding Cost	£ 47,300.09
Total Ordering Cost	£ 14,860.00
Total Cost	£16,034,160.09

4.2.6 Experiment 6: Type 2 Demand – High Ordering Cost - Small Truck Capacity

1. **Joint Replenishment Grouping Results**

GROUP A						
Product ID	Average Daily Demand	Group Demand	Truck Capacity	K	Original T	Rounded T
2	0.09	13.21	15	99	1.14	1
3	0.38					
4	5.09					
6	6.60					
7	0.09					
9	0.47					
10	0.09					
12	0.09					
13	0.09					
14	0.09					
16	0.09					

GROUP B						
Product ID	Average Daily Demand	Group Demand	Truck Capacity	K	Original T	Rounded T
1	0.28	6.79	15	99	2.21	2
5	4.81					
8	0.28					
11	0.19					
15	0.19					
17	0.19					
18	0.19					
19	0.38					

20	0.28					

2. <u>Joint Replenishment Simulation</u>

Performance Measure	Result
Average Inventory Level (Pallets/ day)	26.31
Average Service Level	88.35%
Total Purchase Cost	£ 15,996,000.00
Total Holding Cost	£ 598,881.65
Total Ordering Cost	£ 131,967.00
Total Cost	£ 16,726,848.65

3. <u>Adjusted EOQ Simulation</u>

Performance Measure	Result
Average Inventory Level (Pallets/ day)	5.61
Average Service Level	99.55%
Total Purchase Cost	£ 16,015,000.00
Total Holding Cost	£ 61,612.11
Total Ordering Cost	£ 140,657.00
Total Cost	£16,217,269.11

4.2.7 Experiment 7: Type 2 Demand – Low Ordering Cost - Large Truck Capacity

1. **Joint Replenishment Grouping Results**

GROUP A						
Product ID	Average Daily Demand	Group Demand	Truck Capacity	K	Original T	Rounded T
1	0.28	20	20	7.5	1	1
2	0.09					
3	0.38					
4	5.09					
5	4.81					
6	6.60					
7	0.09					
8	0.28					
9	0.47					
10	0.09					
11	0.19					
12	0.09					
13	0.09					
14	0.09					
15	0.19					
16	0.09					
17	0.19					
18	0.19					
19	0.38					
20	0.28					

2. Joint Replenishment Simulation

Performance Measure	Result
Average Inventory Level (Pallets/ day)	14.67
Average Service Level	98.44%
Total Purchase Cost	£16,000,000.00
Total Holding Cost	£161,183.43
Total Ordering Cost	£ 7,500.00
Total Cost	£16,168,683.43

3. Adjusted EOQ Simulation

Performance Measure	Result
Average Inventory Level (Pallets/ day)	8.07
Average Service Level	99.48%
Total Purchase Cost	£ 16,080,000.00
Total Holding Cost	£88,687.32
Total Ordering Cost	£7,955.00
Total Cost	£16,176,642.32

4.2.8 Experiment 8: Type 2 Demand – High Ordering Cost - Large Truck Capacity

1. <u>Joint Replenishment Grouping Results</u>

GROUP A						
Product ID	Average Daily Demand	Group Demand	Truck Capacity	K	Original T	Rounded T
1	0.28	20	20	82.5	1	1
2	0.09					
3	0.38					
4	5.09					
5	4.81					
6	6.60					
7	0.09					
8	0.28					
9	0.47					
10	0.09					
11	0.19					
12	0.09					
13	0.09					
14	0.09					
15	0.19					
16	0.09					
17	0.19					
18	0.19					
19	0.38					
20	0.28					

2. Joint Replenishment Simulation

Performance Measure	Result
Average Inventory Level (Pallets/ day)	13.05
Average Service Level	99.51%
Total Purchase Cost	£ 16,000,000.00
Total Holding Cost	£ 143,461.28
Total Ordering Cost	£ 82,500.00
Total Cost	£ 16,225,961.28

3. Adjusted EOQ Simulation

Performance Measure	Result
Average Inventory Level (Pallets/ day)	7.79
Average Service Level	99.46%
Total Purchase Cost	£ 16,068,000.00
Total Holding Cost	£85,621.34
Total Ordering Cost	£ 74,030.00
Total Cost	£ 16,227,651.34

CHAPTER 5. Analysis & Discussion

In the course of this project, the principal aim of developing a new heuristic to solve the JRP with quantity discounts and similar purchase costs has been achieved. As discussed in Chapter Two, no notable solutions to this problem had been offered in the research literature to date. Moreover, most of the previous research has ignored the fixed cost of sending a truck and the optimal solution from a truck perspective as well as the significant cost savings that can be achieved by considering this cost.

Furthermore, the project was successful in achieving its set of objectives. First, a joint replenishment grouping heuristic was successfully developed and its efficiency demonstrated in most of the conducted experiments. Second, the joint replenishment model outperformed the adjusted EOQ model in most of the experiments, both in terms of the overall cost and the service level. In this section, a detailed analysis is provided of the results obtained from the simulation. Second, a general evaluation is undertaken of these results and a set of recommendations drawn up and presented. Finally, suggestions regarding how the proposed solution can be used in more complex environments are presented.

5.1 Analysis

5.1.1 Type 1 Demand Experiments: Similar Demand for all Products

In order to analyse the performance of the grouping heuristic under developed in this project, a G(T) vs. average demand rate graph was produced for the four sets of Ks and Qs used for the eight experiments conducted. As mentioned earlier in Chapter Three, this graph describes the relationship between the average demand rate and the total cost when the capacities of the different trucks (and, therefore, Q) are fixed. Figure 27 illustrates the best grouping solution for Experiment 1. One Type 1 truck is denoted as 1T1, while two Type 1 trucks are denoted as 2T1, and so on. For any truck type, the cost function line can be seen to stop when the demand rate is equal to the capacity of the truck or the capacity of a number of trucks of the same type (i.e. when the

replenishment period (T) = 1). For example, since the capacity of the smallest truck in Experiment 1 is 5, a demand rate greater than 5 Pallets/Day will require using another truck of the same type (i.e. 2T1). Furthermore, the best grouping solution is obtained graphically by considering the cost lines of using 1 truck of each type (i.e. 1T1, 1T2, and 1T3). The rest of the graph is useful in indicating when an additional truck of the same type is needed.

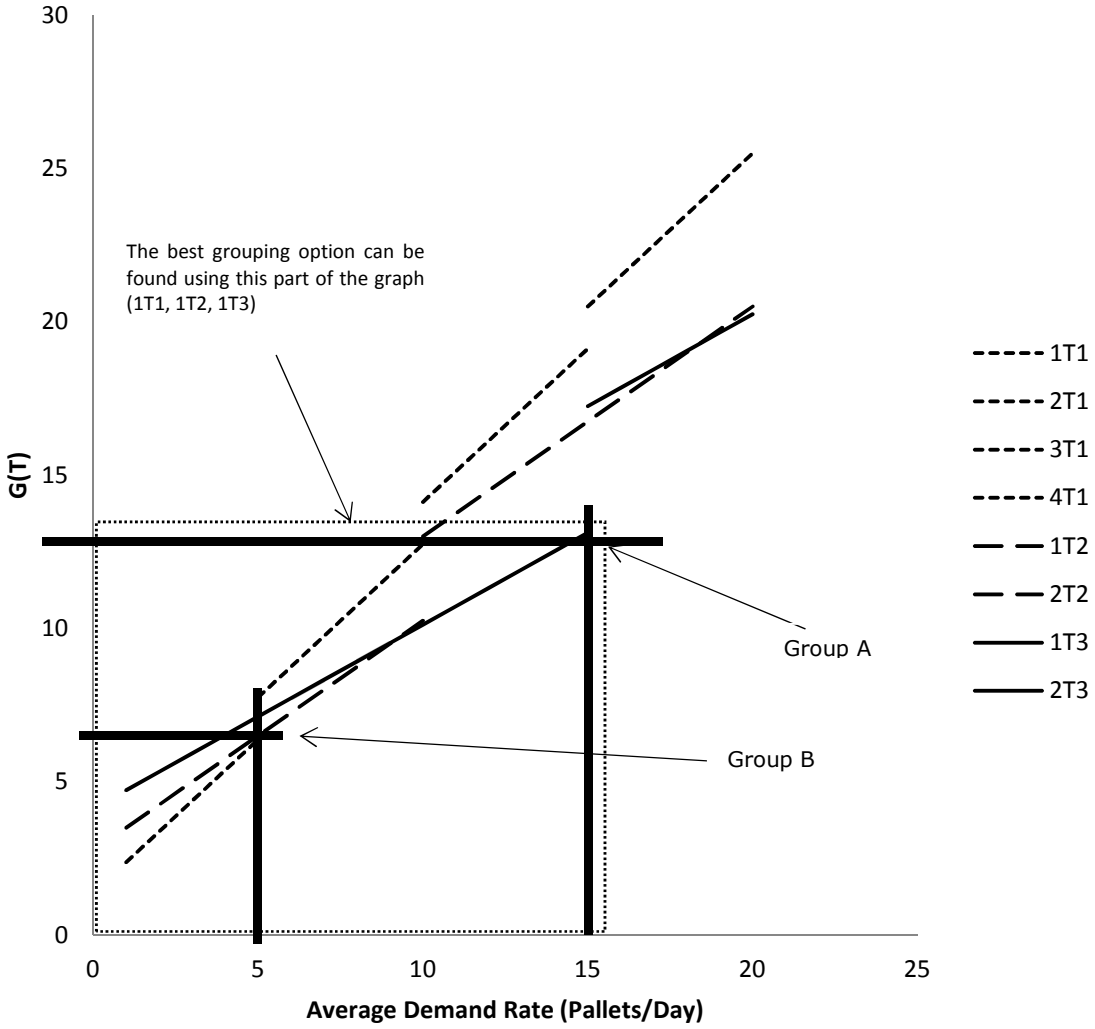


Figure 27: G(T) behaviour with fixed Qs and the best grouping solution

For Experiments 1 and 5

Inspecting Figure 27 above and applying the grouping solution procedure described in Section 3.6.1.4, the best grouping solution for Experiment 1 - in which we examined a situation where the trucks are small and the fixed cost is low- is to send two replenishments per day. The first replenishment would use a 15-pallet truck while the other would use a 5-pallet one. However, given the

individual demand rates of the 20 products set out in Experiment 1, having a group of products that generates a daily demand rate of 5 Pallets/Day and another group of products that generates a demand rate of exactly 15 Pallets/Day is impossible. This is especially the case in light of the assumption that every item must be replenished in a single replenishment in order to get the quantity discount. The model thus produced an alternative grouping solution in which Group A produces a daily demand rate of 5.00006 Pallets/Day and Group B produces a daily demand rate of 14.9994 Pallets/Day. Recall that in the original model, the following constraint is used to avoid $T < 1\,day$ for any replenishment:

$$\sum_{i=1}^{n} \lambda_i \, x_{i,j} \leq Q_j \quad \forall \; replenishment\; j$$

As a result of this constraint, the grouping model failed to recognise that the difference between 5.00006 Pallets/Day and 5 Pallets/Day is an insignificant one in practical terms, and consequently deemed the demand rate of 5.00006 Pallets/Day as unfeasible for a replenishment that uses a truck with a 5-pallet capacity. The model instead used the next best option, which involves using a 15-pallet truck every day and a 10-pallet truck every two days. The impact of this deviation from the best solution is calculated as follows:

$$\frac{G(T_1)}{G(T_2)} = \frac{0.5 \times h \times \lambda \times T_1 + K_1/T}{0.5 \times h \times \lambda \times T_2 + K_2/T} = 0.98$$

Clearly, the impact of the deviation on the cost is insignificant. Nevertheless, performing the joint replenishment simulation using the best grouping option is still possible, and can be done by using the groups identified by LINGO in the simulation. However, instead of using a 10-pallet truck every two days, a 5-pallet truck is used daily. Given a demand rate of 5.00006 Pallets/Day for this group, T can be rounded from 0.9999 days to 1 day without significantly affecting the solution.

The original grouping solution obtained by the LINGO model was used to perform the joint replenishment simulation for this experiment over the 1000-

day period. Looking at Figures 31 to 34, a number of observations can be discerned. For instance, the joint replenishment model results in higher average inventory levels - approximately double the average inventory levels obtained by the adjusted EOQ model. As a result, the total holding cost is also higher in the joint replenishment simulation. As mentioned in the Literature Review, this greater total inventory cost is one of the major drawbacks of (T, S) systems. However, looking at the other performance measures, the joint replenishment system outperformed the EOQ model when solving the small-trucks-low-fixed cost scenario. The service level is slightly higher and the overall cost is £7000 cheaper than that of the adjusted EOQ system.

Furthermore, Figure 28 shows that for Experiment 2, in which we examined a situation where the trucks are large and the fixed cost per order is low, the best grouping solution is to use a 15-pallet truck every day for one group and a 15-pallet truck every three days for the other. The proposed grouping model was successful in identifying these two replenishment options. However, as in the previous experiment, given that individual demand rates for the 20 products in Experiment 2 formed two groups - with a demand rate of 5 Pallets/Day for the first replenishment and 15 Pallets/Day for the second – this was not deemed possible. As a result, the model suggests alternative combinations in which Groups A and B boast average demand rates of 5.32 and 14.68 Pallets/Day, respectively. Group A is thus replenished every 2.82 days using a 15-pallet truck, while Group B is replenished every 1.02 days, also using a 15-pallet truck.

However, since an integer number of days was assumed in the experiment, the T values for Groups A and B were rounded to 3 days and 1 day, respectively. For Group A, this rounding-up resulted in a daily unsatisfied demand of 0.32 pallets. As a result, the overall service level for this group was as low as 92%. Meanwhile, rounding up the value of T for Group B, from 1.02 to 1, resulted in an extra quantity of 0.32 Pallets/Day to be received. The group’s average inventory level was consequently up to 20 Pallets/Day, with a service level of 99.9%. Clearly, the rounded values of T created imbalances between the two groups, and the overall performance of this experiment has been affected as a result.

Looking at Figures 31 to 35, a number of observations can be made. Although the overall cost of the joint replenishment system for this experiment is slightly lower than the overall cost of the EOQ simulation, the overall results are not satisfactory. This is because the savings in the total ordering cost as well as those to the total purchase are mainly attributable to the low service level of 97%. Nevertheless, the joint replenishment results of this experiment can be easily improved upon by, for example, moving Product 9 and Product 13 from Group A to Group B. In doing so, the average daily demand for Groups A and B is 5.06 and 14.94, producing T values of 2.69 and 1.004, respectively. As such, the rounding up process has a lesser impact on the overall performance. Another solution to this problem is to assume that the 0.32 Pallet/Day demand rate is produced by a separate product (take 0.32 Pallets/Day from Product 2, for example), and move this product to Group B. Thus, we can use the best grouping solution for this experiment and improve the overall performance in the process.

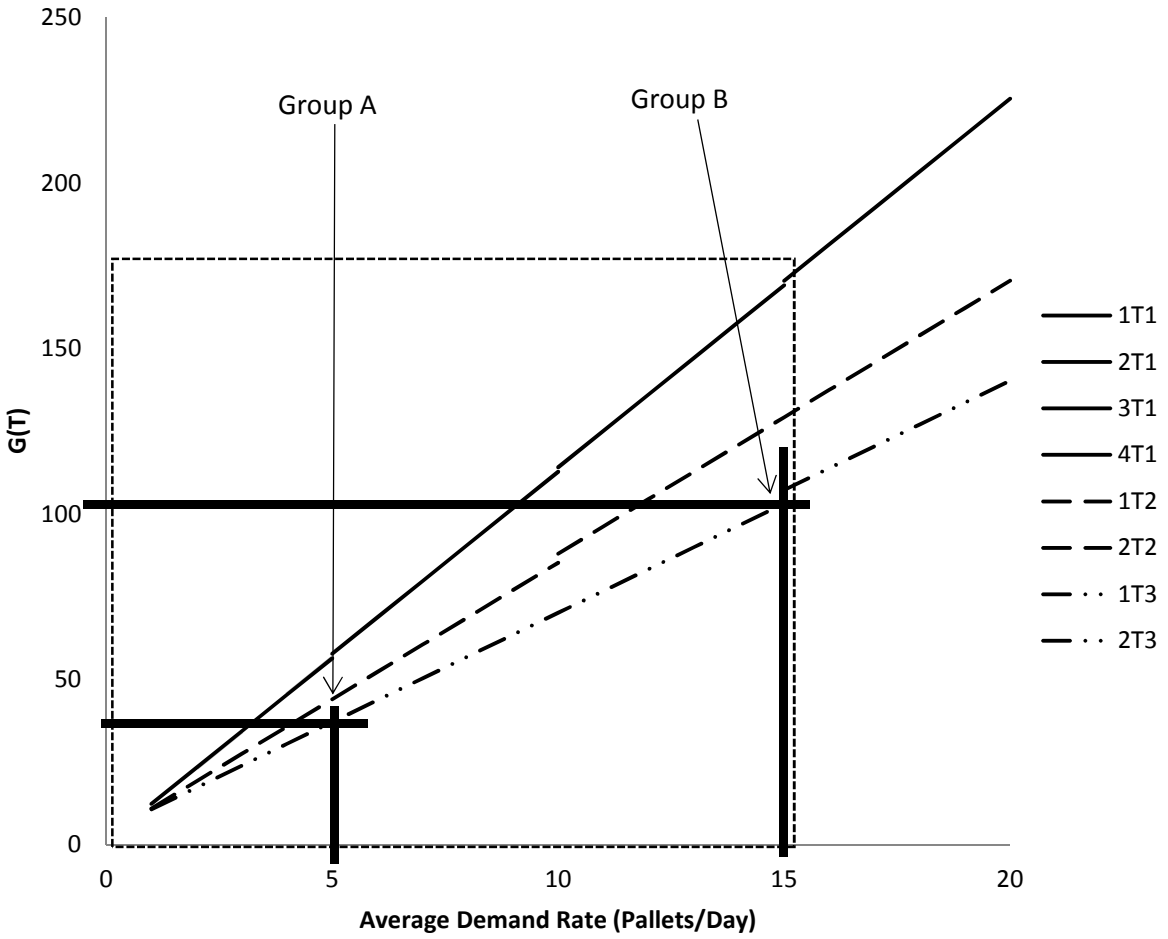


Figure 28: G(T) behaviour with fixed Qs and the best grouping solution

For Experiments 2 and 6

The third experiment involved a high fixed cost per order and a small truck size. Figure 29 shows that the best grouping option is to replenish a quantity of 20 pallets daily using a 20-pallet truck, and the LINGO model was successful in identifying this solution. Further, the proposed joint replenishment solution proved satisfactory when compared to the adjusted EOQ solution. Figures 31 to 35 show how the Joint replenishment system outperformed the adjusted EOQ system across all the performance measures.

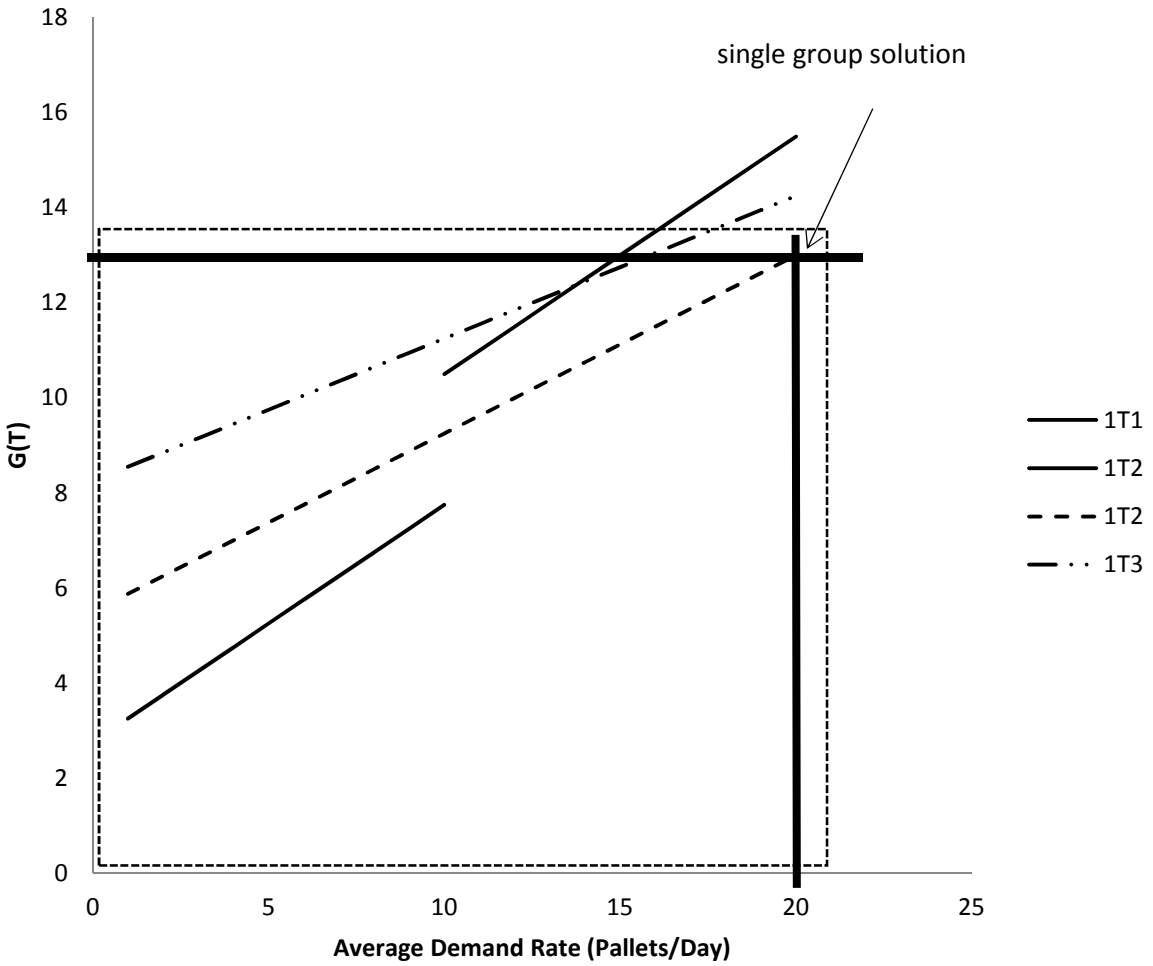


Figure 29: G(T) behaviour with fixed Qs and the best grouping solution

For Experiments 3 and 7

For Experiment 4, meanwhile, a large truck capacity and a high ordering cost are assumed. The best replenishment in this case would be to use a Type 3 truck every 1.5 days. Although the model was successful in identifying this solution; it was not used in the simulation, due to the assumption of T being an integer number. As such, using a replenishment period of 1 day will result in a very low service level, while rounding T to 2 days will result in excess inventory levels. The second best option was thus adopted, which is to use a 20-pallet truck every day. Nonetheless, the overall performance of the joint replenishment simulation outperformed the adjusted EOQ simulation results. The service level, inventory holding cost, average inventory level, and total purchase cost of the joint replenishment simulation were all improved, as shown in Figures 31 to 35. Most

importantly, the overall cost when the joint replenishment system was used is £127,000 less than the corresponding overall cost under the adjusted EOQ.

