MODELING AND ANALYSIS OF A KIT MANAGEMENT PROBLEM

by

Güler Kızılcık

B.S. in Textile Engineering, Istanbul Technical University, 2003

B.S. in Chemical Engineering, Istanbul Technical University, 2004

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2007

ACKNOWLEDGEMENTS

I would like to acknowledge the enthusiastic supervision of Prof. Refik Güllü. His guidance, insightful criticisms and patient encouragement contributed very much for the realization of this thesis.

I am very grateful to my family, especially my sister for the patience, encouragement and all the support she has given me during this long time of studies.

I especially would like thank to my friends Ali Akkaya and Onur Özgün for their support throughout this study.

Finally, I would like to gratefully acknowledge all of my friends and my colleagues for their patience and moral support.

ABSTRACT

MODELING AND ANALYSIS OF A KIT MANAGEMENT PROBLEM

In this study, we consider a multi-item inventory system in which customers may order different but possibly overlapping kits. Customer demand for a kit follows a Poisson process where there is a fixed probability for each kit to be demanded. When the kit is formed, it is sent to the demand location for a common time and one item from the kit is consumed. The unused items in the kit are returned to stock while an order is placed for the consumed item. We consider two supply processes for the consumed item. In the first supply system, replenishment lead times for each component are independent and identically distributed random variables as in an M/G/∞ queue. Second supply system is a load dependent system where items are processed through an M/M/1 queue.

Firstly, we derive the joint probability distribution of outstanding orders to evaluate the availability of a kit to be formed when a demand arrives. Secondly, we develop an efficient heuristic to optimize the base-stock levels of each item subject to a service level constraint. We have conducted many numerical computations to prove the effectiveness of our heuristic approach. For computations, we develop a code in MATLAB computing program and a simulation model in Arena simulation package.

ÖZET

BİR KİT YÖNETİMİ PROBLEMİNİN MODELLENMESİ VE ANALİZİ

Bu çalışmada, müşterilerin farklı fakat çakışan kitler talep ettiği bir çoklu ürün envanter sistemi gözönüne alınmaktadır. Müşterilerin bir kite olan talebi, her kit için belirli bir talep olasılığının bulunduğu bir Poisson dağılımını izlemektedir. Talep edilen bir kit oluşturulduğunda, belirli bir süre için talep edilen yere gönderilmekte ve burada kit içindeki ürünlerden biri kullanılmaktadır. Kit içindeki kullanılmayan ürünler envantere geri gönderilirken, harcanan ürün için bir sipariş verilmektedir. Harcanan ürün için iki ayrı tedarik sistemi incelenmiştir. Birinci tedarik sisteminde, her ürünün tedarik süreleri bir M/G/∞ kuyruğunda olduğu gibi bağımsız ve aynı şekilde dağılan rasgele değişkenler olarak modellenmiştir. İkinci tedarik sistemi ise ürünlerin bir M/M/1 kuyruğunda işlem gördüğü bir yük bağımlı sistemdir.

İlk olarak, bir talep geldiğinde bir kitin hazır bulunma olasılığının değerlendirilmesi için yerine getirilmemiş taleplerin birleşik olasılık dağılımı çıkarılmıştır. İkinci olarak, bir servis seviyesi kısıtı gözönüne alınarak her ürün için temel stok seviyelerini optimize eden deneyimsel bir algoritma geliştirilmiştir. Deneyimsel yaklaşımın geçerliliğinin ispatı için pek çok hesaplama yapılmıştır. Hesaplamalar için, MATLAB programında bir kod yazılmış ve Arena simülasyon paketinde bir simülasyon modeli oluşturulmuştur.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ATO	Assemble-to-order
ETO	Engineer-to-order
FCFS	First Come First Served
MTO	Make-to-order
MTS	Make-to-stock

1. INTRODUCTION

Supplying product to meet customer demand profitably is a crucial objective of all supply chains. Short delivery time and the efficient management of inventory are two important elements that determine the competitiveness of many firms [1].

The control and maintenance of inventory are problems common to all organizations. The major goals of inventory management are to minimize inventory investment, to maximize customer service and to assure efficient plant operation. The criterion most frequently applied to inventory optimization models is cost minimization whereas customer service and efficient plant operation consideration appear as constraints. It is an objective of inventory management when to order and control the lot sizes of each item purchased or manufactured by the organization so that the overall costs associated with the purchase or manufacture are at minimum. Inventory costs are associated with the operation of inventory system. They are basic economic parameter inputs to any inventory decision model. The principal costs are purchase, order/setup, holding and stockout costs [2].

Competitive pressures in today's marketplace are forcing manufacturers to offer quicker response to customer needs. As a result, managers pay close attention to performance measures that reflect the system responsiveness. These trends present some new challenges in inventory management. For example, manufacturers and distributors often manage the stocks of a huge variety of items. A customer order typically consists of several different items in different amounts. Managing multi-item inventory systems is hard, especially when customers order different but possibly overlapping subsets of items and customer satisfaction is based on the fulfillment of the entire order. Manufacturers and distributors of multiple products face this challenge, and so do mail-order and online retailers [3].

Customers are satisfied only if their requests are met at the time they order, or within a short period of time. The order fill rate, probability of filling an entire

customer order immediately from the shelf - or more generally, within a prespecified time window- is an important service measure in industry. The issue is crucial to the recent assemble-to-order (ATO) practice. In the movement toward improved supply-chain management, more and more enterprises have adopted the assemble-to-order (ATO) system. When a customer order arrives, the required components are pulled from inventory and the end product is assembled and delivered to the customer. The assemble-to-order (ATO) system has become a widely accepted business model especially in the electronic industry. Many firms, which face heightening customer expectations, shrinking product life cycles, increasing demand for product varieties, and rapid technology breakthroughs, have successfully used ATO to broaden their customized product offerings, to lower inventory cost, and to reduce time-to-market. The usual inventory-service trade-off becomes even more prominent in the ATO system because each customer order typically involves a large number of components, and the stockout of any component will cause a delay in supplying the order. Hence, it is critically important to characterize the order fill rate and the necessary inventory investment. The analysis, however, is difficult because the production-inventory dynamics among the components are highly correlated, driven by a common demand stream [3, 4].

This study mainly concerns the determination of optimal base stock levels and the evaluation and analysis of order fulfillment performance measures for a multi-item, assemble-to-order inventory system with stochastic leadtimes. Our questions driving this study are: (1) For any given stock level of each item, what is the probability a demand can be satisfied immediately (order fill rate or service level)? and (2) For any given service level, what are the optimal stock levels of each item?

In our study we assume a multivariate compound Poisson demand. Each demand requires a fixed subset (kit) of items and demands are filled on a first-come-first-served (FCFS) basis. If there is enough on-hand inventory for all the items required by a demand (a kit of items) upon its arrival, the demand is filled immediately. When a demand arrives and some of its required items are in stock but others are not, the missing items are obtained from an emergency source immediately or in a very short time. The kit is held at the demand location for a common time applying to all items

and only one of the items in the kit will be consumed while others are returned to stock. If an item is consumed for a demand occurrence, a replenishment order of one unit of the consumed item is given immediately.

Two supply processes are considered in this study. Firstly, we develop models both for a system where replenishment lead times for each item is independent and identically distributed random variables thus supply model is a parallel processing system as in an infinite server queue (M/G/∞). Secondly, we have a load dependent system where the items are produced on separate single-server production facilities with exponentially distributed production times (M/M/1).

In our study, firstly we provide an explicit characterization of kit availability probability. We derive the joint probability distribution of the outstanding orders to evaluate the availability of a kit. Secondly, as the exact computation of the optimum base-stock levels is very exhaustive, we develop a heuristic algorithm. The aim of the study constitutes practical use of the developed model so a MATLAB code was developed to calculate the probability a demand can be satisfied immediately and the exact and heuristic optimal stock levels for a given service level for different system designs.

The study also comprises a simulation framework for the described model. The simualtion model can be used to obtain kit availability probabilities for any given base-stock level. Additionally, this model can be used to define different distributions for the arrival and supply processes and observe how the service levels will respond to different distributions.

The rest of the study is organized as follows. In section 2, the results of the literature survey are provided. At the end of section 2, we describe our model briefly and state how our model differs from the other studies. Section 3 defines the notation, the formulations to compute the order fill rates (service levels) for a given base stock level and optimal stock levels with a service level constraint. This section provides a routine to calculate the exact optimal stock levels in a greedy fashion and a heuristic to

calculate approximately. In section 4, results of our numerical calculations are given. Section 5 provides details of a simulation framework for the study and in Section 6 brief concluding remarks are summarized and future research directions are presented.

2. LITERATURE REVIEW

2.1. Inventory Control

Inventory refers to stocks of anything necessary to do business. Raw materials, goods in process and finished goods- any quantifiable item that you can handle, buy, sell, store, consume, produce, or track- represent various forms of inventory. Inventories constitute an important investment in all type of firms, from a merchandise distributor to a manufacturer of products. These stocks represent a large portion of the business investment and must be well managed in order to maximize profits [5]. Every inventory lies between two activities or processes, usually called the supply process, adding new units to the inventory, and the demand process, subtracting materials from the inventory.

Lack of inventories could have serious negative effects in many situations. Axsäter [6] argues that the two main reasons for inventories are economies of scale and uncertainties. Lambert and Stock [7] mean that inventories serve five purposes within a firm; they enable the firm to achieve economies of scale, they balance supply and demand, they enable specialization in manufacturing, they provide protection from uncertainties in demand and order cycle, and they act as a buffer between critical interfaces within the channel of distribution.

In the production and distribution of goods, inventory is the currency of service. An increase in service can virtually always be achieved through an increase in safety stocks, so a supplier inevitably faces a trade-off between service levels and inventory costs [8]. Huge quantities of materials are sometimes kept in stock to deal with productivity constraints or to fulfill dynamic demand patterns. In this sense, it is vital to have effective information to aid the management for the decision making process to maximize the customer service, minimize total investment and maintain the operating efficiency. The situation turns complex because these objectives are in conflict with each other, and tradeoffs occur when trying to improve one of them. For example, to

maximize customer service, a relatively high investment in inventories is required, and due to capital constraints, these funds could have the opportunity of better profit in some other investment. The conflict finds its solution when applying an efficient inventory control, leveling these trade-offs between investment and costs to find an adequate policy for the operation of the business. This principle is well known and simple in concept, however the complexity of real situations makes it difficult to apply. Most of the real situations not only face a one item problem, but multiple items with several periods of replenishment. These inventories are frequently managed in aggregate due to the complexity of handling each individual item [9].

Successful inventory management involves balancing the costs of inventory with the benefits of inventory. It plays a crucial role in the smooth and efficient running of any organization. Reducing excess inventory and investing in the right inventories leads to better customer service, better inventory turnover-and a healthier bottom line [10].

The three main factors in inventory control decision making process are:

- The cost of holding the stock (e.g., based on the interest rate)
- The cost of placing an order (e.g., for raw material stocks) or the set-up cost of production
- The cost of shortage, i.e., what is lost if the stock is insufficient to meet all demand

The third element is the most difficult to measure and is often handled by establishing a 'service level' policy, e.g, certain percentage of demand will be met from stock without delay [11].

2.2. Product Positioning Strategies and Assemble-to-Order Systems

The product positioning strategies that an organization selects will determine the types of inventory to be held. For each product, a strategy must be developed for satisfying its target market. The product positioning strategy depends on the product

cycle time and the time a customer is willing to wait for product delivery. Inventory requirements are derived from the product positioning strategy. An organization may provide items in three ways: a) an anticipation of receiving an order, b) after receiving an order, c) in some combination of the previous two.

The recognition of the need for product availability is a realization of the importance of time. The cycle time is the total elapsed time required to provide a product. Generally, the only way a customer can obtain delivery of a product in less than the cycle time is through the availability of inventory.

General product positioning strategies may be one or more of the following:

1. Make-to-Stock (MTS): When an order is received, the product is immediately available from inventory and is an off-the-shelf standard item. The product is completely produced and placed in inventory in anticipation of demand.
2. Make-to-Order (MTO): Production does not start until a customer order is received. The customer must wait the entire cycle time for the product. Frequently long lead time components are planned prior to the order arriving in order to reduce the delivery time to the customer.
3. Assemble-to-Order (ATO): When an order is received, the product is assembled from a group of standard subassemblies already in the inventory. Finished goods are not usually available, since the product has numerous optional features and each customer may desire a unique configuration. The customer does not wait the entire cycle time for the product, but only the time required to assemble the product from its various options. The assembly lead time is usually quite short for most products. A similar categorization applies to products that are finished or packaged to order rather than assembled.
4. Engineer-to-Order (ETO): When an order is received, it is necessary to design the product to customer specifications as well as to produce it. This strategy applies to specialty products for unique customer requirements. The customer is usually willing to tolerate a long lead time.

With make-to-stock strategy, finished goods inventory is maintained and the customer wait time is very short. With make-to-order, assemble to order, and engineer-to-order strategies, no finished goods inventory is available, so the customer must place an order and wait for its completion. In make-to-order and engineer-to-order, the customer must wait the entire cycle time; in assemble-to-order, the customer waiting time is less than the cycle time. It is common for organizations to have different strategies for different products [12].

An ATO system is one in which demand is received for products, and supply is ordered in terms of components. Stock is held only for components, and products are assembled when an order is received [13]. To make each product requires a particular selection of components, comprising only a subset of them, but possibly several units of certain ones. Some or all components are shared by several products. The time to assemble a product from its components is negligible. The time to acquire or produce a component, however, is substantial. A product is assembled only in response to demand. In other words, component and subassemblies are replenished in a make-to-stock (MTS) fashion but finished products are assembled in a make-to-order (MTO) manner. Such a hybrid planning approach is particularly advantageous in situations where the assembly time of a product is considerably shorter than the procurement and/or manufacturing time of its components and subassemblies; thus, making a tradeoff between inventory holding cost, product variety, and delivery time achievable [14].

Maximizing the order fulfillment time reliability while keeping the total inventory investment low is an objective voiced by managers of assemble-to-order manufacturing systems [13]. ATO systems in practice help to increase product variety, while achieving quick response times and low cost [1]. It enables a firm to shorten its response time to its customers by staging inventory of components ahead of demand while postponing the final assembly until demand is realized. This strategy is particularly valuable when component supply leadtimes are long or the supply processes are capacitated. Furthermore, by pooling component inventories, ATO can reduce the costs of offering higher product variety, which can be useful when demand for individual end products is variable [15].

An assemble-to-order (ATO) system is an important business model in managing a wide-ranging class of supply chains. Examples of ATO systems can be found in various industries producing consumer goods [14]. The best way to understand and appreciate an ATO system is to consider the manufacturing and distribution of personal computers (PCs). A PC is a complex machine, built with hundreds of components. A PC company typically offers several lines of product which customers can select when placing an order - different combinations of CPU, memory, hard drive, and other components and peripherals (CD ROM, sound card, modem, monitor, keyboard, printer, etc). Whereas each of these components takes a substantial lead time to build, the time it takes to assemble all the components into a PC, following a specific customer order, takes virtually no time - provided all the components are available. Hence, managing the component inventory is of critical importance to the business: The stockout of any component will delay order fulfillment, whereas excess inventory could easily destroy the firm's profit margin and diminish its competitive edge [16].

A good example for an ATO system is Dell Computer's. Dell lets the customer select among several processors, monitors, disk drives, etc. Thus, the number of products (combinations of options) is huge. This approach has been so successful that most other makers of personal computers are adopting similar systems. Indeed, the ATO approach has become widespread throughout the electronics industry. Also, the major U.S. automobile companies are studying ambitious ATO systems for the assembly of cars [17].

Another example is mail-order systems, including online retailing such as in Lands End and amazon.com which maintain inventories of the items in its catalogue. The items correspond to components, and a product is any combination of them [17, 18]. The assembly of a product entails picking out the items in the customer's order and packaging them. Also, the problem of stocking spare parts for the repair of equipment can be another example. The parts are the components, and a product is a particular type of repair job, requiring particular parts. The parts may be located at a central point, where equipment needing service arrives (e.g., vehicles), or the parts may travel to stationary equipment (as in field service of computers, copiers, and factory ma-

chines). In either case, the part requirements of a job are usually unknown in advance [18].

In spite of their prevalence in practice, ATO systems are however notoriously difficult to analyze and manage. The difficulty appears to be due to several factors: (a) demands for the different components being correlated; (b) supply leadtimes for different components being different; and (c) order fulfillment being dependent on the availability of multiple components. These factors make it difficult to manage components independently. For example, continuing to produce one component when there is a shortage of another may do little to improve the ability to fulfill demand. Similarly, deciding to stop production of one component without considering the inventory level of other components may not necessarily reduce overall inventory costs [15].

2.3. Multi-Item Inventory Systems and Model Description

2.3.1. Multi-Item Inventory Systems

In multi-item inventory systems, a customer order typically consists of several different items in different amounts and different orders may have overlapping subset of items. Optimal planning of multi-item inventory systems have been widely studied in the literature in the recent years but they differ in modeling assumptions and approaches. Hausman [13] and Zhang [19] study discrete-time models with multivariate normal demand and constant component replenishment leadtimes where Song [18] concerns continous-review models with multivariate compound Poisson demand and deterministic leadtimes. Song *et al.* [20], Glasserman and Wang [8] consider multivariate Poisson demand but the supply process is modeled as a single-server queue. A parallel processing system supply model is also studied by Lu *et al.* [4]. Many concentrate on performance measures like the item and order fullfillment performances, order-based backorders and customer waiting time.

Hausman *et al.* [13] study a model that maximizes a lower bound on the aggregate order fill rate with an inventory budget constraint. They consider a multi-item

system in which multivariate normal distribution is used to represent the item demands. Each item's inventory is controlled independently through a periodic review order-up-to policy. All items have the same review cycle. The replenishment lead times are assumed to be deterministic and non-negative multiples of the period length. The aim of the paper is to compute the joint demand fulfillment probability within a given time window. Additionally, a heuristic approach to maximize the joint demand fulfillment probability is discussed. They concluded that even the demands are correlated; a heuristic ignoring the correlations can give good approximate results.

A similar analysis is Song's [3] study on the order fill rate over an entire planning horizon in a multi-item, base-stock inventory system in which the demand process forms a multivariate compound Poisson process and the replenishment leadtimes are constant. Each demand requires a fixed kit of items, and the amount requested for each item within the kit is a positive random variable. She studies a computational procedure to compute the exact order fill rates and shows that it can be obtained through a series of convolutions of one dimensional compound Poisson distributions. She also utilizes lower and upper bounds on the order fill rates using item fill rates but concludes that item fill rates are not good indicators of the order fill rate especially when the item correlations are high.

Song [18] studies order-based backorders which can be defined as the average number of customer orders that are not yet completely filled. It is an important service measure because it is proportional to the average customer waiting time, so it determines customer dissatisfaction due to delivery delays. She considers a base-stock inventory system with deterministic lead times and uses a new approach instead of the joint distributions to obtain a closed form expression.

In a study by Lu and Song [21], a multi-item, multi-product assemble-to-order system is reviewed where the authors aim to determine the optimal base-stock levels by introducing order-based backorder costs. They use an unconstrained cost optimization formulation and compare the findings with the standard single-item, newsvendor-type model with item-based backorder cost. They express that the findings of the single-item

newsvendor-type model can be used as a bound for the order-based problem and can be a starting point in searching for the optimal base-stock levels in a greedy fashion. They also study the increase in optimal base-stock levels when lead times are stochastically longer and the effect of leadtime variability and demand correlation in these levels. Finally, they conclude with a formulation to obtain closed-form approximations of the optimal base-stock levels.

Song *et al.* [20] study a multi-component, multi-product production and inventory system and present a procedure to compute the item-based, order-based and system-based performance measures, such as fill rate, service level (the probability that an order will be backlogged and eventually served), waiting time distribution, etc. for a system in which demand forms a multivariate Poisson process. There is a dedicated facility with exponentially distributed processing times to process replenishment orders for any given item and a finite buffer. The procedure relies on computation of steady state joint distribution of outstanding orders. The occupancy in the supply system can be modeled as a quasi birth-death process because of the structure of the demand process.

Lu *et al.* [4] analyze a multi-item system where they model the supply process as a set of queues driven by a common, multiclass batch Poisson input. They aim to derive the joint queue-length distribution and easy-to-compute approximations and bounds for the order fulfillment performance measures. In a similar study by Song and Yao [16], M/G/∞ queues with a common Poisson arrival stream is analyzed. They study the effect of lead time variability on the performance of an assemble-to-order system and show that variability degrades the performance of the system.

While most of the studies consider a system with multiple products, Benfaajar and ElHafsi [15] study a system of a single product consisting of m components and n customer classes. The demand and supply process is similar to those studies as the demand is assumed to be a Poisson process and components are produced on single-server facilities with exponentially distributed processing times. They formulate the model as a Markov decision process and results show that the optimal production policy

for each component is a state-dependent base-stock policy. Additionally, they present that the optimal inventory allocation is a state-dependent multi-level rationing policy where the component rationing level for each class is non-increasing in the inventory level of other components. An order from a customer class is satisfied only if the current inventory level at each component is above a certain rationing level.

Hsu *et al.* [1] develop an optimization model where demand is uncertain in an assemble-to-order environment. Their study differs from the others as they define the price for the final product and the component costs depending on their delivery leadtimes. They also provide solutions for the situation where the manufacturer has the option of not delivering the full quantity, but instead takes the penalty for a delivery shortage.

DeCroix *et al.* [22] employ an ATO system managed using a base stock policy for each item over an infinite horizon where demands and returns arrive according to Poisson process. Demands are satisfied on a FCFS basis. The unsatisfied demand is backlogged and when an item is returned it can be used immediately to satisfy a demand. The complexity of the system arises from the existence of the retuns. They present methods for computing performance measures like order fill rate, fill rate within a time window and average backorders and also include a procedure to calculate a near-optimal base stock policy.

2.3.2. Model Description

In this study, we analyze a multi-item inventory system in which customers may order different but possibly overlapping subsets of items (kits), such as a multi-product assemble-to-order system. Our aim is to determine the probability a demand can be satisfied for a given basestock level and the optimal base stock level for each item when there is a service level constraint for the subset of items.

We model the demand process as a multivariate Poisson process. The overall demand arrives according to a Poisson process, but there is a fixed probability that a

demand requests particular kit of different items. Whenever a kit demand occurs, if there is enough on-hand inventory for all the items required by a demand, the demand is satisfied immediately. When a demand arrives and some of its required items are available in stock but others are not, the missing items are obtained from an emergency source immediately or in a very short time. When the kit is formed, it is sent to demand location (site) and held on site for a common time applying to all items. Only one of the items in the kit will be consumed while others are returned to stock after a common time spent on site. If an item is used for a demand occurance, a replenishment order of one unit of the used item is given immediately. The representative figure of the model is given in Figure 2.1.

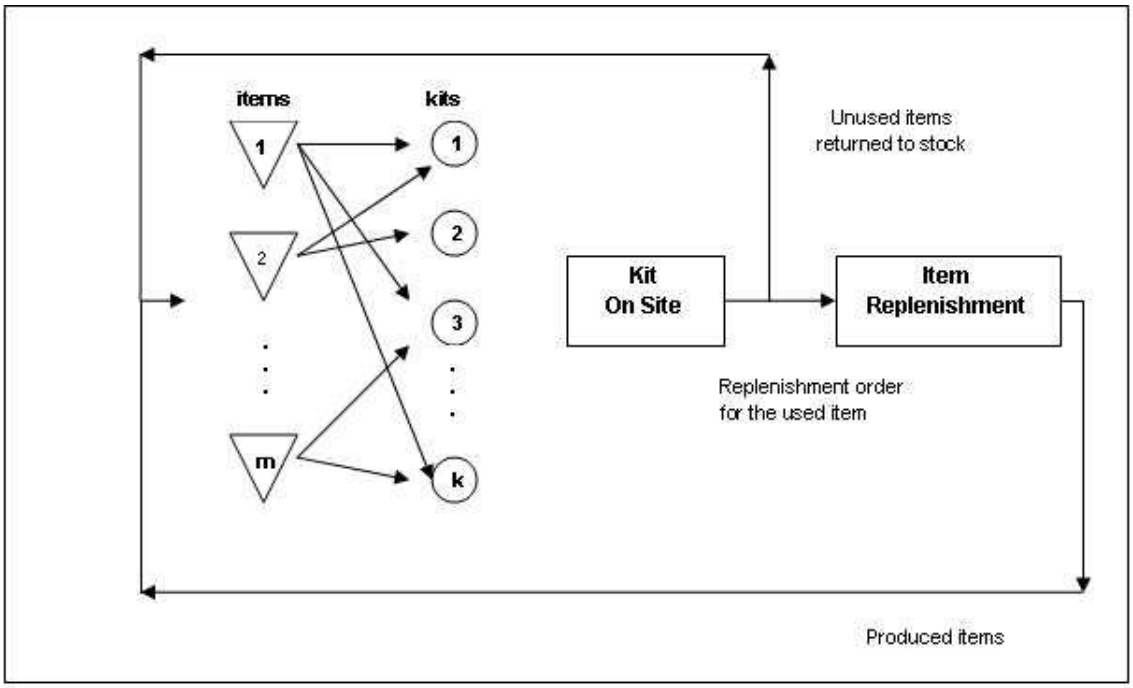

Figure 2.1. Multi-item inventory model

Our motivation for this study is to obtain order fill rates and the optimal stock levels for hospital implants and repair toolkits where only one of the items in a kit is used and the other items are returned to stock after a certain period of time spent on the hospital/repair site. For example, for spinal implant products, a surgent may not know which type of implant to use before the surgery and requests a kit of implants. During the surgery all items in the kit are held in the hospital and one of the implants is used. The unused implants are returned to inventory after the surgery and an order is placed for the used implant.

The contributions of this study and the differences from the related literature can be listed as below:

- We consider a system with returns. The demanded kit is held in the demand location and one of the items in the kit is consumed while others are returned to their stocks.
- As mentioned above, only one of the items in a kit is consumed in our model. In studies related with ATO systems, all items in the kit are considered to be consumed, splitting of items is not studied.
- In our study, we provide an explicit characterization of kit availability probability. We derive the joint probability distribution of the outstanding orders to evaluate the availability of a kit (service level).
- Exact computation of optimum base-stock values of each item is very exhaustive, so in our study we develop a heuristic which gives good results in considerably shorter times when compared to exact values.
- The study also comprises a simulation framework for the described model. We can obtain kit availability probabilities for any given base-stock level using this model. The simulation model can also be used to implement different arrival and supply distributions easily.

3. ANALYSIS OF THE MODEL

In this section, specific model assumptions and basic notation are introduced.

We consider an inventory system of M items. Let $\Omega = \{i = 1, 2, ..., M\}$ be set of all item indexes. There are K kits, $(k = 1, 2, ..., K)$, which can be any subset of item set Ω.