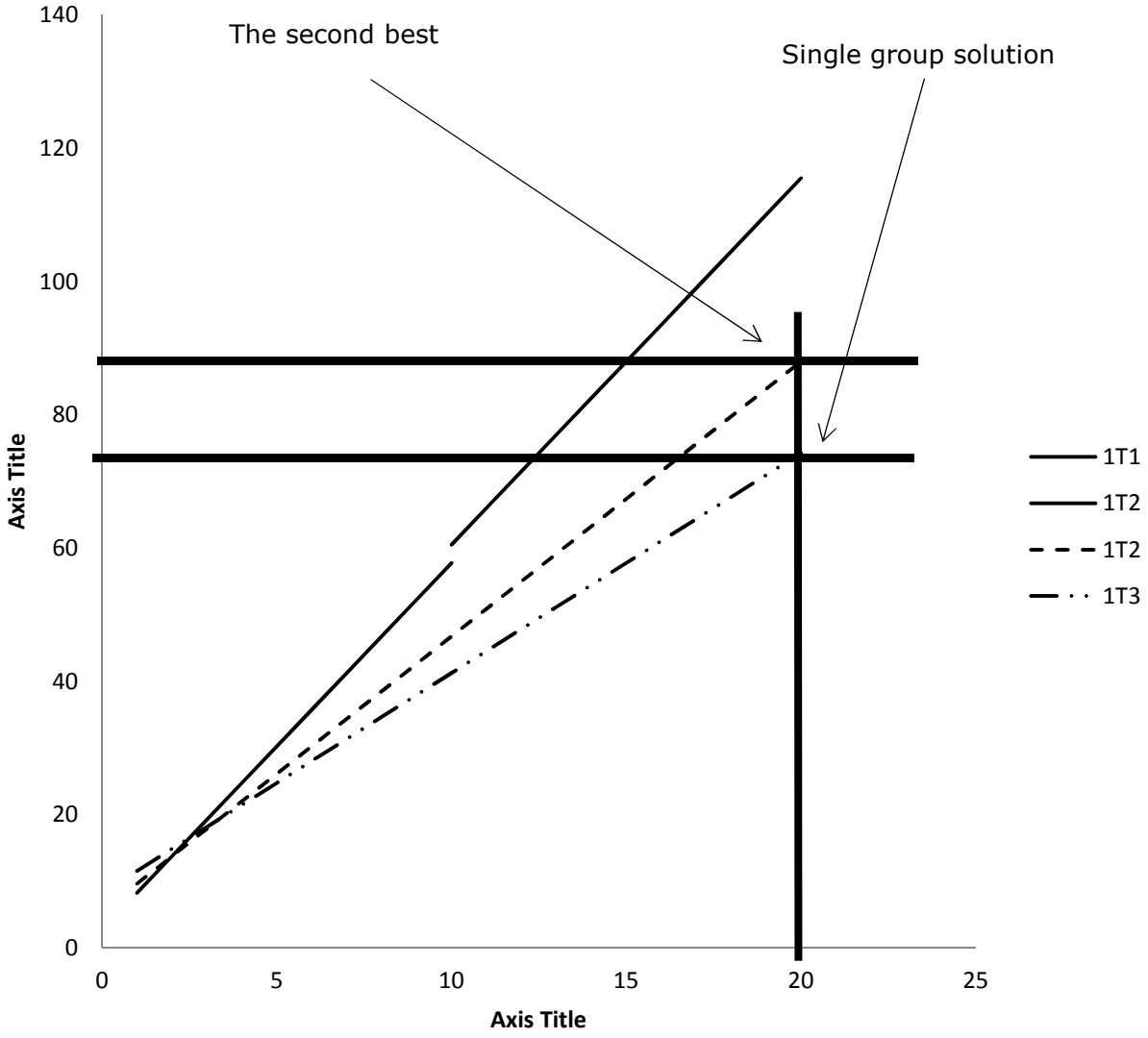


Figure 30: G(T) behaviour with fixed Qs and the best grouping solution in Experiments 4 and 8

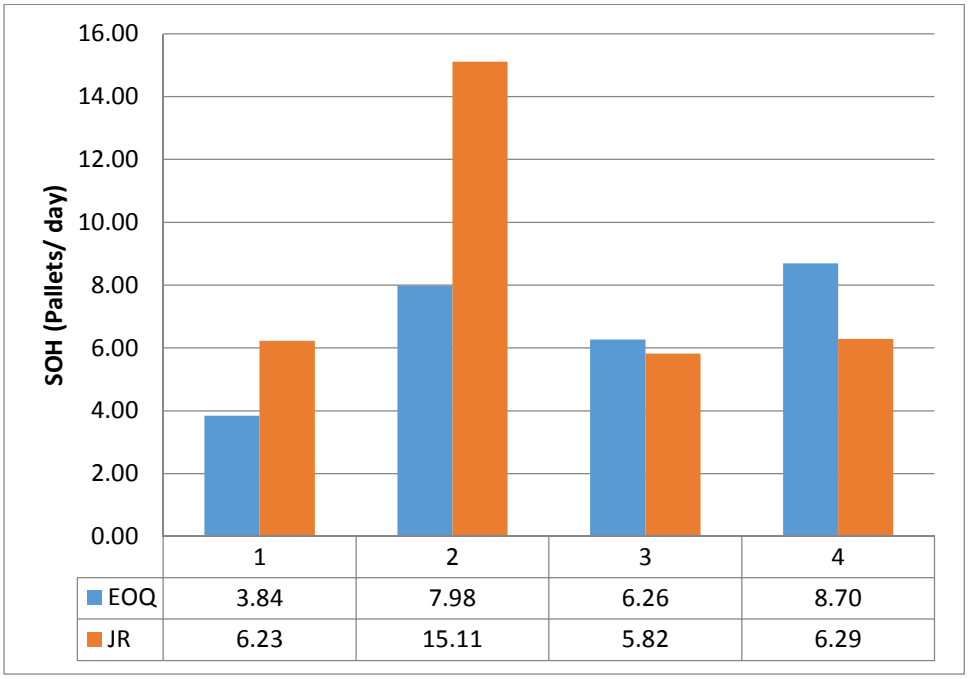


Figure 31: Average Stock on Hand in Experiments 1-4

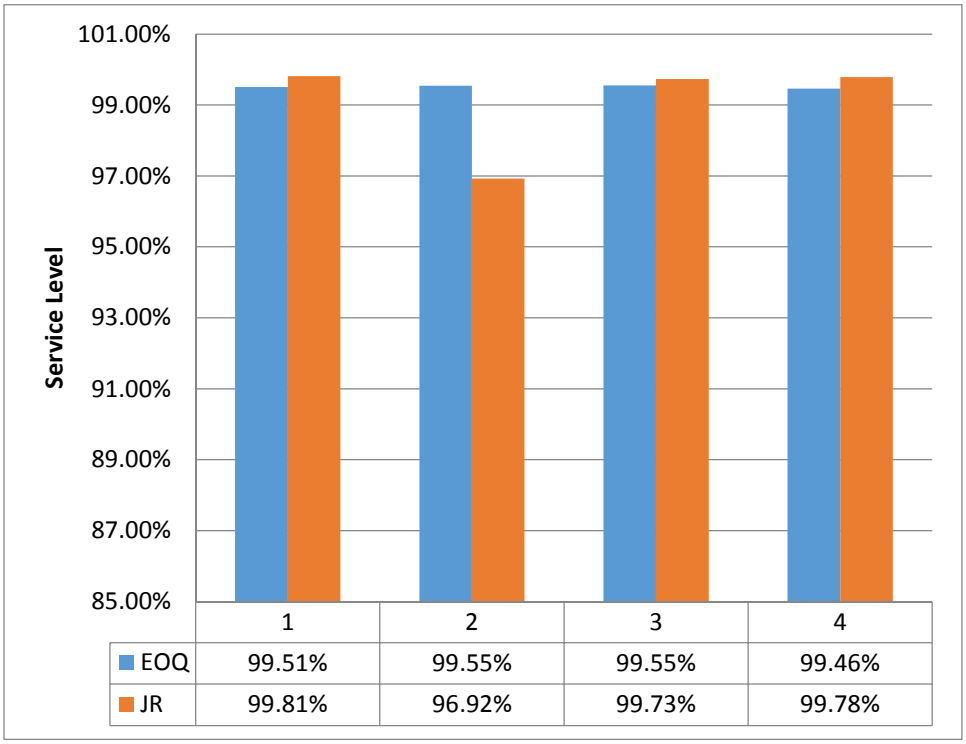


Figure 32: Average Service levels in Experiments 1-4

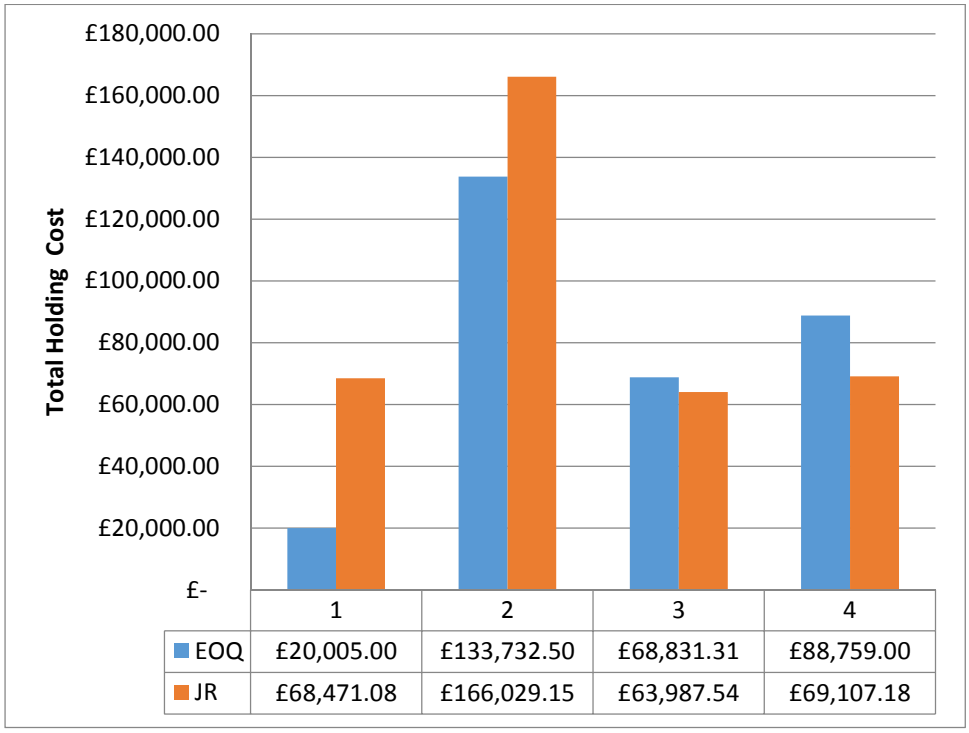


Figure 33: Total holding costs in Experiments 1 to 4

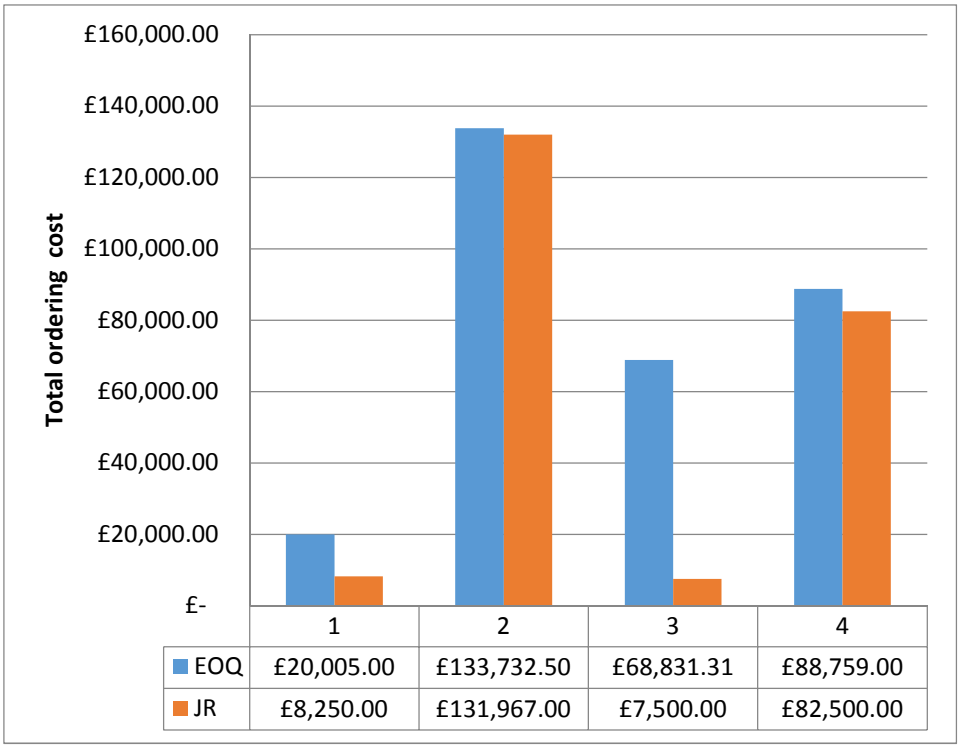


Figure 34: Total ordering costs in Experiments 1 to 4

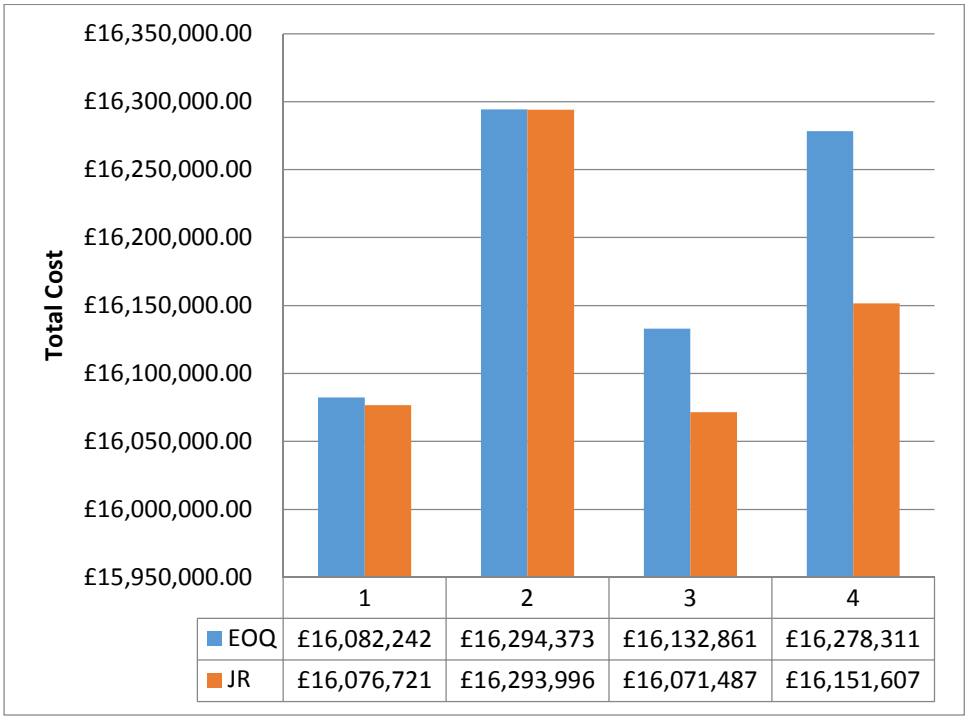


Figure 35: Overall costs in Experiments 1 to 4

5.1.2 Type 2 Demand Experiments: Very low demand for most of the products and high demand for the remaining few products

Since the best grouping solution does not depend on the individual demand rates of the 20 products, Figures 27, 28, 29 and 30 remain applicable for Experiments 5, 6, 7 and 8, respectively, As such, the best grouping option obtained graphically for Experiments 1, 2, 3 and 4 will also be the best grouping option for Experiments 5, 6, 7, 8 respectively. Furthermore, for the remaining four experiments (Experiments 5 to 8), the second type of demand was used for the simulation (Demand is very low for most of the products but high for the remaining few).

With regards to Experiment 5, the best grouping solution, shown in Figure 27, is that for Experiment 1. However, unlike Experiment 1, the model was successful in generating ideal groups in which Group A generates a demand rate of exactly 5 Pallets/Day and Group B generates a demand rate of exactly 15 Pallets/Day. Nevertheless, despite using the best grouping option, the joint replenishment simulation results for this experiment were not satisfactory. As shown in Figure

39, this is due to the average inventory level in the joint replenishment simulation being four times bigger than that in the EOQ simulation. As a result, as Figure 41 indicates, the total inventory holding cost in the joint replenishment simulation was also much higher. Accordingly, the overall cost of the joint replenishment system was £131,000 higher than that for the EOQ system (as shown in Figure 41).

A close examination of the results shows the main issue to be the inventory holding cost of Product 2 in Group A, which was overloaded with stock (as shown in Figure 36). However, Product 2 has a very low demand rate of 0.09 Pallets/Day. Rather, the issue here is in the simulation model itself, and will be discussed later in this chapter (in Section 5.2). Nevertheless, the joint replenishment model outperformed the adjusted EOQ model in terms of the service level in this experiment (as shown in Figure 38).

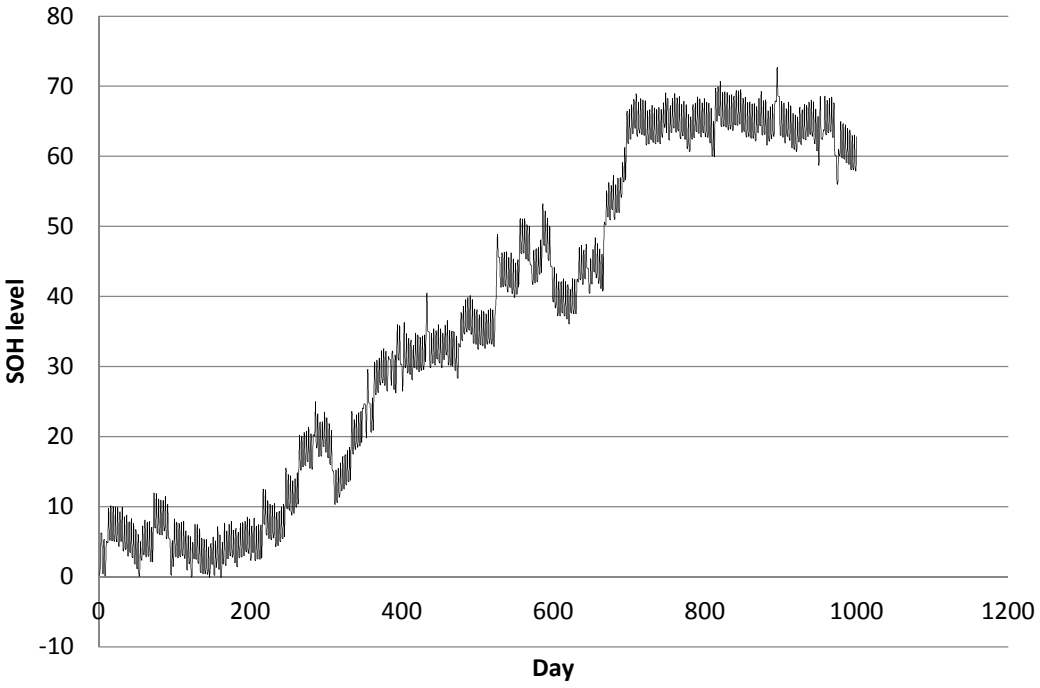


Figure 36: SOH levels for Product 2 in Experiment 5

In Experiment 6, where the ordering cost is high and the Truck size is small, the performance of the joint replenishment model was very poor, with an average inventory level that is 5 times that of the EOQ simulation (as shown in Figure

39). As a result, the inventory holding cost is very high compared to the total inventory holding cost in the adjusted EOQ simulation (as indicated in Figure 41).

Moreover, as shown in Figure 40, the average service level is at 88%, which is significantly lower than that in the adjusted EOQ simulation. Additionally, despite this very low service level, the total cost of the joint replenishment simulation is very high, and exceeds the corresponding figure for EOQ by more than £500,000. Such results clearly signal the presence of a fundamental issue in this experiment.

Looking at the grouping solution and the detailed simulation results, these poor results can be explained. As mentioned earlier, the best grouping solution of this experiment is essentially the same as that for Experiment 2, which involves replenishing the different items using two 15-pallet trucks, in which the first is dispatched daily and the second every 3 days. As with Experiment 2, the problem stems from the grouping model failing to recognise that T must be an integer value. Accordingly, although LINGO found the best grouping option, this option is neither practical nor possible to implement in the simulation model. Indeed, the model grouped the products into two groups, in which Group A boasts an average demand rate of 13.21 Pallets/Day and Group B a daily demand rate of 6.79 Pallets/Day, producing T values of 1.14 Pallets/Day and 2.21 Pallets/Day for Groups A and B, respectively. In performing the simulation, the T values were thus rounded to 1 and 2, respectively. As a result, both groups suffered from excess inventory levels and, consequently, very high inventory holding costs. As shown in Figures 39 and 41, the average inventory level in the joint replenishment simulation is 26.31 Pallets/Day and the total inventory holding cost is £599,000, which is much higher than the average inventory level of 5.61 Pallets/Day and the total inventory holding cost of £62,000 observed in the adjusted EOQ simulation.

In general, with such excess inventory level, one would expect to see very high service levels. However, as was observed in the previous example, the service level in this experiment was at 88%, which is very low compared to the 98% produced by the adjusted EOQ model. This is attributable to the fact the simulation model is not allocating the replenishment quantity to the different

items in an appropriate way. As mentioned earlier, this problem will be discussed in more detail in Section 5.3.