Customer orders arrive at the system following a Poisson process with rate λ. Each order may require several items simultaneously. We assume that there is a fixed probability P_k that an order is of kit type k, $\sum_k P_k = 1$. Each order's type is independent of the other orders' types and all other events. Thus the type-k order stream forms a Poisson process with rate $\lambda_k = P_k\lambda$.

For any $i \in \Omega$ and kit type k $(k = 1, 2, ..., K)$, let

λ = overall demand rate
P_k = probability that a demand is of kit type k, $(\sum_k P_k = 1)$
λ_k = demand rate of kit type k, $(\lambda_k = P_k\ \lambda)$
$U(k)$ = set of items contained in kit k
$V(i)$ = set of kits containing item i
$\Gamma(k)$ = set of kits containing any one of the items in kit k
$p_{k,i}$ = probability that an item is consumed in a demand for kit k, $(\sum_{i \in U(k)} p_{k,i} = 1)$
λ_i = aggregate consumption rate of item i, $(\lambda_i = \sum_{k \in V(i)} \lambda_k p_{k,i})$

Demands are filled on a first-come-first-served (FCFS) basis. A kit demand is always satisfied. If there is enough on-hand inventory for all the items required by a kit, the demand is filled immediately. When a demand arrives and some of its required items are in stock but others are not, the items that are unavailable are supplied exogenously through emergency channels (by borrowing, renting from market, etc.). As soon as a unit of that item becomes available it is returned to the exogenous source.

The significance of this assumption is to be able to maintain a base stock level (and not more) for each item. Alternative backlogging approaches are studied by Lu and Song [4], Zhang [19] and Song [3].

The inventory of each item is controlled by an independent base-stock policy with S_i = the base-stock level for item i, $i = 1, 2, ..., M$. Even in our case the component demands are not independent, we use an independent base stock policy because of its simplicity, and partly because it provides a benchmark on how much inventory is needed to provide a certain service level [4, 16].

Let T_k be the random variable denoting the holding time of the kit at the demand location from the time of demand occurance until the return of a kit. It is a common time applying to all the items including the one which is consumed.

Since at each kit demand a single unit of an item in that kit will be consumed, the base-stock policy implies that every demand will trigger a replenishment order of one unit of consumed item, regardless of whether or not there is stockout at the item inventory. We consider two situations for the supply process. We may have a situation where successive replenishment times of an item are dependent as in a case where an item is supplied through a finite capacity queue (like an M/M/1 queue) or we may have an infinite server model where the replenishment lead times would be independent (like an M/G/∞ queue). Let τ_i be the time it takes to replenish an item from the supplier if there are no other pending replenishment orders for this item. If the successive replenishment times are independent, then the replenishment time for each item would simply be τ_i. However, the realized replenishment time may be considerably higher if there are outstanding replenishment orders of the same item due to waiting. The orders for item i may be sent to a single machine production facility, say facility i, in which they are processed on a FCFS basis. The processing times at facility i are i.i.d. exponentially distributed random variables with rate μ_i. Thus the system can be seen as M parallel stochastic production facilities, where facility accepts Poisson inputs with rate λ_i. In any case the replenishment times and kit holding times on site are independent random variables.

Suppose a kit of type k is demanded. Let i be an item located in the kit. Then the lead time for the item would be $L_i = T_k$, if the item is not consumed and $L_i = T_k + \tau_i +$ possible waiting, if the item is consumed. In particular, the collection $\{L_i, i \in U(k)\}$ is not independent. Note that T_k is common for all items in kit k. For any given time t, let

$I_i(t)$ = on-hand inventory for item i
$B_i(t)$ = number of outstanding emergency supplied units for item i
$N_i(t)$ = number of units on site and outstanding for item i

Then,

$$I_i(t) = (S_i - N_i(t))^+ \tag{3.1}$$

$$B_i(t) = (N_i(t) - S_i)^+ \tag{3.2}$$

For the number of outstanding items, these units are either at the demand location (hospital, repair site etc.) or being replenished. Since the item lead times are not independent, number of outstanding units for the items in the same kit are not independent.

Let N_i, I_i, and B_i be the corresponding steady-state limits of the above mentioned random variables (provided that they exist). Then,

$$I_i = (S_i - N_i)^+$$

$$B_i = (N_i - S_i)^+$$

Normally, one would expect that supplying an item from emergency channels would be much expensive and not desirable. An objective would be to set $(S_1, S_2, ..., S_M)$ in such a way that the kit-readiness probabilities (order fill rate or service level), which is the probability that the items for a kit is available in the inventory, in the long run is at least a prespecified level. So we are interested in computing the order fill rate

(service level) of a kit which can be stated as;

F_k = joint probability that all items in a type k kit are available
$= Pr\{I_i > 0, \forall i \in U(k)\}$
$= Pr\{N_i < S_i, \forall i \in U(k)\}$

We would like to set the base stock levels such that F_k would be at least a prespecified level;

$$F_k = Pr\{N_i < S_i, \forall i \in U(k)\} \geq \alpha_k \tag{3.3}$$

Here, the key is to obtain the joint distribution of $(N_1, N_2, ..., N_M)$. But these are not independent collections. Classical approach ignore the dependency of demands and assumes N_i are independent. So under independence assumption the joint distribution of $(N_1, N_2, ..., N_M)$ can be written as;

$$\{N_i = n_i, \forall i \in U(k)\} = \beta_k = \prod_{i \in U(k)} Pr\{N_i < S_i\} \tag{3.4}$$

But, due to the dependent demands, the probability that all demand will be filled immediately cannot generally be reduced to a product form. In the next section, we derive the joint distribution of $\{N_i, i \in U(k)\}$ for $k = 1, 2, ..., K$.

3.1. Joint Distribuiton

For each component i, the number of outstanding orders N_i is exactly the number of jobs on demand location (site) and in replenishment with Poisson arrival rate λ_i. Let Y_k be the number of kits at the site and X_i be the number of items at replenishment.

When we fix an item i, $\{Y_m, m \in V(i)\}$ represent the number in the system variables for independent M/G/∞ systems with respective arrival rates λ_m, $m \in V(i)$.

Therefore, Y_m are distributed as a Poisson random variable with mean $\lambda_m E[T_m]$. The departure process of an M/G/∞ queue is independent of the number that are already present in the system. Thus, X_i are independent of $\{Y_m, m \in V(i)\}$.

The departures from an M/G/∞ queue is a Poisson process whose rate is the same as the input rate λ_m and each departure from the queue (site) goes to the replenishment process with probability $p_{m,i}$ if the item is used. Therefore, X_i is the number in the system of a Poisson arrival queue with arrival rate $\sum_{m \in V(i)} p_{m,i}\lambda_m$ and service time τ_i. If the successive replenishment times of an item are independent (M/G/∞ type replenishment), then X_i are distributed as Poisson with mean $E[\tau_i] \sum_{m \in V(i)} p_{m,i}\lambda_m$.

N_i are only correlated because they share a common Y_k. For any $i = 1, 2, ..., M$ we can write

$$N_i = X_i + \sum_{m \in V(i)} Y_m \tag{3.5}$$

Now the joint distribution of $\{N_i, i \in U(k)\}$ for $k = 1, 2, ..., K$ can be expressed as

$$\{N_i = n_i, \forall i \in U(k)\} = \bigcup_{\Upsilon_k} (Y_j = y_j, \forall j \in \Gamma(k), X_i = n_i - \sum_{l \in V(i)} y_l, \forall i \in U(k)) \tag{3.6}$$

where,

$$\Upsilon_k = \{y_j \geq 0, j \in \Gamma(k) : \sum_{j \in V(i)} y_j \leq n_i, \forall i \in U(k)\} \tag{3.7}$$

For computing the outstanding orders for an item, one should consider that this item can be demanded by other kits. So, the number of jobs on demand location is the sum of outstanding orders of all kits containing that item. This also implies that for all items in a demanded kit, we consider the outstanding orders of all kits containing any one of the items of the concerning kit for computing the availability. The number of orders for a kit on demand location can be at most minimum of the outstanding

orders of the kits containing any of the items in demanded kit. Consider three kits and five items with $U(1) = \{1, 2, 3\}$, $U(2) = \{1, 4\}$ and $U(3) = \{3, 4, 5\}$. The first kit is demanded. The items in the first kit can also be demanded by kits 2 and 3 because they share common components; $\Gamma(1) = \{1, 2, 3\}$. So while computing the joint distribution of the number of outstanding orders on site, we write for Kit 1;

$$\bigcup_{\Upsilon_1}(Y_1 = y_1, Y_2 = y_2, Y_3 = y_3, X_1 = n_1 - y_1 - y_2, X_2 = n_2 - y_2, X_3 = n_3 - y_1 - y_3)$$
$$\Upsilon_1 = \{y_1 \geq 0, y_2 \geq 0, y_3 \geq 0, y_1 + y_2 \leq n_1, y_1 \leq n_2, y_1 + y_3 \leq n_3\}$$

From the above equation, y_1 can be at most $\min\{n_1 - y_2, n_2, n_3 - y_3\}$, where y_2 can be at most n_1 and y_3 can be at most n_3.

Making use of independence, we can now express the joint distribution in product form,

$$\Pr\{N_i = n_i, \forall i \in U(k)\} = \sum_{\Upsilon_k} \prod_{j \in \Gamma(k)} \Pr(Y_j = y_j) \prod_{i \in U(k)} \Pr(X_i = n_i - \sum_{l \in V(i)} y_l) \quad (3.8)$$

If the successive replenishment times of an item are independent, the general equation above becomes,

$$\Pr\{N_i = n_i, \forall i \in U(k)\} = \sum_{\Upsilon_k} \prod_{j \in \Gamma(k)} \frac{e^{-\lambda_j E[T_j]}(\lambda_j E[T_j])^{y_j}}{y_j!}$$
$$\prod_{i \in U(k)} \frac{e^{-E[\tau_i] \sum_{k \in V(i)} \lambda_k p_{k,i}} (E[\tau_i] \sum_{k \in V(i)} \lambda_k p_{k,i})^{(n_i - \sum_{l \in V(i)} y_l)}}{(n_i - \sum_{l \in V(i)} y_l)!} \quad (3.9)$$

When the successive replenishment times are dependent through an M/M/1 queue and let $\rho_i = E[\tau_i] \sum_{k \in V(i)} \lambda_k p_{k,i} < 1$, we rewrite the equation as,

$$\Pr\{N_i = n_i, \forall i \in U(k)\} = \sum_{\Upsilon_k} \prod_{j \in \Gamma(k)} \frac{e^{-\lambda_j E[T_j]}(\lambda_j E[T_j])^{y_j}}{y_j!}$$

$$\prod_{i\in U(k)} (E[\tau_i] \sum_{k\in V(i)} \lambda_k p_{k,i})^{(n_i - \sum_{l\in V(i)} y_l)} (1 - E[\tau_i] \sum_{k\in V(i)} \lambda_k p_{k,i}) \tag{3.10}$$

Below presented some examples for both independent and dependent replenishment times.

Example 1: We consider a single kit containing M items and the replenishment times are independent. The joint distribution of outstanding items is

$$\Pr(N_1 = n_1, N_2 = n_2, ..., N_M = n_M) =$$
$$e^{-\lambda(E[T] + \sum_{i=1}^{M} p_i E[\tau_i])} \sum_{u=0}^{\min\{n_1,...,n_m\}} \frac{(\lambda E[T])^u \prod_{i=1}^{M} (\lambda p_i E[\tau_i])^{n_i - u}}{u! \prod_{i=1}^{M} (n_i - u)!}$$

Example 2: Same case in Example 1 but the replenishment times are dependent through an M/M/1 queue (let $\rho_i = \lambda p_i (E[\tau_i]) < 1$),

$$\Pr(N_1 = n_1, N_2 = n_2, ..., N_M = n_M) =$$
$$\sum_{u=0}^{\min\{n_1,n_2,...,n_M\}} \frac{e^{-\lambda E[T]} (\lambda E[T])^u}{u!} \prod_{i=1}^{M} (1 - (\lambda p_i E[\tau_i]))(\lambda p_i E[\tau_i])^{n_i - u}$$

Example 3: We consider a case with two kits and 3 items where the replenishment times are independent. Let
$U(1) = \{1, 2\},\ U(2) = \{1, 3\}$, then $V(1) = \{1, 2\},\ V(2) = \{1\},\ V(3) = \{2\},\ \Gamma(1) = \{1, 2\},\ \Gamma(2) = \{1, 2\}$.
Also,

$$\Upsilon_1 = \{y_1 \geq 0, y_2 \geq 0, y_1 + y_2 \leq n_1, y_1 \leq n_2\}$$
$$\Upsilon_2 = \{y_1 \geq 0, y_2 \geq 0, y_1 + y_2 \leq n_1, y_2 \leq n_3\}$$

For the first kit, the joint distribution can be written as:

$$\begin{aligned}&\Pr(N_1 = n_1, N_2 = n_2) = \\ &e^{-\lambda_1 E[T_1] - \lambda_2 E[T_2]} \sum_{y_2=0}^{n_1} \sum_{y_1=0}^{\min\{n_1 - y_2, n_2\}} \frac{(\lambda_1 E[T_1])^{y_1} (\lambda_2 E[T_2])^{y_2}}{y_1! y_2!} \\ &\Pr\{X_1 = n_1 - y_1 - y_2\} \Pr\{X_2 = n_2 - y_1\}\end{aligned}$$

with

$$\begin{aligned}Pr\{X_1 = n\} &= \frac{e^{-E[\tau_1](\lambda_1 p_{1,1} + \lambda_2 p_{2,2})} (E[\tau_1](\lambda_1 p_{1,1} + \lambda_2 p_{2,2}))^n}{n!} \\ Pr\{X_2 = n\} &= \frac{e^{-E[\tau_2](\lambda_1 p_{1,2})} (E[\tau_2]\lambda_1 p_{1,2})^n}{n!}\end{aligned}$$

For the second kit,

$$\begin{aligned}&\Pr(N_1 = n_1, N_3 = n_3) = \\ &e^{-\lambda_1 E[T_1] - \lambda_2 E[T_2]} \sum_{y_2=0}^{\min\{n_1, n_3\}} \sum_{y_1=0}^{n_1 - y_2} \frac{(\lambda_1 E[T_1])^{y_1} (\lambda_2 E[T_2])^{y_2}}{y_1! y_2!} \\ &\Pr\{X_1 = n_1 - y_1 - y_2\} \Pr\{X_3 = n_3 - y_2\}\end{aligned}$$

with

$$\begin{aligned}Pr\{X_1 = n\} &= \frac{e^{-E[\tau_1](\lambda_1 p_{1,1} + \lambda_2 p_{2,2})} (E[\tau_1](\lambda_1 p_{1,1} + \lambda_2 p_{2,2}))^n}{n!} \\ Pr\{X_3 = n\} &= \frac{e^{-E[\tau_3](\lambda_2 p_{2,3})} (E[\tau_3]\lambda_2 p_{2,3})^n}{n!}\end{aligned}$$

3.2. Optimization

One of the objectives of this study is to find the optimal base stock level for each item when there is a service level constraint. The problem can be defined as below:

$$\text{minimize} \qquad C(S_1, S_2, ..., S_M) = \sum_{i=1}^{M} h_i E\left[(S_i - N_i)^+\right] \tag{3.11}$$

$$\text{subject to} \qquad Pr\{N_i < S_i, \forall i \in U(k)\} \geq \alpha_k, \ k = 1, 2, ..., K \tag{3.12}$$

$$S_i \geq 0, \forall i = 1, 2, ..., M \tag{3.13}$$

where h_i is the holding cost per unit of item i.

To obtain the optimum values, firstly we bound the feasible set of the problem. Let $S_1, S_2, ..., S_M$ be a feasible solution to the constaint set of the optimization problem. Then a lower bound for any kit k and $i \in U(k)$ can be defined as,

$$S_i^L(k) = \min\{x : Pr\{N_i < x\} \geq \alpha_k\}. \tag{3.14}$$

Thus, if all other items in kit k are abundantly available, $S_i^L(k)$ many units of item i would be sufficient to satisfy the kit availability constraint for kit k. Define

$$S_i^L = \max_{k \in V(i)} \left\{S_i^L(k)\right\} \tag{3.15}$$

Then, $S_i \geq S_i^L(k)$ for $i = 1, 2, ..., M$ is a lower bound.

The computation of lower bound is easy. The lower bound for an item can be written as;

$$Pr\{N_i < x\} \geq \alpha_k = Pr\{N_i = X_i + \sum_{m \in V(i)} Y_m < x\} \geq \alpha_k\}$$

We can calculate the number of outstanding orders easily. For example for an M/G/∞ system, X_i is distributed as Poisson with mean $E[\tau_i] \sum_{m \in V(i)} p_{m,i}\lambda_m$ and Y_m is distributed as Poisson with mean $\sum_{m \in V(i)} \lambda_m E[T_m]$. So, for a given service level one can compute lower bound for an item using the cumulative Poisson distribution with mean $E[\tau_i] \sum_{m \in V(i)} p_{m,i}\lambda_m + \sum_{m \in V(i)} \lambda_m E[T_m]$.

Consider two kits and four items with $U(1) = \{1, 2, 3\}$ and $U(2) = \{1, 4\}$. Let

$\lambda_1 = 2$ and $\lambda_2 = 1$, and $E[T_1] = 0.5, E[T_2] = 0.2$. Item consumption probabilities are (0.4,0.3,0.3) and (0.3,0.7) for Kits 1 and 2, respectively. Let $E[\tau_i] = 1$ for all i=1,2,3,4. If a 90% kit availability is desired for both kits, we can easily compute that $S_1^L = 5, S_2^L = S_3^L = 4, S_4^L = 3$. We then conclude that for a solution to be feasible, $(S_1, S_2, S_3, S_4) \geq (5, 4, 4, 3)$.

Figure 3.1 and Figure 3.2 present kit availability contours for Kit 1 and Kit 2, respectively for the above mentioned example. Each contour in the figures represents a level of kit probability beginning from 0.1 to 0.9. For example for the defined case, if we have 6 units of type 1, 4 units of type 2 and 3 and 3 units of type 4 on hand, from Figures 3.1 and 3.2 we see that a service level of 86% can be obtained for Kit 1 and 91% for Kit 2.

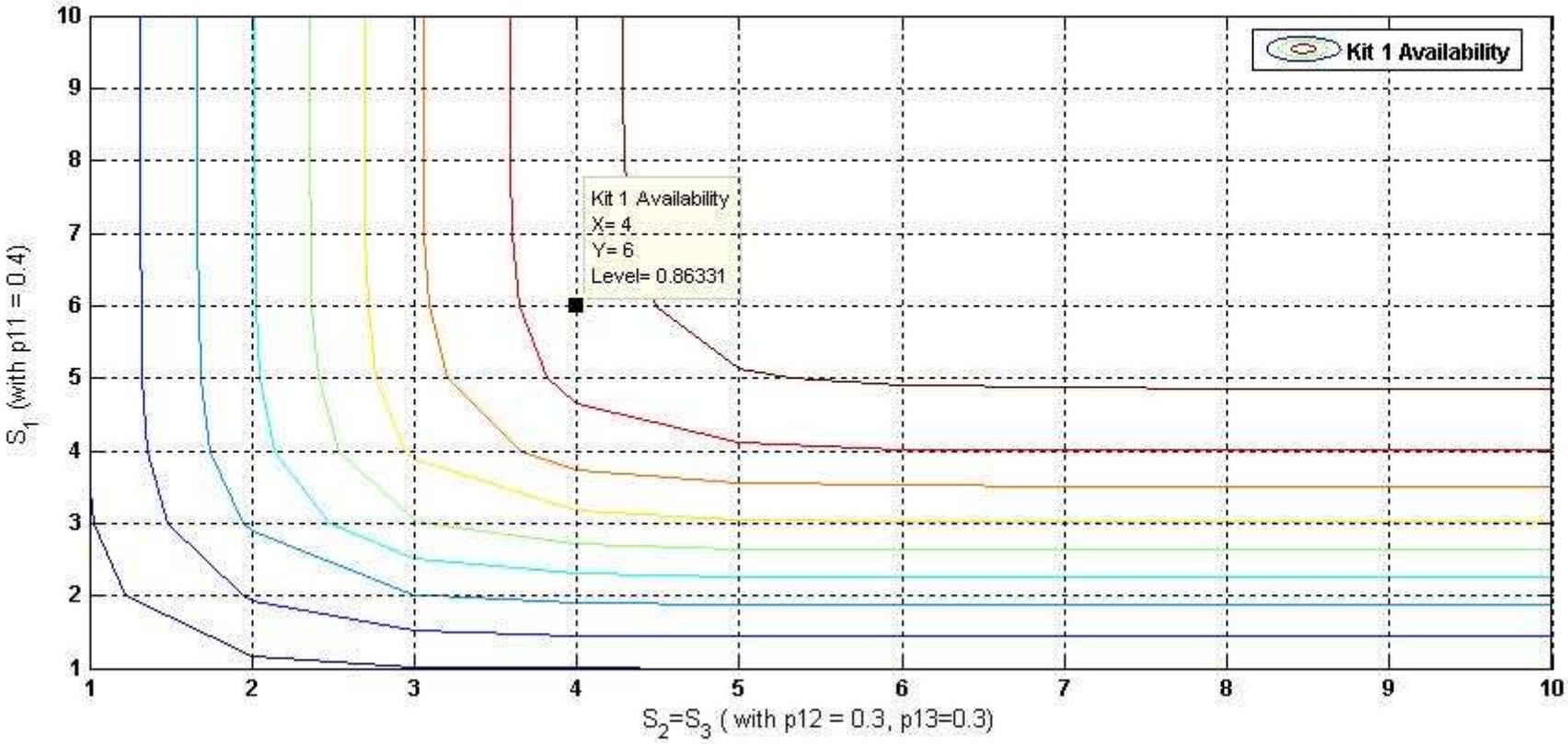

Figure 3.1. Kit availability contours for kit 1 ($\lambda_1 = 2, \lambda_2 = 1, E[T_1] = 0.5, E[T_2] = 0.2, E[\tau_i] = 1$)

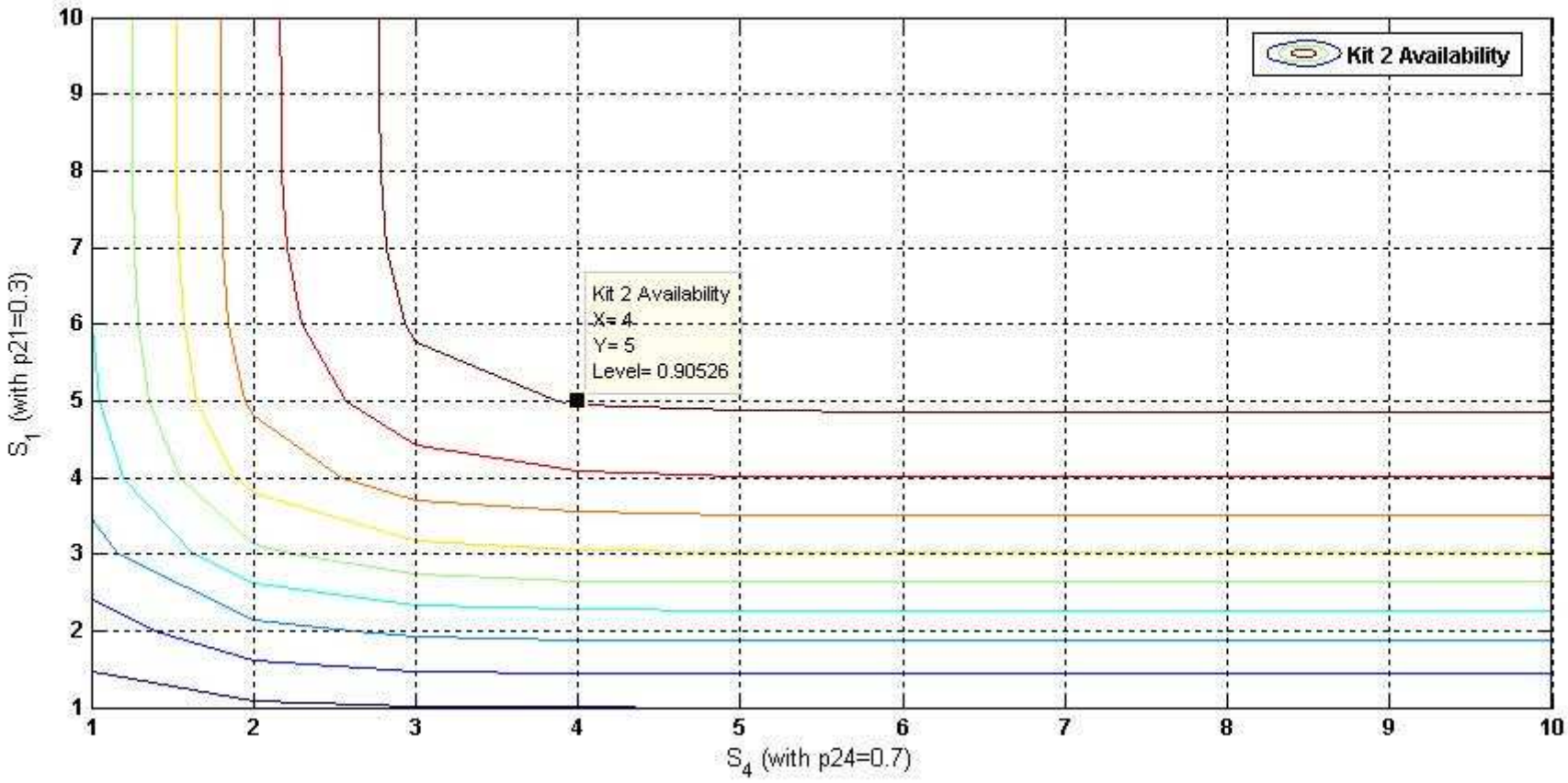

Figure 3.2. Kit availability contours for kit 2 ($\lambda_1 = 2, \lambda_2 = 1, E[T_1] = 0.5, E[T_2] = 0.2, E[\tau_i] = 1$)

We can obtain the set of optimal values for base-stock levels for each item in a greedy fashion. Starting from the lower bound up to a defined upper bound, we can calculate the kit availability for each combination of stock values. Then, we can choose the stock combinations where the kit availability constraint is satisfied. From these combinations one may calculate the cost of each combination and select the combination which has the minimum cost. This procedure will provide us the exact optimal values for base-stock levels for each item. But the greedy approach is difficult and impractical. Therefore, for computing the optimal base-stock values, below we present a heuristic method based on the ratio of increase in holding cost to increase in kit availability.

We rewrite our objective function as below:

$$\begin{aligned}
\text{minimize } C(S_1, S_2, ..., S_M) &= \sum_{i=1}^{M} h_i E\left[(S_i - N_i)^+\right] \\
&= \sum_{i=1}^{M} h_i \sum_{k=0}^{S_i-1} (S_i - k) \Pr\{N_i = k\} \\
&= \sum_{i=1}^{M} h_i \sum_{k=0}^{S_i-1} \sum_{j=k}^{S_i-1} \Pr\{N_i = k\} \\
&= \sum_{i=1}^{M} h_i \sum_{j=0}^{S_i-1} \sum_{k=0}^{j} \Pr\{N_i = k\} \\
&= \sum_{i=1}^{M} \sum_{j=0}^{S_i-1} h_i \Pr\{N_i \leq j\}
\end{aligned}$$

So, finally we obtain our objective function as:

$$\text{minimize} \quad C(S_1, S_2, ..., S_M) = \sum_{i=1}^{M} \sum_{j=0}^{S_i-1} h_i \ Pr\{N_i \leq j\} \tag{3.16}$$

$$\text{subject to} \quad Pr\{N_i < S_i, \forall i \in U(k)\} \geq \alpha_k, \ k = 1, 2, ..., K \tag{3.17}$$

$$S_i \geq 0, \forall i = 1, 2, ..., M \tag{3.18}$$

Let $\mathbf{S} = \{S_1, S_2, ..., S_M\}$ and let $\mathbf{e_i}$ be the unit vector of size M with 1 at i^{th} position.

Then,

$$C(\mathbf{S}+\mathbf{e_i}) - C(\mathbf{S}) = h_i Pr\{N_i \leq S_i\} \tag{3.19}$$

For convenience, let

$$g_k(\mathbf{S}) = Pr\{N_i \leq S_i, \forall i \in U(k)\} \tag{3.20}$$

For a given vector of base stock levels $\mathbf{S}$ define sets:

$$FS(\mathbf{S}) = \{k \in \{1,2,...,K\} : g_k(\mathbf{S}) \geq \alpha_k\} \tag{3.21}$$

$$FS^c(\mathbf{S}) = \{k \in \{1,2,...,K\} : g_k(\mathbf{S}) < \alpha_k\} \tag{3.22}$$

$FS(\mathbf{S})$ is the set of kits for which the kit availability probabilities are satisfied. Likewise, $FS^c(\mathbf{S})$ is the complementary set, those kits for which the kit availability probabilities are not satisfied at $\mathbf{S}$.

Algorithm:

Step 0. Set $\mathbf{S} = \{S_1^L, S_2^L, ..., S_M^L\}$ and $\Omega = \{1,2,...,M\}$, $CM = \infty$, $CS = \emptyset$, $BS = \emptyset$. Compute $C(\mathbf{S})$ and compute $g_k(\mathbf{S})$ for all $k = 1,2,...,K$. If $FS^c(\mathbf{S}) = \emptyset$, stop. $\mathbf{S^h} = \mathbf{S}$. Otherwise, go to Step 1.

Step 1. For all $i \in \Omega$, if $i \in U(FS(\mathbf{S}))$ and $i \notin U(FS^c(\mathbf{S}))$ (i is an item that only belongs to one of the kits whose service level is feasible), then $\Omega = \Omega - i$.

Step 2. For all $i \in \Omega$, compute $C(\mathbf{S}+\mathbf{e_i})$, and $g_k(\mathbf{S}+\mathbf{e_i})$ for $k = 1,2,...,K$. If $FS^c(\mathbf{S}+\mathbf{e_i}) = \emptyset$, then $CS = CS \cup \{i\}$. If $CS = \emptyset$, go to Step 4.

Step 3. Let $i^* \in CS$ be such that $C(\mathbf{S}+\mathbf{e_i})$ is minimum. If $C(\mathbf{S}+\mathbf{e_i}) < CM$, then let $CM = C(\mathbf{S}+\mathbf{e_i}*)$ and $\mathbf{S^h} = \mathbf{S}+\mathbf{e_i}*$. If $CS = \Omega$, then stop. Otherwise, go to Step 4.

Step 4. For all $i \in \Omega - CS$, if $C(\mathbf{S}+\mathbf{e_i}) < CM$, set $BS = BS \cup \{i\}$. If $BS = \emptyset$, stop.

Otherwise, for all $i \in BS$ define

$$R(\mathbf{S}+\mathbf{e_i}) = \frac{C(\mathbf{S}+e_i) - C(\mathbf{S})}{\sum_{k=1}^{K}\left(g_k(\mathbf{S}+\mathbf{e_i}) - g_k(\mathbf{S})\right)} \tag{3.23}$$

Let $i^* \in BS$ be such that $R(\mathbf{S}+\mathbf{e_i})$ is minimum. Let $S = \mathbf{S}+\mathbf{e_{i^*}}$, $CS = \emptyset$, $BS = \emptyset$ and go to Step 1.

We explain how the algorithm works using an example. Consider two kits and four items with $U(1) = \{1,2,3\}$ and $U(2) = \{1,4\}$. Let $\lambda_1 = 2$ and $\lambda_2 = 1$, and $E[T_1] = 0.5, E[T_2] = 0.2$. Item consumption probabilities are (0.4,0.3,0.3) and (0.3,0.7) for Kit 1 and 2, respectively. Let $E[\tau_i] = 1$ for all i=1,2,3,4 and the requested service levels for both kits are 90%.

In Table 3.1 the steps of the algorithm are presented. We begin with the lower bound. If we satisfy the service level (kit availability) constraints for both kits, then lower bound is the optimal solution, if not we proceed to first step where we increase the stock level of one item in each case. For example, in Table 3.1 the lower bound is $(S_1^L, S_2^L, S_3^L, S_4^L) = (5,4,4,3)$, and the service level constraints are not satisfied, so in the first step we will examine the ratio of the incremental cost to incremental availability for the cases $(S_1, S_2, S_3, S_4) = (6,4,4,3), (5,5,4,3), (5,4,5,3)$ and $(5,4,4,4)$. If the kit availability constraints for both kits are satisfied we will simply choose the one with minimum cost. If one of the kit availability constraints is not satisfied, then for the next step we will choose the combination with the minimum ratio. At the end of first step, the service level constraints for both of the kits are not satisfied. The stock combination with minimum ratio which is (6,4,4,3) with ratio 11,958 is chosen for the next step. Another consideration here is that, if one of the service level constraints is satisfied, in the next step we will not increase the stock level of the item which only belongs to this kit but not any of the other kits. For the case below, in the first step the service level constraint for the second kit is satisfied (0.911), so we do not need to increase the stock level of item 4 which only belongs to second kit. If we increase the stock level of this item, in the next step we are sure that the ratio will be higher and we will discard this combination. For example we will not examine the case (6,4,4,4) because it is obvious

that increasing the stock level of 4^{th} item to 4 will increase the cost unnecessarily. In the second step of the example, again the service level constraints for both kits are not satisfied and we will choose the combination with the minimum ratio. There are two combinations which have the same ratio, we will choose arbitrarily. Also, again for this step we will not increase the stock level of the 4^{th} item as the service level constraint for the second kit is already satisfied. We choose (6,5,4,3) for the second step and proceed to next step. In the third step, all combinations satisfy the service level constraints for both kits, so we simply choose the one with minimum cost which is (6,5,5,3) with cost 11.733. Finally, this is reported as the optimal solution of our system obtained using the heuristic approach.

Table 3.1. Example for heuristic approach - Steps of algorithm

	S_1	S_2	S_3	S_4	Cost	Kit 1 Availability	Kit 2 Availability	Incremental Cost	Incremental Availability	Ratio
Lower Bound	5	4	4	3	8.810	0.830	0.863			
First Step	6	4	4	3	9.780	0.863	0.911	0.970	0.081	**11.958**
	5	5	4	3	9.786	0.859	0.863	0.976	0.029	34.229
	5	4	5	3	9.786	0.859	0.863	0.976	0.029	34.229
	5	4	4	4	9.797	0.830	0.905	0.987	0.042	23.315
Second Step	7	4	4	3	10.771	0.873	0.929	0.991	0.028	35.140
	6	5	4	3	10.756	0.896	0.911	0.976	0.033	**29.666**
	6	4	5	3	10.756	0.896	0.911	0.976	0.033	**29.666**
Third Step	7	5	4	3	11.747	0.908	0.929			
	6	6	4	3	11.750	0.903	0.911			
	6	5	5	3	**11.733**	0.940	0.911			
Optimal Solution	6	5	5	3	11.733	0.940	0.911			

4. NUMERICAL COMPUTATIONS

In this section, we present the results of our numerical experiments and discuss our key observations. The implementation of our model is carried out in stages. In the implementation phase of the model, technical computing program MATLAB 7.0 is employed.

Our first aim is to compute the probability a demand can be satisfied immediately for any given stock level of each item. We derive the joint probability distribution equation in the previous chapter. However, we need a practical way to compute the kit availability because as the number of kits and items increase, the computations become harder to handle manually. So, in the first stage of the computation phase we developed a code to calculate the kit availability (service level) for a given stock level of each item. During development stage, the calculations done by the MATLAB code is simultaneously checked manually and with the simulation model explained in Chapter 5.

Optimizing the base-stock levels for a given service level constraint is the second goal of this study. To accomplish this goal, we extend our code computing the kit availability for a given stock level firstly to calculate the same figure for all combinations of stock levels between a lower stock level and an upper stock level. This helps us to determine the exact optimum stock levels in a greedy fashion. The code computes kit availability and cost of all comnibations of stock levels, eliminates the ones which are not satisfying the service level constraint and finally selecting the one with minimum cost from the set of combinations satisfying the service level constraint. The exact approach however is impractical as all calculations are done for each combination from the lower stock level to the upper stock level. This is an exact but a very time consuming method to obtain optimum base-stock levels.

In section 3.2 of the previous chapter, an algorithm for calculating the optimum base-stock levels heuristically is presented. The heuristic approach is based on the ratio

of increase in holding cost to increase in kit availability each time we consider a new stock-level combination. A routine for calculating the base-stock levels heuristically is also developed using MATLAB. MATLAB codes computing the optimum base-stock levels both exactly and heuristically are presented in Appendix B.

To test the performance of our heuristic, we carried out a series of numerical experiments for two systems. The first system we consider is a processing system where replenishment lead times for each item is independent as in an infinite server queue (M/G/∞). The second system is a load dependent system where items are produced on separate single-server production facilities with exponentially distributed production times (M/M/1). For all experiments, there are two kits and demands for these kits are arrived according to Poisson process with overall arrival rate λ=1. We set the service level constraint as 90% for both kits and obtain base-stock levels satisfying this constraint exactly and heuristically.

For M/G/∞ experiments, the parameter values (kit probabilities (P_k), item probabilities in each kit ($p_{i,k}$), expected common time spent on site ($E[T_k]$), expected time to replenish an item ($E[\tau_i]$) and holding cost of each item (h_i)) are generated randomly using the distributions presented in Table 4.1. Number of items and items present in each kit are also determined randomly by a code implemented in MATLAB. Totally, 100 experiments are conducted. The distribution of these experiments depending on the number of items is also given in Table 4.1. As the parameters are randomly generated, to be able to use the same parameters for the heuristic experiments, the parameters are written into a Microsoft Excel file and for the set of heuristic runs parameters are read from this file.

Among 100 runs conducted exactly and heuristically, results of 6 heuristic runs differ from the results of the exact runs. Table 4.2 presents the kits, results of exact and heuristic runs and the amount of deviation in base-stock levels and total cost. The parameters and results of all runs are presented in Tables A.1 to A.11 given in Appendix A. In the first two columns of these tables we see which items are included in Kit 1 and Kit 2. In the next column, in the first part of the tables, the overlapping ratio which

can be defined as ratio of the number of overlapping items to the number of overlapping kits is given. This ratio is an indicator of the dependency of kits. Subsequent columns present the parameter values for kit probabilities (P_k), item probabilities in each kit ($p_{i,k}$), expected common time spent on site ($E[T_k]$), expected time to replenish an item ($E[\tau_i]$) and holding cost of each item (h_i) respectively. In the second part of these tables, results of exact and heuristic runs for these parameters are displayed.

When we examine the deviated results, we observe that the maximum difference in stock levels is 4, where it is 2.3 on the average. The percentage cost difference between the exact and heuristic is 3.05 % on average with a minimum of 0% and a maximum of 7.55%. These figures are demonstrated also in Table 4.1.

Table 4.1. Summary of M/G/∞ run parameters and results

Number of runs	100
Number of heuristic runs different from exact runs	6
Number of 2-item runs	8
Number of 3-item runs	40
Number of 4-item runs	35
Number of 5-item runs	17
λ	1
Kit probabilities	DiscreteUniform[0:0.05:1]
Item probabilities	Uniform[0;1]
T	DiscreteUniform[0.5:0.5:2.5]
τ	DiscreteUniform[0.1 0.25 0.5 1 1.5 2 2.5 3]
h	DiscreteUniform[1 1.5 2]
Maximum stock difference between heuristic runs and exact runs	4
Average stock difference between heuristic runs and exact runs	2.3
Maximum cost difference between heuristic runs and exact runs	0.9750
Average cost difference between heuristic runs and exact runs	0.4758
Percentage of maximum cost difference between heuristic runs and exact runs	7.55%
Percentage of average cost difference between heuristic runs and exact runs	3.05%

Table 4.2. Deviations in M/G/∞ runs

Kit 1	**Kit 2**	**Exact**						**Heuristic**						**Deviation**					
		S_1	S_2	S_3	S_4	S_5	**Cost**	S_1	S_2	S_3	S_4	S_5	**Cost**	ΔS_1	ΔS_2	ΔS_3	ΔS_4	ΔS_5	**ΔCost (%)**
{1,2,3}	{1}	6	4	4	-	-	12.9206	7	3	4	-	-	13.8956	1	-1	0	-	-	7.55
{2,3,4}	{1,3}	3	2	3	4	-	10.0818	3	2	4	3	-	10.5592	0	0	1	-1	-	4.73
{1,3,4}	{1,2,4}	6	3	5	7	-	17.3242	6	4	5	6	-	17.3299	0	1	0	-1	-	0.03
{2,3,4,5}	{1,3}	3	4	3	5	2	18.3529	3	3	4	5	2	18.3546	0	-1	1	0	0	0.01
{4,5}	{1,2,3,4}	3	5	3	6	4	16.8156	4	4	3	5	5	17.7500	1	0	0	-1	0	5.56
{2,3,5}	{1,2,4,5}	5	3	2	3	4	18.1634	4	3	2	3	5	18.6238	-1	0	0	0	1	2.53

We carried out similar experiments for M/M/1 system but in this case instead of determining parameters from randomly generated data, we define two sets (I and II in Table 4.3) for kit probabilities (P_k), item probabilities in each kit ($p_{i,k}$), expected common time spent on site ($E[T_k]$), traffic intensity ($\rho_i = E[\tau_i] \sum_{k \in V(i)} \lambda_k p_{k,i}$) and item holding cost (h_i) and set up a controlled environment with combinations of each level of these parameters. We consider two kits and three items. We study three kit combinations: a) $U(1) = \{1, 2\}$, $U(2) = \{3\}$ b) $U(1) = \{1, 2\}$, $U(2) = \{1, 3\}$ and c) $U(1) = \{1, 2, 3\}$, $U(2) = \{1, 3\}$. We have defined the overlapping ratio as the ratio of the number of overlapping items to the number of overlapping kits, so these cases correspond to overlapping ratios of 0, 0.5 and 1 respectively.

We have conducted 96 experiments. Each experiment corresponds to a different combination of the above kits and parameters. The number of heuristic runs differ from the exact runs is 8. Table 4.4 presents the kits, results of exact and heuristic runs and the amount of deviation in base-stock levels and total cost. The parameters and results of all runs are presented in Tables A.12 to A.35 given in Appendix A. These tables are presented in pairs. The first table in a pair gives the figures computed exactly and the second table heuristically. First two columns of the tables give kit probabilities (P_k). Subsequent columns gives item probabilities ($p_{i,k}$) in each kit. In the next two columns, expected time spent on site ($E[T_k]$) are presented where in the following columns item holding costs (h_i) are displayed. Last columns give the results of the experiments for base-stock levels, total cost and service levels respectively.

Results of the experiments show that the maximum difference in stock levels is 2 where it is again 2 on the average. When we look at the figures, the percentage cost difference between the exact policy and heuristic are relatively small. The percentage cost difference between the exact and heuristic is 0.07 % on average with a maximum of only 0.12%. These figures are represented in Table 4.3.

Table 4.3. Summary of M/M/1 run parameters and results

Number of runs	96	
Number of heuristic runs different from exact runs	8	
	I	**II**
λ	1	1
Kit probabilities	$P_1 = P_2 = 0.5$	$P_1 = 0.8$ $P_2 = 0.2$
Item probabilities	1/(no of items in a kit)	$p_{k,1} = 0.75$ $p_{k,i}$=0.25/(no of items in a kit-1) for all $i \neq 1$
T	$T_1 = T_2 = 1.5$	$T_1 = 2.5$ $T_2 = 1$
ρ	$\rho_1 = \rho_2 = \rho_3 = 0.5$	$\rho_1 = 0.8$ $\rho_2 = \rho_3 = 0.5$
h	$h_1 = h_2 = h_3 = 1.5$	$h_1 = 2$ $h_2 = h_3 = 0.5$
Maximum stock difference between heuristic runs and exact runs	2	
Average stock difference between heuristic runs and exact runs	2	
Maximum cost difference between heuristic runs and exact runs	0.0352	
Average cost difference between heuristic runs and exact runs	0.01688	
Percentage of maximum cost difference between heuristic runs and exact runs	0.12%	
Percentage of average cost difference between heuristic runs and exact runs	0.07%	

Table 4.4. Deviations in M/M/1 runs

Kit 1	**Kit 2**	**Exact**				**Heuristic**				**Deviation**			
		S_1	S_2	S_3	**Cost**	S_1	S_2	S_3	**Cost**	ΔS_1	ΔS_2	ΔS_3	**ΔCost (%)**
{1,2}	{3}	5	6	5	15.3102	6	5	5	15.3102	1	-1	0	0.00
{1,2}	{3}	5	6	5	15.3102	6	5	5	15.3102	1	-1	1	0.00
{1,2,3}	{1,3}	7	7	8	21.0412	8	7	7	21.0412	1	0	1	0.00
{1,2,3}	{1,3}	7	7	8	21.0412	8	7	7	21.0412	1	0	1	0.00
{1,2,3}	{1,3}	16	7	7	29.5769	15	7	8	29.6121	-1	0	1	0.12
{1,2,3}	{1,3}	17	8	8	31.0121	16	8	9	31.0444	-1	0	1	0.10
{1,2,3}	{1,3}	16	7	7	29.5769	15	7	8	29.6121	-1	0	1	0.12
{1,2,3}	{1,3}	17	8	8	31.0121	16	8	9	31.0444	-1	0	1	0.10

The results of experiments for two different systems reveal that the heuristic algorithm works quite well when compared with the exact results. The base-stock levels obtained heuristically are good approximations to those computed exactly.

Another advantage of the heuristic method is that run-time of heuristic method is very small when compared with run-time of exact approach. For example, a 3-item experiment in an M/M/1 system takes about 20 minutes to compute exactly while it takes only 20-30 seconds heuristically. The exact method examines all combinations of stock levels in an exhaustive fashion to select the optimum level so it takes so much time to obtain the result. As the number of items and overlapping of kits increase, the run time also increases drastically. For a 3-item exact run, time to complete the run is approximately 20 minutes but for a 5-item system it may go up to 6-7 hours depending on the traffic intensity.

We may conclude that our heuristic approach is an efficient computational practice to analyze large systems in very short times with a very small error.

In the previous chapter, we derive the joint distribution of N_i. We show that these are not independent collections. Classical approach ignores the dependency of demands and assumes N_i are independent. In our model, by taking the dependency of demands into account, we provide an improved procudure to calculate the base stock levels.

To compare two approaches, we have conducted some experiments in an M/G/∞ system. Kit availability values for both systems are computed and compared for the parameters presented in Table 4.5. The results for both systems are summarized in Table 4.6. Here, SL_1 and SL_2 show the kit availability values for the system where demands are dependent and β_1 and β_1 show for the system where demands are assumed to be independent. We observe from the results of experiments that for the optimum base-stock levels of each item if we consider dependency of demands we obtain higher kit availability figures. This implies that if we have assumed independent demands and solve our system according to this assumption, to obtain the same service levels we

would hold more items in stock and this would increase the holding cost unnecessarily.

Table 4.5. Run parameters of experiments for independence assumption

Kit 1	**Kit 2**	P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$	$p_{1,5}$	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$	$p_{2,5}$
{1,2}	{1,3}	0.25	0.75	0.25	0.75	0	-	-	0.418	0.582	-	-	-
{1,2}	{3,4}	0.8	0.2	0.55	0.45	0	0	-	0.815	0.185	0	0	-
{1,2,3}	{1,4}	0.35	0.65	0.173	0.794	0.033	0	-	0.02	0.98	0	0	-
{1,2,3,4}	{1,4}	0.95	0.05	0.452	0.135	0.052	0.362	-	0.651	0.349	0	0	-
{2,4}	{1,2,3,4}	0.2	0.8	0.382	0.618	0	0	-	0.152	0.477	0.286	0.086	-
{1,4,5}	{2,3,4}	0.85	0.15	0.321	0.095	0.584	0	0	0.146	0.316	0.539	0	0
{1,2,3,4}	{2,4,5}	0.65	0.35	0.26	0.321	0.224	0.195	0	0.194	0.571	0.235	0	0
{1,2,3,4,5}	{2,4}	0.25	0.75	0.264	0.272	0.367	0.075	0.022	0.537	0.463	0	0	0

Kit 1	**Kit 2**	T_1	T_2	τ_1	τ_2	τ_3	τ_4	τ_5	h_1	h_2	h_3	h_4	h_5
{1,2}	{1,3}	2	1.5	1	2.5	2.5	-	-	2	1	1	-	-
{1,2}	{3,4}	1.5	2.5	0.25	2.5	0.1	3	-	1	1	2	1	-
{1,2,3}	{1,4}	2.5	2	1.5	2	3	1.5	-	2	1.5	2	1	-
{1,2,3,4}	{1,4}	1	2.5	0.5	0.1	0.25	0.1	-	1.5	1	1.5	1	-
{2,4}	{1,2,3,4}	1	1.5	1	3	0.25	1	-	1.5	2	1	2	-
{1,4,5}	{2,3,4}	2	1	1	0.1	0.5	2	2.5	1.5	1.5	1	1	2
{1,2,3,4}	{2,4,5}	1	2	1	2	2	1	1	1.5	2	1.5	1.5	2
{1,2,3,4,5}	{2,4}	0.5	1.5	0.5	0.1	1.5	3	1.5	1	2	1	2	2

Table 4.6. Comparison of dependence and independence assumption

Kit 1	**Kit 2**	S_1	S_2	S_3	S_4	S_5	SL_1	SL_2	β_1	β_2
{1,2}	{1,3}	7	3	6	-	-	0.9316	0.914	0.929	0.9029
{1,2}	{3,4}	4	5	2	3	-	0.9015	0.903	0.8777	0.8829
{1,2,3}	{1,4}	6	5	3	5	-	0.9165	0.9071	0.8942	0.8945
{1,2,3,4}	{1,4}	4	3	3	4	-	0.9127	0.9542	0.799	0.939
{2,4}	{1,2,3,4}	4	7	4	4	-	0.9133	0.9024	0.9007	0.8264
{1,4,5}	{2,3,4}	5	2	2	5	7	0.9041	0.9207	0.8566	0.9082
{1,2,3,4}	{2,4,5}	3	5	4	5	3	0.9076	0.9176	0.8684	0.8871
{1,2,3,4,5}	{2,4}	2	4	2	6	2	0.9128	0.9404	0.8818	0.929

5. SIMULATION FRAMEWORK FOR THE MODEL

A simulation model which realistically reflects our research problem and supports our calculations is developed and details of the framework are given in this chapter. Firstly, steps of the model are presented briefly. Secondly, two cases are studied to compare kit availabilities computed exactly by MATLAB and simulated by Arena. Lastly, kit availabilities in systems with different processing distributions are evaluated by simulation.

The simulation software package Arena 9.0 is used for the implementation of the simulation model of the multi-item system studied. The system is modeled by the basic process modules, elements and blocks of Arena 9.0. The input data of the system can be listed as; number of kits, number of items, items present in each kit, stock levels for each item, demand arrival distribution, kit probabilities, item probabilities in each kit, expected common time spent on site and item processing distribution. The simulation model is designed and developed such that arrival times and item processing times (replenishment times) can be randomly generated based on probability distributions defined. Kit availabilities (service level for each kit) for the given item stock levels are obtained as the output of the model.

Figure 5.1 presents a simplified flow of our multi-item system. All items are created at the beginning; their types are assigned immediately and are sent to a queue which will simulate the stock of each item. Then, a kit demand is generated. Any distribution or expression may be used to set the time between arrivals. We study a Poisson arrival system so time between arrivals is set as exponential. When a demand arrival occurs, a kit type is assigned according to a discrete probability distribution. In the next step, program searches the stock whether the items in the demanded kit are present. If all the items are present, they are removed from the stock and sent to grouping to form the kit. If one or more of the items are missing, these items are supplied from an emergency source by duplicating the item needed. If an item of the type supplied from emergency source is replenished meanwhile, this item is returned

to the source. When a kit is formed by grouping the items present in the demanded kit, they are delayed on site for a common time and an item is selected to be used according to a discrete probability distribution. The unused items are sent back to the stock and the used item is ordered. The used item can be replenished through an infinite server system, a single server system or a multi-server system where different distributions may be used to reflect the processing period. We study an M/G/∞ and an M/M/1 system. Finally, the item produced/supplied sent to its stock. Details of the simulation model are presented in Appendix C.