The grouping issue in this experiment can be solved simply by moving some of the products from Group B to Group A. For example, moving products 1, 8, 15, 17, 18, 19 and 20 from Group B to A will result in an average demand rate of exactly 15 Pallets/Day for Group A and exactly 5 Pallets/Day for Group B. This will, indeed, improve the simulation results. In particular, this way of grouping will improve the total holding costs and the inventory levels, since this solution avoids the rounding step.

Implementing these suggestions into Experiment 6, the following results were obtained:

Performance Measure	**Result**
Average Inventory Level (Pallets/ day)	10.7
Average Service Level	99.63%
Total Purchase Cost	£15,996,000.00
Total Holding Cost	£ 117,703.3
Total Ordering Cost	£ 42,237.64
Total Cost	£16,155,940.3

Clearly, these results represent significant improvement in the overall performance. In fact, as opposite to the original experiment, with these adjustments the joint replenishment model, clearly, outperforms the adjusted EOQ model as the total costs have been reduced by £62 thousands. Accordingly, these adjustments can be added as a further step in the proposed solution to guarantee robustness in the solution. Moreover, with these adjustments, the inventory level behaviour of one product (product 2) has been significantly improved as shown in the Figure 37 and Figure 38 below.

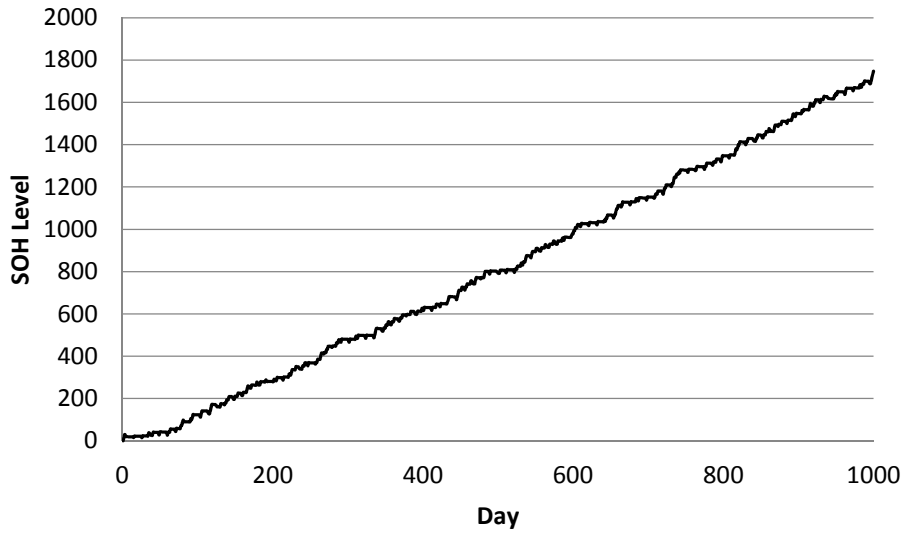


Figure 37: Daily SOH level for product 2 originally

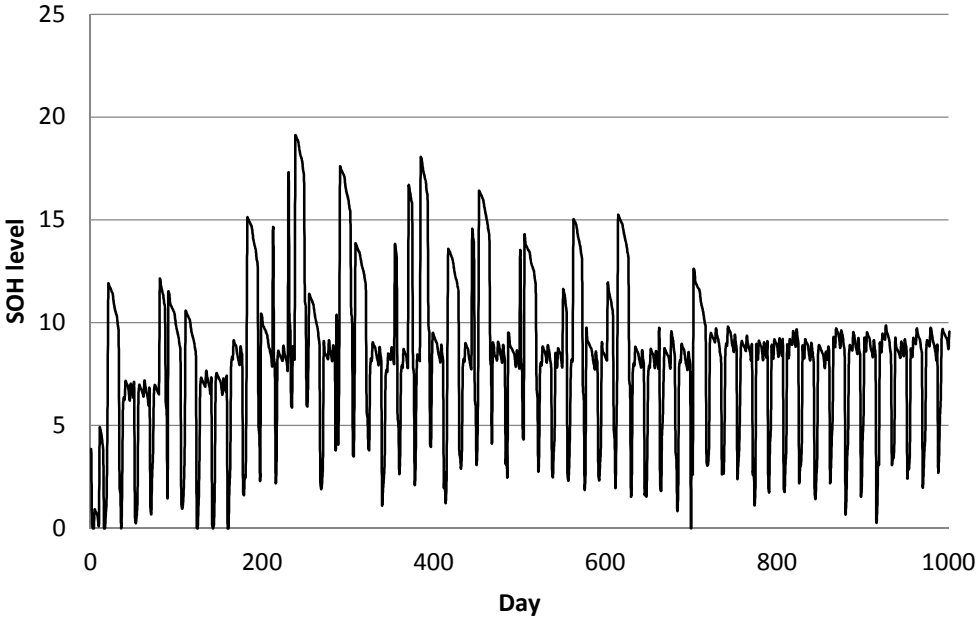
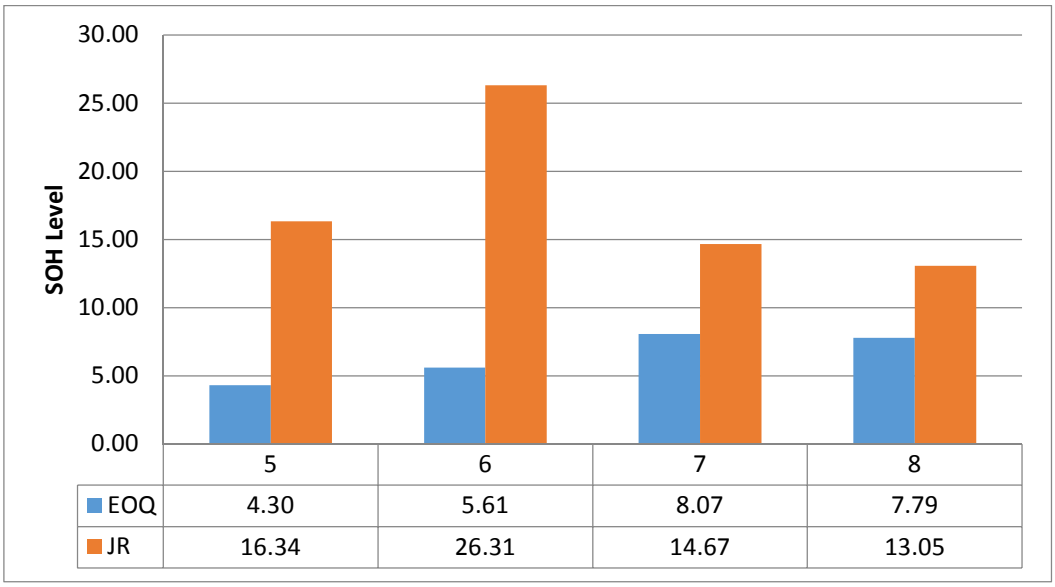


Figure 38: The SOH level for product 2 after adjusting experiment 6

With regards to Experiment 7, which involves low ordering costs and large truck sizes, the overall performance of the joint replenishment simulation outperformed that of the adjusted EOQ simulation, achieving a service level of 99.4%, which is approximately the same as that observed in the adjusted EOQ simulation. Furthermore, although the total inventory cost is higher in the joint replenishment simulation, the overall cost is £12,000 lower (as shown in Figure 43).

Finally, in Experiment 8 the grouping solution used is essentially that for Experiment 4, which involves daily replenishing a 20-pallet truck. The overall joint replenishment simulation results are satisfactory, with a service level of 99.51%, compared to 99.46% in the EOQ simulation, and an overall cost of more than £2,500 lower than the EOQ figure.



	5	6	7	8
EOQ	4.30	5.61	8.07	7.79
JR	16.34	26.31	14.67	13.05

Figure 39: Average Stock on Hand (Experiments 5 to 8)

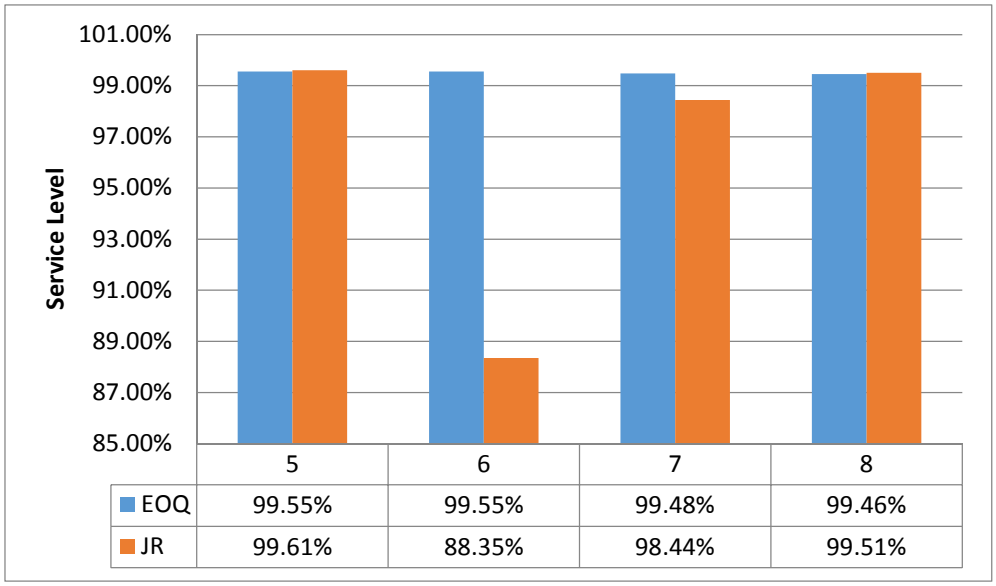


Figure 40: Average Service Levels (Experiments 5 to 8)

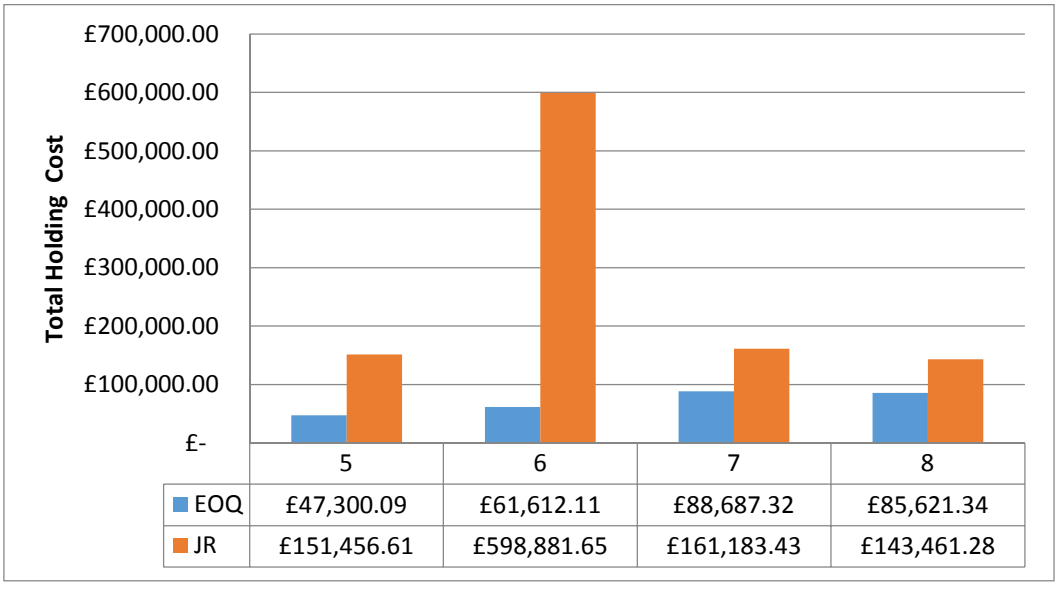


Figure 41: Total Holding Costs (Experiments 5 to 8)

Figure 42: Total Ordering Costs (Experiments 5 to 8)

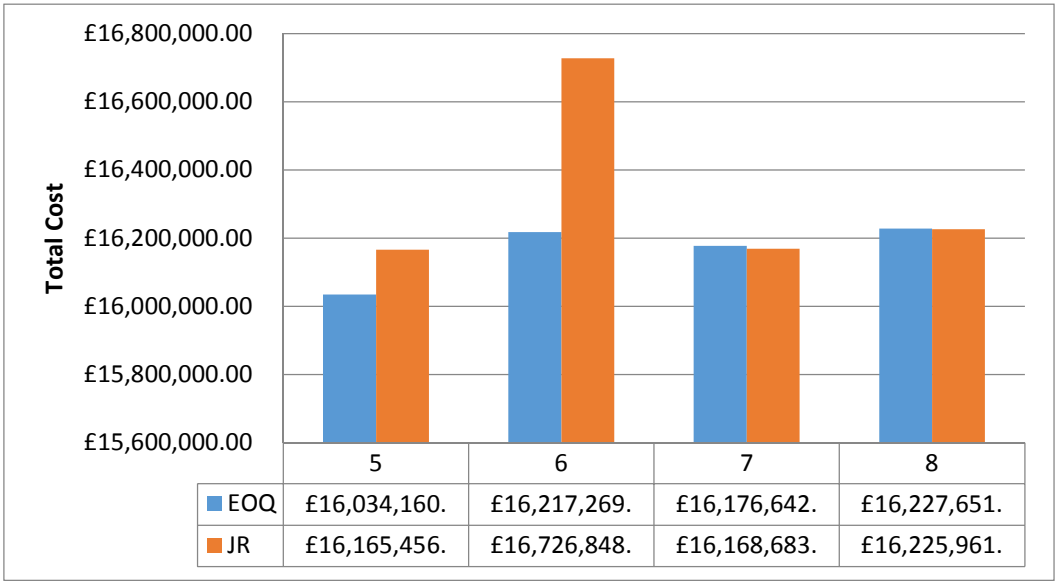


Figure 43: Total Costs (Experiments 5 to 8)

5.2 The impact of changing the ordering cost and truck sizes

In examining the performance of the new joint replenishment solution, a low fixed cost per order was adopted in some situations and a high fixed cost per order in others. As expected, the performance of the model in reducing the overall cost was greater when the fixed cost per order is high. This is because the joint replenishment theory is typically designed to reduce the overall costs of order and, as such, it is designed for situations where the ordering costs are high. Moreover, as discussed in the Literature Review chapter, (T, S) policies are also designed to control environments where the ordering costs are high.

However, the performance of the proposed solution is also improved with larger trucks. This is because using big trucks allows for the ideal joint replenishment situation to be applied, namely to replenish all different items at once at every replenishment instance, thus minimising the total ordering cost.

In general, the proposed solution is more likely to succeed in reducing the overall costs and in improving the overall service level in situations where the fixed order cost is high and the trucks available are large. Moreover, changing the values of the ordering costs and the truck sizes also produces an impact on the way the products are grouped together. In particular, the higher the ordering cost and the larger the available trucks, the more likely it is that the solution will involve very few replenishments.

5.3 The impact of the products' demand rates

In this dissertation, we examined the performance of the proposed solution using two types of demand. Under Type 1 demand, in which all the products exhibited similar demand rates, the performance of the grouping heuristic was efficient in finding the optimal groups. The only issue, as described in Section 5.1, was that raised by the rounding-up step. However, different solutions were offered to address this problem which, if implemented, would improve the overall performance.

Furthermore, under Type 1 demand, the performance of the (T, S) simulation was also acceptable across all experiments, with the overall costs being reduced

and the average service levels improved. On the other hand, running the experiments under Type 2 demand, where most of the products had low demand rates but a remaining few exhibited high demand levels, it was realised that the (T, S) model needed further improvements. This is because the (T, S) simulation model attempts to minimise the total deviation from the order-up-to level, but without considering the differences in the demand rates among the different products. Accordingly, when a mix of high and low demand products is present in a group, the model fails in creating the proper balance when allocating the truck capacity to the different products. This issue manifested itself in Experiments 5 and 6 in particular, and a simple solution would be to divide each high demand product into several products. Indeed, this step was performed after running the grouping model. For example, in Experiment 5, Product 5 in Group A had an average demand rate of 4.81 Pallets/Day. Dividing the Product into 10 sub-products in turn divided the demand rate by 10; yielding 10 products with a demand rate of 0.481 Pallets/Day each. This way, the unbalanced allocation of the truck capacity, and therefore the ensuing excess inventory levels for some products and very low service levels for others, can be avoided. The order-up-to-level for each new product was then re-calculated and the simulation re-run.

5.4 The solution for items with different holding costs

The proposed solution in this project is particularly designed for joint replenishment environments where the holding costs for different items is the same. However, this is not always the case in real life situations. As such, using the following procedure, the proposed solution can be adapted to situations where the holding costs are different for different items:

We know that for any truck,

$$G(T) = \frac{\sum_{i=1}^{n} \lambda_i h_i}{2} T + \frac{K}{T} \quad (5.1)$$

Therefore,

$$T^* = \sqrt{\frac{2K}{\sum_{i=1}^{n} \lambda_i h_i}} \quad (5.2)$$

Further, we also know that,

$$T^* = \frac{Q^*}{\sum_{i=1}^{n} \lambda_i} \quad (5.3)$$

Therefore,

$$\frac{Q^*}{\sum_{i=1}^{n} \lambda_i} = \sqrt{\frac{2K}{\sum_{i=1}^{n} \lambda_i h_i}} \quad (5.4)$$

Hence,

$$\sum_{i=1}^{n} \lambda_i h_i = \frac{2K(\sum_{i=1}^{n} \lambda_i)^2}{{Q^*}^2} \quad (5.5)$$

Supposing that α_i= is the % of λ_i in $\sum_{i=1}^{n} \lambda_i$, then we have:

$$\lambda_i = \alpha_i \sum_{i=1}^{n} \lambda_i \quad (5.6)$$

Accordingly,

$$\sum_{i=1}^{n} \lambda_i h_i = h_1 \alpha_1 \sum_{i=1}^{n} \lambda_i + h_2 \alpha_2 \sum_{i=1}^{n} \lambda_i + \cdots + h_n \alpha_n \sum_{i=1}^{n} \lambda_i \quad (5.7)$$

Therefore,

$$\sum_{i=1}^{n} \lambda_i h_i = (h_1 \alpha_1 + h_2 \alpha + \cdots + h_n \alpha_n) \sum_{i=1}^{n} \lambda_i \quad (5.8)$$

Using Equation 5.1, we obtain:

$$(h_1\alpha_1 + h_2\alpha + \cdots + h_n\alpha_n)\sum_{i=1}^{n}\lambda_i = \frac{2K(\sum_{i=1}^{n}\lambda_i)^2}{Q^{*2}} \quad (5.9)$$

As such, the ideal demand rate for any truck of size Q^* can be expressed as:

$$\sum_{i=1}^{n}\lambda_i = \frac{Q^{*2}}{2K}(h_1\alpha_1 + h_2\alpha + \cdots + h_n\alpha_n) \quad (5.10)$$

Given the ideal demand rate for each truck, we propose the following grouping heuristic for this problem:

1. Set values for $\alpha_i s$.

2. Find $\sum_{i=1}^{n}\lambda_i$ using Equation 5.10.

3. Find $\lambda_i's$ using Equation 5.6.

4. Group the products using the Bin-Packing approach. This is done by allocating the different $\lambda_i s$ to the ideal demand rates of the different trucks.

Supposing that $h_i = h_1 \, or \, h_2$, $n = 2$, $\alpha_1 = \alpha$ and $\alpha_2 = \beta$ we then obtain:

- The total cost is at a minimum at $\alpha = 1$ and $\beta = 0$ when $h_1 < h_2$ or $\beta = 1$ and $\alpha = 0$ when $h_2 < h_1$

- The total cost is at a maximum at $\alpha = 1$ and $\beta = 0$ when $h_1 > h_2$ or $\beta = 1$ and $\alpha = 0$ when $h_2 > h_1$

- These two conditions set the upper and lower boundaries of $\sum_{i=1}^{n}\lambda_i h_i$

- This proves that there is no optimum value for $\sum_{i=1}^{n} \lambda_i h_i$ and that it is rather dependent on the values allocated for α and β

Therefore, in this heuristic solution, realistic and feasible values of the different $\alpha_i s$ can be allocated according to the actual percentages of each product from the overall real demand. Once the products are grouped, the (T, S) system adopted in this dissertation can be used.

It must be noted, however, that while the proposed solution presented in this project to the joint replenishment problem under consideration has been shown to be reliable, it nevertheless represents the first step towards the development of more efficient and versatile future solutions to the problem, particularly by integrating novel angles and approaches that have thus far been unexplored in past efforts.

CHAPTER 6. Conclusion

In this final chapter, an overview will first be provided of the main objectives that were successfully fulfilled in the course of undertaking the present project. Second, a summary of the key results obtained during the experimental process (i.e. the joint replenishment simulation) will be presented. Finally, potential areas of study and further research relevant to the topic will be examined, and a set of recommendations and suggestions in this regard will be offered.

It must be noted at the outset that this dissertation has accomplished its principal aim, namely the development of a new heuristic model to solve the Joint Replenishment Problem (JRP) for a configuration involving stochastic characteristics of demand, products with similar holding and purchase costs, quantity discounts and a fixed cost charged with each dispatched truck. To that purpose, the dissertation successfully fulfilled its overall objective of minimising the total inventory control cost for joint replenishment environments - in which a fixed cost is charged for each truck used to replenish the various items.

The first step of the heuristic solution developed in this project involves efficiently placing the different items into the appropriate groups, in order to achieve quantity discounts and to benefit from economies of scale. To this effect, an integer programming grouping model was successfully developed based on a full truckload policy and total demand. The grouping heuristic was programmed using the LINGO optimisation software. The widely-used Bin Packing approach was implemented into the LINGO model in order to allocate the different products to the appropriate groups, so that each group is replenished every T using a single full truck.

The second step of the solution involves developing a replenishment policy where the choice of order size must help ensure full truckloads, so as to exploit quantity discounts. For this purpose, an adjusted (T, S) policy was used in which, at each review instant, T, the inventory position of each item, i, within group j is reviewed. When the order is placed, thus, its inventory position is raised to $S_i + d_i^- - d_i^+$, whereby d_i^- and d_i^+ are the over-achievement and under-achievement variables.

Furthermore, to test the performance of the heuristic solution, experimental data based on a real-life context was generated. Using this data, different groups of products were set up and a 1000-day simulation (using LINGO) was conducted, based on the adjusted (T, Q) policy. This simulation aimed to determine the order quantities of the different products.

The results obtained from the joint replenishment simulation were subsequently compared against the simulation results of an adjusted EOQ model that were performed using Microsoft Excel. In order to conduct this comparative analysis, a set of key performance measures were adopted: Total inventory cost, Average Inventory Level and the Service Level.

With regards to the key findings of the simulation, the proposed solution to the JPR was found to perform better in situations where the fixed cost per order is high. In addition, it was found that, in most cases, the proposed solution outperformed that of the adjusted EOQ model. Moreover, in performing the (T, S) simulation, it was shown that the simulation model programmed in LINGO attempts to minimise the total deviation from the order-up-to levels of the different items without taking into consideration the individual needs of the different items in terms of their order quantities. This is notably the case when Type 2 demand is adopted when running the simulation. As a result, some products were overloaded with stock while others experienced very low service levels due to the unbalanced supply.

As such, a simple solution to this issue was suggested, according to which products with very high demand rates are divided into several sub-products. Moreover, the grouping heuristic has been shown not to recognise that the replenishment cycle (T) must be an integer number of days, which impacted on the overall results. Nevertheless, a number of simple solutions to this problem were provided which, if applied, would improve the overall performance and bolster the robustness of the proposed solution.

The topic of joint replenishment and the Joint Replenishment Problem specifically, will continue to present interesting issues and challenges for the

academic and business community. In this context, the present research work, while successfully achieving its stated aim and objectives, has also traced a way towards further work into the topic. Indeed, a number of avenues for future work into the topic of joint replenishment can prove greatly beneficial in terms both of academic and practical advantages.

For instance, there is great potential for extending the present solution to allow it to handle items with different holding costs. Moreover, the solution can be adapted in a number of interesting ways so as to suit a variety of purposes, for instance by adopting different policies such as the Can-Order policy. Further research may also be carried out to help improve and refine the current solution, such as improving the implementation or conducting more extensive experimentation.

References

Aksoy, Y. & Erenguc, S., 1989. Multi-item inventory models with coordinated replenishments: A survey.. *International Journal of Operations and Production Management, Volume* 8, pp. 63-73.

Andreas, N., 2006. *Essays on Joint Replenishment and Multi-Echlon Inventory Systems,* Doctoral dissertation, Luleå University of Technology.

Arkin, E., Joneja, D. & Roundy, R., 1989. Computational complexity of uncapacitated multi-echelon production planning problems. *Operations Research Letters,* 8(2), p. 61–66.

Atkins, D., 1991. The Inventory Joint Replenishment Problem with a General Class of Joint Costs. *European Journal of Operational Research,* Volume 51, pp. 310-312.

Atkins, D. & Iyogun, P., 1988. Periodic versus 'can-order' policies for coordinated multi-item inventory systems. *Management Science,* Volume 34, pp. 791-796.

Balakrishnan, N., 2012. *Managerial Decision Modeling with Spreadsheets.* 3rd ed. :Prentice Hall.

Balintfy, J., 1964. On a Basic Class of Multi-Item Inventory Problems. *Management Science,* 10(2), pp. 287-297.

Benton, W., 1991. Quantity discounts under conditions of multiple items, multiple suppliers and resource limitations. *International Journal of Production Research ,* Volume 29, p. 1953– 1961.

Brown, G., 1967. *Decision Rules for Inventory Management.* New York: Rinehart & Winston.

Cachon, G., 2001. Managing a retailer's shelf space, inventory, and transportation. *Manufacturing and Service Operations,* 3(3), p. 211–229.

Carlson, M. & Miltenburg, G., 1988. Using the service point model to control large group of items. *Omega,* 16(5), p. 481–489.

Cha, C. & Moon, I., 2005. The joint replenishment problem with quantity discounts under constant demand. *OR Spectrum,* 27(4), pp. 569-581.

Chakravarty, A., 1984. Joint inventory replenishments with group discounts based on invoice value. *Management Science ,* Volume 30, p. 1105–1112.

Chen, C. & Min, K., 1994. A Multi-Product EOQ Model with Pricing Consideration - T.C.E Cheng's Model Revisited. *Journal of Computers and Industrial Engineering,* 26(4), pp. 787-794.

Chung, C., Hum, S. & Kirca, O., 1996. The coordinated replenishment dynamic lot-sizing problem with quantity discounts. *European Journal of Operational Research,* Volume 94, p. 122–133.

Dolan, J., 1987. Quantity discounts: managerial issues and research opportunities. *Marketing Science,* 6(1), pp. 1-22.

Doll, L. & Whybark, C., 1973. An Iterative Procedure for the Single-Machine Multi-Product Lot Scheduling Problem. *Management Science ,* 20(1).

Donal, J., 1998. Quantity Discounts: Managerial Issues and Research Opportunities. *Marketing Scince,* 1-22(1), p. 6.

Eijs, M., 1994. Multi-item inventory systems with joint ordering and transortation decisions. *International Journal of Production Economics,* Volume 35, pp. 285-292.

Eijs, M., 1994. Multi-Item Inventory Systems with Joint Ordering and Transportation Decisions. *International Journal of Production Economics,* Volume 35, pp. 285-292.

Eijs, V., 1993. A note on the joint replenishment problem under constant demand. *Journal of the Operational Research Society,* Volume 44 , pp. 185-191.

Eijs, V., 1993. A note on the joint replenishment problem under constant demand. *Journal of the Operational Research Society,* Volume 44, pp. 185-191.

Eijs, V., Heuts, J. & Kleijnen, C., 1992. Analysis and comparison of two strategies for multi-item inventory systems with joint replenishment costs. *European Journal of Operational Research,* Volume 59, p. 405–412.

Eppen, G. & Martin, R., 1987. Solving multi-item capacitated lot-sizing problems using variable redefinition. *Operations Research,* Volume 35, p. 832–848.

Federgruen, A., Groenevelt, H. & Tijms, H., 1984. Coordinated replenishments in a multi-item inventory system with compound Poisson demand. *Management Science,* Volume 30, pp. 344-357.

Ghiani, G., Laporte, G. & Musmanno, R., 2013. *Introduction to Logistics systems Management.* NY: Wiley.

Glock, C., 2012. The Joint Ecoomic Lot Size Problem: A Review. *International Journal of Production Economics,* 135(2), pp. 671-686.

Goyal, S., 1973. Determination of Economic Packaging Frequency for Items Jointly Replenished. *Management Science,* 20(2), pp. 232-235.

Goyal, S., 1974a. Optimum ordering policy for a multi-item single supplier. *Operational Research Quarterly,* Volume 25, pp. 293-298.

Goyal, S., 1974b. Determination of optimal packaging frequency of jointly replenished items. *Management Science,* Volume 21, pp. 436-443.

Goyal, S., 1988. Determining the optimum production-packaging policy for jointly replenished items. *Engineering Costs and Production Economics,* Volume 15, pp. 339-341.

Goyal, S. & Belton, A., 1979. A Simple Method of Determining Order Quantities in Joint Replenishment Under Deterministic Demand. *Management Science,* Volume 25, p. 604.

Goyal, S. & Deshmukh, S., 1993. A note on "the economic ordering quantity for jointly replenished items''. *International Journal of Production,* Volume 31, pp. 2959-2961.

Goyal, S., Harir, A. & Abou-el-ata, M., 1995. The resource constrained multi-item inventory problem with price discount: a geometric programming approach. *Production Planning & Control ,* Volume 6, p. 374–377.

Goyal, S. & Satir, S., 1989. Joint Replenishment Inventory Control: Deterministic and Stochastic Models. *European Journal of Operational Research,* Volume 38, pp. 2-13.

Guder, F. & Zydiak, J., 1997. Non-stationary ordering policies for multi-item inventory systems subject to a single resource constraint and quantity discounts. *Computers & Operations Research,* 24(1), p. 61–71.

Guder, F. & Zydiak, J., 1999. Ordering Policies for Multi-item Inventory Systems Subject to Multiple Resource Constraints. *Computers and Operations Research,* Volume 26, pp. 583-597.

Guder, F., Zydiak, J. & Chaudhry, S., 1994. Capacitated multiple item ordering with incremental quantity discounts. *Journal of the Operational Research Society,* Volume 45, p. 1197–1205.

Hariri, A., Abou-El-Ata, M. & Goyal, S., 1995. The resource constrained multi-item inventory problem with price discount: a geometric programming approach. *Production Planning & Control,* 6(4), pp. 374-377.

Hoque, M., 2006. European Journal of Operational Research. *An Optimal Solution Technique for the Joint Replenishment Problem with Storage and Transport Capacities and Budget Constraints,* Volume 175, pp. 1033-1042.

Ignall, E., 1969. Optimal continuous review policies for two product inventory. *Management Science,* Volume 15, pp. 278-283.

Johansen, S. & Melchiors, P., 2003. Can-order policy for the periodic-review joint replenishment problem. *Journal of the Operational Research Society,* 54(3), pp. 283-290.

Kamalia, A., Ghomia, S. & Jolaib, F., 2011. A multi-objective quantity discount and joint optimization model for coordination of a single-buyer multi-vendor supply chain. *Computers & Mathematics with Applications,* 62(8), p. 3251–3269.

Kang, H. & Lee, A., 2012. A stochastic lot-sizing model with multi-supplier and quantity discounts. *International Journal of Production Research,* 51(1), pp. 245-263.

Kaspi, M. & Rosenblatt, M., 1983. An Improvement of Silver's Algorithm for the Joint Replenishment Problem. *IIE Transactions,* Volume 15, pp. 264-269.

Kaspi, M. & Rosenblatt, M., 1991. On the economic ordering quantity for jointly replenished items. *International Journal of Production Research,* Volume 29, pp. 107-114.

Keerthana, R., 2013. *Implementation of Joint Replenishment Inventory Model with Quantity Discounts.*MSc Dissertation thesis. University of Nottingham.

Khouja, M. & Goyal, S., 2008. A Review of the Joint Replenishment Problem Literature: 1989-2005. *European Journal of Operational Research ,* Volume 186, pp. 1-16.

Khouja, M. & Saydam, S., 2005. Joint replenishment problem under continuous unit cost change. *International Journal of Production Research,* 43(2), pp. 311-326.

Kiesmuller, G., 2009. A multi-item periodic replenishment policy with full truck loads. *International Journal of Production Economics ,* Volume 118, pp. 275-281.

Lee, A., Kang, H., Lai, C. & Hong, W., 2013. An integrated model for lot sizing with supplier selection and quantity discounts. *Applied Mathematical Modelling,* 37(7), p. 4733–4746.

Li, J. & Liu, L., 2006. Supply chain coordination with quantity discount policy. *International Journal of Production Economics,* 101(1), p. 89–98.

Lu, L., 1995. A One Vendor Multi-Buyer Integrated Inventory Model. *European Journal of Operational Research,* Volume 81 , pp. 312-323.

McGee, V. & David, P., 1996. Periodic production scheduling at a fastener manufacturer. *International Journal of Production Economics,* Volume 46, pp. 65-87.

Melchiors, P., 2002. Calculating can-order policies for the joint replenishment problem by the compensation approach. *European Journal of Operational Research,* 141(3), pp. 587-595.

Miltenburg, G. & Silver, E., 1989. A microcomputer inventory control package for controlling families of items. *Engineering Costs and Production Economics,* Volume 15, p. 201–209.

Miltenburg, J., 1985 . Allocating a Replenishment Order Among a Family of Items. *IEE Transactions ,* Volume 17, pp. 261-267.

Miltenburg, J., 1987. Co-ordinated control of a family of discount-related items. *INFOR,* Volume 25, pp. 97-116.

Moon, I. & Cha, B., 2005. The Joint Replenishment Problem with Quantity Discounts Under Constant Demand. *OR Spectrum ,* Volume 27, pp. 569-581.

Moon, I. & Cha, B., 2006. The joint replenishment problem with resource restriction. *European Journal of Operational Research,* Volume 173, pp. 190-198.

Moon, I. & Cha, B., 2006. The Joint Replenishment Problem with Resource Restrictions. *European Journal of Operational Research ,* Volume 173, pp. 190-198.

Moon, I., Goyal, S. & Cha, B., 2008. The Joint Replenishment Problem Involving Multiple Suppliers Offering Quantity Discounts.. *International Journal of Systems Science,* 39 (6), pp. 629-637.

Moussourakis, J. & Haksever, C., 2013. Models for Ordering Multiple Products Subject to Multiple Constraints, Quantity and Freight Discounts. *American Journal of Operations Research ,* Volume 3, pp. 521-535.

Nagasawa, K. & Irohara, T., 2013. Joint Replineshment Problem in Multi-item Inventory Control with Carrier Capacity and Receiving Inspection Cost. *Operations and Supply Chain Management ,* Volume 6, pp. 111-116.

Nahmias, S., 2009. *Production and Operations Analysis.* 6th ed. NY: McGraw-Hill.

Nielsen, C. & Larsen, C., 2005. An analytical study of the Q(s, S) policy applied. *European Journal of Operational Research,* Volume 163, pp. 721-732.

Nilsson, A., Segerstedt, A. & Sluis, E., 2007. A New Iterative Heuristic to Solve the Joint Replenishment Problem Using Spreadsheet Technique. *International Journal of Production Economics,* Volume 108, pp. 399-405.

Olsen, A., 2005. An Evolutionary Algorithm to Solve the Joint Replenishment Problem using Direct Grouping. *Computers and Industrial Engineering ,* Volume 48, pp. 223-235.

P&H, 2014. *Palmer and Harvey.* [Online]
Available at: http://www.palmerharvey.co.uk/
[Accessed 2014 6 30].

Pantumsinchai, P., 1992. A Comparison of Three Joint Ordering Inventory Policies. *Decision Science,* 23(1), pp. 111-127.

Paul, S., Wahab, M. & Ongkunaruk, P., 2014. Joint replenishment with imperfect items and price discount. *Computers & Industrial Engineering,* Volume 74, p. 179–185.

Pirkul, H. & Aras, O., 1985. Capacitated multiple item ordering problem with quantity discounts. *IIE Transactions,* Volume 17, p. 206–211.

Pirkul, H. & Aras, O., 1985. Capacitated multiple item ordering problem with quantity discounts. *IIE Transactions,* Volume 17, p. 206–211.

Porras, E. & Dekker, R., 2006. An Efficient Optimal Solution Method for The Joint Replenishment Problem with Minimum Order Quantities. *European Journal of Operational Research,* Volume 174, pp. 1595-1615.

Porras, E. & Dekker, R., 2006b. On the efficiency of optimal algorithms for the joint replenishment problem: A comparative study. *Econometric Institute Report EI,* Volume 33.

Porras, E. & Dekker, R., 2008. A Solution Method for the Joint Replenishment Problem with Correction Factor. *International Journal of Production Economics ,* Volume 113, pp. 843-851.

Queyranne, M. & Sun, D., 1993. The Performance Ratio of Grouping Policies for the Joint Replenishment Problem. *Discrete Applied Mathematics ,* Volume 46, pp. 43-72.

Qu, W., Bookbinder, J. & Iyogun, P., 1999. An Integrated Inventory-Transportation System with Modified Periodic Policy for Multiple Products. *European Journal of Operational Research ,* Volume 115, pp. 245-269.

Schouten, F., Eijs, V. & Heuts, R., 1994. Coordinated replenishment systems with discount opportunities. *International Journal of Production Research,* Volume 32, p. 2879–2895.

Schultz, H. & Johansen, S., 1999. Can-order policies for coordinated inventory replenishment with Erlang distributed times between ordering. *European Journal of Operational Research,* 16(1), pp. 30-41.

Shu, F., 1971. Economic ordering frequency for two items jointly replenished. *Management Science,* Volume 17, pp. 406-410.

Shumnij, J., 2010. *Investigating Joint Replenishment Inventory Models.* MSc dissertation. The University of Nottingham.

Silver, E., 1965. Some characteristics of a special joint-order inventory model. *Operations Research,* Volume 14, pp. 319-322.

Silver, E., 1973. Three ways of obtaining the average cost expressions in a problem related to joint replenishment inventory control. *Naval Research Logistics Quarterly,* Volume 20, pp. 241-254.

Silver, E., 1974. A control system for co-ordinated inventory replenishment. *International Journal of Production Research,* Volume 12, pp. 647-671.

Silver, E., 1976. A simple method of determining order quantities. *Management Science,* Volume 22, pp. 1351-1361.

Silver, E., 1981. Establishing reorder points in the (S,c,s) co-ordinated control system under compound Poisson demand. *International Journal of Production,* Volume 19, pp. 743-750.

Silver, E. & Bischak, D., 2011. The exact fill rate in a periodic review base stock system under normally distributed demand. *Omega,* 39(3), pp. 346-349.

Silver, E., Pyke, D. & Peterson, R., 1998. *Inventory Management and Production Planning and Scheduling.* 3rd ed. NY: Wiley.

Simpson, C. & Erenguc, S., 1995. Multiple Stage Production Planning Research: History and Opportunities. *International Journal of Production and Operational Science ,* 6(16), pp. 25-40.

Starr, M. & Miller, D., 1962. *Inventory Control: Theory and Practice.* NJ: Prentice Hall.

Tayur, S., Ganeshan, R. & Magazine, M., 1999. *Quantitative Models for Supply Chain Management.* Boston: Kluwer Academic Publishers.

Thompstone, R. & Silver, E., 1975. A co-ordinated inventory control system for compound Poisson demand and zero lead time. *International Journal of Production Research,* Volume 581-602, p. 13.

Toptal, A., Cetinkaya, A. & Lee, C., 2003. The Buyer-Vendor Coordination Problem: Modeling Inbound and Outbound Cargo Capacity and Costs. *IEE Transactions ,* Volume 35, pp. 987-1002.

Tsai, C. & Huang, P., 2009. An association clustering algorithm for can-order policies in the joint replenishment problem. *International Journal of Production Economics,* 117(1), pp. 30-41.

Viswanathan, S., 1996. A new optimal algorithm for the joint replenishment. *Journal of the Operational Research Society,* Volume 47, pp. 936-44.

Viswanathan, S., 1997. Periodic review (s,S) policies for joint replenishment inventory systems. *Management Science,* Volume 43, p. 1447–1454.

Viswanathan, S., 2002. On Optimal Algorithms for the Joint Replenishment Problem. *Journal of the Operational Research Society,* Volume 53, pp. 1286-1290.

Wee, H., 2013. *Inventory Modelling Systems: Modelling and Research Methods.* s.l.:Nova.

Wilson, R., 1993. *Nonlinear Pricing.* NY: Oxford University Press.

Xu, J., Lu, L. & Glover, F., 2000. The deterministic multi-item dynamic lot size problem with joint business volume discount. *Annals of Operations Research ,* Volume 96, p. 317–337.

Appendices

(Note that due to the enormous number of appendices, the full list is submitted online and in the attached CD)

Grouping EXP1

```
!EXP1;

Model:

Sets:

PRODUCT/1..20/:

DEMAND;

REP/1..17/:
y,
Q,
IDEAL_DEMAND,
D1,
D2,
K;

PXR(PRODUCT,REP):
X;

endsets

data:

DEMAND          =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','AVG_DEMAND');
Q               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','Q');
IDEAL_DEMAND    =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','IDEAL_DEMAND');
H               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','HOLDING_COST');
K               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','FIXED_COST');

enddata

!Objective function;

MIN = @SUM (REP(J): (0.5*H*Q(J)*Y(J))+(K/Q(J))*( @SUM (PRODUCT(I): DEMAND(I)*X(I,J))));

!constraints;

!every item should be allocated to one replenishment only;
@FOR ( PRODUCT(I): @SUM (REP(J): X(I,J)) =1);

!x is an intejer = 0 or 1 (i.e. x is binary;
@FOR (PXR (I,J): @BIN (X(I,J)));

!y is binary;
@FOR (REP(J): @BIN (Y(J)));

!logical condition;
@FOR(PXR(I,J): X(I,J) <= Y(J));

!loading/capacity constraint;

@FOR(REP(J): (D1(J)- D2(J)+ @SUM (PRODUCT (I): DEMAND(I)*X(I,J)))= IDEAL_DEMAND(J)*Y(J));
@for (REP(J): (@SUM (PRODUCT (I): DEMAND(I)*X(I,J))) <= Q(J));

end
```

Grouping EXP 5

```
!EXP1 X;

Model:

Sets:

PRODUCT/1..20/:

DEMAND;

REP/1..7/:
Y,
Q,
IDEAL_DEMAND,
D1,
D2,
K;

PXR(PRODUCT,REP):
X;

endsets

data:

DEMAND         =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','AVG_DEMAND');
Q              =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','Q');
IDEAL_DEMAND   =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','IDEAL_DEMAND');
H              =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','HOLDING_COST');
K              =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','FIXED_COST');

enddata

!Objective function;

MIN = @SUM (REP(J): (0.5*H*Q(J)*Y(J))+(K/Q(J))*( @SUM (PRODUCT(I): DEMAND(I)*X(I,J))));

!constraints;

!every item should be allocated to one replenishment only;
@FOR ( PRODUCT(I): @SUM (REP(J): X(I,J)) =1);

!x is an intejer = 0 or 1 (i.e. x is binary;
@FOR (PXR (I,J): @BIN (X(I,J)));

!y is binary;
@FOR (REP(J): @BIN (Y(J)));

!logical condition;
@FOR(PXR(I,J): X(I,J) <= Y(J));

!loading/capacity constraint;

@FOR(REP(J): (D1(J)- D2(J)+ @SUM (PRODUCT (I): DEMAND(I)*X(I,J)))= IDEAL_DEMAND(J)*Y(J));
@for (REP(J): (@SUM (PRODUCT (I): DEMAND(I)*X(I,J))) <= Q(J));

end
```