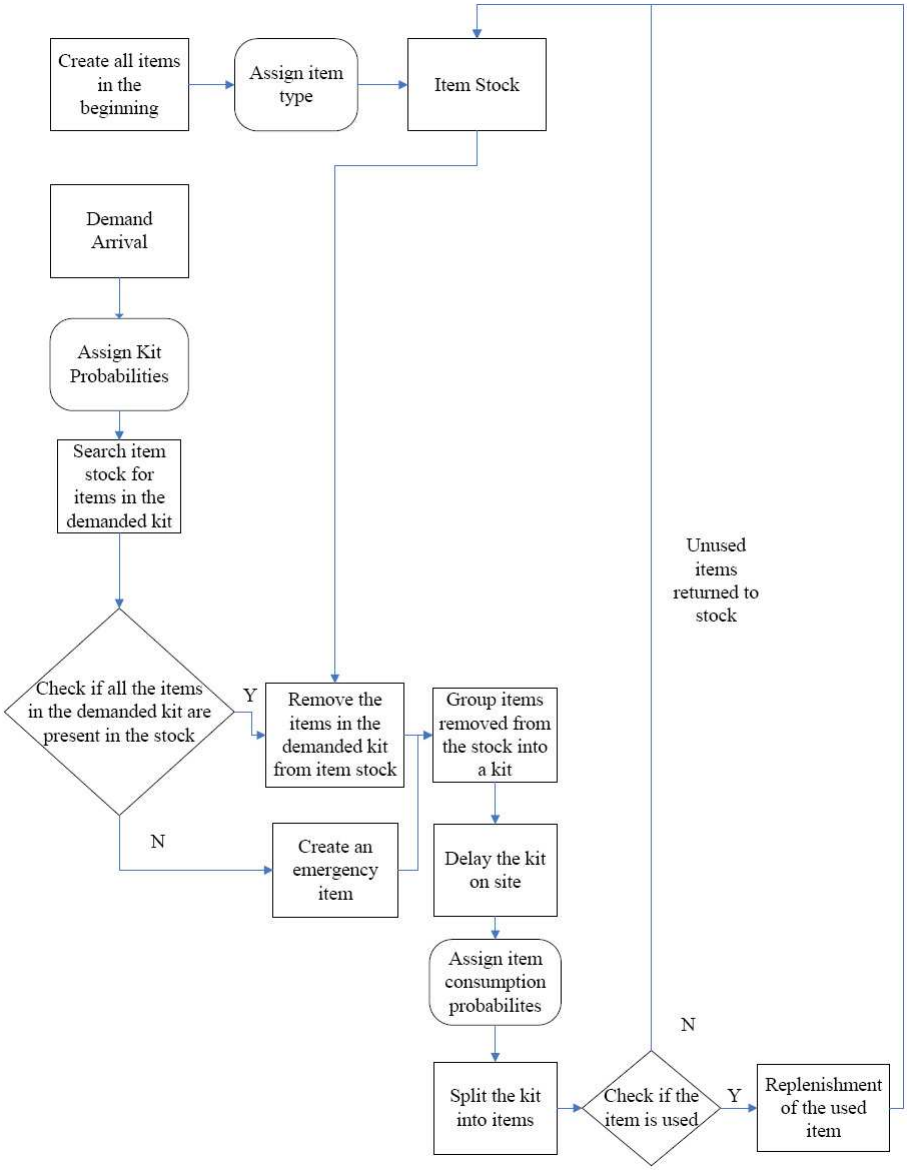

Figure 5.1. Arena flow chart (simplified)

The performance of our simulation model is evaluated by comparing the kit availabilities computed exactly using the MATLAB routine explained in the previous chapter and simulated by Arena. Two different cases are studied, one with a single kit and the other with two kits sharing a common item. The details of the cases and the results obtained exactly and by simulation are presented below. The system runs for 10,000 days for 10 replications.

Case 1: Consider a single kit with 3 items with consumption probabilities (0.5, 0.2, 0.3). Demand arrives according to a Poisson process with rate $\lambda = 0.3$ and $E[T] = 0.5$. Let $E[\tau_1] = 2$, $E[\tau_2] = 1$ and $E[\tau_3] = 2$. We compute kit availability for the stock levels of (2,1,1) using both Arena and MATLAB firstly in a system with independent replenishment times (through an M/G/∞ queue) and secondly in a system with dependent replenishment times (through an M/M/1 queue).

Kit availability in an M/G/∞ system computed using MATLAB is 0.6519 while it is 0.6520 when computed using Arena simulation. In an M/M/1 system the kit availability calculated using Matlab is 0.6037 and 0.6035 when simulated by Arena.

Case 2: Consider two kits and four items with $U(1) = \{1, 2, 3\}$ and $U(2) = \{1, 4\}$. Let $\lambda = 1$, kit demand probabilities (0.6, 0.4) and $E[T_1] = 0.5, E[T_2] = 0.2$. Item consumption probabilities are (0.4,0.3,0.3) and (0.3,0.7) for kits 1 and 2, respectively. Let $E[\tau_i] = 1$ for all i=1,2,3,4. We compute kit availabilities for kits 1 and 2 for the stock levels of (5,5,5,5) using both Arena and MATLAB in an M/G/∞ system and in an M/M/1 system. Table 5.1 gives kit availability for an M/G/∞ system and Table 5.2 for an M/M/1 system.

Table 5.1. Comparison of kit availabilities computed using MATLAB and ARENA in an M/G/∞ system

	MATLAB	ARENA
Kit Availability 1	0.7673	0.7669
Kit Availability 2	0.7950	0.7940

Table 5.2. Comparison of kit availabilities computed using MATLAB and ARENA in an M/M/1 system

	MATLAB	ARENA
Kit Availability 1	0.6784	0.6782
Kit Availability 2	0.6948	0.6947

From the results of the above two cases, it is seen that the kit availability values computed exactly using MATLAB are well approximated by simulation. One may choose to evaluate the kit availability values for a given stock level by simulation instead of using the exact computation.

This framework also provides a platform to analyze different arrival and processing distributions. We consider the second case above and use different distributions for item replenishment (exponential, deterministic, uniform and erlang) in an infinite server and a single server system. The results of both systems are presented in Table 5.3 and Table 5.4.

Table 5.3. Comparison of different distributions in an infinite server queue by simulation

	Exponential	Deterministic	Erlang-2	Uniform
Kit Availability 1	0.76687	0.76835	0.58739	0.76696
Kit Availability 2	0.79403	0.79505	0.61175	0.79175

Table 5.4. Comparison of different distributions in a single server queue by simulation

	Exponential	Deterministic	Erlang-2	Uniform
Kit Availability 1	0.67815	0.72674	0.32243	0.69935
Kit Availability 2	0.69470	0.74988	0.32496	0.72444

The results of the above examples shed light on how different replenishment models affect the kit availabilities. The advantage of the simulation model is to obtain the kit availabilities in varying situations easily without reconsidering how to derive kit availability equation for these varying situations (different arrival distributions, single, multiple or infinite server queues, various processing times etc.).

6. CONCLUSION

In this study, we model and analyze a kit management problem. We consider a multi-item inventory system in which customers may order different but possibly overlapping kits. The aim of the study is to evaluate the availability of a kit to be formed when a demand arrives, and to determine the optimal base-stock levels for the multi-item inventory subject to a service level constraint.

Customer demand follows a Poisson process where there is a fixed probability for each kit to be demanded. Demands are filled on a first-come-first-served basis. When a demand for a kit is arrived, if the on-hand inventory for all the items in the requested kit is available in the stock, the demand is filled immediately but if there are missing items, these items are supplied from an exogenous source. The kit is then sent to the demand location for a common time applying to all items in the kit and one of the items is used in the demand location. The unused items in the kit are returned to stock while an order is placed for the used item. We consider two supply processes for the used item. In the first supply system replenishment lead times for each component are independent random variables as in an M/G/∞ queue. Second supply system is a load dependent system where items are processed through an M/M/1 queue.

The availability of a kit for a given base-stock level can be computed by evaluating the joint distribution of outstanding orders. We developed a model for calculating the joint probability of outstanding orders. In the implementation stage, we built up a code in MATLAB technical computing program and a simulation model in simulation package Arena. With the use of both MATLAB code and Arena model, for a given base-stock level the probability a demand can be filled is computed easily. Arena model also gives us the flexibility to define different arrival and replenishment distributions without a need to derive the joint probability equation every time.

Secondly, we developed an efficient algorithm to optimize the base-stock levels of each item heuristically. We have carried out many numerical computations to prove the

effectiveness of our heuristic approach. The numerical examples are conducted both in a greedy fashion to obtain exact results and by using the heuristic method. Results of these studies indicate the heuristic approach provide efficient solutions in relatively very shorter times when compared to exact approach.

Lastly, to present the improvement provided by our model where we take the dependency of demands into account, we have carried out experiments for our model and a classical model which ignores the dependency of demands. We have concluded from the results of experiments that for obtaining the service levels provided by our model, in a classical system more items have to be hold in stock which would lead an increase in holding costs.

There are some points open to further research. In our model, we consider only one item of a specific type is available in a kit. This may be extended to multiple items of the same item type. Another approach for a further study may be to allow backorders and obtain new performance measures like the average number of backorders for the studied model.

APPENDIX A: RUN PARAMETERS AND RESULTS

This appendix presents the parameters and results of the experiments conducted for an M/G/∞ and an M/M/1 system.

Tables A.1 -A.11 show the run parameters and results of both exact and heuristic runs for an M/G/∞ system. For all experiments, two kits are considered but the items in the kits are randomly determined using a routine performed in MATLAB. Parameters for M/G/∞ experiments are also randomly determined according to the distributions presented in Table 4.1 in Chapter 4. In the tables, first two columns represent the items in Kit 1 and 2. Subsequent columns of the first part of the table give all the parameters which are randomly selected. In the second part exact base-stock levels, cost and the service levels are presented firstly. These columns are followed by the same figures computed heuristically. The tables are classified according to the total number of items.

Tables A.12 -A.35 show the run parameters and results of exact and heuristic runs for an M/M/1 system. A set of controlled experiments are conducted for an M/M/1 system. Experiments are conducted according to all combinations of parameters given in Table 4.3 in Chapter 4. There are again two kits. The tables are classified according to the items in the kits, overlapping ratio and traffic intensity and are presented in pairs. The first table in a pair gives the figures computed exactly and the second table heuristically. One may compare the results of the exact and heuristic runs for the same parameters easily from these tables.

Table A.1. M/G/∞ Run parameters and results - 2-item runs

Kit 1	**Kit 2**	**OR**	P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	τ_1	τ_2	h_1	h_2
{2}	{1}	0	0.3	0.7	1	0	1	0	2	2.5	3	2.5	1.5	1.5
{1}	{2}	0	0.6	0.4	1	0	1	0	2	2.5	2.5	1	1	1.5
{1}	{2}	0	0.25	0.75	1	0	1	0	1.5	2	2.5	2	1	1
{2}	{1,2}	0.5	0.55	0.45	1	0	0.259	0.741	1.5	2.5	2	1	2	1
{1}	{2}	0	0.6	0.4	1	0	1	0	0.5	0.5	1	1.5	1	2
{1}	{2}	0	0.3	0.7	1	0	1	0	0.5	0.5	1.5	2	1	1.5
{2}	{1}	0	0.4	0.6	1	0	1	0	1.5	2	1.5	0.25	1	1.5
{2}	{1}	0	0.55	0.45	1	0	1	0	1	0.5	1	0.1	2	2

Kit 1	**Kit 2**	**Exact**					**Heuristic**				
		S_1	S_2	**Cost**	SL_1	SL_2	S_1	S_2	**Cost**	SL_1	SL_2
{2}	{1}	7	4	8.4067	0.9518	0.9044	7	4	8.4067	0.9518	0.9044
{1}	{2}	6	4	6.8200	0.9433	0.9463	6	4	6.8200	0.9433	0.9463
{1}	{2}	3	6	4.6564	0.9197	0.9161	3	6	4.6564	0.9197	0.9161
{2}	{1,2}	4	6	7.9474	0.9319	0.9033	4	6	7.9474	0.9319	0.9033
{1}	{2}	3	3	5.2324	0.9371	0.9526	3	3	5.2324	0.9371	0.9526
{1}	{2}	3	5	6.4872	0.9769	0.9671	3	5	6.4872	0.9769	0.9671
{2}	{1}	5	3	5.5208	0.9659	0.9379	5	3	5.5208	0.9659	0.9379
{2}	{1}	3	3	7.3490	0.9764	0.9688	3	3	7.3490	0.9764	0.9688

Table A.2. M/G/∞ run parameters and results - 3-item runs

Kit 1	**Kit 2**	**OR**	P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	T_1	T_2	τ_1	τ_2	τ_3	h_1	h_2	h_3
{2}	{1,3}	0	0	1	1	0	0	0.767	0.233	0	1	2	2	2	0.1	1.5	1.5	1.5
{1}	{2,3}	0	0.5	0.5	1	0	0	0.411	0.589	0	2	1	2	3	1.5	2	2	2
{3}	{1,2}	0	0.75	0.25	1	0	0	0.678	0.322	0	1	0.5	0.1	2.5	0.5	1.5	2	1
{1,2}	{3}	0	0.2	0.8	0.946	0.054	0	1	0	0	1	1	0.25	0.1	0.1	2	1	2
{2,3}	{1}	0	0.3	0.7	0.519	0.481	0	1	0	0	2.5	0.5	1.5	0.25	0.5	1	1	1.5
{1,2}	{3}	0	0.85	0.15	0.545	0.455	0	1	0	0	2	2.5	2	2	0.5	2	1.5	1
{2}	{1,3}	0	0.9	0.1	1	0	0	0.653	0.347	0	0.5	2	3	0.25	0.1	1.5	1	1.5
{1}	{2,3}	0	1	0	1	0	0	0.352	0.648	0	1	1.5	2.5	3	1	1.5	2	1.5
{1,3}	{2}	0	0.65	0.35	0.209	0.791	0	1	0	0	2.5	2	1	0.25	1	1	1	1
{2}	{1,3}	0	0.95	0.05	1	0	0	0.606	0.394	0	2	0.5	2.5	1	3	1	1	2

Kit 1	**Kit 2**	**Exact**						**Heuristic**					
		S_1	S_2	S_3	**Cost**	SL_1	SL_2	S_1	S_2	S_3	**Cost**	SL_1	SL_2
{2}	{1,3}	7	2	5	11.0217	1	0.9080	7	2	5	11.0217	1	0.9080
{1}	{2,3}	5	4	3	14.2745	0.9473	0.9147	5	4	3	14.2745	0.9473	0.9147
{3}	{1,2}	2	2	4	5.9580	0.9724	0.9554	2	2	4	5.9580	0.9724	0.9554
{1,2}	{3}	2	2	3	6.3707	0.9739	0.9404	2	2	3	6.3707	0.9739	0.9404
{2,3}	{1}	4	3	3	6.7638	0.9444	0.9463	4	3	3	6.7638	0.9444	0.9463
{1,2}	{3}	6	6	2	12.7702	0.9270	0.9246	6	6	2	12.7702	0.9270	0.9246
{2}	{1,3}	2	3	2	4.7039	0.9688	0.9391	2	3	2	4.7039	0.9688	0.9391
{1}	{2,3}	7	2	2	8.7667	0.9347	1	7	2	2	8.7667	0.9347	1
{1,3}	{2}	5	3	5	7.6210	0.9284	0.9544	5	3	5	7.6210	0.9284	0.9544
{2}	{1,3}	2	6	2	6.1199	0.9304	0.9923	2	6	2	6.1199	0.9304	0.9923

Table A.3. M/G/∞ run parameters and results - 3-item runs (cont'd)

Kit 1	**Kit 2**	**OR**	P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	T_1	T_2	τ_1	τ_2	τ_3	h_1	h_2	h_3
{3}	{1,2,3}	0.5	0.45	0.55	1	0	0	0.647	0.185	0.168	2.5	2.5	0.25	0.25	0.1	1	1.5	1
{1}	{1,2,3}	0.5	0.65	0.35	1	0	0	0.067	0.731	0.202	2.5	1.5	2	1	3	1	1.5	1
{1,2}	{1,3}	0.5	0.25	0.75	0.250	0.750	0	0.418	0.582	0	1.5	2.5	1	2.5	2.5	1	1.5	1.5
{1,2,3}	{2}	0.5	0.95	0.05	0.177	0.317	0.506	1	0	0	2	1.5	2	1.5	0.5	2	1	1
{2}	{1,2,3}	0.5	0.85	0.15	1	0	0	0.136	0.516	0.348	1.5	1	0.1	0.5	0.25	2	1	1
{1,2,3}	{3}	0.5	0.85	0.15	0.418	0.196	0.386	1	0	0	1.5	2.5	0.1	0.1	1	1.5	1.5	1.5
{2}	{1,2,3}	0.5	0.8	0.2	1	0	0	0.667	0.149	0.183	1.5	0.5	1.5	1	0.25	1	1	1
{1,2,3}	{1}	0.5	0.65	0.35	0.679	0.015	0.306	1	0	0	1	2	3	1	2.5	1	2	1
{2}	{1,2,3}	0.5	0.5	0.5	1	0	0	0.155	0.576	0.269	1	0.5	0.25	2.5	2	1	1	1.5
{3}	{1,2,3}	0.5	0.85	0.15	1	0	0	0.142	0.151	0.706	1.5	2.5	1	0.25	2	1.5	1.5	1

Kit 1	**Kit 2**	**Exact**						**Heuristic**					
		S_1	S_2	S_3	**Cost**	SL_1	SL_2	S_1	S_2	S_3	**Cost**	SL_1	SL_2
{3}	{1,2,3}	4	4	6	9.2739	0.9543	0.9144	4	4	6	9.2739	0.9543	0.9144
{1}	{1,2,3}	8	3	3	8.9361	0.9734	0.9162	8	3	3	8.9361	0.9734	0.9162
{1,2}	{1,3}	7	3	6	11.4635	0.9316	0.9140	7	3	6	11.4635	0.9316	0.9140
{1,2,3}	{2}	5	6	6	12.5757	0.9100	0.9579	5	6	6	12.5757	0.9100	0.9579
{2}	{1,2,3}	2	5	2	5.9443	0.9568	0.9475	2	5	2	5.9443	0.9568	0.9475
{1,2,3}	{3}	4	4	5	11.4948	0.9179	0.9351	4	4	5	11.4948	0.9179	0.9351
{2}	{1,2,3}	2	5	2	4.7389	0.9349	0.9022	2	5	2	4.7389	0.9349	0.9022
{1,2,3}	{1}	8	3	4	10.4740	0.9264	0.9636	8	3	4	10.4740	0.9264	0.9636
{2}	{1,2,3}	2	6	3	7.0472	0.9417	0.9113	2	6	3	7.0472	0.9417	0.9113
{3}	{1,2,3}	2	2	8	7.2505	0.9708	0.9157	2	2	8	7.2505	0.9708	0.9157

Table A.4. M/G/∞ run parameters and results - 3-item runs (cont'd)

Kit 1	**Kit 2**	**OR**	P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	T_1	T_2	τ_1	τ_2	τ_3	h_1	h_2	h_3
{1,2,3}	{1}	0.5	0.5	0.5	0.266	0.648	0.086	1	0	0	2	1	2	0.1	1.5	2	1	1.5
{1,3}	{1,2}	0.5	0.5	0.5	0.070	0.930	0	0.164	0.836	0	2	2.5	0.25	2.5	2.5	2	2	2
{1,2,3}	{2}	0.5	0.05	0.95	0.682	0.299	0.019	1	0	0	2	1.5	1	0.25	1.5	2	2	2
{1,2,3}	{2}	0.5	0.85	0.15	0.182	0.207	0.611	1	0	0	2	0.5	1	1.5	2	1.5	1	2
{1,2,3}	{3}	0.5	0.05	0.95	0.338	0.333	0.330	1	0	0	1	0.5	0.5	0.5	1	1	1	1
{2,3}	{1,3}	0.5	0.95	0.05	0.519	0.481	0	0.668	0.332	0	1.5	1	0.1	1	2.5	1	1	1
{1,2}	{1,3}	0.5	0.45	0.55	0.164	0.836	0	0.689	0.311	0	1.5	1	0.25	2.5	2.5	2	1	1.5
{1}	{1,2,3}	0.5	0.55	0.45	1	0	0	0.078	0.475	0.447	0.5	2	1	1.5	0.5	1	1	1.5
{1}	{1,2,3}	0.5	0.35	0.65	1	0	0	0.636	0.163	0.201	0.5	2.5	1	1.5	1.5	2	1	2

Kit 1	**Kit 2**	**Exact**						**Heuristic**					
		S_1	S_2	S_3	**Cost**	SL_1	SL_2	S_1	S_2	S_3	**Cost**	SL_1	SL_2
{1,2,3}	{1}	**6**	**4**	**4**	**12.9206**	**0.9254**	**0.9378**	**7**	**3**	**4**	**13.8956**	**0.9029**	**0.9769**
{1,3}	{1,2}	6	5	5	18.0644	0.9130	0.9024	6	5	5	18.0644	0.9130	0.9024
{1,2,3}	{2}	2	5	2	10.1244	0.9591	0.9660	2	5	2	10.1244	0.9591	0.9660
{1,2,3}	{2}	5	6	6	14.6080	0.9165	0.9719	5	6	6	14.6080	0.9165	0.9719
{1,2,3}	{3}	2	2	4	4.3038	0.9340	0.9354	2	2	4	4.3038	0.9340	0.9354
{2,3}	{1,3}	2	5	6	7.2504	0.9198	0.9454	2	5	6	7.2504	0.9198	0.9454
{1,2}	{1,3}	5	4	3	11.5254	0.9132	0.9178	5	4	3	11.5254	0.9132	0.9178
{1}	{1,2,3}	6	4	3	9.0481	0.9906	0.9092	6	4	3	9.0481	0.9906	0.9092
{1}	{1,2,3}	6	5	5	15.8886	0.9536	0.9292	6	5	5	15.8886	0.9536	0.9292

Table A.5. M/G/∞ run parameters and results - 3-item runs (cont'd)

Kit 1	**Kit 2**	**OR**	P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	T_1	T_2	τ_1	τ_2	τ_3	h_1	h_2	h_3
{1,2}	{1,2,3}	1	0.7	0.3	0.440	0.560	0	0.220	0.307	0.473	1.5	2.5	0.25	3	0.5	1	2	2
{1,3}	{1,2,3}	1	0.95	0.05	0.449	0.551	0	0.203	0.600	0.198	2	2	0.5	1.5	1	1	1	1
{1,2,3}	{1,3}	1	0.65	0.35	0.190	0.391	0.419	0.558	0.442	0	1	1.5	0.5	1	0.5	1.5	1.5	2
{1,3}	{1,2,3}	1	0.85	0.15	0.410	0.590	0	0.419	0.267	0.315	2	2	1.5	2	1.5	1.5	2	2
{2,3}	{1,2,3}	1	0.95	0.05	0.471	0.529	0	0.313	0.420	0.267	2	1.5	1.5	0.25	0.25	1	1	1
{2,3}	{1,2,3}	1	0.65	0.35	0.423	0.577	0	0.608	0.059	0.333	1	2	0.1	3	0.25	1	1.5	1
{1,2,3}	{1,3}	1	0	1	0.537	0.364	0.098	0.615	0.385	0	2.5	0.5	0.5	3	2	1.5	1.5	1
{1,2,3}	{2,3}	1	0.65	0.35	0.097	0.454	0.449	0.581	0.419	0	1.5	1.5	2.5	2.5	0.25	1.5	1	1.5
{2,3}	{1,2,3}	1	0.75	0.25	0.057	0.943	0	0.286	0.352	0.361	2	2.5	3	3	0.25	2	2	1
{1,3}	{1,2,3}	1	0.5	0.5	0.580	0.420	0	0.356	0.294	0.350	0.5	2.5	1.5	0.1	0.25	1	2	2
{1,3}	{1,2,3}	1	0.95	0.05	0.449	0.551	0	0.203	0.600	0.198	2	2	0.5	1.5	1	1	1	1

Kit 1	**Kit 2**	**Exact**						**Heuristic**					
		S_1	S_2	S_3	**Cost**	SL_1	SL_2	S_1	S_2	S_3	**Cost**	SL_1	SL_2
{1,2}	{1,2,3}	6	7	3	14.9359	0.9487	0.9125	6	7	3	14.9359	0.9487	0.9125
{1,3}	{1,2,3}	5	2	6	7.1082	0.9145	0.9076	5	2	6	7.1082	0.9145	0.9076
{1,2,3}	{1,3}	4	4	4	12.4257	0.9297	0.9350	4	4	4	12.4257	0.9297	0.9350
{1,3}	{1,2,3}	6	3	6	15.1864	0.9110	0.9070	6	3	6	15.1864	0.9110	0.9070
{2,3}	{1,2,3}	2	5	5	6.6102	0.9277	0.9245	2	5	5	6.6102	0.9277	0.9245
{2,3}	{1,2,3}	3	5	5	9.1433	0.9210	0.9031	3	5	5	9.1433	0.9210	0.9031
{1,2,3}	{1,3}	3	2	4	6.5980	0.9240	0.9240	3	2	4	6.5980	0.9240	0.9240
{1,2,3}	{2,3}	4	7	4	11.3627	0.9064	0.9127	4	7	4	11.3627	0.9064	0.9127
{2,3}	{1,2,3}	3	6	6	13.9176	0.9496	0.9093	3	6	6	13.9176	0.9496	0.9093
{1,3}	{1,2,3}	6	4	4	13.0919	0.9161	0.9151	6	4	4	13.0919	0.9161	0.9151
{1,3}	{1,2,3}	5	2	6	7.1082	0.9145	0.9076	5	2	6	7.1082	0.9145	0.9076

Table A.6. M/G/∞ run parameters and results - 4-item runs

Kit 1	**Kit 2**	**OR**	P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$	T_1	T_2	τ_1	τ_2	τ_3	τ_4	h_1	h_2	h_3	h_4
{1,2,4}	{3}	0	0.85	0.15	0.350	0.329	0.322	0	1	0	0	0	2.5	0.5	0.5	0.1	2	2	2	1.5	1.5	1
{1,2}	{3,4}	0	0	1	0.383	0.617	0	0	0.617	0.383	0	0	0.5	1.5	3	3	1.5	2	2	1	1	1.5
{2,4}	{1,3}	0	0.2	0.8	0.804	0.196	0	0	0.740	0.260	0	0	2	1	0.5	2	3	0.25	1	1	1	2
{1,3}	{2,4}	0	0.75	0.25	0.985	0.015	0	0	0.483	0.517	0	0	2.5	2.5	0.25	2	1.5	3	2	1	1.5	1
{3.4}	{1.2}	0	0	1	0.824	0.176	0	0	0.458	0.542	0	0	2.5	2	2.5	0.25	0.25	0.5	2	2	1.5	1.5
{2,4}	{1,3}	0	0.9	0.1	0.033	0.967	0	0	0.932	0.068	0	0	2	2	2.5	2.5	1.5	1	1.5	1.5	1	1.5
{1,2}	{3,4}	0	0.8	0.2	0.450	0.550	0	0	0.815	0.185	0	0	1.5	2.5	0.25	2.5	0.1	3	1	1	2	1
{1,3}	{2,4}	0	0.1	0.9	0.091	0.909	0	0	0.523	0.477	0	0	2.5	1	0.5	0.5	2	0.25	2	1.5	2	2
{1,3}	{2,4}	0	0.65	0.35	0.274	0.726	0	0	0.810	0.190	0	0	2	2	0.25	0.1	0.5	0.1	1	2	1	1