Grouping EXP 2

```
!EXP2;

Model:

Sets:

PRODUCT/1..20/:

DEMAND;

REP/1..6/:
y,
Q,
IDEAL_DEMAND,
D1,
D2,
K;

PXR(PRODUCT,REP):
X;

endsets

data:

DEMAND          =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','AVG_DEMAND');
Q               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','Q');
IDEAL_DEMAND    =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','IDEAL_DEMAND');
H               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','HOLDING_COST');
K               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','FIXED_COST');

enddata

!Objective function;

MIN = @SUM (REP(J): (0.5*H*Q(J)*Y(J))+(K/Q(J))*( @SUM (PRODUCT(I): DEMAND(I)*X(I,J))));

!constraints;

!every item should be allocated to one replenishment only;
@FOR ( PRODUCT(I): @SUM (REP(J): X(I,J)) =1);

!x is an intejer = 0 or 1 (i.e. x is binary;
@FOR (PXR (I,J): @BIN (X(I,J)));

!y is binary;
@FOR (REP(J): @BIN (Y(J)));

!logical condition;
@FOR(PXR(I,J): X(I,J) <= Y(J));

!loading/capacity constraint;

@FOR(REP(J): (D1(J)- D2(J)+ @SUM (PRODUCT (I): DEMAND(I)*X(I,J)))= IDEAL_DEMAND(J)*Y(J));
@for (REP(J): (@SUM (PRODUCT (I): DEMAND(I)*X(I,J))) <= Q(J));

end
```

EXP 5

```
!EXP2X;

Model:

Sets:

PRODUCT/1..20/:

DEMAND;

REP/1..16/:
y,
Q,
IDEAL_DEMAND,
D1,
D2,
K;

PXR(PRODUCT,REP):
X;

endsets

data:

DEMAND          =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','AVG_DEMAND');
Q               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','Q');
IDEAL_DEMAND    =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','IDEAL_DEMAND');
H               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','HOLDING_COST');
K               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','FIXED_COST');

enddata

!Objective function;

MIN = @SUM (REP(J): (0.5*H*Q(J)*Y(J))+(K/Q(J))*( @SUM (PRODUCT(I): DEMAND(I)*X(I,J))));

!constraints;

!every item should be allocated to one replenishment only;
@FOR ( PRODUCT(I): @SUM (REP(J): X(I,J)) =1);

!x is an intejer = 0 or 1 (i.e. x is binary;
@FOR (PXR (I,J): @BIN (X(I,J)));

!y is binary;
@FOR (REP(J): @BIN (Y(J)));

!logical condition;
@FOR(PXR(I,J): X(I,J) <= Y(J));

!loading/capacity constraint;

@FOR(REP(J): (D1(J)- D2(J)+ @SUM (PRODUCT (I): DEMAND(I)*X(I,J)))= IDEAL_DEMAND(J)*Y(J));
@for (REP(J): (@SUM (PRODUCT (I): DEMAND(I)*X(I,J))) <= Q(J));

end
```

Grouping EXP3

```
!EXP4;

Model:

Sets:

PRODUCT/1..20/:

DEMAND;

REP/1..16/:
y,
Q,
IDEAL_DEMAND,
D1,
D2,
K;

PXR(PRODUCT,REP):
X;

endsets

data:

DEMAND          =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','AVG_DEMAND');
Q               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','Q');
IDEAL_DEMAND    =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','IDEAL_DEMAND');
H               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','HOLDING_COST');
K               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','FIXED_COST');

enddata

!Objective function;

MIN = @SUM (REP(J): (0.5*H*Q(J)*Y(J))+(K/Q(J))*( @SUM (PRODUCT(I): DEMAND(I)*X(I,J))));

!constraints;

!every item should be allocated to one replenishment only;
@FOR ( PRODUCT(I): @SUM (REP(J): X(I,J)) =1);

!x is an intejer = 0 or 1 (i.e. x is binary;
@FOR (PXR (I,J): @BIN (X(I,J)));

!y is binary;
@FOR (REP(J): @BIN (Y(J)));

!logical condition;
@FOR(PXR(I,J): X(I,J) <= Y(J));

!loading/capacity constraint;

@FOR(REP(J): (D1(J)- D2(J)+ @SUM (PRODUCT (I): DEMAND(I)*X(I,J)))= IDEAL_DEMAND(J)*Y(J));
@for (REP(J): (@SUM (PRODUCT (I): DEMAND(I)*X(I,J))) <= Q(J));

end
```

Grouping EXP 4

```
!EXP5;

Model:

Sets:

PRODUCT/1..20/:

DEMAND;

REP/1..8/:
y,
Q,
IDEAL_DEMAND,
D1,
D2,
K;

PXR(PRODUCT,REP):
X;

endsets

data:

DEMAND          =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5.xlsx','AVG_DEMAND');
Q               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5.xlsx','Q');
IDEAL_DEMAND    =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5.xlsx','IDEAL_DEMAND');
H               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5.xlsx','HOLDING_COST');
K               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5.xlsx','FIXED_COST');

enddata

!Objective function;

MIN = @SUM (REP(J): (0.5*H*Q(J)*Y(J))+(K/Q(J))*( @SUM (PRODUCT(I): DEMAND(I)*X(I,J))));

!constraints;

!every item should be allocated to one replenishment only;
@FOR ( PRODUCT(I): @SUM (REP(J): X(I,J)) =1);

!x is an intejer = 0 or 1 (i.e. x is binary;
@FOR (PXR (I,J): @BIN (X(I,J)));

!y is binary;
@FOR (REP(J): @BIN (Y(J)));

!logical condition;
@FOR(PXR(I,J): X(I,J) <= Y(J));

!loading/capacity constraint;

@FOR(REP(J): (D1(J)- D2(J)+ @SUM (PRODUCT (I): DEMAND(I)*X(I,J)))= IDEAL_DEMAND(J)*Y(J));
@for (REP(J): (@SUM (PRODUCT (I): DEMAND(I)*X(I,J))) <= Q(J));

end
```

Grouping EXP5 X

```
!EXP5X;

Model:

Sets:

PRODUCT/1..20/:

DEMAND;

REP/1..12/:
Y,
Q,
IDEAL_DEMAND,
D1,
D2,
K;

PXR(PRODUCT,REP):
X;

endsets

data:

DEMAND          =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5X.xlsx','AVG_DEMAND');
Q               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5X.xlsx','Q');
IDEAL_DEMAND    =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5X.xlsx','IDEAL_DEMAND');
H               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5X.xlsx','HOLDING_COST');
K               =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5X.xlsx','FIXED_COST');

enddata

!Objective function;

MIN = @SUM (REP(J): (0.5*H*Q(J)*Y(J))+(K/Q(J))*( @SUM (PRODUCT(I): DEMAND(I)*X(I,J))));

!constraints;

!every item should be allocated to one replenishment only;
@FOR ( PRODUCT(I): @SUM (REP(J): X(I,J)) =1);

!x is an intejer = 0 or 1 (i.e. x is binary;
@FOR (PXR (I,J): @BIN (X(I,J)));

!y is binary;
@FOR (REP(J): @BIN (Y(J)));

!logical condition;
@FOR(PXR(I,J): X(I,J) <= Y(J));

!loading/capacity constraint;

@FOR(REP(J): (D1(J)- D2(J)+ @SUM (PRODUCT (I): DEMAND(I)*X(I,J)))= IDEAL_DEMAND(J)*Y(J));
@for (REP(J): (@SUM (PRODUCT (I): DEMAND(I)*X(I,J))) <= Q(J));

end
```

EXP 1 - Group A

```
! EXP 1 - Group A;

MODEL:

SETS:

! Generate a set of 6 products;
PRODUCT/1..6/:

! An order-up-to level, S, is assigned to each product, I;
S;

! Generate a set of 500 days;
DAY/1..1000/:

! Modulo operation is used to deteminr if day J is a review day or not and to determine if a
previously
  ordered batch will be recevied.

  The reminder of division of day, J, by the review period of the group, R ( Modulo
calculations);
MOD,

! The reminder of division of day, J, by the review period of the group, R, plus the lead
time, L;
MODRL;

! Product I at day J set;
PXD(PRODUCT,DAY):

! Product i at day j will have the following parameters;

INV,    ! Inventory position of product I at day J ;
INVE,   ! Inventory Position of product I at the end of day J ;
SOH,    ! Stock on Hand of product I at the end of day J;
SOHE,   ! Stock on Hand of product I at the end of day J;
Q,      ! Order quantity of product I at the end of day J;
D,! Demand of product I at day J;
D1,     ! Over Achivement;
D2,     ! Uner Achivement;
REC,    ! Item recevied at day J;
DEL,    ! Satesfied demand of product I at day J;
NDEL,   ! Un satedfied demand of product I at day J;
SLEVEl;! Seriveice level;


ENDSETS

DATA:

! Import Order-up-to level data for each product from Excelsheet -(R,S) Table;
S =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','S_LEVEL_GA');

! Given mean of demand and the standard deviation of demand,
  use excel to generate random demand for the needed number of days.
  Then use @OLE function to import demand data;
D = @OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','REAL_DEMAND_GA');

Cap= 10; ! The capacity of the truck allocated to this group;
L = 1;   ! Lead time;
R = 2;   ! Group review time calculated in the Excelsheet - (R,S) Table;

!Output to excel;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','SL') = SLEVEL;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','SOH') = SOH;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','Q_OUTPUT') = Q;

ENDDATA

! The following submodel is used to allocate the available truck capacity to the different
products in the group.
  The model will minimise the total deviation from the pre-identified order-up-to level;
```

```
SUBMODEL FIND_Q:

! At each day minimise the sum of the deveiation from S as a percenage of S of each product;
@FOR (DAY (J)| J #EQ# DAYN: MIN = @SUM ( PRODUCT(I): (D1(I,J)+D2(I,J))));

! Subject to the following constraints:

  1- The order quantity is a positive integer value;
@for ( pxd (i,J)| J #EQ# DAYN: @gin ( Q(i,J)));

! 2- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (J)| J #EQ# DAYN: @sum (product(i): q(i,j)) = cap);

! 3- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (j)| J #EQ# DAYN: @for (product (i): INV (I,J) + d1(i,j) - d2(i,j) + Q(i,j) =
s(i)));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D1(I,J) >= 0));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D2(I,J) >= 0));

ENDSUBMODEL


! This simple model is used if day j is not a review day to set Q(i,j) values to 0;
SUBMODEL NO_Q:

@FOR (DAY (J)| J #EQ# DAYN: @FOR (PRODUCT (I): Q(I,J) = 0));

ENDSUBMODEL


CALC:

! define the size of the repleishment period;
MXDAY = @SIZE (DAY);

! Set day to day 1;
DAYN = 1;

! Initiate a while loop to apply the submodels (when needed);
@WHILE ( DAYN #LE# MXDAY+1:

! First we distinguesh between day 1 and all the othr days;
@IFC ( DAYN #EQ# 1:

! For day 1, calculate the reminder of division of day, J, by the review period of the group,
R;
@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!for day 1, calculate he reminder of division of day, J, by the review period of the group, R,
plus the lead time, L;
@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

! Initiate the data for the first day;
! The next 4 functions will set SOH(i,j) = SOHE(I, J) = INV (I, J) = INVE(I, J) = S(I);
@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOHE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INV (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INVE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: REC (i,j) = 0));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SLEVEL (i,j) = 0));

! the seconf branch of the first @IFC function applies to days greater than 1;
! Here a set of calculations will be performed for any particular day greater than 1;
@ELSE@IFC (DAYN #GT# 1:

@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

@FOR (PRODUCT (I): @FOR (DAY (J) | J #EQ# DAYN: @IFC (DAYN #GE# L+1: REC(I,J) = Q(I, J-L);
```

```
@ELSE@IFC (DAYN #LT# L+1: REC (I,J) = 0;););));

@for ( product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = REC(I,J) +SOHE(i, j-1)));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): @IFC ( SOH(I,J) #GE# D(I,J): DEL(I,J) = D(I,J);
NDEL(I,J) = 0; SOHE (I,J) = SOH(I,J) - DEL (I,J);

@ELSE@IFC ( SOH(I,J) #LT# D(I,J): DEL(I,J) = SOH (I,J); NDEL(I,J) = D(I,J) - DEL(I,J); SOHE
(I,J) = 0;););));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): INV (I,J) = INVE (I,J-1)));

););

@for ( Day (j) | J #EQ# DAYN: @IFC ( MOD(J) #EQ# 0:

@SOLVE (FIND_Q);

@ELSE@IFC ( MOD(J) #GT# 0:

@SOLVE (NO_Q););););

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): INVE (I,J) = INV(I,J) -
D(I,J)+ Q(I,J)));

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): SLEVEL (I,J) = DEL (I,J)/ D
(I,J) ));

DAYN= DAYN + 1;);

ENDCALC

END
```

EXP 1 - Group B

```
! EXP 1 - Group B;

MODEL:

SETS:

! Generate a set of 6 products;
PRODUCT/1..14/:

! An order-up-to level, S, is assigned to each product, I;
S;

! Generate a set of 500 days;
DAY/1..1000/:

! Modulo operation is used to deteminr if day J is a review day or not and to determine if a
previously
  ordered batch will be recevied.

  The reminder of division of day, J, by the review period of the group, R ( Modulo
calculations);
MOD,

! The reminder of division of day, J, by the review period of the group, R, plus the lead
time, L;
MODRL;

! Product I at day J set;
PXD(PRODUCT,DAY):

! Product i at day j will have the following parameters;

INV,    ! Inventory position of product I at day J ;
INVE,   ! Inventory Position of product I at the end of day J ;
SOH,    ! Stock on Hand of product I at the end of day J;
SOHE,   ! Stock on Hand of product I at the end of day J;
Q,      ! Order quantity of product I at the end of day J;
D,! Demand of product I at day J;
D1,     ! Over Achivement;
D2,     ! Uner Achivement;
REC,    ! Item recevied at day J;
DEL,    ! Satesfied demand of product I at day J;
NDEL,   ! Un satedfied demand of product I at day J;
SLEVEl;! Seriveice level;


ENDSETS

DATA:

! Import Order-up-to level data for each product from Excelsheet -(R,S) Table;
S =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','S_LEVEL_GB');

! Given mean of demand and the standard deviation of demand,
  use excel to generate random demand for the needed number of days.
  Then use @OLE function to import demand data;
D = @OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','REAL_DEMAND_GB');

Cap= 15; ! The capacity of the truck allocated to this group;
L = 1;   ! Lead time;
R = 1;   ! Group review time calculated in the Excelsheet - (R,S) Table;

!Output to excel;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','SLGB') = SLEVEL;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','SOHGB') = SOH;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1.xlsx','QGB') = Q;

ENDDATA

! The following submodel is used to allocate the available truck capacity to the different
products in the group.
  The model will minimise the total deviation from the pre-identified order-up-to level;
```

```
SUBMODEL FIND_Q:

! At each day minimise the sum of the deveiation from S as a percenage of S of each product;
@FOR (DAY (J)| J #EQ# DAYN: MIN = @SUM ( PRODUCT(I): (D1(I,J)+D2(I,J))));

! Subject to the following constraints:

  1- The order quantity is a positive integer value;
@for ( pxd (i,J)| J #EQ# DAYN: @gin ( Q(i,J)));

! 2- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (J)| J #EQ# DAYN: @sum (product(i): q(i,j)) = cap);

! 3- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (j)| J #EQ# DAYN: @for (product (i): INV (I,J) + d1(i,j) - d2(i,j) + Q(i,j) =
s(i)));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D1(I,J) >= 0));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D2(I,J) >= 0));

ENDSUBMODEL


! This simple model is used if day j is not a review day to set Q(i,j) values to 0;
SUBMODEL NO_Q:

@FOR (DAY (J)| J #EQ# DAYN: @FOR (PRODUCT (I): Q(I,J) = 0));

ENDSUBMODEL


CALC:

! define the size of the repleishment period;
MXDAY = @SIZE (DAY);

! Set day to day 1;
DAYN = 1;

! Initiate a while loop to apply the submodels (when needed);
@WHILE ( DAYN #LE# MXDAY+1:

! First we distinguesh between day 1 and all the othr days;
@IFC ( DAYN #EQ# 1:

! For day 1, calculate the reminder of division of day, J, by the review period of the group,
R;
@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!for day 1, calculate he reminder of division of day, J, by the review period of the group, R,
plus the lead time, L;
@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

! Initiate the data for the first day;
! The next 4 functions will set SOH(i,j) = SOHE(I, J) = INV (I, J) = INVE(I, J) = S(I);
@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOHE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INV (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INVE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: REC (i,j) = 0));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SLEVEL (i,j) = 0));

! the seconf branch of the first @IFC function applies to days greater than 1;
! Here a set of calculations will be performed for any particular day greater than 1;
@ELSE@IFC (DAYN #GT# 1:

@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

@FOR (PRODUCT (I): @FOR (DAY (J) | J #EQ# DAYN: @IFC (DAYN #GE# L+1: REC(I,J) = Q(I, J-L);
```

```
@ELSE@IFC (DAYN #LT# L+1: REC (I,J) = 0;););));

@for ( product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = REC(I,J) +SOHE(i, j-1)));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): @IFC ( SOH(I,J) #GE# D(I,J): DEL(I,J) = D(I,J);
NDEL(I,J) = 0; SOHE (I,J) = SOH(I,J) - DEL (I,J);

@ELSE@IFC ( SOH(I,J) #LT# D(I,J): DEL(I,J) = SOH (I,J); NDEL(I,J) = D(I,J) - DEL(I,J); SOHE
(I,J) = 0;););));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): INV (I,J) = INVE (I,J-1)));

););

@for ( Day (j) | J #EQ# DAYN: @IFC ( MOD(J) #EQ# 0:

@SOLVE (FIND_Q);

@ELSE@IFC ( MOD(J) #GT# 0:

@SOLVE (NO_Q););););

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): INVE (I,J) = INV(I,J) -
D(I,J)+ Q(I,J)));

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): SLEVEL (I,J) = DEL (I,J)/ D
(I,J) ));

DAYN= DAYN + 1;);

ENDCALC

END
```

EXP 1X - Group A

```
! EXP 1X - Group A;

MODEL:

SETS:

! Generate a set of 6 products;
PRODUCT/1..3/:

! An order-up-to level, S, is assigned to each product, I;
S;

! Generate a set of 500 days;
DAY/1..1000/:

! Modulo operation is used to deteminr if day J is a review day or not and to determine if a
previously
  ordered batch will be recevied.

  The reminder of division of day, J, by the review period of the group, R ( Modulo
calculations);
MOD,

! The reminder of division of day, J, by the review period of the group, R, plus the lead
time, L;
MODRL;

! Product I at day J set;
PXD(PRODUCT,DAY):

! Product i at day j will have the following parameters;

INV,    ! Inventory position of product I at day J ;
INVE,   ! Inventory Position of product I at the end of day J ;
SOH,    ! Stock on Hand of product I at the end of day J;
SOHE,   ! Stock on Hand of product I at the end of day J;
Q,      ! Order quantity of product I at the end of day J;
D,! Demand of product I at day J;
D1,     ! Over Achivement;
D2,     ! Uner Achivement;
REC,    ! Item recevied at day J;
DEL,    ! Satesfied demand of product I at day J;
NDEL,   ! Un satedfied demand of product I at day J;
SLEVEl;! Seriveice level;


ENDSETS

DATA:

! Import Order-up-to level data for each product from Excelsheet -(R,S) Table;
S =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','S_LEVEL_GA');

! Given mean of demand and the standard deviation of demand,
  use excel to generate random demand for the needed number of days.
  Then use @OLE function to import demand data;
D = @OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','REAL_DEMAND_GA');

Cap= 5; ! The capacity of the truck allocated to this group;
L = 1;    ! Lead time;
R = 1;    ! Group review time calculated in the Excelsheet - (R,S) Table;

!Output to excel;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','SLGA') = SLEVEL;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','SOHGA') = SOHE;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','QGA') = Q;

ENDDATA

! The following submodel is used to allocate the available truck capacity to the different
products in the group.
  The model will minimise the total deviation from the pre-identified order-up-to level;
```

```
SUBMODEL FIND_Q:

! At each day minimise the sum of the deveiation from S as a percenage of S of each product;
@FOR (DAY (J)| J #EQ# DAYN: MIN = @SUM ( PRODUCT(I): (D1(I,J)+D2(I,J))));

! Subject to the following constraints:

  1- The order quantity is a positive integer value;
@for ( pxd (i,J)| J #EQ# DAYN: @gin ( Q(i,J)));

! 2- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (J)| J #EQ# DAYN: @sum (product(i): q(i,j)) = cap);

! 3- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (j)| J #EQ# DAYN: @for (product (i): INV (I,J) + d1(i,j) - d2(i,j) + Q(i,j) =
s(i)));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D1(I,J) >= 0));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D2(I,J) >= 0));

ENDSUBMODEL


! This simple model is used if day j is not a review day to set Q(i,j) values to 0;
SUBMODEL NO_Q:

@FOR (DAY (J)| J #EQ# DAYN: @FOR (PRODUCT (I): Q(I,J) = 0));

ENDSUBMODEL


CALC:

! define the size of the repleishment period;
MXDAY = @SIZE (DAY);

! Set day to day 1;
DAYN = 1;

! Initiate a while loop to apply the submodels (when needed);
@WHILE ( DAYN #LE# MXDAY+1:

! First we distinguesh between day 1 and all the othr days;
@IFC ( DAYN #EQ# 1:

! For day 1, calculate the reminder of division of day, J, by the review period of the group,
R;
@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!for day 1, calculate he reminder of division of day, J, by the review period of the group, R,
plus the lead time, L;
@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

! Initiate the data for the first day;
! The next 4 functions will set SOH(i,j) = SOHE(I, J) = INV (I, J) = INVE(I, J) = S(I);
@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOHE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INV (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INVE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: REC (i,j) = 0));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SLEVEL (i,j) = 0));

! the seconf branch of the first @IFC function applies to days greater than 1;
! Here a set of calculations will be performed for any particular day greater than 1;
@ELSE@IFC (DAYN #GT# 1:

@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

@FOR (PRODUCT (I): @FOR (DAY (J) | J #EQ# DAYN: @IFC (DAYN #GE# L+1: REC(I,J) = Q(I, J-L);
```

```
@ELSE@IFC (DAYN #LT# L+1: REC (I,J) = 0;););));

@for ( product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = REC(I,J) +SOHE(i, j-1)));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): @IFC ( SOH(I,J) #GE# D(I,J): DEL(I,J) = D(I,J);
NDEL(I,J) = 0; SOHE (I,J) = SOH(I,J) - DEL (I,J);

@ELSE@IFC ( SOH(I,J) #LT# D(I,J): DEL(I,J) = SOH (I,J); NDEL(I,J) = D(I,J) - DEL(I,J); SOHE
(I,J) = 0;););));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): INV (I,J) = INVE (I,J-1)));

););

@for ( Day (j) | J #EQ# DAYN: @IFC ( MOD(J) #EQ# 0:

@SOLVE (FIND_Q);

@ELSE@IFC ( MOD(J) #GT# 0:

@SOLVE (NO_Q););););

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): INVE (I,J) = INV(I,J) -
D(I,J)+ Q(I,J)));

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): SLEVEL (I,J) = DEL (I,J)/ D
(I,J) ));

DAYN= DAYN + 1;);

ENDCALC

END
```

EXP 1X - Group A

```
! EXP 1X - Group A;

MODEL:

SETS:

! Generate a set of 6 products;
PRODUCT/1..17/:

! An order-up-to level, S, is assigned to each product, I;
S;

! Generate a set of 500 days;
DAY/1..1000/:

! Modulo operation is used to deteminr if day J is a review day or not and to determine if a
previously
  ordered batch will be recevied.

  The reminder of division of day, J, by the review period of the group, R ( Modulo
calculations);
MOD,

! The reminder of division of day, J, by the review period of the group, R, plus the lead
time, L;
MODRL;

! Product I at day J set;
PXD(PRODUCT,DAY):

! Product i at day j will have the following parameters;

INV,    ! Inventory position of product I at day J ;
INVE,   ! Inventory Position of product I at the end of day J ;
SOH,    ! Stock on Hand of product I at the end of day J;
SOHE,   ! Stock on Hand of product I at the end of day J;
Q,      ! Order quantity of product I at the end of day J;
D,! Demand of product I at day J;
D1,     ! Over Achivement;
D2,     ! Uner Achivement;
REC,    ! Item recevied at day J;
DEL,    ! Satesfied demand of product I at day J;
NDEL,   ! Un satedfied demand of product I at day J;
SLEVEl;! Seriveice level;


ENDSETS

DATA:

! Import Order-up-to level data for each product from Excelsheet -(R,S) Table;
S =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','S_LEVEL_GB');

! Given mean of demand and the standard deviation of demand,
  use excel to generate random demand for the needed number of days.
  Then use @OLE function to import demand data;
D = @OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','REAL_DEMAND_GB');

Cap= 15; ! The capacity of the truck allocated to this group;
L = 1;   ! Lead time;
R = 1;   ! Group review time calculated in the Excelsheet - (R,S) Table;

!Output to excel;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','SLGB') = SLEVEL;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','SOHGB') = SOHE;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP1X.xlsx','QGB') = Q;

ENDDATA

! The following submodel is used to allocate the available truck capacity to the different
products in the group.
  The model will minimise the total deviation from the pre-identified order-up-to level;
```

```
SUBMODEL FIND_Q:

! At each day minimise the sum of the deveiation from S as a percenage of S of each product;
@FOR (DAY (J)| J #EQ# DAYN: MIN = @SUM ( PRODUCT(I): (D1(I,J)+D2(I,J))));

! Subject to the following constraints:

  1- The order quantity is a positive integer value;
@for ( pxd (i,J)| J #EQ# DAYN: @gin ( Q(i,J)));

! 2- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (J)| J #EQ# DAYN: @sum (product(i): q(i,j)) = cap);

! 3- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (j)| J #EQ# DAYN: @for (product (i): INV (I,J) + d1(i,j) - d2(i,j) + Q(i,j) =
s(i)));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D1(I,J) >= 0));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D2(I,J) >= 0));

ENDSUBMODEL


! This simple model is used if day j is not a review day to set Q(i,j) values to 0;
SUBMODEL NO_Q:

@FOR (DAY (J)| J #EQ# DAYN: @FOR (PRODUCT (I): Q(I,J) = 0));

ENDSUBMODEL


CALC:

! define the size of the repleishment period;
MXDAY = @SIZE (DAY);

! Set day to day 1;
DAYN = 1;

! Initiate a while loop to apply the submodels (when needed);
@WHILE ( DAYN #LE# MXDAY+1:

! First we distinguesh between day 1 and all the othr days;
@IFC ( DAYN #EQ# 1:

! For day 1, calculate the reminder of division of day, J, by the review period of the group,
R;
@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!for day 1, calculate he reminder of division of day, J, by the review period of the group, R,
plus the lead time, L;
@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

! Initiate the data for the first day;
! The next 4 functions will set SOH(i,j) = SOHE(I, J) = INV (I, J) = INVE(I, J) = S(I);
@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOHE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INV (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INVE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: REC (i,j) = 0));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SLEVEL (i,j) = 0));

! the seconf branch of the first @IFC function applies to days greater than 1;
! Here a set of calculations will be performed for any particular day greater than 1;
@ELSE@IFC (DAYN #GT# 1:

@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

@FOR (PRODUCT (I): @FOR (DAY (J) | J #EQ# DAYN: @IFC (DAYN #GE# L+1: REC(I,J) = Q(I, J-L);
```

```
@ELSE@IFC (DAYN #LT# L+1: REC (I,J) = 0;););));

@for ( product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = REC(I,J) +SOHE(i, j-1)));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): @IFC ( SOH(I,J) #GE# D(I,J): DEL(I,J) = D(I,J);
NDEL(I,J) = 0; SOHE (I,J) = SOH(I,J) - DEL (I,J);

@ELSE@IFC ( SOH(I,J) #LT# D(I,J): DEL(I,J) = SOH (I,J); NDEL(I,J) = D(I,J) - DEL(I,J); SOHE
(I,J) = 0;););));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): INV (I,J) = INVE (I,J-1)));

););

@for ( Day (j) | J #EQ# DAYN: @IFC ( MOD(J) #EQ# 0:

@SOLVE (FIND_Q);

@ELSE@IFC ( MOD(J) #GT# 0:

@SOLVE (NO_Q););););

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): INVE (I,J) = INV(I,J) -
D(I,J)+ Q(I,J)));

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): SLEVEL (I,J) = DEL (I,J)/ D
(I,J) ));

DAYN= DAYN + 1;);

ENDCALC

END
```

EXP 2 - Group A

```
! EXP 1 - Group A;

MODEL:

SETS:

! Generate a set of 6 products;
PRODUCT/1..8/:

! An order-up-to level, S, is assigned to each product, I;
S;

! Generate a set of 500 days;
DAY/1..1000/:

! Modulo operation is used to deteminr if day J is a review day or not and to determine if a
previously
  ordered batch will be recevied.

  The reminder of division of day, J, by the review period of the group, R ( Modulo
calculations);
MOD,

! The reminder of division of day, J, by the review period of the group, R, plus the lead
time, L;
MODRL;

! Product I at day J set;
PXD(PRODUCT,DAY):

! Product i at day j will have the following parameters;

INV,    ! Inventory position of product I at day J ;
INVE,   ! Inventory Position of product I at the end of day J ;
SOH,    ! Stock on Hand of product I at the end of day J;
SOHE,   ! Stock on Hand of product I at the end of day J;
Q,      ! Order quantity of product I at the end of day J;
D,! Demand of product I at day J;
D1,     ! Over Achivement;
D2,     ! Uner Achivement;
REC,    ! Item recevied at day J;
DEL,    ! Satesfied demand of product I at day J;
NDEL,   ! Un satedfied demand of product I at day J;
SLEVEl;! Seriveice level;


ENDSETS

DATA:

! Import Order-up-to level data for each product from Excelsheet -(R,S) Table;
S =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','S_LEVEL_GA');

! Given mean of demand and the standard deviation of demand,
  use excel to generate random demand for the needed number of days.
  Then use @OLE function to import demand data;
D = @OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','REAL_DEMAND_GA');

Cap= 15; ! The capacity of the truck allocated to this group;
L = 1;   ! Lead time;
R = 3;   ! Group review time calculated in the Excelsheet - (R,S) Table;

!Output to excel;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','SL') = SLEVEL;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','SOH') = SOH;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','QGA') = Q;

ENDDATA

! The following submodel is used to allocate the available truck capacity to the different
products in the group.
  The model will minimise the total deviation from the pre-identified order-up-to level;
```

```
SUBMODEL FIND_Q:

! At each day minimise the sum of the deveiation from S as a percenage of S of each product;
@FOR (DAY (J)| J #EQ# DAYN: MIN = @SUM ( PRODUCT(I): (D1(I,J)+D2(I,J))));

! Subject to the following constraints:

  1- The order quantity is a positive integer value;
@for ( pxd (i,J)| J #EQ# DAYN: @gin ( Q(i,J)));

! 2- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (J)| J #EQ# DAYN: @sum (product(i): q(i,j)) = cap);

! 3- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (j)| J #EQ# DAYN: @for (product (i): INV (I,J) + d1(i,j) - d2(i,j) + Q(i,j) =
s(i)));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D1(I,J) >= 0));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D2(I,J) >= 0));

ENDSUBMODEL


! This simple model is used if day j is not a review day to set Q(i,j) values to 0;
SUBMODEL NO_Q:

@FOR (DAY (J)| J #EQ# DAYN: @FOR (PRODUCT (I): Q(I,J) = 0));

ENDSUBMODEL


CALC:

! define the size of the repleishment period;
MXDAY = @SIZE (DAY);

! Set day to day 1;
DAYN = 1;

! Initiate a while loop to apply the submodels (when needed);
@WHILE ( DAYN #LE# MXDAY+1:

! First we distinguesh between day 1 and all the othr days;
@IFC ( DAYN #EQ# 1:

! For day 1, calculate the reminder of division of day, J, by the review period of the group,
R;
@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!for day 1, calculate he reminder of division of day, J, by the review period of the group, R,
plus the lead time, L;
@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

! Initiate the data for the first day;
! The next 4 functions will set SOH(i,j) = SOHE(I, J) = INV (I, J) = INVE(I, J) = S(I);
@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOHE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INV (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INVE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: REC (i,j) = 0));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SLEVEL (i,j) = 0));

! the seconf branch of the first @IFC function applies to days greater than 1;
! Here a set of calculations will be performed for any particular day greater than 1;
@ELSE@IFC (DAYN #GT# 1:

@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

@FOR (PRODUCT (I): @FOR (DAY (J) | J #EQ# DAYN: @IFC (DAYN #GE# L+1: REC(I,J) = Q(I, J-L);
```

```
@ELSE@IFC (DAYN #LT# L+1: REC (I,J) = 0;););));

@for ( product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = REC(I,J) +SOHE(i, j-1)));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): @IFC ( SOH(I,J) #GE# D(I,J): DEL(I,J) = D(I,J);
NDEL(I,J) = 0; SOHE (I,J) = SOH(I,J) - DEL (I,J);

@ELSE@IFC ( SOH(I,J) #LT# D(I,J): DEL(I,J) = SOH (I,J); NDEL(I,J) = D(I,J) - DEL(I,J); SOHE
(I,J) = 0;););));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): INV (I,J) = INVE (I,J-1)));

););

@for ( Day (j) | J #EQ# DAYN: @IFC ( MOD(J) #EQ# 0:

@SOLVE (FIND_Q);

@ELSE@IFC ( MOD(J) #GT# 0:

@SOLVE (NO_Q););););

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): INVE (I,J) = INV(I,J) -
D(I,J)+ Q(I,J)));

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): SLEVEL (I,J) = DEL (I,J)/ D
(I,J) ));

DAYN= DAYN + 1;);

ENDCALC

END
```

EXP 2 - Group B

```
! EXP 2 - Group B;

MODEL:

SETS:

! Generate a set of 6 products;
PRODUCT/1..12/:

! An order-up-to level, S, is assigned to each product, I;
S;

! Generate a set of 500 days;
DAY/1..1000/:

! Modulo operation is used to deteminr if day J is a review day or not and to determine if a
previously
  ordered batch will be recevied.

  The reminder of division of day, J, by the review period of the group, R ( Modulo
calculations);
MOD,

! The reminder of division of day, J, by the review period of the group, R, plus the lead
time, L;
MODRL;

! Product I at day J set;
PXD(PRODUCT,DAY):

! Product i at day j will have the following parameters;

INV,    ! Inventory position of product I at day J ;
INVE,   ! Inventory Position of product I at the end of day J ;
SOH,    ! Stock on Hand of product I at the end of day J;
SOHE,   ! Stock on Hand of product I at the end of day J;
Q,      ! Order quantity of product I at the end of day J;
D,! Demand of product I at day J;
D1,     ! Over Achivement;
D2,     ! Uner Achivement;
REC,    ! Item recevied at day J;
DEL,    ! Satesfied demand of product I at day J;
NDEL,   ! Un satedfied demand of product I at day J;
SLEVEl;! Seriveice level;


ENDSETS

DATA:

! Import Order-up-to level data for each product from Excelsheet -(R,S) Table;
S =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','S_LEVEL_GB');

! Given mean of demand and the standard deviation of demand,
  use excel to generate random demand for the needed number of days.
  Then use @OLE function to import demand data;
D = @OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','REAL_DEMAND_GB');

Cap= 15; ! The capacity of the truck allocated to this group;
L = 1;   ! Lead time;
R = 1;   ! Group review time calculated in the Excelsheet - (R,S) Table;

!Output to excel;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','SLGB') = SLEVEL;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','SOHGB') = SOH;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2.xlsx','QGB') = Q;

ENDDATA

! The following submodel is used to allocate the available truck capacity to the different
products in the group.
  The model will minimise the total deviation from the pre-identified order-up-to level;
```

```
SUBMODEL FIND_Q:

! At each day minimise the sum of the deveiation from S as a percenage of S of each product;
@FOR (DAY (J)| J #EQ# DAYN: MIN = @SUM ( PRODUCT(I): (D1(I,J)+D2(I,J))));

! Subject to the following constraints:

  1- The order quantity is a positive integer value;
@for ( pxd (i,J)| J #EQ# DAYN: @gin ( Q(i,J)));

! 2- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (J)| J #EQ# DAYN: @sum (product(i): q(i,j)) = cap);

! 3- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (j)| J #EQ# DAYN: @for (product (i): INV (I,J) + d1(i,j) - d2(i,j) + Q(i,j) =
s(i)));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D1(I,J) >= 0));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D2(I,J) >= 0));

ENDSUBMODEL


! This simple model is used if day j is not a review day to set Q(i,j) values to 0;
SUBMODEL NO_Q:

@FOR (DAY (J)| J #EQ# DAYN: @FOR (PRODUCT (I): Q(I,J) = 0));

ENDSUBMODEL


CALC:

! define the size of the repleishment period;
MXDAY = @SIZE (DAY);

! Set day to day 1;
DAYN = 1;

! Initiate a while loop to apply the submodels (when needed);
@WHILE ( DAYN #LE# MXDAY+1:

! First we distinguesh between day 1 and all the othr days;
@IFC ( DAYN #EQ# 1:

! For day 1, calculate the reminder of division of day, J, by the review period of the group,
R;
@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!for day 1, calculate he reminder of division of day, J, by the review period of the group, R,
plus the lead time, L;
@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

! Initiate the data for the first day;
! The next 4 functions will set SOH(i,j) = SOHE(I, J) = INV (I, J) = INVE(I, J) = S(I);
@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOHE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INV (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INVE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: REC (i,j) = 0));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SLEVEL (i,j) = 0));

! the seconf branch of the first @IFC function applies to days greater than 1;
! Here a set of calculations will be performed for any particular day greater than 1;
@ELSE@IFC (DAYN #GT# 1:

@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

@FOR (PRODUCT (I): @FOR (DAY (J) | J #EQ# DAYN: @IFC (DAYN #GE# L+1: REC(I,J) = Q(I, J-L);
```

```
@ELSE@IFC (DAYN #LT# L+1: REC (I,J) = 0;););));

@for ( product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = REC(I,J) +SOHE(i, j-1)));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): @IFC ( SOH(I,J) #GE# D(I,J): DEL(I,J) = D(I,J);
NDEL(I,J) = 0; SOHE (I,J) = SOH(I,J) - DEL (I,J);

@ELSE@IFC ( SOH(I,J) #LT# D(I,J): DEL(I,J) = SOH (I,J); NDEL(I,J) = D(I,J) - DEL(I,J); SOHE
(I,J) = 0;););));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): INV (I,J) = INVE (I,J-1)));

););

@for ( Day (j) | J #EQ# DAYN: @IFC ( MOD(J) #EQ# 0:

@SOLVE (FIND_Q);

@ELSE@IFC ( MOD(J) #GT# 0:

@SOLVE (NO_Q););););

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): INVE (I,J) = INV(I,J) -
D(I,J)+ Q(I,J)));

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): SLEVEL (I,J) = DEL (I,J)/ D
(I,J) ));

DAYN= DAYN + 1;);

ENDCALC

END
```

EXP 2X - Group A

```
! EXP 2X - Group A;

MODEL:

SETS:

! Generate a set of 6 products;
PRODUCT/1..18/:

! An order-up-to level, S, is assigned to each product, I;
S;

! Generate a set of 500 days;
DAY/1..1000/:

! Modulo operation is used to deteminr if day J is a review day or not and to determine if a
previously
  ordered batch will be recevied.

  The reminder of division of day, J, by the review period of the group, R ( Modulo
calculations);
MOD,

! The reminder of division of day, J, by the review period of the group, R, plus the lead
time, L;
MODRL;

! Product I at day J set;
PXD(PRODUCT,DAY):

! Product i at day j will have the following parameters;

INV,    ! Inventory position of product I at day J ;
INVE,   ! Inventory Position of product I at the end of day J ;
SOH,    ! Stock on Hand of product I at the end of day J;
SOHE,   ! Stock on Hand of product I at the end of day J;
Q,      ! Order quantity of product I at the end of day J;
D,! Demand of product I at day J;
D1,     ! Over Achivement;
D2,     ! Uner Achivement;
REC,    ! Item recevied at day J;
DEL,    ! Satesfied demand of product I at day J;
NDEL,   ! Un satedfied demand of product I at day J;
SLEVEl;! Seriveice level;


ENDSETS

DATA:

! Import Order-up-to level data for each product from Excelsheet -(R,S) Table;
S =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','S_LEVEL_GA');

! Given mean of demand and the standard deviation of demand,
  use excel to generate random demand for the needed number of days.
  Then use @OLE function to import demand data;
D = @OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','REAL_DEMAND_GA');

Cap= 15; ! The capacity of the truck allocated to this group;
L = 1;   ! Lead time;
R = 1;   ! Group review time calculated in the Excelsheet - (R,S) Table;

!Output to excel;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','SLGA') = SLEVEL;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','SOHGA') = SOHE;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','QGA') = Q;

ENDDATA

! The following submodel is used to allocate the available truck capacity to the different
products in the group.
  The model will minimise the total deviation from the pre-identified order-up-to level;
```

```
SUBMODEL FIND_Q:

! At each day minimise the sum of the deveiation from S as a percenage of S of each product;
@FOR (DAY (J)| J #EQ# DAYN: MIN = @SUM ( PRODUCT(I): (D1(I,J)+D2(I,J))));

! Subject to the following constraints:

  1- The order quantity is a positive integer value;
@for ( pxd (i,J)| J #EQ# DAYN: @gin ( Q(i,J)));

! 2- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (J)| J #EQ# DAYN: @sum (product(i): q(i,j)) = cap);

! 3- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (j)| J #EQ# DAYN: @for (product (i): INV (I,J) + d1(i,j) - d2(i,j) + Q(i,j) =
s(i)));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D1(I,J) >= 0));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D2(I,J) >= 0));

ENDSUBMODEL


! This simple model is used if day j is not a review day to set Q(i,j) values to 0;
SUBMODEL NO_Q:

@FOR (DAY (J)| J #EQ# DAYN: @FOR (PRODUCT (I): Q(I,J) = 0));

ENDSUBMODEL


CALC:

! define the size of the repleishment period;
MXDAY = @SIZE (DAY);

! Set day to day 1;
DAYN = 1;

! Initiate a while loop to apply the submodels (when needed);
@WHILE ( DAYN #LE# MXDAY+1:

! First we distinguesh between day 1 and all the othr days;
@IFC ( DAYN #EQ# 1:

! For day 1, calculate the reminder of division of day, J, by the review period of the group,
R;
@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!for day 1, calculate he reminder of division of day, J, by the review period of the group, R,
plus the lead time, L;
@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

! Initiate the data for the first day;
! The next 4 functions will set SOH(i,j) = SOHE(I, J) = INV (I, J) = INVE(I, J) = S(I);
@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOHE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INV (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INVE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: REC (i,j) = 0));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SLEVEL (i,j) = 0));

! the seconf branch of the first @IFC function applies to days greater than 1;
! Here a set of calculations will be performed for any particular day greater than 1;
@ELSE@IFC (DAYN #GT# 1:

@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

@FOR (PRODUCT (I): @FOR (DAY (J) | J #EQ# DAYN: @IFC (DAYN #GE# L+1: REC(I,J) = Q(I, J-L);
```

```
@ELSE@IFC (DAYN #LT# L+1: REC (I,J) = 0;););));

@for ( product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = REC(I,J) +SOHE(i, j-1)));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): @IFC ( SOH(I,J) #GE# D(I,J): DEL(I,J) = D(I,J);
NDEL(I,J) = 0; SOHE (I,J) = SOH(I,J) - DEL (I,J);

@ELSE@IFC ( SOH(I,J) #LT# D(I,J): DEL(I,J) = SOH (I,J); NDEL(I,J) = D(I,J) - DEL(I,J); SOHE
(I,J) = 0;););));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): INV (I,J) = INVE (I,J-1)));

););

@for ( Day (j) | J #EQ# DAYN: @IFC ( MOD(J) #EQ# 0:

@SOLVE (FIND_Q);

@ELSE@IFC ( MOD(J) #GT# 0:

@SOLVE (NO_Q););););

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): INVE (I,J) = INV(I,J) -
D(I,J)+ Q(I,J)));

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): SLEVEL (I,J) = DEL (I,J)/ D
(I,J) ));

DAYN= DAYN + 1;);

ENDCALC

END
```