Kit 1	**Kit 2**	**Exact**							**Heuristic**						
		S_1	S_2	S_3	S_4	**Cost**	SL_1	SL_2	S_1	S_2	S_3	S_4	**Cost**	SL_1	SL_2
{1,2,4}	{3}	5	5	2	6	14.1768	0.9038	0.9450	5	5	2	6	14.1768	0.9038	0.9450
{1,2}	{3,4}	2	2	6	5	10.5084	1	0.9036	2	2	6	5	10.5084	1	0.9036
{2,4}	{1,3}	4	3	4	2	8.6021	0.9218	0.9324	4	3	4	2	8.6021	0.9218	0.9324
{1,3}	{2,4}	5	3	5	4	14.4954	0.9407	0.9350	5	3	5	4	14.4954	0.9407	0.9350
{3,4}	{1,2}	7	5	2	2	16.2224	1	0.9184	7	5	2	2	16.2224	1	0.9184
{2,4}	{1,3}	2	5	2	6	11.7906	0.9302	0.9280	2	5	2	6	11.7906	0.9302	0.9280
{1,2}	{3,4}	4	5	2	3	8.7493	0.9015	0.9030	4	5	2	3	8.7493	0.9015	0.9030
{1,3}	{2,4}	2	4	2	3	10.9339	0.9290	0.9141	2	4	2	3	10.9339	0.9290	0.9141
{1,3}	{2,4}	4	3	4	3	10.0857	0.9261	0.9615	4	3	4	3	10.0857	0.9261	0.9615

Table A.7. M/G/∞ run parameters and results - 4-item runs (cont'd)

Kit 1	**Kit 2**	**OR**	P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$	T_1	T_2	τ_1	τ_2	τ_3	τ_4	h_1	h_2	h_3	h_4
{4}	{1,2,3,4}	0.5	0.05	0.95	1	0	0	0	0.208	0.461	0.250	0.081	2.5	2	1	1.5	3	2	1	1.5	1	1.5
{2,3}	{1,3,4}	0.5	0.8	0.2	0.860	0.140	0	0	0.376	0.515	0.109	0	1.5	2.5	2	3	3	0.1	1.5	1.5	1	1.5
{1,3,4}	{2,4}	0.5	0.35	0.65	0.345	0.437	0.218	0	0.439	0.561	0	0	0.5	1.5	1	2.5	2.5	1	2	1.5	1	1
{1,3}	{1,2,4}	0.5	0.25	0.75	0.252	0.748	0	0	0.040	0.443	0.517	0	1.5	0.5	0.5	0.25	3	2	2	1.5	2	1.5
{1,2,3}	{1,4}	0.5	0.35	0.65	0.173	0.794	0.033	0	0.020	0.980	0	0	2.5	2	1.5	2	3	1.5	2	1.5	2	1
{1}	{1,2,3,4}	0.5	0.15	0.85	1	0	0	0	0.245	0.229	0.099	0.428	0.5	2.5	1	0.25	0.5	0.25	1	1	2	2
{2}	{1,2,3,4}	0.5	0.85	0.15	1	0	0	0	0.100	0.567	0.332	0.001	2.5	2.5	0.1	0.1	0.5	3	1.5	2	1,5	2
{1,3,4}	{1,2}	0.5	0.35	0.65	0.516	0.226	0.258	0	0.501	0.499	0	0	2.5	2	2	2.5	0.1	0.5	1	1.5	1	2
{3,4}	{1,2,4}	0.5	0.95	0.05	0.436	0.564	0	0	0.386	0.008	0.606	0	0.5	1.5	1	2	2.5	0.25	1.5	2	1	2

Kit 1	**Kit 2**	**Exact**							**Heuristic**						
		S_1	S_2	S_3	S_4	**Cost**	SL_1	SL_2	S_1	S_2	S_3	S_4	**Cost**	SL_1	SL_2
{4}	{1,2,3,4}	5	6	6	6	16.6748	0.9711	0.9016	5	6	6	6	16.6748	0.9711	0.9016
{2,3}	{1,3,4}	3	7	7	2	14.2644	0.9447	0.9005	3	7	7	2	14.2644	0.9447	0.9005
{1,3,4}	{2,4}	2	4	3	6	11.2567	0.9461	0.9072	2	4	3	6	11.2567	0.9461	0.9072
{1,3}	{1,2,4}	4	2	3	4	14.0842	0.9265	0.9035	4	2	3	4	14.0842	0.9265	0.9035
{1,2,3}	{1,4}	6	5	3	5	18.3404	0.9165	0.9071	6	5	3	5	18.3404	0.9165	0.9071
{1}	{1,2,3,4}	6	5	5	5	17.0572	0.9540	0.9092	6	5	5	5	17.0572	0.9540	0.9092
{2}	{1,2,3,4}	2	6	2	2	11.4262	0.9514	0.9015	2	6	2	2	11.4262	0.9514	0.9015
{1,3,4}	{1,2}	7	5	3	3	13.1073	0.9045	0.9078	7	5	3	3	13.1073	0.9045	0.9078
{3,4}	{1,2,4}	2	2	4	3	9.4104	0.9130	0.9641	2	2	4	3	9.4104	0.9130	0.9641

Table A.8. M/G/∞ run parameters and results - 4-item runs (cont'd)

Kit 1	**Kit 2**	**OR**	P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$	T_1	T_2	τ_1	τ_2	τ_3	τ_4	h_1	h_2	h_3	h_4
{2,3,4}	{1,3}	0.5	0.15	0.85	0.157	0.139	0.704	0	0.483	0.517	0	0	2	0.5	0.5	1.5	0.5	2	1.5	2	1.5	1
{1,2,3,4}	{4}	0.5	0.55	0.45	0.048	0.311	0.359	0.281	1	0	0	0	2.5	0.5	0.25	2.5	2	1	1	1.5	1	1.5
{1,2,4}	{2,3}	0.5	0.6	0.4	0.051	0.568	0.382	0	0.076	0.924	0	0	0.5	0.5	1.5	2.5	2.5	2.5	1	2	1	2
{1,3,4}	{1,2}	0.5	0.4	0.6	0.539	0.037	0.424	0	0.292	0.708	0	0	1.5	2.5	1	2.5	1	2	2	1.5	1.5	1.5
{1,2,3,4}	{2}	0.5	0.3	0.7	0.032	0.158	0.799	0.010	1	0	0	0	2	1	0.25	3	3	3	1.5	1.5	1.5	1
{1}	{1,2,3,4}	0.5	0.05	0.95	1	0	0	0	0.276	0.244	0.205	0.275	2	1.5	2	0.25	1	2.5	2	1.5	1.5	2
{1,2,4}	{1,3}	0.5	0.65	0.35	0.188	0.174	0.638	0	0.541	0.459	0	0	2	2.5	0.1	0.5	2.5	0.5	2	2	1	1
{1,2}	{1,3,4}	0.5	0.15	0.85	0.695	0.305	0	0	0.160	0.462	0.378	0	1.5	1	1.5	0.25	1.5	2	1	2	2	2

Kit 1	**Kit 2**	**Exact**							**Heuristic**						
		S_1	S_2	S_3	S_4	**Cost**	SL_1	SL_2	S_1	S_2	S_3	S_4	**Cost**	SL_1	SL_2
{2,3,4}	{1,3}	**3**	**2**	**3**	**4**	**10.0818**	**0.9031**	**0.9159**	**3**	**2**	**4**	**3**	**10.5592**	**0.9403**	**0.9630**
{1,2,3,4}	{4}	5	5	5	5	15.0929	0.9023	0.9270	5	5	5	5	15.0929	0.9023	0.9270
{1,2,4}	{2,3}	2	5	4	3	13.6332	0.9037	0.9583	2	5	4	3	13.6332	0.9037	0.9583
{1,3,4}	{1,2}	6	6	3	3	17.2734	0.9002	0.9271	6	6	3	3	17.2734	0.9002	0.9271
{1,2,3,4}	{2}	3	8	4	3	14.9401	0.9224	0.9716	3	8	4	3	14.9401	0.9224	0.9716
{1}	{1,2,3,4}	6	4	5	5	21.3893	0.9774	0.9054	6	4	5	5	21.3893	0.9774	0.9054
{1,2,4}	{1,3}	5	4	4	5	15.9756	0.9075	0.9032	5	4	4	5	15.9756	0.9075	0.9032
{1,2}	{1,3,4}	5	2	4	5	16.5479	0.9635	0.9264	5	2	4	5	16.5479	0.9635	0.9264

Table A.9. M/G/∞ run parameters and results - 4-item runs (cont'd)

Kit 1	**Kit 2**	**OR**	P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$	T_1	T_2	τ_1	τ_2	τ_3	τ_4	h_1	h_2	h_3	h_4
{1,2,3,4}	{1,4}	1	0.95	0.05	0.452	0.135	0.052	0.362	0.651	0.349	0	0	1	2.5	0.5	0.1	0.25	0.1	1.5	1	1.5	1
{1,2}	{1,2,3,4}	1	0.5	0.5	0.543	0.457	0	0	0.112	0.306	0.305	0.276	0.5	0.5	1	0.5	1	3	1.5	1	1.5	1.5
{1,3,4}	{1,2,4}	1	0.65	0.35	0.336	0.628	0.036	0	0.495	0.128	0.377	0	2	2.5	1	0.25	1	3	1.5	1	2	1
{1,2,3}	{1,2,4}	1	1	0	0.162	0.326	0.513	0	0.213	0.519	0.267	0	1	1	1.5	0.25	2	0.5	1.5	1.5	2	1
{2,4}	{1,2,3,4}	1	0.2	0.8	0.382	0.618	0	0	0.152	0.477	0.286	0.086	1	1.5	1	3	0.25	1	1.5	2	1	2
{1,2,3}	{1,2,3,4}	1.5	0.3	0.7	0.326	0.281	0.392	0	0.395	0.448	0.070	0.088	2	2	2.5	1	1	1	1.5	1	1	2
{1,2,3,4}	{1,3,4}	1.5	0.9	0.1	0.315	0.340	0.084	0.261	0.456	0.386	0.157	0	0.5	0.5	0.5	1	1	3	1.5	1	1.5	1.5
{1,2,3,4}	{1,2,3}	1.5	0.3	0.7	0.002	0.340	0.380	0.278	0.307	0.578	0.114	0	2.5	1.5	0.1	3	0.1	1.5	2	2	1.5	2
{1,2,3,4}	{1,3,4}	1.5	0.95	0.05	0.028	0.447	0.223	0.301	0.345	0.276	0.379	0	2	2	0.25	0.1	0.1	0.5	2	2	1	2

Kit 1	**Kit 2**	**Exact**							**Heuristic**						
		S_1	S_2	S_3	S_4	**Cost**	SL_1	SL_2	S_1	S_2	S_3	S_4	**Cost**	SL_1	SL_2
{1,2,3,4}	{1,4}	4	3	3	4	10.4119	0.9127	0.9542	4	3	3	4	10.4119	0.9127	0.9542
{1,2}	{1,2,3,4}	3	3	3	3	10.0697	0.9337	0.9094	3	3	3	3	10.0697	0.9337	0.9094
{1,3,4}	{1,2,4}	**6**	**3**	**5**	**7**	**17.3242**	**0.9330**	**0.9081**	**6**	**4**	**5**	**6**	**17.3299**	**0.9168**	**0.9266**
{1,2,3}	{1,2,4}	4	4	5	2	14.3317	0.9217	0.9581	4	4	5	2	14.3317	0.9217	0.9581
{2,4}	{1,2,3,4}	4	7	4	4	18.9301	0.9133	0.9024	4	7	4	4	18.9301	0.9133	0.9024
{1,2,3}	{1,2,3,4}	7	6	6	4	17.9521	0.9453	0.9121	7	6	6	4	17.9521	0.9453	0.9121
{1,2,3,4}	{1,3,4}	3	3	3	4	11.0057	0.9118	0.9345	3	3	3	4	11.0057	0.9118	0.9345
{1,2,3,4}	{1,2,3}	5	8	5	3	23.3641	0.9105	0.9511	5	8	5	3	23.3641	0.9105	0.9511
{1,2,3,4}	{1,3,4}	5	5	5	5	19.9361	0.9269	0.9282	5	5	5	5	19.9361	0.9269	0.9282

Table A.10. M/G/∞ run parameters and results - 5-item runs

Kit 1	**Kit 2**	**OR**	P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$	$p_{1,5}$	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$	T_1	T_2	τ_1	τ_2	τ_3	τ_4	τ_5	h_1	h_2	h_3	h_4	h_5
{1,2,3,4}	{5}	0	0.85	0.15	0.123	0.304	0.081	0.492	0	1	0	0	0	1	1	1	2	0.25	1	1	1	2	1	1	2
{2,3,4,5}	{1,3}	0.5	0.65	0.35	0.463	0.132	0.389	0.016	0	0.896	0.104	0	0	0.5	1.5	0.25	2	0.5	3	0.25	1.5	2	2	1.5	1.5
{4,5}	{1,2,3,4}	0.5	0.55	0.45	0.319	0.681	0	0	0	0.397	0.192	0.172	0.240	1	2	1	3	0.25	0.5	3	1.5	1	2	1	1.5
{1,4,5}	{2,3,4}	0.5	0.85	0.15	0.321	0.095	0.584	0	0	0.146	0.316	0.539	0	2	1	1	0.1	0.5	2	2.5	1.5	1.5	1	1	1.5
{2,3,4,5}	{1,3,5}	1	0.6	0.4	0.278	0.259	0.142	0.321	0	0.453	0.087	0.460	0	2.5	0.5	0.5	2.5	0.1	0.25	0.25	1	1.5	1	1	1
{1,2,3,4}	{2,4,5}	1	0.65	0.35	0.260	0.321	0.224	0.195	0	0.194	0.571	0.235	0	1	2	1	2	2	1	1	1.5	2	1.5	1.5	1.5
{1,2,3,4,5}	{2,4}	1	0.25	0.75	0.264	0.272	0.367	0.075	0.022	0.537	0.463	0	0	0.5	1.5	0.5	0.1	1.5	3	1.5	1	2	1	2	1.5
{2,3,5}	{1,2,4,5}	1	0.1	0.9	0.474	0.107	0.419	0	0	0.270	0.223	0.075	0.432	1	0.5	3	0.25	1	1	2.5	1.5	2	1	2	2
{1,2,3}	{1,3,4,5}	1	0.8	0.2	0.233	0.341	0.426	0	0	0.290	0.137	0.051	0.521	2.5	2	0.5	1	0.5	0.5	3	1.5	2	2	2	2

Kit 1	**Kit 2**	**Exact**								**Heuristic**							
		S_1	S_2	S_3	S_4	S_5	**Cost**	SL_1	SL_2	S_1	S_2	S_3	S_4	S_5	**Cost**	SL_1	SL_2
{1,2,3,4}	{5}	4	4	3	4	2	13.5698	0.9080	0.9631	4	4	3	4	2	13.5698	0.9080	0.9631
{2,3,4,5}	{1,3}	**3**	**4**	**3**	**5**	**2**	**18.3529**	**0.9040**	**0.9302**	**3**	**3**	**4**	**5**	**2**	**18.3546**	**0.9022**	**0.9696**
{4,5}	{1,2,3,4}	**3**	**5**	**3**	**6**	**4**	**16.8156**	**0.9073**	**0.9006**	**4**	**4**	**3**	**5**	**5**	**17.7500**	**0.9524**	**0.9185**
{1,4,5}	{2,3,4}	5	2	2	5	7	15.6137	0.9041	0.9207	5	2	2	5	7	15.6137	0.9041	0.9207
{2,3,4,5}	{1,3,5}	2	5	5	4	5	13.8489	0.9156	0.9334	2	5	5	4	5	13.8489	0.9156	0.9334
{1,2,3,4}	{2,4,5}	3	5	4	5	3	19.9404	0.9076	0.9176	3	5	4	5	3	19.9404	0.9076	0.9176
{1,2,3,4,5}	{2,4}	2	4	2	6	2	15.4724	0.9128	0.9404	2	4	2	6	2	15.4724	0.9128	0.9404
{2,3,5}	{1,2,4,5}	**5**	**3**	**2**	**3**	**4**	**18.1634**	**0.9050**	**0.9009**	**4**	**3**	**2**	**3**	**5**	**18.6238**	**0.9541**	**0.9312**
{1,2,3}	{1,3,4,5}	6	5	6	3	3	24.6511	0.9030	0.9167	6	5	6	3	3	24.6511	0.9030	0.9167

Table A.11. M/G/∞ run parameters and results - 5-item runs (cont'd)

Kit 1	**Kit 2**	**OR**	P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$	T_1	T_2	τ_1	τ_2	τ_3	τ_4	τ_5	h_1	h_2	h_3	h_4	h_5
{2,3,4,5}	{1,3,5}	1	0.6	0.4	0.278	0.259	0.142	0.321	0.453	0.087	0.460	0	2.5	0.5	0.5	2.5	0.1	0.25	0.25	1	1.5	1	1	1
{2,3,4,5}	{1,4,5}	1	0.15	0.85	0.021	0.461	0.332	0.186	0.212	0.159	0.628	0	1	2.5	2	2	2.5	1	0.25	1	2	1.5	1.5	1.5
{2,3,4,5}	{1,3,5}	1	0.5	0.5	0.384	0.413	0.166	0.037	0.239	0.516	0.246	0	2.5	2	3	0.1	1	1	0.5	2	1	1.5	2	1
{1,3,5}	{1,2,4,5}	1	0.85	0.15	0.449	0.336	0.214	0	0.394	0.143	0.390	0.073	1	1	0.25	0.5	1	1.5	3	2	1	1.5	2	1.5
{2,3,4,5}	{1,2,4,5}	1.5	0.45	0.55	0.334	0.239	0.172	0.255	0.188	0.259	0.365	0.187	2	2.5	1	1	3	3	2	1	2	2	1	2
{1,3,4,5}	{1,2,3,4}	1.5	0.9	0.1	0.207	0.323	0.412	0.058	0.360	0.265	0.069	0.305	1	1.5	0.5	0.1	0.5	0.25	3	2	1.5	1	1	1
{1,2,3,4}	{2,3,4,5}	1.5	0	1	0.401	0.376	0.182	0.041	0.228	0.306	0.157	0.309	2	2	3	1	1	2.5	0.1	1.5	1	1.5	1	1.5
{1,2,3,4}	{1,2,3,5}	1.5	0	1	0.339	0.187	0.465	0.009	0.154	0.348	0.182	0.316	0.5	0.5	2	0.25	0.1	0.1	2	1	2	1	1.5	1.5

Kit 1	**Kit 2**	**Exact**								**Heuristic**							
		S_1	S_2	S_3	S_4	S_5	**Cost**	SL_1	SL_2	S_1	S_2	S_3	S_4	S_5	**Cost**	SL_1	SL_2
{2,3,4,5}	{1,3,5}	2	5	5	4	5	13.8489	0.9156	0.9334	2	5	5	4	5	13.8489	0.9156	0.9334
{2,3,4,5}	{1,4,5}	6	2	2	6	6	17.3414	0.9158	0.9395	6	2	2	6	6	17.3414	0.9158	0.9395
{2,3,4,5}	{1,3,5}	4	4	6	4	6	20.5695	0.9139	0.9075	4	4	6	4	6	20.5695	0.9139	0.9075
{1,3,5}	{1,2,4,5}	4	2	4	2	4	15.2661	0.9092	0.9010	4	2	4	2	4	15.2661	0.9092	0.9010
{2,3,4,5}	{1,2,4,5}	5	6	4	8	6	26.4010	0.9054	0.9222	5	6	4	8	6	26.4010	0.9054	0.9222
{1,3,4,5}	{1,2,3,4}	4	2	4	4	4	14.2042	0.9469	0.9463	4	2	4	4	4	14.2042	0.9469	0.9463
{1,2,3,4}	{2,3,4,5}	2	6	5	6	5	16.9554	0.9083	0.9064	2	6	5	6	5	16.9554	0.9083	0.9064
{1,2,3,4}	{1,2,3,5}	3	3	3	2	4	12.6969	0.9446	0.9279	3	3	3	2	4	12.6969	0.9446	0.9279

Table A.12. M/M/1 run parameters and results - Exact (Kit 1={1,2} Kit 2={3} OR=0, $\lambda = 1$, $\rho_1 = \rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.5	0.5	0.5	0.5	1	1.5	1.5	1.5	1.5	1.5	**5**	**6**	**5**	**15.3102**	**0.9053**	**0.9340**
0.5	0.5	0.5	0.5	1	1.5	1.5	2	0.5	0.5	5	6	5	9.7233	0.9053	0.9340
0.5	0.5	0.5	0.5	1	2.5	1	1.5	1.5	1.5	6	6	5	15.8560	0.9001	0.9485
0.5	0.5	0.5	0.5	1	2.5	1	2	0.5	0.5	6	6	5	10.7771	0.9001	0.9485
0.5	0.5	0.75	0.25	1	1.5	1.5	1.5	1.5	1.5	**5**	**6**	**5**	**15.3102**	**0.9053**	**0.9340**
0.5	0.5	0.75	0.25	1	1.5	1.5	2	0.5	0.5	5	6	5	9.7233	0.9053	0.9340
0.5	0.5	0.75	0.25	1	2.5	1	1.5	1.5	1.5	6	6	5	15.8560	0.9001	0.9485
0.5	0.5	0.75	0.25	1	2.5	1	2	0.5	0.5	6	6	5	10.7771	0.9001	0.9485

Table A.13. M/M/1 run parameters and results - Heuristic (Kit 1={1,2} Kit 2={3} OR=0, $\lambda = 1$, $\rho_1 = \rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.5	0.5	0.5	0.5	1	1.5	1.5	1.5	1.5	1.5	**6**	**5**	**5**	**15.3102**	**0.9053**	**0.9340**
0.5	0.5	0.5	0.5	1	1.5	1.5	2	0.5	0.5	5	6	5	9.7233	0.9053	0.9340
0.5	0.5	0.5	0.5	1	2.5	1	1.5	1.5	1.5	6	6	5	15.8560	0.9001	0.9485
0.5	0.5	0.5	0.5	1	2.5	1	2	0.5	0.5	6	6	5	10.7771	0.9001	0.9485
0.5	0.5	0.75	0.25	1	1.5	1.5	1.5	1.5	1.5	**6**	**5**	**5**	**15.3102**	**0.9053**	**0.9340**
0.5	0.5	0.75	0.25	1	1.5	1.5	2	0.5	0.5	5	6	5	9.7233	0.9053	0.9340
0.5	0.5	0.75	0.25	1	2.5	1	1.5	1.5	1.5	6	6	5	15.8560	0.9001	0.9485
0.5	0.5	0.75	0.25	1	2.5	1	2	0.5	0.5	6	6	5	10.7771	0.9001	0.9485

Table A.14. M/M/1 run parameters and results - Exact (Kit 1={1,2} Kit 2={3} OR=0, $\lambda = 1$, $\rho_1 = \rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.8	0.2	0.5	0.5	1	1.5	1.5	1.5	1.5	1.5	6	6	4	14.7246	0.9043	0.9157
0.8	0.2	0.5	0.5	1	1.5	1.5	2	0.5	0.5	6	6	4	10.4600	0.9043	0.9157
0.8	0.2	0.5	0.5	1	2.5	1	1.5	1.5	1.5	7	7	4	15.6692	0.9005	0.9237
0.8	0.2	0.5	0.5	1	2.5	1	2	0.5	0.5	7	7	4	11.2075	0.9005	0.9237
0.8	0.2	0.75	0.25	1	1.5	1.5	1.5	1.5	1.5	6	6	4	14.7246	0.9043	0.9157
0.8	0.2	0.75	0.25	1	1.5	1.5	2	0.5	0.5	6	6	4	10.4600	0.9043	0.9157
0.8	0.2	0.75	0.25	1	2.5	1	1.5	1.5	1.5	7	7	4	15.6692	0.9005	0.9237
0.8	0.2	0.75	0.25	1	2.5	1	2	0.5	0.5	7	7	4	11.2075	0.9005	0.9237

Table A.15. M/M/1 run parameters and results - Heuristic (Kit 1={1,2} Kit 2={3} OR=0, $\lambda = 1$, $\rho_1 = \rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.8	0.2	0.5	0.5	1	1.5	1.5	1.5	1.5	1.5	6	6	4	14.7246	0.9043	0.9157
0.8	0.2	0.5	0.5	1	1.5	1.5	2	0.5	0.5	6	6	4	10.4600	0.9043	0.9157
0.8	0.2	0.5	0.5	1	2.5	1	1.5	1.5	1.5	7	7	4	15.6692	0.9005	0.9237
0.8	0.2	0.5	0.5	1	2.5	1	2	0.5	0.5	7	7	4	11.2075	0.9005	0.9237
0.8	0.2	0.75	0.25	1	1.5	1.5	1.5	1.5	1.5	6	6	4	14.7246	0.9043	0.9157
0.8	0.2	0.75	0.25	1	1.5	1.5	2	0.5	0.5	6	6	4	10.4600	0.9043	0.9157
0.8	0.2	0.75	0.25	1	2.5	1	1.5	1.5	1.5	7	7	4	15.6692	0.9005	0.9237
0.8	0.2	0.75	0.25	1	2.5	1	2	0.5	0.5	7	7	4	11.2075	0.9005	0.9237

Table A.16. M/M/1 run parameters and results - Exact (Kit 1={1,2} Kit 2={3} OR=0, $\lambda = 1$, $\rho_1 = 0.8$ $\rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.5	0.5	0.5	0.5	1	1.5	1.5	1.5	1.5	1.5	13	6	5	23.3214	0.9033	0.9340
0.5	0.5	0.5	0.5	1	1.5	1.5	2	0.5	0.5	12	7	5	19.0293	0.9022	0.9340
0.5	0.5	0.5	0.5	1	2.5	1	1.5	1.5	1.5	13	7	5	23.8133	0.9004	0.9485
0.5	0.5	0.5	0.5	1	2.5	1	2	0.5	0.5	12	9	5	19.0677	0.9001	0.9485
0.5	0.5	0.75	0.25	1	1.5	1.5	1.5	1.5	1.5	13	6	5	23.3214	0.9033	0.9340
0.5	0.5	0.75	0.25	1	1.5	1.5	2	0.5	0.5	12	7	5	19.0293	0.9022	0.9340
0.5	0.5	0.75	0.25	1	2.5	1	1.5	1.5	1.5	13	7	5	23.8133	0.9004	0.9485
0.5	0.5	0.75	0.25	1	2.5	1	2	0.5	0.5	12	9	5	19.0677	0.9001	0.9485

Table A.17. M/M/1 run parameters and results - Heuristic (Kit 1={1,2} Kit 2={3} OR=0, $\lambda = 1$, $\rho_1 = 0.8$ $\rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.5	0.5	0.5	0.5	1	1.5	1.5	1.5	1.5	1.5	13	6	5	23.3214	0.9033	0.9340
0.5	0.5	0.5	0.5	1	1.5	1.5	2	0.5	0.5	12	7	5	19.0293	0.9022	0.9340
0.5	0.5	0.5	0.5	1	2.5	1	1.5	1.5	1.5	13	7	5	23.8133	0.9004	0.9485
0.5	0.5	0.5	0.5	1	2.5	1	2	0.5	0.5	12	9	5	19.0677	0.9001	0.9485
0.5	0.5	0.75	0.25	1	1.5	1.5	1.5	1.5	1.5	13	6	5	23.3214	0.9033	0.9340
0.5	0.5	0.75	0.25	1	1.5	1.5	2	0.5	0.5	12	7	5	19.0293	0.9022	0.9340
0.5	0.5	0.75	0.25	1	2.5	1	1.5	1.5	1.5	13	7	5	23.8133	0.9004	0.9485
0.5	0.5	0.75	0.25	1	2.5	1	2	0.5	0.5	12	9	5	19.0677	0.9001	0.9485