EXP 2X - Group B

```
! EXP 2X - Group B;

MODEL:

SETS:

! Generate a set of 6 products;
PRODUCT/1..9/:

! An order-up-to level, S, is assigned to each product, I;
S;

! Generate a set of 500 days;
DAY/1..1000/:

! Modulo operation is used to deteminr if day J is a review day or not and to determine if a
previously
  ordered batch will be recevied.

  The reminder of division of day, J, by the review period of the group, R ( Modulo
calculations);
MOD,

! The reminder of division of day, J, by the review period of the group, R, plus the lead
time, L;
MODRL;

! Product I at day J set;
PXD(PRODUCT,DAY):

! Product i at day j will have the following parameters;

INV,    ! Inventory position of product I at day J ;
INVE,   ! Inventory Position of product I at the end of day J ;
SOH,    ! Stock on Hand of product I at the end of day J;
SOHE,   ! Stock on Hand of product I at the end of day J;
Q,      ! Order quantity of product I at the end of day J;
D,! Demand of product I at day J;
D1,     ! Over Achivement;
D2,     ! Uner Achivement;
REC,    ! Item recevied at day J;
DEL,    ! Satesfied demand of product I at day J;
NDEL,   ! Un satedfied demand of product I at day J;
SLEVEl;! Seriveice level;


ENDSETS

DATA:

! Import Order-up-to level data for each product from Excelsheet -(R,S) Table;
S =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','S_LEVEL_GB');

! Given mean of demand and the standard deviation of demand,
  use excel to generate random demand for the needed number of days.
  Then use @OLE function to import demand data;
D = @OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','REAL_DEMAND_GB');

Cap= 15; ! The capacity of the truck allocated to this group;
L = 1;   ! Lead time;
R = 3;   ! Group review time calculated in the Excelsheet - (R,S) Table;

!Output to excel;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','SLGB') = SLEVEL;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','SOHGB') = SOH;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP2X.xlsx','QGB') = Q;

ENDDATA

! The following submodel is used to allocate the available truck capacity to the different
products in the group.
  The model will minimise the total deviation from the pre-identified order-up-to level;
```

```
SUBMODEL FIND_Q:

! At each day minimise the sum of the deveiation from S as a percenage of S of each product;
@FOR (DAY (J)| J #EQ# DAYN: MIN = @SUM ( PRODUCT(I): (D1(I,J)+D2(I,J))));

! Subject to the following constraints:

  1- The order quantity is a positive integer value;
@for ( pxd (i,J)| J #EQ# DAYN: @gin ( Q(i,J)));

! 2- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (J)| J #EQ# DAYN: @sum (product(i): q(i,j)) = cap);

! 3- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (j)| J #EQ# DAYN: @for (product (i): INV (I,J) + d1(i,j) - d2(i,j) + Q(i,j) =
s(i)));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D1(I,J) >= 0));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D2(I,J) >= 0));

ENDSUBMODEL


! This simple model is used if day j is not a review day to set Q(i,j) values to 0;
SUBMODEL NO_Q:

@FOR (DAY (J)| J #EQ# DAYN: @FOR (PRODUCT (I): Q(I,J) = 0));

ENDSUBMODEL


CALC:

! define the size of the repleishment period;
MXDAY = @SIZE (DAY);

! Set day to day 1;
DAYN = 1;

! Initiate a while loop to apply the submodels (when needed);
@WHILE ( DAYN #LE# MXDAY+1:

! First we distinguesh between day 1 and all the othr days;
@IFC ( DAYN #EQ# 1:

! For day 1, calculate the reminder of division of day, J, by the review period of the group,
R;
@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!for day 1, calculate he reminder of division of day, J, by the review period of the group, R,
plus the lead time, L;
@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

! Initiate the data for the first day;
! The next 4 functions will set SOH(i,j) = SOHE(I, J) = INV (I, J) = INVE(I, J) = S(I);
@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOHE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INV (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INVE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: REC (i,j) = 0));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SLEVEL (i,j) = 0));

! the seconf branch of the first @IFC function applies to days greater than 1;
! Here a set of calculations will be performed for any particular day greater than 1;
@ELSE@IFC (DAYN #GT# 1:

@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

@FOR (PRODUCT (I): @FOR (DAY (J) | J #EQ# DAYN: @IFC (DAYN #GE# L+1: REC(I,J) = Q(I, J-L);
```

```
@ELSE@IFC (DAYN #LT# L+1: REC (I,J) = 0;););));

@for ( product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = REC(I,J) +SOHE(i, j-1)));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): @IFC ( SOH(I,J) #GE# D(I,J): DEL(I,J) = D(I,J);
NDEL(I,J) = 0; SOHE (I,J) = SOH(I,J) - DEL (I,J);

@ELSE@IFC ( SOH(I,J) #LT# D(I,J): DEL(I,J) = SOH (I,J); NDEL(I,J) = D(I,J) - DEL(I,J); SOHE
(I,J) = 0;););));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): INV (I,J) = INVE (I,J-1)));

););

@for ( Day (j) | J #EQ# DAYN: @IFC ( MOD(J) #EQ# 0:

@SOLVE (FIND_Q);

@ELSE@IFC ( MOD(J) #GT# 0:

@SOLVE (NO_Q););););

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): INVE (I,J) = INV(I,J) -
D(I,J)+ Q(I,J)));

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): SLEVEL (I,J) = DEL (I,J)/ D
(I,J) ));

DAYN= DAYN + 1;);

ENDCALC

END
```

```
EXP 4
! EXP 4;

MODEL:

SETS:

! Generate a set of 6 products;
PRODUCT/1..20/:

! An order-up-to level, S, is assigned to each product, I;
S;

! Generate a set of 500 days;
DAY/1..1000/:

! Modulo operation is used to deteminr if day J is a review day or not and to determine if a
previously
  ordered batch will be recevied.

  The reminder of division of day, J, by the review period of the group, R ( Modulo
calculations);
MOD,

! The reminder of division of day, J, by the review period of the group, R, plus the lead
time, L;
MODRL;

! Product I at day J set;
PXD(PRODUCT,DAY):

! Product i at day j will have the following parameters;

INV,    ! Inventory position of product I at day J ;
INVE,   ! Inventory Position of product I at the end of day J ;
SOH,    ! Stock on Hand of product I at the end of day J;
SOHE,   ! Stock on Hand of product I at the end of day J;
Q,      ! Order quantity of product I at the end of day J;
D,! Demand of product I at day J;
D1,     ! Over Achivement;
D2,     ! Uner Achivement;
REC,    ! Item recevied at day J;
DEL,    ! Satesfied demand of product I at day J;
NDEL,   ! Un satedfied demand of product I at day J;
SLEVEl;! Seriveice level;


ENDSETS

DATA:

! Import Order-up-to level data for each product from Excelsheet -(R,S) Table;
S =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP4.xlsx','S_LEVEL_GA');

! Given mean of demand and the standard deviation of demand,
  use excel to generate random demand for the needed number of days.
  Then use @OLE function to import demand data;
D = @OLE('\Users\Cripps Hire Laptop\Desktop\EXP4.xlsx','REAL_DEMAND_GA');

Cap= 20; ! The capacity of the truck allocated to this group;
L = 1;   ! Lead time;
R = 1;   ! Group review time calculated in the Excelsheet - (R,S) Table;

!Output to excel;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP4.xlsx','SL') = SLEVEL;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP4.xlsx','SOH') = SOH;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP4.xlsx','QGA') = Q;

ENDDATA

! The following submodel is used to allocate the available truck capacity to the different
products in the group.
  The model will minimise the total deviation from the pre-identified order-up-to level;
SUBMODEL FIND_Q:
```

```
! At each day minimise the sum of the deveiation from S as a percenage of S of each product;
@FOR (DAY (J)| J #EQ# DAYN: MIN = @SUM ( PRODUCT(I): (D1(I,J)+D2(I,J))));

! Subject to the following constraints:

  1- The order quantity is a positive integer value;
@for ( pxd (i,J)| J #EQ# DAYN: @gin ( Q(i,J)));

! 2- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (J)| J #EQ# DAYN: @sum (product(i): q(i,j)) = cap);

! 3- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (j)| J #EQ# DAYN: @for (product (i): INV (I,J) + d1(i,j) - d2(i,j) + Q(i,j) =
s(i)));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D1(I,J) >= 0));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D2(I,J) >= 0));

!@for ( day (j)| J #EQ# DAYN #and# dayn #gt# 1: @for (product (i): (inv(i,j)+Q(i,j))>=
(D(i,j))));

!@for ( day (j)| J #EQ# DAYN #and# dayn #gt# 1: @for (product (i): SOH(i,j) > 1));

ENDSUBMODEl


! This simple model is used if day j is not a review day to set Q(i,j) values to 0;
SUBMODEL NO_Q:

@FOR (DAY (J)| J #EQ# DAYN: @FOR (PRODUCT (I): Q(I,J) = 0));

ENDSUBMODEL


CALC:

! define the size of the repleishment period;
MXDAY = @SIZE (DAY);

! Set day to day 1;
DAYN = 1;

! Initiate a while loop to apply the submodels (when needed);
@WHILE ( DAYN #LE# MXDAY+1:

! First we distinguesh between day 1 and all the othr days;
@IFC ( DAYN #EQ# 1:

! For day 1, calculate the reminder of division of day, J, by the review period of the group,
R;
@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!for day 1, calculate he reminder of division of day, J, by the review period of the group, R,
plus the lead time, L;
@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

! Initiate the data for the first day;
! The next 4 functions will set SOH(i,j) = SOHE(I, J) = INV (I, J) = INVE(I, J) = S(I);
@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOHE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INV (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INVE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: REC (i,j) = 0));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SLEVEL (i,j) = 0));

! the seconf branch of the first @IFC function applies to days greater than 1;
! Here a set of calculations will be performed for any particular day greater than 1;
@ELSE@IFC (DAYN #GT# 1:

@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));
```

```
!@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

@FOR (PRODUCT (I): @FOR (DAY (J) | J #EQ# DAYN: @IFC (DAYN #GE# L+1: REC(I,J) = Q(I, J-L);

@ELSE@IFC (DAYN #LT# L+1: REC (I,J) = 0;););));

@for ( product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = REC(I,J) +SOHE(i, j-1)));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): @IFC ( SOH(I,J) #GE# D(I,J): DEL(I,J) = D(I,J);
NDEL(I,J) = 0; SOHE (I,J) = SOH(I,J) - DEL (I,J);

@ELSE@IFC ( SOH(I,J) #LT# D(I,J): DEL(I,J) = SOH (I,J); NDEL(I,J) = D(I,J) - DEL(I,J); SOHE
(I,J) = 0;););));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): INV (I,J) = INVE (I,J-1)));

););

@for ( Day (j) | J #EQ# DAYN: @IFC ( MOD(J) #EQ# 0:

@SOLVE (FIND_Q);

@ELSE@IFC ( MOD(J) #GT# 0:

@SOLVE (NO_Q););););

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): INVE (I,J) = INV(I,J) -
D(I,J)+ Q(I,J)));

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): SLEVEL (I,J) = DEL (I,J)/ D
(I,J) ));

DAYN= DAYN + 1;);

ENDCALC

END
```

EXP 4X

```
! EXP 4X;

MODEL:

SETS:

! Generate a set of 6 products;
PRODUCT/1..20/:

! An order-up-to level, S, is assigned to each product, I;
S;

! Generate a set of 500 days;
DAY/1..1000/:

! Modulo operation is used to deteminr if day J is a review day or not and to determine if a
previously
  ordered batch will be recevied.

  The reminder of division of day, J, by the review period of the group, R ( Modulo
calculations);
MOD,

! The reminder of division of day, J, by the review period of the group, R, plus the lead
time, L;
MODRL;

! Product I at day J set;
PXD(PRODUCT,DAY):

! Product i at day j will have the following parameters;

INV,    ! Inventory position of product I at day J ;
INVE,   ! Inventory Position of product I at the end of day J ;
SOH,    ! Stock on Hand of product I at the end of day J;
SOHE,   ! Stock on Hand of product I at the end of day J;
Q,      ! Order quantity of product I at the end of day J;
D,! Demand of product I at day J;
D1,     ! Over Achivement;
D2,     ! Uner Achivement;
REC,    ! Item recevied at day J;
DEL,    ! Satesfied demand of product I at day J;
NDEL,   ! Un satedfied demand of product I at day J;
SLEVEl;! Seriveice level;


ENDSETS

DATA:

! Import Order-up-to level data for each product from Excelsheet -(R,S) Table;
S =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP4X.xlsx','S_LEVEL_GA');

! Given mean of demand and the standard deviation of demand,
  use excel to generate random demand for the needed number of days.
  Then use @OLE function to import demand data;
D = @OLE('\Users\Cripps Hire Laptop\Desktop\EXP4X.xlsx','REAL_DEMAND_GA');

Cap= 20; ! The capacity of the truck allocated to this group;
L = 1;   ! Lead time;
R = 1;   ! Group review time calculated in the Excelsheet - (R,S) Table;

!Output to excel;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP4X.xlsx','SLGA') = SLEVEL;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP4X.xlsx','SOHGA') = SOH;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP4X.xlsx','QGA') = Q;

ENDDATA

! The following submodel is used to allocate the available truck capacity to the different
products in the group.
  The model will minimise the total deviation from the pre-identified order-up-to level;
```

```
SUBMODEL FIND_Q:

! At each day minimise the sum of the deveiation from S as a percenage of S of each product;
@FOR (DAY (J)| J #EQ# DAYN: MIN = @SUM ( PRODUCT(I): (D1(I,J)+D2(I,J))));

! Subject to the following constraints:

  1- The order quantity is a positive integer value;
@for ( pxd (i,J)| J #EQ# DAYN: @gin ( Q(i,J)));

! 2- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (J)| J #EQ# DAYN: @sum (product(i): q(i,j)) = cap);

! 3- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (j)| J #EQ# DAYN: @for (product (i): INV (I,J) + d1(i,j) - d2(i,j) + Q(i,j) =
s(i)));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D1(I,J) >= 0));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D2(I,J) >= 0));

ENDSUBMODEL


! This simple model is used if day j is not a review day to set Q(i,j) values to 0;
SUBMODEL NO_Q:

@FOR (DAY (J)| J #EQ# DAYN: @FOR (PRODUCT (I): Q(I,J) = 0));

ENDSUBMODEL


CALC:

! define the size of the repleishment period;
MXDAY = @SIZE (DAY);

! Set day to day 1;
DAYN = 1;

! Initiate a while loop to apply the submodels (when needed);
@WHILE ( DAYN #LE# MXDAY+1:

! First we distinguesh between day 1 and all the othr days;
@IFC ( DAYN #EQ# 1:

! For day 1, calculate the reminder of division of day, J, by the review period of the group,
R;
@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!for day 1, calculate he reminder of division of day, J, by the review period of the group, R,
plus the lead time, L;
@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

! Initiate the data for the first day;
! The next 4 functions will set SOH(i,j) = SOHE(I, J) = INV (I, J) = INVE(I, J) = S(I);
@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOHE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INV (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INVE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: REC (i,j) = 0));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SLEVEL (i,j) = 0));

! the seconf branch of the first @IFC function applies to days greater than 1;
! Here a set of calculations will be performed for any particular day greater than 1;
@ELSE@IFC (DAYN #GT# 1:

@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));
```

```
@FOR (PRODUCT (I): @FOR (DAY (J) | J #EQ# DAYN: @IFC (DAYN #GE# L+1: REC(I,J) = Q(I, J-L);

@ELSE@IFC (DAYN #LT# L+1: REC (I,J) = 0;););));

@for ( product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = REC(I,J) +SOHE(i, j-1)));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): @IFC ( SOH(I,J) #GE# D(I,J): DEL(I,J) = D(I,J);
NDEL(I,J) = 0; SOHE (I,J) = SOH(I,J) - DEL (I,J);

@ELSE@IFC ( SOH(I,J) #LT# D(I,J): DEL(I,J) = SOH (I,J); NDEL(I,J) = D(I,J) - DEL(I,J); SOHE
(I,J) = 0;););));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): INV (I,J) = INVE (I,J-1)));

););

@for ( Day (j) | J #EQ# DAYN: @IFC ( MOD(J) #EQ# 0:

@SOLVE (FIND_Q);

@ELSE@IFC ( MOD(J) #GT# 0:

@SOLVE (NO_Q););););

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): INVE (I,J) = INV(I,J) -
D(I,J)+ Q(I,J)));

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): SLEVEL (I,J) = DEL (I,J)/ D
(I,J) ));

DAYN= DAYN + 1;);

ENDCALC

END
```

```
EXP 5
! EXP 5;

MODEL:

SETS:

! Generate a set of 6 products;
PRODUCT/1..20/:

! An order-up-to level, S, is assigned to each product, I;
S;

! Generate a set of 500 days;
DAY/1..1000/:

! Modulo operation is used to deteminr if day J is a review day or not and to determine if a
previously
  ordered batch will be recevied.

  The reminder of division of day, J, by the review period of the group, R ( Modulo
calculations);
MOD,

! The reminder of division of day, J, by the review period of the group, R, plus the lead
time, L;
MODRL;

! Product I at day J set;
PXD(PRODUCT,DAY):

! Product i at day j will have the following parameters;

INV,    ! Inventory position of product I at day J ;
INVE,   ! Inventory Position of product I at the end of day J ;
SOH,    ! Stock on Hand of product I at the end of day J;
SOHE,   ! Stock on Hand of product I at the end of day J;
Q,      ! Order quantity of product I at the end of day J;
D,! Demand of product I at day J;
D1,     ! Over Achivement;
D2,     ! Uner Achivement;
REC,    ! Item recevied at day J;
DEL,    ! Satesfied demand of product I at day J;
NDEL,   ! Un satedfied demand of product I at day J;
SLEVEl;! Seriveice level;


ENDSETS

DATA:

! Import Order-up-to level data for each product from Excelsheet -(R,S) Table;
S =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5.xlsx','S_LEVEL_GA');

! Given mean of demand and the standard deviation of demand,
  use excel to generate random demand for the needed number of days.
  Then use @OLE function to import demand data;
D = @OLE('\Users\Cripps Hire Laptop\Desktop\EXP5.xlsx','REAL_DEMAND_GA');

Cap= 20; ! The capacity of the truck allocated to this group;
L = 1;   ! Lead time;
R = 1;   ! Group review time calculated in the Excelsheet - (R,S) Table;

!Output to excel;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5.xlsx','SL') = SLEVEL;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5.xlsx','SOH') = SOH;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5.xlsx','QGA') = Q;

ENDDATA

! The following submodel is used to allocate the available truck capacity to the different
products in the group.
  The model will minimise the total deviation from the pre-identified order-up-to level;
```

```
SUBMODEL FIND_Q:

! At each day minimise the sum of the deveiation from S as a percenage of S of each product;
@FOR (DAY (J)| J #EQ# DAYN: MIN = @SUM ( PRODUCT(I): (D1(I,J)+D2(I,J))));

! Subject to the following constraints:

  1- The order quantity is a positive integer value;
@for ( pxd (i,J)| J #EQ# DAYN: @gin ( Q(i,J)));

! 2- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (J)| J #EQ# DAYN: @sum (product(i): q(i,j)) = cap);

! 3- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (j)| J #EQ# DAYN: @for (product (i): INV (I,J) + d1(i,j) - d2(i,j) + Q(i,j) =
s(i)));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D1(I,J) >= 0));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D2(I,J) >= 0));

ENDSUBMODEL


! This simple model is used if day j is not a review day to set Q(i,j) values to 0;
SUBMODEL NO_Q:

@FOR (DAY (J)| J #EQ# DAYN: @FOR (PRODUCT (I): Q(I,J) = 0));

ENDSUBMODEL


CALC:

! define the size of the repleishment period;
MXDAY = @SIZE (DAY);

! Set day to day 1;
DAYN = 1;

! Initiate a while loop to apply the submodels (when needed);
@WHILE ( DAYN #LE# MXDAY+1:

! First we distinguesh between day 1 and all the othr days;
@IFC ( DAYN #EQ# 1:

! For day 1, calculate the reminder of division of day, J, by the review period of the group,
R;
@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!for day 1, calculate he reminder of division of day, J, by the review period of the group, R,
plus the lead time, L;
@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

! Initiate the data for the first day;
! The next 4 functions will set SOH(i,j) = SOHE(I, J) = INV (I, J) = INVE(I, J) = S(I);
@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOHE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INV (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INVE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: REC (i,j) = 0));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SLEVEL (i,j) = 0));

! the seconf branch of the first @IFC function applies to days greater than 1;
! Here a set of calculations will be performed for any particular day greater than 1;
@ELSE@IFC (DAYN #GT# 1:

@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

@FOR (PRODUCT (I): @FOR (DAY (J) | J #EQ# DAYN: @IFC (DAYN #GE# L+1: REC(I,J) = Q(I, J-L);
```

```
@ELSE@IFC (DAYN #LT# L+1: REC (I,J) = 0;););));

@for ( product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = REC(I,J) +SOHE(i, j-1)));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): @IFC ( SOH(I,J) #GE# D(I,J): DEL(I,J) = D(I,J);
NDEL(I,J) = 0; SOHE (I,J) = SOH(I,J) - DEL (I,J);

@ELSE@IFC ( SOH(I,J) #LT# D(I,J): DEL(I,J) = SOH (I,J); NDEL(I,J) = D(I,J) - DEL(I,J); SOHE
(I,J) = 0;););));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): INV (I,J) = INVE (I,J-1)));

););

@for ( Day (j) | J #EQ# DAYN: @IFC ( MOD(J) #EQ# 0:

@SOLVE (FIND_Q);

@ELSE@IFC ( MOD(J) #GT# 0:

@SOLVE (NO_Q););););

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): INVE (I,J) = INV(I,J) -
D(I,J)+ Q(I,J)));

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): SLEVEL (I,J) = DEL (I,J)/ D
(I,J) ));

DAYN= DAYN + 1;);

ENDCALC

END
```

```
EXP 5
! EXP 5;

MODEL:

SETS:

! Generate a set of 6 products;
PRODUCT/1..20/:

! An order-up-to level, S, is assigned to each product, I;
S;

! Generate a set of 500 days;
DAY/1..1000/:

! Modulo operation is used to deteminr if day J is a review day or not and to determine if a
previously
  ordered batch will be recevied.

  The reminder of division of day, J, by the review period of the group, R ( Modulo
calculations);
MOD,

! The reminder of division of day, J, by the review period of the group, R, plus the lead
time, L;
MODRL;

! Product I at day J set;
PXD(PRODUCT,DAY):

! Product i at day j will have the following parameters;

INV,    ! Inventory position of product I at day J ;
INVE,   ! Inventory Position of product I at the end of day J ;
SOH,    ! Stock on Hand of product I at the end of day J;
SOHE,   ! Stock on Hand of product I at the end of day J;
Q,      ! Order quantity of product I at the end of day J;
D,! Demand of product I at day J;
D1,     ! Over Achivement;
D2,     ! Uner Achivement;
REC,    ! Item recevied at day J;
DEL,    ! Satesfied demand of product I at day J;
NDEL,   ! Un satedfied demand of product I at day J;
SLEVEl;! Seriveice level;


ENDSETS

DATA:

! Import Order-up-to level data for each product from Excelsheet -(R,S) Table;
S =@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5X.xlsx','S_LEVEL_GA');

! Given mean of demand and the standard deviation of demand,
  use excel to generate random demand for the needed number of days.
  Then use @OLE function to import demand data;
D = @OLE('\Users\Cripps Hire Laptop\Desktop\EXP5X.xlsx','REAL_DEMAND_GA');

Cap= 20; ! The capacity of the truck allocated to this group;
L = 1;   ! Lead time;
R = 1;   ! Group review time calculated in the Excelsheet - (R,S) Table;

!Output to excel;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5X.xlsx','SL') = SLEVEL;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5X.xlsx','SOH') = SOHE;

@OLE('\Users\Cripps Hire Laptop\Desktop\EXP5X.xlsx','QGA') = Q;

ENDDATA

! The following submodel is used to allocate the available truck capacity to the different
products in the group.
  The model will minimise the total deviation from the pre-identified order-up-to level;
```

```
SUBMODEL FIND_Q:

! At each day minimise the sum of the deveiation from S as a percenage of S of each product;
@FOR (DAY (J)| J #EQ# DAYN: MIN = @SUM ( PRODUCT(I): (D1(I,J)+D2(I,J))));

! Subject to the following constraints:

  1- The order quantity is a positive integer value;
@for ( pxd (i,J)| J #EQ# DAYN: @gin ( Q(i,J)));

! 2- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (J)| J #EQ# DAYN: @sum (product(i): q(i,j)) = cap);

! 3- At day J the, the sum of the ordered quantities must fill the truck;
@for ( day (j)| J #EQ# DAYN: @for (product (i): INV (I,J) + d1(i,j) - d2(i,j) + Q(i,j) =
s(i)));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D1(I,J) >= 0));

@for ( day (j)| J #EQ# DAYN: @for (product (i): D2(I,J) >= 0));


ENDSUBMODEL


! This simple model is used if day j is not a review day to set Q(i,j) values to 0;
SUBMODEL NO_Q:

@FOR (DAY (J)| J #EQ# DAYN: @FOR (PRODUCT (I): Q(I,J) = 0));

ENDSUBMODEL


CALC:

! define the size of the repleishment period;
MXDAY = @SIZE (DAY);

! Set day to day 1;
DAYN = 1;

! Initiate a while loop to apply the submodels (when needed);
@WHILE ( DAYN #LE# MXDAY+1:

! First we distinguesh between day 1 and all the othr days;
@IFC ( DAYN #EQ# 1:

! For day 1, calculate the reminder of division of day, J, by the review period of the group,
R;
@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!for day 1, calculate he reminder of division of day, J, by the review period of the group, R,
plus the lead time, L;
@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));

! Initiate the data for the first day;
! The next 4 functions will set SOH(i,j) = SOHE(I, J) = INV (I, J) = INVE(I, J) = S(I);
@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SOHE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INV (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: INVE (i,j) = S(i)));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: REC (i,j) = 0));

@for(product (i): @ for ( Day (j) | J #EQ# DAYN: SLEVEL (i,j) = 0));

! the seconf branch of the first @IFC function applies to days greater than 1;
! Here a set of calculations will be performed for any particular day greater than 1;
@ELSE@IFC (DAYN #GT# 1:

@for ( Day (j) | J #EQ# DAYN: MOD(J) = @MOD(DAYN,R));

!@for ( Day (j) | J #EQ# DAYN: MODRL(J) = @MOD(DAYN,(R+L)));
```

```
@FOR (PRODUCT (I): @FOR (DAY (J) | J #EQ# DAYN: @IFC (DAYN #GE# L+1: REC(I,J) = Q(I, J-L);

@ELSE@IFC (DAYN #LT# L+1: REC (I,J) = 0;););));

@for ( product (i): @ for ( Day (j) | J #EQ# DAYN: SOH (i,j) = REC(I,J) +SOHE(i, j-1)));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): @IFC ( SOH(I,J) #GE# D(I,J): DEL(I,J) = D(I,J);
NDEL(I,J) = 0; SOHE (I,J) = SOH(I,J) - DEL (I,J);

@ELSE@IFC ( SOH(I,J) #LT# D(I,J): DEL(I,J) = SOH (I,J); NDEL(I,J) = D(I,J) - DEL(I,J); SOHE
(I,J) = 0;););));

@FOR (DAY(J) | J #EQ# DAYN: @FOR (PRODUCT (I): INV (I,J) = INVE (I,J-1)));

););

@for ( Day (j) | J #EQ# DAYN: @IFC ( MOD(J) #EQ# 0:

@SOLVE (FIND_Q);

@ELSE@IFC ( MOD(J) #GT# 0:

@SOLVE (NO_Q););););

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): INVE (I,J) = INV(I,J) -
D(I,J)+ Q(I,J)));

@FOR (DAY(J) | J #EQ# DAYN #and# dayn #gt# 1: @FOR (PRODUCT (I): SLEVEL (I,J) = DEL (I,J)/ D
(I,J) ));

DAYN= DAYN + 1;);

ENDCALC

END
```