Table A.18. M/M/1 run parameters and results - Exact (Kit 1={1,2} Kit 2={3} OR=0, $\lambda = 1$, $\rho_1 = 0.8$ $\rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.8	0.2	0.5	0.5	1	1.5	1.5	1.5	1.5	1.5	13	7	4	22.6888	0.9025	0.9157
0.8	0.2	0.5	0.5	1	1.5	1.5	2	0.5	0.5	12	9	4	18.7565	0.9016	0.9157
0.8	0.2	0.5	0.5	1	2.5	1	1.5	1.5	1.5	14	8	4	23.5366	0.9023	0.9237
0.8	0.2	0.5	0.5	1	2.5	1	2	0.5	0.5	13	10	4	19.3742	0.9032	0.9237
0.8	0.2	0.75	0.25	1	1.5	1.5	1.5	1.5	1.5	13	7	4	22.6888	0.9025	0.9157
0.8	0.2	0.75	0.25	1	1.5	1.5	2	0.5	0.5	12	9	4	18.7565	0.9016	0.9157
0.8	0.2	0.75	0.25	1	2.5	1	1.5	1.5	1.5	14	8	4	23.5366	0.9023	0.9237
0.8	0.2	0.75	0.25	1	2.5	1	2	0.5	0.5	13	10	4	19.3742	0.9032	0.9237

Table A.19. M/M/1 run parameters and results - Heuristic (Kit 1={1,2} Kit 2={3} OR=0, $\lambda = 1$, $\rho_1 = 0.8$ $\rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.8	0.2	0.5	0.5	1	1.5	1.5	1.5	1.5	1.5	13	7	4	22.6888	0.9025	0.9157
0.8	0.2	0.5	0.5	1	1.5	1.5	2	0.5	0.5	12	9	4	18.7565	0.9016	0.9157
0.8	0.2	0.5	0.5	1	2.5	1	1.5	1.5	1.5	14	8	4	23.5366	0.9023	0.9237
0.8	0.2	0.5	0.5	1	2.5	1	2	0.5	0.5	13	10	4	19.3742	0.9032	0.9237
0.8	0.2	0.75	0.25	1	1.5	1.5	1.5	1.5	1.5	13	7	4	22.6888	0.9025	0.9157
0.8	0.2	0.75	0.25	1	1.5	1.5	2	0.5	0.5	12	9	4	18.7565	0.9016	0.9157
0.8	0.2	0.75	0.25	1	2.5	1	1.5	1.5	1.5	14	8	4	23.5366	0.9023	0.9237
0.8	0.2	0.75	0.25	1	2.5	1	2	0.5	0.5	13	10	4	19.3742	0.9032	0.9237

Table A.20. M/M/1 run parameters and results - Exact (Kit 1={1,2} Kit 2={1,3} OR=0.5, $\lambda = 1$, $\rho_1 = \rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.5	0.5	0.5	0.5	0.5	0.5	1.5	1.5	1.5	1.5	1.5	7	5	5	15.8750	0.9037	0.9037
0.5	0.5	0.5	0.5	0.5	0.5	1.5	1.5	2	0.5	0.5	6	6	6	10.9632	0.9027	0.9027
0.5	0.5	0.5	0.5	0.5	0.5	2.5	1	1.5	1.5	1.5	7	6	5	16.6761	0.9083	0.9079
0.5	0.5	0.5	0.5	0.5	0.5	2.5	1	2	0.5	0.5	7	6	5	11.8706	0.9083	0.9079
0.5	0.5	0.75	0.25	0.75	0.25	1.5	1.5	1.5	1.5	1.5	7	5	5	15.8750	0.9037	0.9037
0.5	0.5	0.75	0.25	0.75	0.25	1.5	1.5	2	0.5	0.5	6	6	6	10.9632	0.9027	0.9027
0.5	0.5	0.75	0.25	0.75	0.25	2.5	1	1.5	1.5	1.5	7	6	5	16.6761	0.9083	0.9079
0.5	0.5	0.75	0.25	0.75	0.25	2.5	1	2	0.5	0.5	7	6	5	11.8706	0.9083	0.9079

Table A.21. M/M/1 run parameters and results - Heuristic (Kit 1={1,2} Kit 2={1,3} OR=0.5, $\lambda = 1$, $\rho_1 = \rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.5	0.5	0.5	0.5	0.5	0.5	1.5	1.5	1.5	1.5	1.5	7	5	5	15.8750	0.9037	0.9037
0.5	0.5	0.5	0.5	0.5	0.5	1.5	1.5	2	0.5	0.5	6	6	6	10.9632	0.9027	0.9027
0.5	0.5	0.5	0.5	0.5	0.5	2.5	1	1.5	1.5	1.5	7	6	5	16.6761	0.9083	0.9079
0.5	0.5	0.5	0.5	0.5	0.5	2.5	1	2	0.5	0.5	7	6	5	11.8706	0.9083	0.9079
0.5	0.5	0.75	0.25	0.75	0.25	1.5	1.5	1.5	1.5	1.5	7	5	5	15.8750	0.9037	0.9037
0.5	0.5	0.75	0.25	0.75	0.25	1.5	1.5	2	0.5	0.5	6	6	6	10.9632	0.9027	0.9027
0.5	0.5	0.75	0.25	0.75	0.25	2.5	1	1.5	1.5	1.5	7	6	5	16.6761	0.9083	0.9079
0.5	0.5	0.75	0.25	0.75	0.25	2.5	1	2	0.5	0.5	7	6	5	11.8706	0.9083	0.9079

Table A.22. M/M/1 run parameters and results - Exact (Kit 1={1,2} Kit 2={1,3} OR=0.5, $\lambda = 1$, $\rho_1 = \rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.8	0.2	0.5	0.5	0.5	0.5	1.5	1.5	1.5	1.5	1.5	7	6	5	17.2446	0.9187	0.9250
0.8	0.2	0.5	0.5	0.5	0.5	1.5	1.5	2	0.5	0.5	6	7	6	11.4293	0.9102	0.9122
0.8	0.2	0.5	0.5	0.5	0.5	2.5	1	1.5	1.5	1.5	8	7	5	18.2971	0.9175	0.9290
0.8	0.2	0.5	0.5	0.5	0.5	2.5	1	2	0.5	0.5	7	8	6	12.3136	0.9120	0.9155
0.8	0.2	0.75	0.25	0.75	0.25	1.5	1.5	1.5	1.5	1.5	7	6	5	17.2446	0.9187	0.9250
0.8	0.2	0.75	0.25	0.75	0.25	1.5	1.5	2	0.5	0.5	6	7	6	11.4293	0.9102	0.9122
0.8	0.2	0.75	0.25	0.75	0.25	2.5	1	1.5	1.5	1.5	8	7	5	18.2971	0.9175	0.9290
0.8	0.2	0.75	0.25	0.75	0.25	2.5	1	2	0.5	0.5	7	8	6	12.3136	0.9120	0.9155

Table A.23. M/M/1 run parameters and results - Heuristic (Kit 1={1,2} Kit 2={1,3} OR=0.5, $\lambda = 1$, $\rho_1 = \rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.8	0.2	0.5	0.5	0.5	0.5	1.5	1.5	1.5	1.5	1.5	7	6	5	17.2446	0.9187	0.9250
0.8	0.2	0.5	0.5	0.5	0.5	1.5	1.5	2	0.5	0.5	6	7	6	11.4293	0.9102	0.9122
0.8	0.2	0.5	0.5	0.5	0.5	2.5	1	1.5	1.5	1.5	8	7	5	18.2971	0.9175	0.9290
0.8	0.2	0.5	0.5	0.5	0.5	2.5	1	2	0.5	0.5	7	8	6	12.3136	0.9120	0.9155
0.8	0.2	0.75	0.25	0.75	0.25	1.5	1.5	1.5	1.5	1.5	7	6	5	17.2446	0.9187	0.9250
0.8	0.2	0.75	0.25	0.75	0.25	1.5	1.5	2	0.5	0.5	6	7	6	11.4293	0.9102	0.9122
0.8	0.2	0.75	0.25	0.75	0.25	2.5	1	1.5	1.5	1.5	8	7	5	18.2971	0.9175	0.9290
0.8	0.2	0.75	0.25	0.75	0.25	2.5	1	2	0.5	0.5	7	8	6	12.3136	0.9120	0.9155

Table A.24. M/M/1 run parameters and results - Exact (Kit 1={1,2} Kit 2={1,3} OR=0.5, $\lambda = 1$, $\rho_1 = 0.8$ $\rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.5	0.5	0.5	0.5	0.5	0.5	1.5	1.5	1.5	1.5	1.5	14	6	6	25.2077	0.9055	0.9055
0.5	0.5	0.5	0.5	0.5	0.5	1.5	1.5	2	0.5	0.5	13	7	7	20.5810	0.9051	0.9051
0.5	0.5	0.5	0.5	0.5	0.5	2.5	1	1.5	1.5	1.5	16	6	5	25.9486	0.9055	0.9074
0.5	0.5	0.5	0.5	0.5	0.5	2.5	1	2	0.5	0.5	13	8	7	20.5267	0.9028	0.9032
0.5	0.5	0.75	0.25	0.75	0.25	1.5	1.5	1.5	1.5	1.5	14	6	6	25.2077	0.9055	0.9055
0.5	0.5	0.75	0.25	0.75	0.25	1.5	1.5	2	0.5	0.5	13	7	7	20.5810	0.9051	0.9051
0.5	0.5	0.75	0.25	0.75	0.25	2.5	1	1.5	1.5	1.5	16	6	5	25.9486	0.9055	0.9074
0.5	0.5	0.75	0.25	0.75	0.25	2.5	1	2	0.5	0.5	13	8	7	20.5267	0.9028	0.9032

Table A.25. M/M/1 run parameters and results - Heuristic (Kit 1={1,2} Kit 2={1,3} OR=0.5, $\lambda = 1$, $\rho_1 = 0.8$ $\rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.5	0.5	0.5	0.5	0.5	0.5	1.5	1.5	1.5	1.5	1.5	14	6	6	25.2077	0.9055	0.9055
0.5	0.5	0.5	0.5	0.5	0.5	1.5	1.5	2	0.5	0.5	13	7	7	20.5810	0.9051	0.9051
0.5	0.5	0.5	0.5	0.5	0.5	2.5	1	1.5	1.5	1.5	16	6	5	25.9486	0.9055	0.9074
0.5	0.5	0.5	0.5	0.5	0.5	2.5	1	2	0.5	0.5	13	8	7	20.5267	0.9028	0.9032
0.5	0.5	0.75	0.25	0.75	0.25	1.5	1.5	1.5	1.5	1.5	14	6	6	25.2077	0.9055	0.9055
0.5	0.5	0.75	0.25	0.75	0.25	1.5	1.5	2	0.5	0.5	13	7	7	20.5810	0.9051	0.9051
0.5	0.5	0.75	0.25	0.75	0.25	2.5	1	1.5	1.5	1.5	16	6	5	25.9486	0.9055	0.9074
0.5	0.5	0.75	0.25	0.75	0.25	2.5	1	2	0.5	0.5	13	8	7	20.5267	0.9028	0.9032

Table A.26. M/M/1 run parameters and results - Exact (Kit 1={1,2} Kit 2={1,3} OR=0.5, $\lambda = 1$, $\rho_1 = 0.8$ $\rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.8	0.2	0.5	0.5	0.5	0.5	1.5	1.5	1.5	1.5	1.5	15	6	5	25.0997	0.9008	0.9090
0.8	0.2	0.5	0.5	0.5	0.5	1.5	1.5	2	0.5	0.5	13	8	6	20.5572	0.9084	0.9007
0.8	0.2	0.5	0.5	0.5	0.5	2.5	1	1.5	1.5	1.5	15	8	5	26.1175	0.9133	0.9033
0.8	0.2	0.5	0.5	0.5	0.5	2.5	1	2	0.5	0.5	14	9	6	21.3437	0.9112	0.9062
0.8	0.2	0.75	0.25	0.75	0.25	1.5	1.5	1.5	1.5	1.5	15	6	5	25.0997	0.9008	0.9090
0.8	0.2	0.75	0.25	0.75	0.25	1.5	1.5	2	0.5	0.5	13	8	6	20.5572	0.9084	0.9007
0.8	0.2	0.75	0.25	0.75	0.25	2.5	1	1.5	1.5	1.5	15	8	5	26.1175	0.9133	0.9033
0.8	0.2	0.75	0.25	0.75	0.25	2.5	1	2	0.5	0.5	14	9	6	21.3437	0.9112	0.9062

Table A.27. M/M/1 run parameters and results - Heuristic (Kit 1={1,2} Kit 2={1,3} OR=0.5, $\lambda = 1$, $\rho_1 = 0.8$ $\rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.8	0.2	0.5	0.5	0.5	0.5	1.5	1.5	1.5	1.5	1.5	15	6	5	25.0997	0.9008	0.9090
0.8	0.2	0.5	0.5	0.5	0.5	1.5	1.5	2	0.5	0.5	13	8	6	20.5572	0.9084	0.9007
0.8	0.2	0.5	0.5	0.5	0.5	2.5	1	1.5	1.5	1.5	15	8	5	26.1175	0.9133	0.9033
0.8	0.2	0.5	0.5	0.5	0.5	2.5	1	2	0.5	0.5	14	9	6	21.3437	0.9112	0.9062
0.8	0.2	0.75	0.25	0.75	0.25	1.5	1.5	1.5	1.5	1.5	15	6	5	25.0997	0.9008	0.9090
0.8	0.2	0.75	0.25	0.75	0.25	1.5	1.5	2	0.5	0.5	13	8	6	20.5572	0.9084	0.9007
0.8	0.2	0.75	0.25	0.75	0.25	2.5	1	1.5	1.5	1.5	15	8	5	26.1175	0.9133	0.9033
0.8	0.2	0.75	0.25	0.75	0.25	2.5	1	2	0.5	0.5	14	9	6	21.3437	0.9112	0.9062

Table A.28. M/M/1 run parameters and results - Exact (Kit 1={1,2,3} Kit 2={1,3} OR=1, $\lambda = 1$, $\rho_1 = \rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	1.5	1.5	1.5	7	6	7	19.3405	0.9056	0.9347
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	2	0.5	0.5	6	7	8	12.1345	0.9038	0.9178
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	1.5	1.5	1.5	**7**	**7**	**8**	**21.0412**	**0.9146**	**0.9374**
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	2	0.5	0.5	6	9	9	12.8982	0.9007	0.9059
0.5	0.5	0.75	0.125	0.125	0.75	0.25	1.5	1.5	1.5	1.5	1.5	7	6	7	19.3405	0.9056	0.9347
0.5	0.5	0.75	0.125	0.125	0.75	0.25	1.5	1.5	2	0.5	0.5	6	7	8	12.1345	0.9038	0.9178
0.5	0.5	0.75	0.125	0.125	0.75	0.25	2.5	1	1.5	1.5	1.5	**7**	**7**	**8**	**21.0412**	**0.9146**	**0.9374**
0.5	0.5	0.75	0.125	0.125	0.75	0.25	2.5	1	2	0.5	0.5	6	9	9	12.8982	0.9007	0.9059

Table A.29. M/M/1 run parameters and results - Heuristic (Kit 1={1,2,3} Kit 2={1,3} OR=1, $\lambda = 1$, $\rho_1 = \rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	1.5	1.5	1.5	7	6	7	19.3405	0.9056	0.9347
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	2	0.5	0.5	6	7	8	12.1345	0.9038	0.9178
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	1.5	1.5	1.5	**8**	**7**	**7**	**21.0412**	**0.9146**	**0.9374**
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	2	0.5	0.5	6	9	9	12.8982	0.9007	0.9059
0.5	0.5	0.75	0.125	0.125	0.75	0.25	1.5	1.5	1.5	1.5	1.5	7	6	7	19.3405	0.9056	0.9347
0.5	0.5	0.75	0.125	0.125	0.75	0.25	1.5	1.5	2	0.5	0.5	6	7	8	12.1345	0.9038	0.9178
0.5	0.5	0.75	0.125	0.125	0.75	0.25	2.5	1	1.5	1.5	1.5	**8**	**7**	**7**	**21.0412**	**0.9146**	**0.9374**
0.5	0.5	0.75	0.125	0.125	0.75	0.25	2.5	1	2	0.5	0.5	6	9	9	12.8982	0.9007	0.9059

Table A.30. M/M/1 run parameters and results - Exact (Kit 1={1,2,3} Kit 2={1,3} OR=1, $\lambda = 1$, $\rho_1 = \rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	1.5	1.5	1.5	7	7	7	20.2832	0.9131	0.9347
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	2	0.5	0.5	6	8	8	12.4505	0.9075	0.9178
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	1.5	1.5	1.5	8	8	8	21.7806	0.9151	0.9370
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	2	0.5	0.5	7	8	9	12.9899	0.9004	0.9207
0.8	0.2	0.75	0.125	0.125	0.75	0.25	1.5	1.5	1.5	1.5	1.5	7	7	7	20.2832	0.9131	0.9347
0.8	0.2	0.75	0.125	0.125	0.75	0.25	1.5	1.5	2	0.5	0.5	6	8	8	12.4505	0.9075	0.9178
0.8	0.2	0.75	0.125	0.125	0.75	0.25	2.5	1	1.5	1.5	1.5	8	8	8	21.7806	0.9151	0.9370
0.8	0.2	0.75	0.125	0.125	0.75	0.25	2.5	1	2	0.5	0.5	7	8	9	12.9899	0.9004	0.9207

Table A.31. M/M/1 run parameters and results - Heuristic (Kit 1={1,2,3} Kit 2={1,3} OR=1, $\lambda = 1$, $\rho_1 = \rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	1.5	1.5	1.5	7	7	7	20.2832	0.9131	0.9347
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	2	0.5	0.5	6	8	8	12.4505	0.9075	0.9178
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	1.5	1.5	1.5	8	8	8	21.7806	0.9151	0.9370
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	2	0.5	0.5	7	8	9	12.9899	0.9004	0.9207
0.8	0.2	0.75	0.125	0.125	0.75	0.25	1.5	1.5	1.5	1.5	1.5	7	7	7	20.2832	0.9131	0.9347
0.8	0.2	0.75	0.125	0.125	0.75	0.25	1.5	1.5	2	0.5	0.5	6	8	8	12.4505	0.9075	0.9178
0.8	0.2	0.75	0.125	0.125	0.75	0.25	2.5	1	1.5	1.5	1.5	8	8	8	21.7806	0.9151	0.9370
0.8	0.2	0.75	0.125	0.125	0.75	0.25	2.5	1	2	0.5	0.5	7	8	9	12.9899	0.9004	0.9207

Table A.32. M/M/1 run parameters and results - Exact (Kit 1={1,2,3} Kit 2={1,3} OR=1, $\lambda = 1$, $\rho_1 = 0.8$ $\rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	1.5	1.5	1.5	15	6	8	28.6695	0.9027	0.9326
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	2	0.5	0.5	13	8	9	21.7603	0.9049	0.9123
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	1.5	1.5	1.5	15	7	8	28.8792	0.9014	0.9250
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	2	0.5	0.5	13	9	10	22.0028	0.9041	0.9100
0.5	0.5	0.75	0.125	0.125	0.75	0.25	1.5	1.5	1.5	1.5	1.5	15	6	8	28.6695	0.9027	0.9326
0.5	0.5	0.75	0.125	0.125	0.75	0.25	1.5	1.5	2	0.5	0.5	13	8	9	21.7603	0.9049	0.9123
0.5	0.5	0.75	0.125	0.125	0.75	0.25	2.5	1	1.5	1.5	1.5	15	7	8	28.8792	0.9014	0.9250
0.5	0.5	0.75	0.125	0.125	0.75	0.25	2.5	1	2	0.5	0.5	13	9	10	22.0028	0.9041	0.9100

Table A.33. M/M/1 run parameters and results - Heuristic (Kit 1={1,2,3} Kit 2={1,3} OR=1, $\lambda = 1$, $\rho_1 = 0.8$ $\rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	1.5	1.5	1.5	15	6	8	28.6695	0.9027	0.9326
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	2	0.5	0.5	13	8	9	21.7603	0.9049	0.9123
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	1.5	1.5	1.5	15	7	8	28.8792	0.9014	0.9250
0.5	0.5	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	2	0.5	0.5	13	9	10	22.0028	0.9041	0.9100
0.5	0.5	0.75	0.125	0.125	0.75	0.25	1.5	1.5	1.5	1.5	1.5	15	6	8	28.6695	0.9027	0.9326
0.5	0.5	0.75	0.125	0.125	0.75	0.25	1.5	1.5	2	0.5	0.5	13	8	9	21.7603	0.9049	0.9123
0.5	0.5	0.75	0.125	0.125	0.75	0.25	2.5	1	1.5	1.5	1.5	15	7	8	28.8792	0.9014	0.9250
0.5	0.5	0.75	0.125	0.125	0.75	0.25	2.5	1	2	0.5	0.5	13	9	10	22.0028	0.9041	0.9100

Table A.34. M/M/1 run parameters and results - Exact (Kit 1={1,2,3} Kit 2={1,3} OR=1, $\lambda = 1$, $\rho_1 = 0.8$ $\rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	1.5	1.5	1.5	**16**	**7**	**7**	**29.5769**	**0.9042**	**0.9263**
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	2	0.5	0.5	13	8	9	21.5804	0.9010	0.9123
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	1.5	1.5	1.5	**17**	**8**	**8**	**31.0121**	**0.9060**	**0.9287**
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	2	0.5	0.5	14	9	10	22.5156	0.9041	0.9161
0.8	0.2	0.75	0.125	0.125	0.75	0.25	1.5	1.5	1.5	1.5	1.5	**16**	**7**	**7**	**29.5769**	**0.9042**	**0.9263**
0.8	0.2	0.75	0.125	0.125	0.75	0.25	1.5	1.5	2	0.5	0.5	13	8	9	21.5804	0.9010	0.9123
0.8	0.2	0.75	0.125	0.125	0.75	0.25	2.5	1	1.5	1.5	1.5	**17**	**8**	**8**	**31.0121**	**0.9060**	**0.9287**
0.8	0.2	0.75	0.125	0.125	0.75	0.25	2.5	1	2	0.5	0.5	14	9	10	22.5156	0.9041	0.9161

Table A.35. M/M/1 run parameters and results - Heuristic (Kit 1={1,2,3} Kit 2={1,3} OR=1, $\lambda = 1$, $\rho_1 = 0.8$ $\rho_2 = \rho_3 = 0.5$)

P_1	P_2	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{2,1}$	$p_{2,2}$	T_1	T_2	h_1	h_2	h_3	S_1	S_2	S_3	**Cost**	SL_1	SL_2
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	1.5	1.5	1.5	**15**	**7**	**8**	**29.6121**	**0.9097**	**0.9326**
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	1.5	1.5	2	0.5	0.5	13	8	9	21.5804	0.9010	0.9123
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	1.5	1.5	1.5	**16**	**8**	**9**	**31.0444**	**0.9110**	**0.9352**
0.8	0.2	0.3333	0.3333	0.3334	0.5	0.5	2.5	1	2	0.5	0.5	14	9	10	22.5156	0.9041	0.9161
0.8	0.2	0.75	0.125	0.125	0.75	0.25	1.5	1.5	1.5	1.5	1.5	**15**	**7**	**8**	**29.6121**	**0.9097**	**0.9326**
0.8	0.2	0.75	0.125	0.125	0.75	0.25	1.5	1.5	2	0.5	0.5	13	8	9	21.5804	0.9010	0.9123
0.8	0.2	0.75	0.125	0.125	0.75	0.25	2.5	1	1.5	1.5	1.5	**16**	**8**	**9**	**31.0444**	**0.9110**	**0.9352**
0.8	0.2	0.75	0.125	0.125	0.75	0.25	2.5	1	2	0.5	0.5	14	9	10	22.5156	0.9041	0.9161

APPENDIX B: MATLAB CODES FOR OPTIMAL STOCK LEVELS

The programs described in this appendix was written to calculate the optimal stock levels for a system where replenishment lead times for each item is independent thus supply model is a parallel processing system as in a infinite server queue (M/G/∞) and for a system with a load dependent lead times where the items are produced on separate single-server production facilities with exponentially distributed production times (M/M/1).

The input data of the system are items present in each kit, demand arrival rate (λ), kit probabilities (P_k), item probabilities in each kit ($p_{i,k}$), expected common time spent on site ($E[T]$), expected time to replenish an item ($E[\tau]$) for an M/G/∞ system or traffic intensity (ρ) for an M/M/1 system, holding cost of each item (h_i) and the desired service level. We developed two routines; one computing the optimum stock levels exactly by looking at all combinations of stock levels satisfying the service level constraint and choosing the one with minimum cost and the other computing the stock levels heuristically using the algorithm defined in Chapter 3. Both routines write optimal stock levels, optimum cost and service levels in an Excel sheet as the output of the runs.