	Group	Product No.	Average daily demand	Average Group Daily Demand	Truck Type	K	h	T (group)	T to the nearest integer	Exp Demand Leatime & R	SD λi	SD Leadtime & R	G(k)	k (from tables)	SS	Si
3	A	1	1.210431	5.00006	B	7.5	0.55	1.999976	2	3.631294	0.201739	0.349421	0.034641	1.43	0.499673	4.13
4		5	1.36124							4.083721	0.226873	0.392956	0.034641	1.43	0.561927	4.65
5		6	0.055264							0.165793	0.009211	0.015953	0.034641	1.43	0.022813	0.19
6		11	1.56872							4.70616	0.261453	0.45285	0.034641	1.43	0.647576	5.35
7		13	0.020149							0.060448	0.003358	0.005817	0.034641	1.43	0.008318	0.07
8		19	0.784255							2.352764	0.130709	0.226395	0.034641	1.43	0.323745	2.68
9	B	2	1.78133	14.99994	C	9		1.000004	1	3.562661	0.296888	0.419864	0.0173205	1.73	0.726364	4.29
10		3	0.757515							1.51503	0.126253	0.178548	0.0173205	1.73	0.308888	1.82
11		4	0.697857							1.395714	0.11631	0.164487	0.0173205	1.73	0.284562	1.68
12		7	0.848593							1.697187	0.141432	0.200015	0.0173205	1.73	0.346027	2.04
13		8	1.457873							2.915746	0.242979	0.343624	0.0173205	1.73	0.594469	3.51
14		9	0.240776							0.481552	0.040129	0.056751	0.0173205	1.73	0.09818	0.58
15		10	1.028612							2.057224	0.171435	0.242446	0.0173205	1.73	0.419432	2.48
16		12	0.656513							1.313026	0.109419	0.154742	0.0173205	1.73	0.267703	1.58
17		14	0.774823							1.549645	0.129137	0.182627	0.0173205	1.73	0.315946	1.87
18		15	2.71098							5.42196	0.45183	0.638984	0.0173205	1.73	1.105443	6.53
19		16	0.562826							1.125653	0.093804	0.132659	0.0173205	1.73	0.229501	1.36
20		17	1.247419							2.494838	0.207903	0.294019	0.0173205	1.73	0.508654	3
21		18	0.395753							0.791505	0.065959	0.09328	0.0173205	1.73	0.161374	0.95
22		20	1.839069							3.678138	0.306512	0.433473	0.0173205	1.73	0.749908	4.43

(R,S) TABLE | LINGO_OUTPUT GA | LINGO_OUTPUT GB | RESULTS

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		review period & order-up-to level calculations															
2		Group	Product No.	Average daily demand	Average Group Daily Demand	Truck Type	K	h	T (group)	T to the nearest integer	Exp Demand Leatime & R	SD λi	SD Leadtime & R	G(k)	k (from tables)	SS	Si
3		A	2	1.78133	5.317427	c	99	0.55	5.641826	3	7.125321	0.296888	0.593777	0.045	1.31	0.777848	7.9
4			3	0.757515							3.03006	0.126253	0.252505	0.045	1.31	0.330782	3.36
5			9	0.240776							0.963104	0.040129	0.080259	0.045	1.31	0.105139	1.07
6			13	0.020149							0.080597	0.003358	0.006716	0.045	1.31	0.008799	0.09
7			14	0.774823							3.099291	0.129137	0.258274	0.045	1.31	0.338339	3.44
8			16	0.562826							2.251305	0.093804	0.187609	0.045	1.31	0.245767	2.5
9			18	0.395753							1.583011	0.065959	0.131918	0.045	1.31	0.172812	1.76
10			19	0.784255							3.137019	0.130709	0.261418	0.045	1.31	0.342458	3.48
11		b	1	1.210431	14.68257	c	99		2.043239	1	2.420862	0.201739	0.285301	0.0212132	1.65	0.470747	2.89
12			4	0.697857							1.395714	0.11631	0.164487	0.0212132	1.65	0.271403	1.67
13			5	1.36124							2.722481	0.226873	0.320847	0.0212132	1.65	0.529398	3.25
14			6	0.055264							0.110529	0.009211	0.013026	0.0212132	1.65	0.021493	0.13
15			7	0.848593							1.697187	0.141432	0.200015	0.0212132	1.65	0.330025	2.03
16			8	1.457873							2.915746	0.242979	0.343624	0.0212132	1.65	0.56698	3.48
17			10	1.028612							2.057224	0.171435	0.242446	0.0212132	1.65	0.400036	2.46
18			11	1.56872							3.13744	0.261453	0.369751	0.0212132	1.65	0.610089	3.75
19			12	0.656513							1.313026	0.109419	0.154742	0.0212132	1.65	0.255324	1.57
20			15	2.71098							5.42196	0.45183	0.638984	0.0212132	1.65	1.054324	6.48
21			17	1.247419							2.494838	0.207903	0.294019	0.0212132	1.65	0.485132	2.98
22			20	1.839069							3.678138	0.306512	0.433473	0.0212132	1.65	0.71523	4.39

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		review period & order-up-to level calculations															
2		Group	Product No.	Average daily demand	Average Group Daily Demand	Truck Type	K	h	T (group)	T to the nearest integer	Exp Demand Leatime & R	SD λi	SD Leadtime & R	G(k)	k (from tables)	SS	Si
3		A	1	1.210431	20	B	7.5	0.55	0.5	1	2.420862	0.201739	0.285301	0.015	1.78	0.507836	2.93
4			2	1.78133							3.562661	0.296888	0.419864	0.015	1.78	0.747357	4.31
5			3	0.757515							1.51503	0.126253	0.178548	0.015	1.78	0.317815	1.83
6			4	0.697857							1.395714	0.11631	0.164487	0.015	1.78	0.292786	1.69
7			5	1.36124							2.722481	0.226873	0.320847	0.015	1.78	0.571108	3.29
8			6	0.055264							0.110529	0.009211	0.013026	0.015	1.78	0.023186	0.13
9			7	0.848593							1.697187	0.141432	0.200015	0.015	1.78	0.356027	2.05
10			8	1.457873							2.915746	0.242979	0.343624	0.015	1.78	0.611651	3.53
11			9	0.240776							0.481552	0.040129	0.056751	0.015	1.78	0.101018	0.58
12			10	1.028612							2.057224	0.171435	0.242446	0.015	1.78	0.431554	2.49
13			11	1.56872							3.13744	0.261453	0.369751	0.015	1.78	0.658157	3.8
14			12	0.656513							1.313026	0.109419	0.154742	0.015	1.78	0.27544	1.59
15			13	0.020149							0.040299	0.003358	0.004749	0.015	1.78	0.008454	0.05
16			14	0.774823							1.549645	0.129137	0.182627	0.015	1.78	0.325077	1.87
17			15	2.71098							5.42196	0.45183	0.638984	0.015	1.78	1.137392	6.56
18			16	0.562826							1.125653	0.093804	0.132659	0.015	1.78	0.236134	1.36
19			17	1.247419							2.494838	0.207903	0.294019	0.015	1.78	0.523355	3.02
20			18	0.395753							0.791505	0.065959	0.09328	0.015	1.78	0.166038	0.96
21			19	0.784255							1.56851	0.130709	0.184851	0.015	1.78	0.329034	1.9
22			20	1.839069							3.678138	0.306512	0.433473	0.015	1.78	0.771581	4.45

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		review period & order-up-to level calculations															
2		Group	Product No.	Average daily demand	Average Group Daily Demand	Truck Type	K	h	T (group)	T to the nearest integer	Exp Demand Leatime & R	SD λi	SD Leadtime & R	G(k)	k (from tables)	SS	Si
3		A	1	1.210431	20	C	99	0.55	1.5	2	3.631294	0.201739	0.349421	0.034641	1.4	0.48919	4.12
4			2	1.78133							5.343991	0.296888	0.514226	0.034641	1.4	0.719916	6.06
5			3	0.757515							2.272545	0.126253	0.218676	0.034641	1.4	0.306146	2.58
6			4	0.697857							2.093572	0.11631	0.201454	0.034641	1.4	0.282036	2.38
7			5	1.36124							4.083721	0.226873	0.392956	0.034641	1.4	0.550139	4.63
8			6	0.055264							0.165793	0.009211	0.015953	0.034641	1.4	0.022335	0.19
9			7	0.848593							2.54578	0.141432	0.244968	0.034641	1.4	0.342955	2.89
10			8	1.457873							4.373619	0.242979	0.420852	0.034641	1.4	0.589192	4.96
11			9	0.240776							0.722328	0.040129	0.069506	0.034641	1.4	0.097308	0.82
12			10	1.028612							3.085836	0.171435	0.296935	0.034641	1.4	0.415709	3.5
13			11	1.56872							4.70616	0.261453	0.45285	0.034641	1.4	0.633991	5.34
14			12	0.656513							1.969539	0.109419	0.189519	0.034641	1.4	0.265327	2.23
15			13	0.020149							0.060448	0.003358	0.005817	0.034641	1.4	0.008143	0.07
16			14	0.774823							2.324468	0.129137	0.223672	0.034641	1.4	0.313141	2.64
17			15	2.71098							8.13294	0.45183	0.782593	0.034641	1.4	1.09563	9.23
18			16	0.562826							1.688479	0.093804	0.162474	0.034641	1.4	0.227464	1.92
19			17	1.247419							3.742257	0.207903	0.360099	0.034641	1.4	0.504138	4.25
20			18	0.395753							1.187258	0.065959	0.114244	0.034641	1.4	0.159942	1.35
21			19	0.784255							2.352764	0.130709	0.226395	0.034641	1.4	0.316953	2.67
22			20	1.839069							5.517207	0.306512	0.530894	0.034641	1.4	0.743251	6.26

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		review period & order-up-to level calculations															
2		Group	Product No.	Average daily demand	Average Group Daily Demand	Truck Type	K	h	T (group)	T to the nearest integer	Exp Demand Leatime & R	SD λi	SD Leadtime & R	G(k)	k (from tables)	SS	Si
3		A	2	0.09434	5	A	5	0.55	1	1	0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
4			5	4.811321							9.622642	0.801887	1.134039	0.0212132	1.63	1.848484	11.47
5			7	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
6		B	1	0.283019	15	C	9		1	1	0.566038	0.04717	0.066708	0.0212132	1.63	0.108734	0.67
7			3	0.377358							0.754717	0.062893	0.088944	0.0212132	1.63	0.144979	0.9
8			4	5.09434							10.18868	0.849057	1.200747	0.0212132	1.63	1.957218	12.15
9			6	6.603774							13.20755	1.100629	1.556524	0.0212132	1.63	2.537135	15.74
10			8	0.283019							0.566038	0.04717	0.066708	0.0212132	1.63	0.108734	0.67
11			9	0.471698							0.943396	0.078616	0.11118	0.0212132	1.63	0.181224	1.12
12			10	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
13			11	0.188679							0.377358	0.031447	0.044472	0.0212132	1.63	0.07249	0.45
14			12	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
15			13	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
16			14	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
17			15	0.188679							0.377358	0.031447	0.044472	0.0212132	1.63	0.07249	0.45
18			16	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
19			17	0.188679							0.377358	0.031447	0.044472	0.0212132	1.63	0.07249	0.45
20			18	0.188679							0.377358	0.031447	0.044472	0.0212132	1.63	0.07249	0.45
21			19	0.377358							0.754717	0.062893	0.088944	0.0212132	1.63	0.144979	0.9
22			20	0.283019							0.566038	0.04717	0.066708	0.0212132	1.63	0.108734	0.67

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		review period & order-up-to level calculations															
2		Group	Product No.	Average daily demand	Average Group Daily Demand	Truck Type	K	h	T (group)	T to the nearest integer	Exp Demand Leatime & R	SD λi	SD Leadtime & R	G(k)	k (from tables)	SS	Si
3			2	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
4			3	0.377358							0.754717	0.062893	0.088944	0.0212132	1.63	0.144979	0.9
5			4	5.09434							10.18868	0.849057	1.200747	0.0212132	1.63	1.957218	12.15
6			6	6.603774							13.20755	1.100629	1.556524	0.0212132	1.63	2.537135	15.74
7			7	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
8		A	9	0.471698	13.20755	C	99		2.271429	1	0.943396	0.078616	0.11118	0.0212132	1.63	0.181224	1.12
9			10	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
10			12	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
11			13	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
12			14	0.09434				0.549451			0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
13			16	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
14			1	0.283019							1.132075	0.04717	0.09434	0.045	1.31	0.123585	1.26
15			5	4.811321							19.24528	0.801887	1.603774	0.045	1.31	2.100943	21.35
16			8	0.283019							1.132075	0.04717	0.09434	0.045	1.31	0.123585	1.26
17			11	0.188679							0.754717	0.031447	0.062893	0.045	1.31	0.08239	0.84
18		B	15	0.188679	6.792453	C	99		4.416667	3	0.754717	0.031447	0.062893	0.045	1.31	0.08239	0.84
19			17	0.188679							0.754717	0.031447	0.062893	0.045	1.31	0.08239	0.84
20			18	0.188679							0.754717	0.031447	0.062893	0.045	1.31	0.08239	0.84
21			19	0.377358							1.509434	0.062893	0.125786	0.045	1.31	0.16478	1.67
22			20	0.283019							1.132075	0.04717	0.09434	0.045	1.31	0.123585	1.26

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		review period & order-up-to level calculations															
2		Group	Product No.	Average daily demand	Average Group Daily Demand	Truck Type	K	h	T (group)	T to the nearest integer	Exp Demand Leatime & R	SD λi	SD Leadtime & R	G(k)	k (from tables)	SS	Si
3		A	1	0.283019	20	B	7.5	0.55	0.5	1	0.566038	0.04717	0.066708	0.0212132	1.4	0.093391	0.66
4			2	0.09434							0.188679	0.015723	0.022236	0.0212132	1.4	0.03113	0.22
5			3	0.377358							0.754717	0.062893	0.088944	0.0212132	1.4	0.124522	0.88
6			4	5.09434							10.18868	0.849057	1.200747	0.0212132	1.4	1.681046	11.87
7			5	4.811321							9.622642	0.801887	1.134039	0.0212132	1.4	1.587655	11.21
8			6	6.603774							13.20755	1.100629	1.556524	0.0212132	1.4	2.179134	15.39
9			7	0.09434							0.188679	0.015723	0.022236	0.0212132	1.4	0.03113	0.22
10			8	0.283019							0.566038	0.04717	0.066708	0.0212132	1.4	0.093391	0.66
11			9	0.471698							0.943396	0.078616	0.11118	0.0212132	1.4	0.155652	1.1
12			10	0.09434							0.188679	0.015723	0.022236	0.0212132	1.4	0.03113	0.22
13			11	0.188679							0.377358	0.031447	0.044472	0.0212132	1.4	0.062261	0.44
14			12	0.09434							0.188679	0.015723	0.022236	0.0212132	1.4	0.03113	0.22
15			13	0.09434							0.188679	0.015723	0.022236	0.0212132	1.4	0.03113	0.22
16			14	0.09434							0.188679	0.015723	0.022236	0.0212132	1.4	0.03113	0.22
17			15	0.188679							0.377358	0.031447	0.044472	0.0212132	1.4	0.062261	0.44
18			16	0.09434							0.188679	0.015723	0.022236	0.0212132	1.4	0.03113	0.22
19			17	0.188679							0.377358	0.031447	0.044472	0.0212132	1.4	0.062261	0.44
20			18	0.188679							0.377358	0.031447	0.044472	0.0212132	1.4	0.062261	0.44
21			19	0.377358							0.754717	0.062893	0.088944	0.0212132	1.4	0.124522	0.88
22			20	0.283019							0.566038	0.04717	0.066708	0.0212132	1.4	0.093391	0.66

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		review period & order-up-to level calculations															
2		Group	Product No.	Average daily demand	Average Group Daily Demand	Truck Type	K	h	T (group)	T to the nearest integer	Exp Demand Leatime & R	SD λi	SD Leadtime & R	G(k)	k (from tables)	SS	Si
3		A	1	0.283019	20	B	82.5	0.55	1	1	0.566038	0.04717	0.066708	0.0212132	1.63	0.108734	0.67
4			2	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
5			3	0.377358							0.754717	0.062893	0.088944	0.0212132	1.63	0.144979	0.9
6			4	5.09434							10.18868	0.849057	1.200747	0.0212132	1.63	1.957218	12.15
7			5	4.811321							9.622642	0.801887	1.134039	0.0212132	1.63	1.848484	11.47
8			6	6.603774							13.20755	1.100629	1.556524	0.0212132	1.63	2.537135	15.74
9			7	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
10			8	0.283019							0.566038	0.04717	0.066708	0.0212132	1.63	0.108734	0.67
11			9	0.471698							0.943396	0.078616	0.11118	0.0212132	1.63	0.181224	1.12
12			10	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
13			11	0.188679							0.377358	0.031447	0.044472	0.0212132	1.63	0.07249	0.45
14			12	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
15			13	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
16			14	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
17			15	0.188679							0.377358	0.031447	0.044472	0.0212132	1.63	0.07249	0.45
18			16	0.09434							0.188679	0.015723	0.022236	0.0212132	1.63	0.036245	0.22
19			17	0.188679							0.377358	0.031447	0.044472	0.0212132	1.63	0.07249	0.45
20			18	0.188679							0.377358	0.031447	0.044472	0.0212132	1.63	0.07249	0.45
21			19	0.377358							0.754717	0.062893	0.088944	0.0212132	1.63	0.144979	0.9
22			20	0.283019							0.566038	0.04717	0.066708	0.0212132	1.63	0.108734	0.67