B.1. Code for an M/G/∞ System

B.1.1. Exact Routine

```
clear;

items = [1 2 3 4];
kits = [1 2 3;1 4 0];
ServiceLevel = [0.9, 0.7];
lambda = 3;
pKit = [0.667, 0.333];
pItem = [0.4 0.3 0.3;0.3 0.7 0];
tau = [1, 1, 1, 1];
rate = zeros(size(pItem,1),size(pItem,2));
outstandingItem = zeros(1,size(items,2));
kitRate = lambda * pKit;
T = [0.5,0.2];
```

```
h = [1, 1, 1, 1];

for i=1:size(pItem,1)
    for j=1:size(pItem,2)
        rate(i,j) = kitRate(i) * pItem(i,j);
    end
end

% S_item = set of kits containing item i
S_item = zeros(size(items,2),size(kits,1));
for kitItemCount=1:size(items,2)
    kitItemSetCount=1;
    for kitCount=1:size(kits,1)
        for kitCountIndex=1:size(kits,2)
            if kits(kitCount,kitCountIndex)==items(kitItemCount)
                S_item(kitItemCount,kitItemSetCount)=kitCount;
                kitItemSetCount=kitItemSetCount+1;
            end
        end
    end
end

for item=1:size(outstandingItem,2)
    sum = 0;
    for i2 = 1:size(kits,1)
        for i3 = 1:size(kits,2)
            if (kits(i2, i3) == item)
                sum = sum + rate(i2, i3) * tau(item);
            end
        end
    end

    sum1 = 0;
    for c=1:size(S_item,2)
        if (S_item(item,c) ~= 0)
            index = 0;
            for (b=1:size(kits,2))
                if (kits(S_item(item,c), b) == item)
                    index = b;
                end
            end
            sum1 = sum1 + (kitRate(S_item(item,c)) * T(S_item(item,c)));
        end
    end
    outstandingItem(item) = sum + sum1;
end

% U_kit = Set of kits containing any one of the items in kit k
U_kit = zeros(size(kits,1),size(kits,1));
for i=1:size(kits,1)
    for j=1:size(kits,2)
        if kits(i,j)~=0
            c = (nonzeros(S_item(kits(i,j),:)))';
            for k=1:size(c,2)
                U_kit(i,c(k))=1;
            end
        end
    end
end

% M_numItemsInKit = number of items in kit k
M_numItemsInKit = zeros(size(kits,1),1);
for i=1:size(kits,1)
    count = 0;
    for j=1:size(kits,2)
        if kits(i,j)~=0
            count = count + 1;
        end
    end
    M_numItemsInKit(i) = count;
end
```

```
% V = |U|
V = zeros(size(U_kit,1),1);
for i=1:size(U_kit,1)
    count = 0;
    for j=1:size(U_kit,2)
        if U_kit(i,j)~=0
            count = count + 1;
        end
    end
    V(i) = count;
end

% U_bothKits = U(l,k), items that are contained both in kit l and kit k
U_bothKits = zeros(size(kits,1),size(kits,1),size(items,2));
for i=1:size(kits,1)
    for j=i:size(kits,1)
        c = 1;
        for k=1:size(kits,2)
            for m=1:size(kits,2)
                if (kits(i,k) == kits(j,m))
                    U_bothKits(i,j,c) = kits(i,k);
                    U_bothKits(j,i,c) = kits(i,k);
                    c  = c+1;
                end
            end
        end
    end
end



lowerStock = zeros(1,size(items,2));
tmpLowerStock = zeros(1,size(items,2));
availability = zeros(1,size(items,2));

for t1=1:size(items,2)

    tmpLowerStock(t1) = 0;

    isProbMet = 0;
    while (isProbMet == 0)
        tmpLowerStock(t1) = tmpLowerStock(t1) + 1;
        availability(t1)= poisscdf(tmpLowerStock(t1),outstandingItem(t1));
        isProbMet = 1;
        maxServiceLevel = 0;
        for t3=1:size(S_item,2)
            if (S_item(t1,t3) ~= 0)
                if (maxServiceLevel == 0)
                    maxServiceLevel = ServiceLevel(S_item(t1,t3));
                else
                    if (ServiceLevel(S_item(t1,t3)) > maxServiceLevel)
                        maxServiceLevel = ServiceLevel(S_item(t1,t3));
                    end
                end
            end
        end

        if (availability(t1) < maxServiceLevel)
            isProbMet = 0;
        end
    end
    lowerStock(t1) = tmpLowerStock(t1)+1;

end
upperStockSize = 10;
upperStock = lowerStock + upperStockSize;

display(lowerStock);
display(upperStock);
```

```
tmpItem = items;

c = 0;
optimumRow = 0;
n = combinationsDouble(lowerStock, upperStock);
minCost = -1;
optimumStock = 0;
for i=1:size(n,1)
tmpRow = n(i,:);
    kitProb = thesis(n(i,:), kits, U_kit, U_bothKits, rate, kitRate, tau,
    T, S_item);
    isProbMet = 1;
    for i2=1:size(kitProb,2)
        if (kitProb(i2) < ServiceLevel(i2))
            isProbMet = 0;
        end
    end
    if (isProbMet == 1)
        cost = 0;
        itemCosts = zeros(1,size(items,2));

S=n(i,:);
for i=1:size(S,2)
    tmpCost = 0;
    m=S(i)-1;
    for j=0:m
        if (j==0)
            tmpCost = 0;
        else
           tmpCost = tmpCost+h(i)*poisscdf(j,outstandingItem(i));
        end
    end
    cost = cost + tmpCost;
    itemCosts(i) = tmpCost;
end
       if (minCost == -1)
            minCost = cost;
            optimumStock = S;
            optimumRow = [optimumStock cost kitProb itemCosts];
        else
            if (cost < minCost)
                minCost = cost;
                optimumStock = S;
                optimumRow = [optimumStock cost kitProb itemCosts];
            end
        end
        if (c == 0)
            c = [tmpRow cost kitProb itemCosts];
        else
            c = [c ; tmpRow cost kitProb itemCosts];
        end

    end

end
l = zeros(1,size(c,2)-size(lowerStock,2));
l2 = [lowerStock l];
c = [c; l2];
u = zeros(1,size(c,2)-size(upperStock,2));
u2 = [upperStock u];
c = [c; u2];
c = [c; optimumRow];
display(optimumStock);
display(minCost);
display(c);
xlswrite('MGInfinity_Exact.xls',c);

function n = combinations(t)
    m = t(1)+1;
    for i=2:size(t,2)
        m = m * (t(i)+1);
```

```
    end
    n = zeros(m, size(t,2));
    for c=1:size(t,2)
        mm = 1;
        if (c ~= size(t,2))
            for i=c+1:size(t,2)
                mm = mm * (t(i)+1);
            end
        end
        c1 = 1;
        while c1 < m
            for nc=0:t(c)
                for c2=1:mm
                    n(c1,c) = nc;
                    c1 = c1+1;
                end
            end
        end
    end
end

function n = combinationsDouble(lower, upper)
    m = upper(1)-lower(1)+1;
    for i=2:size(lower,2)
        m = m * (upper(i)-lower(i)+1);
    end
    n = zeros(m, size(lower,2));
    for c=1:size(lower,2)
        mm = 1;
        if (c ~= size(lower,2))
            for i=c+1:size(lower,2)
                mm = mm * (upper(i)-lower(i)+1);
            end
        end
        c1 = 1;
        while c1 < m
            for nc=lower(c):upper(c)
                for c2=1:mm
                    n(c1,c) = nc;
                    c1 = c1+1;
                end
            end
        end
    end
end

function kitProb = thesis(Stock, kits, U_kit, U_bothKits, rate, kitRate, tau,
 T, S_item)
kitProb = zeros(1,size(kits,1));
for kit=1:size(kits,1)
    sum = 0;
    tmpStock = zeros(1,size(Stock,2));
    for nc=1:size(kits,2)
        item = kits(kit, nc);
        if (item ~= 0)
            tmpStock(1,item) = Stock(item) - 1;
        end
    end
    n = combinations(tmpStock);
    for i=1:size(n,1)
        tmp = calculateKitProb(n(i,:), kit, kits, U_kit, U_bothKits, rate,
        kitRate, tau, T, S_item);
        sum = sum + tmp;
    end
    kitProb(1, kit) =  sum;
end
end

function prob = calculateP(y, kit, kits, U_kit, rate, kitRate, tau,
 T, n, S_item)
    multiply = 1;
```

```
    for j=1:size(kits,2)
        i = kits(kit,j);
        if (i ~= 0)
            sum1 = 0;
            sum2 = 0;
            for c=1:size(S_item,2)
                if (S_item(i,c) ~= 0)
                    index = 0;
                    for (b=1:size(kits,2))
                        if (kits(S_item(i,c), b) == i)
                            index = b;
                        end
                    end
                    sum1 = sum1 + (rate(S_item(i,c), index));
                    sum2 = sum2 + y(S_item(i,c));
                end
            end
            sum2 = n(i) - sum2;
            if (sum2<0)
                value = 0;
            else
                value = ( exp(-tau(i) * sum1) * (tau(i) * sum1)^sum2 )
                /  factorial(sum2);
            end
            multiply = multiply * value;
        end
    end
    for j=1:size(U_kit,2)
        if (U_kit(kit,j) ~= 0)
            value = ( exp(-kitRate(j) * T(j)) * ((kitRate(j) * T(j))^y(j) ) )
            / factorial(y(j));
            multiply = multiply * value;
        end
    end
    prob = multiply;
end

function kitProbability = calculateKitProb (n, kit, kits, U_kit, U_bothKits,
 rate, kitRate, tau, T, S_item)
    R = zeros(1,size(kits,1));
    for l=1:size(kits,1)
        if (U_kit(kit,l) == 1)
            minN = -1;
            for i=1:size(U_bothKits,3)
                tt = U_bothKits(l,kit, i);
                if (tt ~= 0)
                    if (minN == -1)
                        minN = n(1,tt);
                    else
                        if (n(1,tt) < minN)
                            minN = n(1,tt);
                        end
                    end
                end
            end
            if minN < 0
                minN = 0;
            end
            R(1,l) = minN;
        end
    end
    sum = 0;
    y = combinations(R);
    for t=1:size(y,1)
        tmp = calculateP(y(t,:), kit, kits, U_kit, rate, kitRate, tau, T,
        n, S_item);
        sum = sum + tmp;
    end
    kitProbability = sum;
    end
```

B.1.2. Heuristic Routine

```
clear;

items = [1 2 3 4];
kits = [1 2 3;1 4 0];
ServiceLevel = [0.9, 0.7];
lambda = 3;
pKit = [0.667, 0.333];
pItem = [0.4 0.3 0.3;0.3 0.7 0];
tau = [1, 1, 1, 1];
rate = zeros(size(pItem,1),size(pItem,2));
outstandingItem = zeros(1,size(items,2));
kitRate = lambda * pKit;
T = [0.5,0.2];
h = [1, 1, 1, 1];

for i=1:size(pItem,1)
    for j=1:size(pItem,2)
        rate(i,j) = kitRate(i) * pItem(i,j);
    end
end

% S_item = set of kits containing item i
S_item = zeros(size(items,2),size(kits,1));
for kitItemCount=1:size(items,2)
    kitItemSetCount=1;
    for kitCount=1:size(kits,1)
        for kitCountIndex=1:size(kits,2)
            if kits(kitCount,kitCountIndex)==items(kitItemCount)
                S_item(kitItemCount,kitItemSetCount)=kitCount;
                kitItemSetCount=kitItemSetCount+1;
            end
        end
    end
end

for item=1:size(outstandingItem,2)
    sum = 0;
    for i2 = 1:size(kits,1)
        for i3 = 1:size(kits,2)
            if (kits(i2, i3) == item)
                sum = sum + rate(i2, i3) * tau(item);
            end
        end
    end

    sum1 = 0;
    for c=1:size(S_item,2)
        if (S_item(item,c) ~= 0)
            index = 0;
            for (b=1:size(kits,2))
                if (kits(S_item(item,c), b) == item)
                    index = b;
                end
            end
            sum1 = sum1 + (kitRate(S_item(item,c)) * T(S_item(item,c)));
        end
    end
    outstandingItem(item) = sum + sum1;
end

% U_kit = Set of kits containing any one of the items in kit k
U_kit = zeros(size(kits,1),size(kits,1));
for i=1:size(kits,1)
    for j=1:size(kits,2)
        if kits(i,j)~=0
            c = (nonzeros(S_item(kits(i,j),:)))';
            for k=1:size(c,2)
                U_kit(i,c(k))=1;
```

```
            end
        end
    end
end

% M_numItemsInKit = number of items in kit k
M_numItemsInKit = zeros(size(kits,1),1);
for i=1:size(kits,1)
    count = 0;
    for j=1:size(kits,2)
        if kits(i,j)~=0
            count = count + 1;
        end
    end
    M_numItemsInKit(i) = count;
end

% V = |U|
V = zeros(size(U_kit,1),1);
for i=1:size(U_kit,1)
    count = 0;
    for j=1:size(U_kit,2)
        if U_kit(i,j)~=0
            count = count + 1;
        end
    end
    V(i) = count;
end

% U_bothKits = U(l,k), items that are contained both in kit l and kit k
U_bothKits = zeros(size(kits,1),size(kits,1),size(items,2));
for i=1:size(kits,1)
    for j=i:size(kits,1)
        c = 1;
        for k=1:size(kits,2)
            for m=1:size(kits,2)
                if (kits(i,k) == kits(j,m))
                    U_bothKits(i,j,c) = kits(i,k);
                    U_bothKits(j,i,c) = kits(i,k);
                    c  = c+1;
                end
            end
        end
    end
end

lowerStock = zeros(1,size(items,2));
tmpLowerStock = zeros(1,size(items,2));
availability = zeros(1,size(items,2));

for t1=1:size(items,2)

    tmpLowerStock(t1) = 0;

    isProbMet = 0;
    while (isProbMet == 0)
        tmpLowerStock(t1) = tmpLowerStock(t1) + 1;
        availability(t1)= poisscdf(tmpLowerStock(t1),outstandingItem(t1));
        isProbMet = 1;
        maxServiceLevel = 0;
        for t3=1:size(S_item,2)
            if (S_item(t1,t3) ~= 0)
                if (maxServiceLevel == 0)
                    maxServiceLevel = ServiceLevel(S_item(t1,t3));
                else
                    if (ServiceLevel(S_item(t1,t3)) > maxServiceLevel)
                        maxServiceLevel = ServiceLevel(S_item(t1,t3));
                    end
                end
            end
        end
```

```
        %display(maxServiceLevel);
        %display(availability(t1));
        if (availability(t1) < maxServiceLevel)
            isProbMet = 0;
        end
    end
    lowerStock(t1) = tmpLowerStock(t1)+1;

end

tmpItem = items;
c = 0;

optimumFound = 0;
currentStock = lowerStock;
currentCost = 0;
currentKitAvailability = 0;
kitProb = thesis(lowerStock, kits, U_kit, U_bothKits, rate, kitRate,
tau, T, S_item);
for t = 1:size(kitProb,2)
    currentKitAvailability = currentKitAvailability + kitProb(t);
end
S=lowerStock;
cost = 0;
for i=1:size(S,2)
    tmpCost = 0;
    m=S(i)-1;
    for j=0:m
        if (j==0)
            tmpCost = 0;
        else
           tmpCost = tmpCost+h(i)*poisscdf(j,outstandingItem(i));
        end
    end
    cost = cost + tmpCost;
%    itemCosts(i) = tmpCost;
end
currentCost = cost;
validStock = ones(1,size(lowerStock,2));
optimumCost = 0;
optimumRow = 0;

lowerProbMet = 1;
for i4=1:size(kitProb,2)
    if (kitProb(1,i4) > ServiceLevel(i4))
        for i5=1:size(S_item,1)
            if (S_item(i5,1) == i4)
                if (size(S_item,2) == 1)
                    validStock(1,i5) = 0;
                else
                    if (S_item(i5,2) == 0)
                        validStock(1,i5) = 0;
                    end
                end
            end
        end
    end
    if (kitProb(1,i4) < ServiceLevel(i4))
       lowerProbMet = 0;
    end
end

if (lowerProbMet == 1)
optimumFound = 1;
end

optimumRow = [lowerStock cost kitProb [0 0 0]];

if (c == 0)
    c = [lowerStock cost kitProb [0 0 0]];
else
```

```
    c = [c ; lowerStock cost kitProb [0 0 0]];
end
c = [c ; zeros(1,size(c,2))];

while (optimumFound == 0)
    n = combinationsIncremental(currentStock, validStock);
    costs = zeros(size(n,1),1);
    kitProbs = 0;
    kitProbSums = zeros(size(n,1),1);
    incCosts = zeros(size(n,1),1);
    incAvail = zeros(size(n,1),1);
    ratio = zeros(size(n,1),1);
    for i=1:size(n,1)
        kitProb = thesis(n(i,:), kits, U_kit, U_bothKits, rate, kitRate,
        tau, T, S_item);
        S=n(i,:);
        cost = 0;
        for i2=1:size(S,2)
            tmpCost = 0;
            m=S(i2)-1;
            for j=0:m
                if (j==0)
                    tmpCost = 0;
                else
                   tmpCost = tmpCost+h(i2)*poisscdf(j,outstandingItem(i2));
                end
            end
            cost = cost + tmpCost;
        end
        costs(i) = cost;
        for t = 1:size(kitProb,2)
            kitProbSums(i) = kitProbSums(i) + kitProb(1,t);
        end
        if (kitProbs == 0)
            kitProbs = kitProb;
        else
            kitProbs = [kitProbs ; kitProb];
        end
    end
    tmpMinCost = 0;
    tmpIndex = zeros(size(n,1),1);
    isAllProbMet = 1;
    for i2=1:size(kitProbs,1)
        rowProbMet = 1;
        for i3=1:size(kitProbs,2)
            if (kitProbs(i2,i3) < ServiceLevel(i3))
                isAllProbMet = 0;
                rowProbMet = 0;
            end
        end
        if (rowProbMet == 1)
            if (optimumCost == 0)
                optimumCost = costs(i2,1);
                optimumRow = [n(i2,:) costs(i2,1) kitProbs(i2,:) [0 0 0]];
            else
                if (costs(i2, 1) < optimumCost)
                    optimumCost = costs(i2,1);
                    optimumRow = [n(i2,:) costs(i2,1) kitProbs(i2,:) [0 0 0]];
                end
            end
        else
            if (optimumCost == 0)
                tmpIndex(i2) = 1;
            else
                if (costs(i2,1) < optimumCost)
                    tmpIndex(i2) = 1;
                end
            end
        end

    end
```

```
    if (isAllProbMet == 1)
        optimumFound = 1;
    else
        incCosts = costs - currentCost;
        incAvail = kitProbSums - currentKitAvailability;
        ratio = incCosts ./ incAvail;
        minRatio = 0;
        minRatioIndex = -1;
        for t1 = 1:size(ratio,1)
            if (tmpIndex(t1) == 1)
                if (minRatio == 0)
                    minRatio = ratio(t1);
                     minRatioIndex = t1;
                else
                    if (ratio(t1) < minRatio)
                        minRatio = ratio(t1);
                        minRatioIndex = t1;
                    end
                end
            end

        end
        if (minRatioIndex == -1)
            optimumFound = 1;
        else
            currentStock = n(minRatioIndex,:);
            currentCost = costs(minRatioIndex);
            currentKitAvailability = 0;
            for t = 1:size(kitProbs,2)
                currentKitAvailability = currentKitAvailability +
                kitProbs(minRatioIndex,t);
            end
            for i4=1:size(kitProbs,2)
                if (kitProbs(minRatioIndex,i4) > ServiceLevel(i4))
                    for i5=1:size(S_item,1)
                        if (S_item(i5,1) == i4)
                            if (size(S_item,2) == 1)
                                validStock(1,i5) = 0;
                            else
                                if (S_item(i5,2) == 0)
                                    validStock(1,i5) = 0;
                                end
                            end
                        end
                    end
                end
            end

        end
    end
    if (c == 0)
        c = [n costs kitProbs incCosts incAvail ratio];
    else
        c = [c ; n costs kitProbs incCosts incAvail ratio];
    end
        c = [c ; zeros(1,size(c,2))];

end
    c = [c ; optimumRow];
display(c);
xlswrite('MGInfinity_heuristic.xls',c);

function n = combinations(t)
    m = t(1)+1;
    for i=2:size(t,2)
        m = m * (t(i)+1);
    end
    n = zeros(m, size(t,2));
    for c=1:size(t,2)
        mm = 1;
```

```
        if (c ~= size(t,2))
            for i=c+1:size(t,2)
                mm = mm * (t(i)+1);
            end
        end
        c1 = 1;
        while c1 < m
            for nc=0:t(c)
                for c2=1:mm
                    n(c1,c) = nc;
                    c1 = c1+1;
                end
            end
        end
    end
end

function n = combinationsDouble(lower, upper)
    m = upper(1)-lower(1)+1;
    for i=2:size(lower,2)
        m = m * (upper(i)-lower(i)+1);
    end
    n = zeros(m, size(lower,2));
    for c=1:size(lower,2)
        mm = 1;
        if (c ~= size(lower,2))
            for i=c+1:size(lower,2)
                mm = mm * (upper(i)-lower(i)+1);
            end
        end
        c1 = 1;
        while c1 < m
            for nc=lower(c):upper(c)
                for c2=1:mm
                    n(c1,c) = nc;
                    c1 = c1+1;
                end
            end
        end
    end
end

function kitProb = thesis(Stock, kits, U_kit, U_bothKits, rate, kitRate, tau,
 T, S_item)
kitProb = zeros(1,size(kits,1));
for kit=1:size(kits,1)
    sum = 0;
    tmpStock = zeros(1,size(Stock,2));
    for nc=1:size(kits,2)
        item = kits(kit, nc);
        if (item ~= 0)
            tmpStock(1,item) = Stock(item) - 1;
        end
    end
    n = combinations(tmpStock);
    for i=1:size(n,1)
        tmp = calculateKitProb(n(i,:), kit, kits, U_kit, U_bothKits, rate,
        kitRate, tau, T, S_item);
        sum = sum + tmp;
    end
    kitProb(1, kit) =  sum;
end
end

function prob = calculateP(y, kit, kits, U_kit, rate, kitRate, tau,
 T, n, S_item)
    multiply = 1;
    for j=1:size(kits,2)
        i = kits(kit,j);
        if (i ~= 0)
            sum1 = 0;
```

```
            sum2 = 0;
            for c=1:size(S_item,2)
                if (S_item(i,c) ~= 0)
                    index = 0;
                    for (b=1:size(kits,2))
                        if (kits(S_item(i,c), b) == i)
                            index = b;
                        end
                    end
                    sum1 = sum1 + (rate(S_item(i,c), index));
                    sum2 = sum2 + y(S_item(i,c));
                end
            end
            sum2 = n(i) - sum2;
            if (sum2<0)
                value = 0;
            else
                value = ( exp(-tau(i) * sum1) * (tau(i) * sum1)^sum2 )
                /  factorial(sum2);
            end
            multiply = multiply * value;
        end
    end
    for j=1:size(U_kit,2)
        if (U_kit(kit,j) ~= 0)
            value = ( exp(-kitRate(j) * T(j)) * ((kitRate(j) * T(j))^y(j) ) )
            / factorial(y(j));
            multiply = multiply * value;
        end
    end
    prob = multiply;
end

function kitProbability = calculateKitProb (n, kit, kits, U_kit, U_bothKits,
 rate, kitRate, tau, T, S_item)
    R = zeros(1,size(kits,1));
    for l=1:size(kits,1)
        if (U_kit(kit,l) == 1)
            minN = -1;
            for i=1:size(U_bothKits,3)
                tt = U_bothKits(l,kit, i);
                if (tt ~= 0)
                    if (minN == -1)
                        minN = n(1,tt);
                    else
                        if (n(1,tt) < minN)
                            minN = n(1,tt);
                        end
                    end
                end
            end
            if minN < 0
                minN = 0;
            end
            R(1,l) = minN;
        end
    end
    sum = 0;
    y = combinations(R);
    for t=1:size(y,1)
        tmp = calculateP(y(t,:), kit, kits, U_kit, rate, kitRate, tau, T,
        n, S_item);
        sum = sum + tmp;
    end
    kitProbability = sum;
    end
```

B.2. Code for an M/M/1 System

B.2.1. Exact Routine

```
clear;

items = [1 2 3];
kits = [1 2;3 0];
ServiceLevel = [0.9, 0.9];
lambda = 1;
pKit = [0.5, 0.5];
pItem = [0.5 0.5;1 0];
ro=[0.5 0.5 0.5];
rate = zeros(size(pItem,1),size(pItem,2));
outstandingItem = zeros(1,size(items,2));
itemOnSite = zeros(1,size(items,2));
kitRate = lambda * pKit;
T = [1.5,1.5];
h = [1.5 1.5 1.5];

for i=1:size(pItem,1)
    for j=1:size(pItem,2)
        rate(i,j) = kitRate(i) * pItem(i,j);
    end
end

% S_item = set of kits containing item i
S_item = zeros(size(items,2),size(kits,1));
for kitItemCount=1:size(items,2)
    kitItemSetCount=1;
    for kitCount=1:size(kits,1)
        for kitCountIndex=1:size(kits,2)
            if kits(kitCount,kitCountIndex)==items(kitItemCount)
                S_item(kitItemCount,kitItemSetCount)=kitCount;
                kitItemSetCount=kitItemSetCount+1;
            end
        end
    end
end

itemLambda = 0;
for i=1:size(items,2)
    sum1 = 0;
    for c=1:size(S_item,2)
        if (S_item(i,c) ~= 0)
            index = 0;
            for (b=1:size(kits,2))
                if (kits(S_item(i,c), b) == i)
                    index = b;
                end
            end
            sum1 = sum1 + (rate(S_item(i,c), index));
        end
    end
    if (itemLambda == 0)
        itemLambda = sum1;
    else
        itemLambda = [itemLambda sum1];
    end

end
tau = ro./itemLambda;
display(tau);

for item=1:size(outstandingItem,2)
    sum = 0;
    for i2 = 1:size(kits,1)
        for i3 = 1:size(kits,2)
```

```
            if (kits(i2, i3) == item)
                sum = sum + rate(i2, i3) * tau(item);
            end
        end
    end

    sum1 = 0;
    for c=1:size(S_item,2)
        if (S_item(item,c) ~= 0)
            index = 0;
            for (b=1:size(kits,2))
                if (kits(S_item(item,c), b) == item)
                    index = b;
                end
            end
            sum1 = sum1 + (kitRate(S_item(item,c)) * T(S_item(item,c)));
        end
    end
    outstandingItem(item) = sum + sum1;
end

for item=1:size(itemOnSite,2)
    sum1 = 0;
    for c=1:size(S_item,2)
        if (S_item(item,c) ~= 0)
            index = 0;
            for (b=1:size(kits,2))
                if (kits(S_item(item,c), b) == item)
                    index = b;
                end
            end
            sum1 = sum1 + (kitRate(S_item(item,c)) * T(S_item(item,c)));
        end
    end
    itemOnSite(item) = sum1;
end

% U_kit = Set of kits containing any one of the items in kit k
U_kit = zeros(size(kits,1),size(kits,1));
for i=1:size(kits,1)
    for j=1:size(kits,2)
        if kits(i,j)~=0
            c = (nonzeros(S_item(kits(i,j),:)))';
            for k=1:size(c,2)
                U_kit(i,c(k))=1;
            end
        end
    end
end

% M_numItemsInKit = number of items in kit k
M_numItemsInKit = zeros(size(kits,1),1);
for i=1:size(kits,1)
    count = 0;
    for j=1:size(kits,2)
        if kits(i,j)~=0
            count = count + 1;
        end
    end
    M_numItemsInKit(i) = count;
end

% V = |U|
V = zeros(size(U_kit,1),1);
for i=1:size(U_kit,1)
    count = 0;
    for j=1:size(U_kit,2)
        if U_kit(i,j)~=0
            count = count + 1;
        end
    end
```

```
    V(i) = count;
end

% U_bothKits = U(l,k), items that are contained both in kit l and kit k
U_bothKits = zeros(size(kits,1),size(kits,1),size(items,2));
for i=1:size(kits,1)
    for j=i:size(kits,1)
        c = 1;
        for k=1:size(kits,2)
            for m=1:size(kits,2)
                if (kits(i,k) == kits(j,m))
                    U_bothKits(i,j,c) = kits(i,k);
                    U_bothKits(j,i,c) = kits(i,k);
                    c  = c+1;
                end
            end
        end
    end
end

lowerStock = zeros(1,size(items,2));
tmpLowerStock = zeros(1,size(items,2));
availability = zeros(1,size(items,2));

for t1=1:size(items,2)

    tmpLowerStock(t1) = 0;

    isProbMet = 0;
    while (isProbMet == 0)
        tmpLowerStock(t1) = tmpLowerStock(t1) + 1;
        availability(t1) = 0;
        for k=0:tmpLowerStock(t1)
            availability(t1)= availability(t1)+poisspdf(k,itemOnSite(t1))*
            (1 -(ro(t1)^(tmpLowerStock(t1)-k+1)));
        end
        isProbMet = 1;
        maxServiceLevel = 0;
        for t3=1:size(S_item,2)
            if (S_item(t1,t3) ~= 0)
                if (maxServiceLevel == 0)
                    maxServiceLevel = ServiceLevel(S_item(t1,t3));
                else
                    if (ServiceLevel(S_item(t1,t3)) > maxServiceLevel)
                        maxServiceLevel = ServiceLevel(S_item(t1,t3));
                    end
                end
            end
        end
        %display(maxServiceLevel);
        %display(availability(t1));
        if (availability(t1) < maxServiceLevel)
            isProbMet = 0;
        end
    end
    lowerStock(t1) = tmpLowerStock(t1)+1;

end

upperStockSize = 10;
upperStock = lowerStock + upperStockSize;

display(lowerStock);
%display(upperStock);

tmpItem = items;

c = 0;
optimumRow = 0;
n = combinationsDouble(lowerStock, upperStock);
minCost = -1;
```

```
optimumStock = 0;
for i=1:size(n,1)
    display(i);
    display(size(n,1));
    tmpRow = n(i,:);
    kitProb = thesis(n(i,:), kits, U_kit, U_bothKits, rate, kitRate, ro,
    T, S_item);
    isProbMet = 1;
    for i2=1:size(kitProb,2)
        if (kitProb(i2) < ServiceLevel(i2))
            isProbMet = 0;
        end
    end
    if (isProbMet == 1)
        cost = 0;
        itemCosts = zeros(1,size(items,2));

S=n(i,:);
for i=1:size(S,2)
    tmpCost = 0;
    m=S(i)-1;
    for j=0:m
        if (j==0)
            tmpCost = 0;
        else
            for k=0:j
                tmpCost = tmpCost+h(i)*poisspdf(k,itemOnSite(i))*(1 - (ro(i)^(j-k+1)));
            end
        end
    end
    cost = cost + tmpCost;
    itemCosts(i) = tmpCost;
end
        if (minCost == -1)
            minCost = cost;
            optimumStock = S;
            optimumRow = [optimumStock cost kitProb itemCosts];
        else
            if (cost < minCost)
                minCost = cost;
                optimumStock = S;
                optimumRow = [optimumStock cost kitProb itemCosts];
            end
        end
        if (c == 0)
            c = [tmpRow cost kitProb itemCosts];
        else
            c = [c ; tmpRow cost kitProb itemCosts];
        end

    end

end
l = zeros(1,size(c,2)-size(lowerStock,2));
l2 = [lowerStock l];
c = [c; l2];
u = zeros(1,size(c,2)-size(upperStock,2));
u2 = [upperStock u];
c = [c; u2];
c = [c; optimumRow];
display(optimumStock);
display(minCost);
display(c);
xlswrite('mm1exact.xls',c);

function n = combinations(t)
    m = t(1)+1;
    for i=2:size(t,2)
        m = m * (t(i)+1);
    end
    n = zeros(m, size(t,2));
```

```
    for c=1:size(t,2)
        mm = 1;
        if (c ~= size(t,2))
            for i=c+1:size(t,2)
                mm = mm * (t(i)+1);
            end
        end
        c1 = 1;
        while c1 < m
            for nc=0:t(c)
                for c2=1:mm
                    n(c1,c) = nc;
                    c1 = c1+1;
                end
            end
        end
    end
end

function n = combinationsIncremental(current, valid)
    n = zeros(size(current,2), size(current,2));
    for i=1:size(current,2)
        for j=1:size(current,2)
            if (i==j)
                n(i,j) = current(j)+1;
            else
                n(i,j) = current(j);
            end
        end
    end
    tmp = 0;
    for i=1:size(valid,2)
        if (valid(1,i) == 1)
            if (tmp == 0)
                tmp = n(i,:);
            else
                tmp = [tmp; n(i,:)];
            end
        end
    end
    n = tmp;
end

function kitProb = thesis(Stock, kits, U_kit, U_bothKits, rate, kitRate, ro,
 T, S_item)
kitProb = zeros(1,size(kits,1));
for kit=1:size(kits,1)
    sum = 0;
    tmpStock = zeros(1,size(Stock,2));
    for nc=1:size(kits,2)
        item = kits(kit, nc);
        if (item ~= 0)
            tmpStock(1,item) = Stock(item) - 1;
        end
    end
    n = combinations(tmpStock);
    for i=1:size(n,1)
        tmp = calculateKitProb(n(i,:), kit, kits, U_kit, U_bothKits, rate,
        kitRate, ro, T, S_item);
        sum = sum + tmp;
    end
    kitProb(1, kit) =  sum;
end
end

function prob = calculateP(y, kit, kits, U_kit, rate, kitRate, ro, T, n, S_item)
    multiply = 1;
    for j=1:size(kits,2)
        i = kits(kit,j);
        if (i ~= 0)
            sum2 = 0;
```

```
            for c=1:size(S_item,2)
                if (S_item(i,c) ~= 0)
                    sum2 = sum2 + y(S_item(i,c));
                end
            end
            sum2 = n(i) - sum2;
            if (sum2<0)
                value = 0;
            else
                value = (ro(i)^sum2) * (1 - ro(i)) ;
            end
            multiply = multiply * value;
        end
    end
    for j=1:size(U_kit,2)
        if (U_kit(kit,j) ~= 0)
            value = ( exp(-kitRate(j) * T(j)) * ((kitRate(j) * T(j))^y(j) ) )
             / factorial(y(j));
            multiply = multiply * value;
        end
    end
    prob = multiply;
end

function kitProbability = calculateKitProb (n, kit, kits, U_kit, U_bothKits, rate,
 kitRate, ro, T, S_item)
    R = zeros(1,size(kits,1));
    for l=1:size(kits,1)
        if (U_kit(kit,l) == 1)
            minN = -1;
            for i=1:size(U_bothKits,3)
                tt = U_bothKits(l,kit, i);
                if (tt ~= 0)
                    if (minN == -1)
                        minN = n(1,tt);
                    else
                        if (n(1,tt) < minN)
                            minN = n(1,tt);
                        end
                    end
                end
            end
            if minN < 0
                minN = 0;
            end
            R(1,l) = minN;
        end
    end
    sum = 0;
    y = combinations(R);
    for t=1:size(y,1)
        tmp = calculateP(y(t,:), kit, kits, U_kit, rate, kitRate, ro, T,
        n, S_item);
        sum = sum + tmp;
    end
    kitProbability = sum;
    end
```

B.2.2. Heuristic Routine

```
clear;

items = [1 2 3];
kits = [1 2;3 0];
ServiceLevel = [0.9, 0.9];
lambda = 1;
pKit = [0.5, 0.5];
pItem = [0.5 0.5;1 0];
```

```
%tau = [0.5, 0.6, 0.6, 0.5];
ro=[0.5 0.5 0.5];
rate = zeros(size(pItem,1),size(pItem,2));
outstandingItem = zeros(1,size(items,2));
itemOnSite = zeros(1,size(items,2));
kitRate = lambda * pKit;
T = [1.5,1.5];
h = [1.5, 1.5, 1.5];

for i=1:size(pItem,1)
    for j=1:size(pItem,2)
        rate(i,j) = kitRate(i) * pItem(i,j);
    end
end

% S_item = set of kits containing item i
S_item = zeros(size(items,2),size(kits,1));
for kitItemCount=1:size(items,2)
    kitItemSetCount=1;
    for kitCount=1:size(kits,1)
        for kitCountIndex=1:size(kits,2)
            if kits(kitCount,kitCountIndex)==items(kitItemCount)
                S_item(kitItemCount,kitItemSetCount)=kitCount;
                kitItemSetCount=kitItemSetCount+1;
            end
        end
    end
end

itemLambda = 0;
for i=1:size(items,2)
    sum1 = 0;
    for c=1:size(S_item,2)
        if (S_item(i,c) ~= 0)
            index = 0;
            for (b=1:size(kits,2))
                if (kits(S_item(i,c), b) == i)
                    index = b;
                end
            end
            sum1 = sum1 + (rate(S_item(i,c), index));
        end
    end
    if (itemLambda == 0)
        itemLambda = sum1;
    else
        itemLambda = [itemLambda sum1];
    end

end
tau = ro./itemLambda;
display(tau);

for item=1:size(outstandingItem,2)
    sum = 0;
    for i2 = 1:size(kits,1)
        for i3 = 1:size(kits,2)
            if (kits(i2, i3) == item)
                sum = sum + rate(i2, i3) * tau(item);
            end
        end
    end

    sum1 = 0;
    for c=1:size(S_item,2)
        if (S_item(item,c) ~= 0)
            index = 0;
            for (b=1:size(kits,2))
                if (kits(S_item(item,c), b) == item)
                    index = b;
                end
```

```
            end
            sum1 = sum1 + (kitRate(S_item(item,c)) * T(S_item(item,c)) );
        end
    end
    outstandingItem(item) = sum + sum1;
end

for item=1:size(itemOnSite,2)

    sum1 = 0;
    for c=1:size(S_item,2)
        if (S_item(item,c) ~= 0)
            index = 0;
            for (b=1:size(kits,2))
                if (kits(S_item(item,c), b) == item)
                    index = b;
                end
            end
            sum1 = sum1 + (kitRate(S_item(item,c)) * T(S_item(item,c)));
        end
    end
    itemOnSite(item) = sum1;
end

% U_kit = Set of kits containing any one of the items in kit k
U_kit = zeros(size(kits,1),size(kits,1));
for i=1:size(kits,1)
    for j=1:size(kits,2)
        if kits(i,j)~=0
            c = (nonzeros(S_item(kits(i,j),:)))';
            for k=1:size(c,2)
                U_kit(i,c(k))=1;
            end
        end
    end
end

% M_numItemsInKit = number of items in kit k
M_numItemsInKit = zeros(size(kits,1),1);
for i=1:size(kits,1)
    count = 0;
    for j=1:size(kits,2)
        if kits(i,j)~=0
            count = count + 1;
        end
    end
    M_numItemsInKit(i) = count;
end

% V = |U|
V = zeros(size(U_kit,1),1);
for i=1:size(U_kit,1)
    count = 0;
    for j=1:size(U_kit,2)
        if U_kit(i,j)~=0
            count = count + 1;
        end
    end
    V(i) = count;
end

% U_bothKits = U(l,k), items that are contained both in kit l and kit k
U_bothKits = zeros(size(kits,1),size(kits,1),size(items,2));
for i=1:size(kits,1)
    for j=i:size(kits,1)
        c = 1;
        for k=1:size(kits,2)
            for m=1:size(kits,2)
                if (kits(i,k) == kits(j,m))
                    U_bothKits(i,j,c) = kits(i,k);
                    U_bothKits(j,i,c) = kits(i,k);
```

```
                    c  = c+1;
                end
            end
        end
    end
end

lowerStock = zeros(1,size(items,2));
tmpLowerStock = zeros(1,size(items,2));
availability = zeros(1,size(items,2));

for t1=1:size(items,2)
    tmpLowerStock(t1) = 0;
    isProbMet = 0;
    while (isProbMet == 0)
        tmpLowerStock(t1) = tmpLowerStock(t1) + 1;
        availability(t1) = 0;
        for k=0:tmpLowerStock(t1)
            availability(t1)= availability(t1)+poisspdf(k,itemOnSite(t1))
            *(1 - (ro(t1)^(tmpLowerStock(t1)-k+1)));
        end
        %display(tmpLowerStock);
        isProbMet = 1;
        maxServiceLevel = 0;
        for t3=1:size(S_item,2)
            if (S_item(t1,t3) ~= 0)
                if (maxServiceLevel == 0)
                    maxServiceLevel = ServiceLevel(S_item(t1,t3));
                else
                    if (ServiceLevel(S_item(t1,t3)) > maxServiceLevel)
                        maxServiceLevel = ServiceLevel(S_item(t1,t3));
                    end
                end
            end
        end
        %display(maxServiceLevel);
        %display(availability(t1));
        if (availability(t1) < maxServiceLevel)
            isProbMet = 0;
        end
    end
    lowerStock(t1) = tmpLowerStock(t1)+1;

end

tmpItem = items;
c = 0;
optimumFound = 0;
currentStock = lowerStock;
currentCost = 0;
currentKitAvailability = 0;
kitProb = thesis(lowerStock, kits, U_kit, U_bothKits, rate, kitRate,
tau, T, S_item);
for t = 1:size(kitProb,2)
    currentKitAvailability = currentKitAvailability + kitProb(t);
end
S=lowerStock;
cost = 0;
for i=1:size(S,2)
    tmpCost = 0;
    m=S(i)-1;
    for j=0:m
        if (j==0)
            tmpCost = 0;
        else
            for k=0:j
                %tmpCost = tmpCost+h(i)*(1 - (ro(i))^j);
                tmpCost = tmpCost+h(i)*poisspdf(k,itemOnSite(i))*
                (1 - (ro(i)^(j-k+1)));
            end
        end
```

```
    end
    cost = cost + tmpCost;
end
currentCost = cost;
validStock = ones(1,size(lowerStock,2));
optimumCost = 0;
optimumRow = 0;

lowerProbMet = 1;
for i4=1:size(kitProb,2)
    if (kitProb(1,i4) > ServiceLevel(i4))
        for i5=1:size(S_item,1)
            if (S_item(i5,1) == i4)
                if (size(S_item,2) == 1)
                    validStock(1,i5) = 0;
                else
                    if (S_item(i5,2) == 0)
                        validStock(1,i5) = 0;
                    end
                end
            end
        end
    end
    if (kitProb(1,i4) < ServiceLevel(i4))
       lowerProbMet = 0;
    end
end

if (lowerProbMet == 1)
optimumFound = 1;
end

optimumRow = [lowerStock cost kitProb [0 0 0]];

if (c == 0)
    c = [lowerStock cost kitProb [0 0 0]];
else
    c = [c ; lowerStock cost kitProb [0 0 0]];
end
c = [c ; zeros(1,size(c,2))];

while (optimumFound == 0)
    n = combinationsIncremental(currentStock, validStock);
    costs = zeros(size(n,1),1);
    kitProbs = 0;
    kitProbSums = zeros(size(n,1),1);
    incCosts = zeros(size(n,1),1);
    incAvail = zeros(size(n,1),1);
    ratio = zeros(size(n,1),1);
    for i=1:size(n,1)
        kitProb = thesis(n(i,:), kits, U_kit, U_bothKits, rate, kitRate,
        tau, T, S_item);
        S=n(i,:);
        cost = 0;
        for i2=1:size(S,2)
            tmpCost = 0;
            m=S(i2)-1;
            for j=0:m
                if (j==0)
                    tmpCost = 0;
                else
                    for k=0:j
                        tmpCost = tmpCost+h(i2)*poisspdf(k,itemOnSite(i2))*
                        (1 - (ro(i2)^(j-k+1)));
                    end
                end
            end
            cost = cost + tmpCost;
        end
        costs(i) = cost;
        for t = 1:size(kitProb,2)
```

```
        kitProbSums(i) = kitProbSums(i) + kitProb(1,t);
    end
    if (kitProbs == 0)
        kitProbs = kitProb;
    else
        kitProbs = [kitProbs ; kitProb];
    end
end
tmpMinCost = 0;
tmpIndex = zeros(size(n,1),1);
isAllProbMet = 1;
for i2=1:size(kitProbs,1)
    rowProbMet = 1;
    for i3=1:size(kitProbs,2)
        if (kitProbs(i2,i3) < ServiceLevel(i3))
            isAllProbMet = 0;
            rowProbMet = 0;
        end
    end
    if (rowProbMet == 1)
        if (optimumCost == 0)
            optimumCost = costs(i2,1);
            optimumRow = [n(i2,:) costs(i2,1) kitProbs(i2,:) [0 0 0]];
        else
            if (costs(i2, 1) < optimumCost)
                optimumCost = costs(i2,1);
                optimumRow = [n(i2,:) costs(i2,1) kitProbs(i2,:) [0 0 0]];
            end
        end
    else
        if (optimumCost == 0)
            tmpIndex(i2) = 1;
        else
            if (costs(i2,1) < optimumCost)
                tmpIndex(i2) = 1;
            end
        end
    end

end

if (isAllProbMet == 1)
    optimumFound = 1;
else
    incCosts = costs - currentCost;
    incAvail = kitProbSums - currentKitAvailability;
    ratio = incCosts ./ incAvail;
    minRatio = 0;
    minRatioIndex = -1;
    for t1 = 1:size(ratio,1)
        if (tmpIndex(t1) == 1)
            if (minRatio == 0)
                minRatio = ratio(t1);
                 minRatioIndex = t1;
            else
                if (ratio(t1) < minRatio)
                    minRatio = ratio(t1);
                    minRatioIndex = t1;
                end
            end
        end

    end
    if (minRatioIndex == -1)
        optimumFound = 1;
    else
        currentStock = n(minRatioIndex,:);
        currentCost = costs(minRatioIndex);
        currentKitAvailability = 0;
        for t = 1:size(kitProbs,2)
            currentKitAvailability = currentKitAvailability +
```

```
                kitProbs(minRatioIndex,t);
            end
            for i4=1:size(kitProbs,2)
                if (kitProbs(minRatioIndex,i4) > ServiceLevel(i4))
                    for i5=1:size(S_item,1)
                        if (S_item(i5,1) == i4)
                            if (size(S_item,2) == 1)
                                validStock(1,i5) = 0;
                            else
                                if (S_item(i5,2) == 0)
                                    validStock(1,i5) = 0;
                                end
                            end
                        end
                    end
                end
            end

        end
    end
    if (c == 0)
        c = [n costs kitProbs incCosts incAvail ratio];
    else
        c = [c ; n costs kitProbs incCosts incAvail ratio];
    end
        c = [c ; zeros(1,size(c,2))];
end
    c = [c ; optimumRow];
display(c);
xlswrite('mm1_heuristic.xls',c);

function n = combinations(t)
    m = t(1)+1;
    for i=2:size(t,2)
        m = m * (t(i)+1);
    end
    n = zeros(m, size(t,2));
    for c=1:size(t,2)
        mm = 1;
        if (c ~= size(t,2))
            for i=c+1:size(t,2)
                mm = mm * (t(i)+1);
            end
        end
        c1 = 1;
        while c1 < m
            for nc=0:t(c)
                for c2=1:mm
                    n(c1,c) = nc;
                    c1 = c1+1;
                end
            end
        end
    end
end

function n = combinationsIncremental(current, valid)
    n = zeros(size(current,2), size(current,2));
    for i=1:size(current,2)
        for j=1:size(current,2)
            if (i==j)
                n(i,j) = current(j)+1;
            else
                n(i,j) = current(j);
            end
        end
    end
    tmp = 0;
    for i=1:size(valid,2)
        if (valid(1,i) == 1)
            if (tmp == 0)
```

```
                tmp = n(i,:);
            else
                tmp = [tmp; n(i,:)];
            end
        end
    end
    n = tmp;
end

function kitProb = thesis(Stock, kits, U_kit, U_bothKits, rate, kitRate, ro,
 T, S_item)
kitProb = zeros(1,size(kits,1));
for kit=1:size(kits,1)
    sum = 0;
    tmpStock = zeros(1,size(Stock,2));
    for nc=1:size(kits,2)
        item = kits(kit, nc);
        if (item ~= 0)
            tmpStock(1,item) = Stock(item) - 1;
        end
    end
    n = combinations(tmpStock);
    for i=1:size(n,1)
        tmp = calculateKitProb(n(i,:), kit, kits, U_kit, U_bothKits, rate,
        kitRate, ro, T, S_item);
        sum = sum + tmp;
    end
    kitProb(1, kit) =  sum;
end
end

function prob = calculateP(y, kit, kits, U_kit, rate, kitRate, ro, T, n, S_item)
    multiply = 1;
    for j=1:size(kits,2)
        i = kits(kit,j);
        if (i ~= 0)
            sum2 = 0;
            for c=1:size(S_item,2)
                if (S_item(i,c) ~= 0)
                    sum2 = sum2 + y(S_item(i,c));
                end
            end
            sum2 = n(i) - sum2;
            if (sum2<0)
                value = 0;
            else
                value = (ro(i)^sum2) * (1 - ro(i)) ;
            end
            multiply = multiply * value;
        end
    end
    for j=1:size(U_kit,2)
        if (U_kit(kit,j) ~= 0)
            value = ( exp(-kitRate(j) * T(j)) * ((kitRate(j) * T(j))^y(j) ) )
             / factorial(y(j));
            multiply = multiply * value;
        end
    end
    prob = multiply;
end

function kitProbability = calculateKitProb (n, kit, kits, U_kit, U_bothKits, rate,
 kitRate, ro, T, S_item)
    R = zeros(1,size(kits,1));
    for l=1:size(kits,1)
        if (U_kit(kit,l) == 1)
            minN = -1;
            for i=1:size(U_bothKits,3)
                tt = U_bothKits(l,kit, i);
                if (tt ~= 0)
                    if (minN == -1)
```

```
                    minN = n(1,tt);
                else
                    if (n(1,tt) < minN)
                        minN = n(1,tt);
                    end
                end
            end
        end
        if minN < 0
            minN = 0;
        end
        R(1,l) = minN;
    end
end
sum = 0;
y = combinations(R);
for t=1:size(y,1)
    tmp = calculateP(y(t,:), kit, kits, U_kit, rate, kitRate, ro, T,
    n, S_item);
    sum = sum + tmp;
end
kitProbability = sum;
end
```

APPENDIX C: DETAILS OF SIMULATION MODEL

One of the objectives of this study is to create a functional simulation model in order to provide a practical environment to evaluate the kit availabilities for a given base-stock level. The simulation software package Arena 9.0 is employed for the implementation of the system. The details of the system are presented in this chapter by examining the steps of the simulation model.

At the beginning, entities as much as the number of item types is created. We use the flow shown in Figure C.1. After the creation of entities, the counter block counts the item types and for each entity assign block looks the base-stock level of that item type and adds the stock levels together to find the total number of all items to be created. This provides us to have a flexible system where there is no need to define the total number of items each time we study a different system. After obtaining the total number of items to be created, entities are disposed.

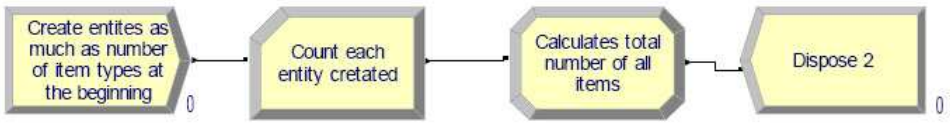

Figure C.1. Detailed Arena flow - 1

Figure C.2 shows the flow where all items are created and their item type attributes are assigned. To assign the item types, firstly base-stock level of each item is checked and the number of entities counted by the counter block is compared with this stock level. For example, consider we keep five items of type 1. We will count the first 5 entities, assign attribute of being type 1 item and then zero the counter. For item type 2, we will check its base-stock level and assign to the next that much amount of entities item type 2 and continue this loop until all items have an item type attribute. All items having an attribute are sent to a queue holding all items in the inventory.

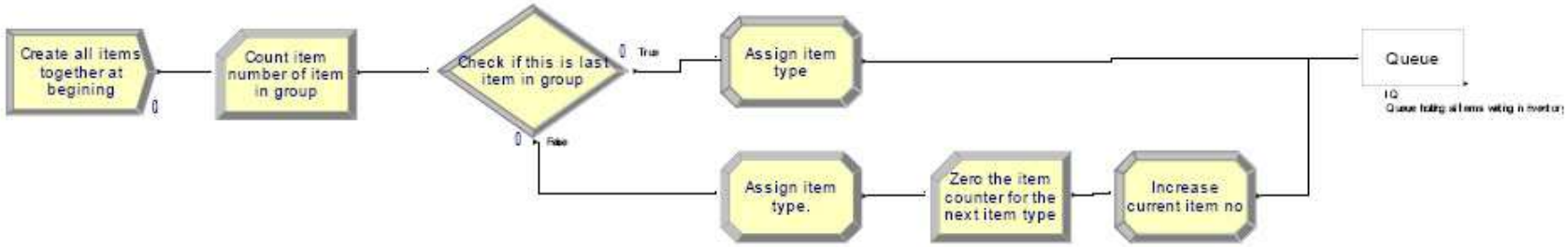

Figure C.2. Detailed Arena flow - 2

In the next step, demand for a kit is generated. Any distribution or expression may be used to set the time between kit demand arrivals. We study a Poisson arrival system so time between arrivals is set as exponential. When a demand arrival occurs, a kit type is assigned according to a discrete probability distribution. For example, consider a system of two kits. An expression like DISC(0.6,1,1,2) is used to assign kit probability of 0.6 to Kit 1 and 0.4 to Kit 2. A while block is then used to define a loop to select the items in the demanded kit. Search and decide modules check whether the items in the demanded kit are available in the stock queue. If one of the items in the demanded kit is not available, a new entity is created by duplicating the original entity having the needed item type attribute. If an item of the type supplied by duplicating is replenished meanwhile, this item is returned to the source. Figure C.3 and Figure C.4 present this flow of demand generation.

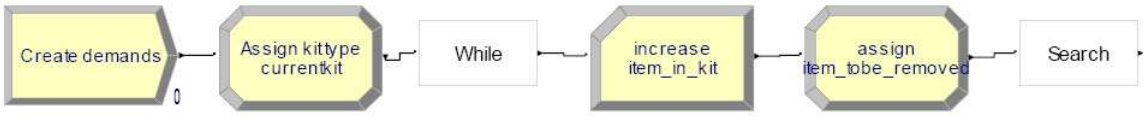

Figure C.3. Detailed Arena flow - 3

When all items are present, they are removed from the queue and sent to grouping to form the demanded kit as shown in Figure C.5 and Figure C.6. Then, the kit is delayed on site for a common time according to its kit type attribute. An expression like,

$$(kittype == 1) * expo(0.5) + (kittype == 2) * expo(0.2)$$

is used to define exponential waiting times with means 0.5 and 0.2. Meanwhile, an item to be used is selected according to a discrete probability distribution. Below expression may be used for a system of two kits; first kit containing items {1,2,3} and second kit containing {1,4}:

$$(kittype == 1) * DISC(0.4, 1, 0.7, 2, 1, 3) + (kittype == 2) * DISC(0.3, 1, 1, 4)$$

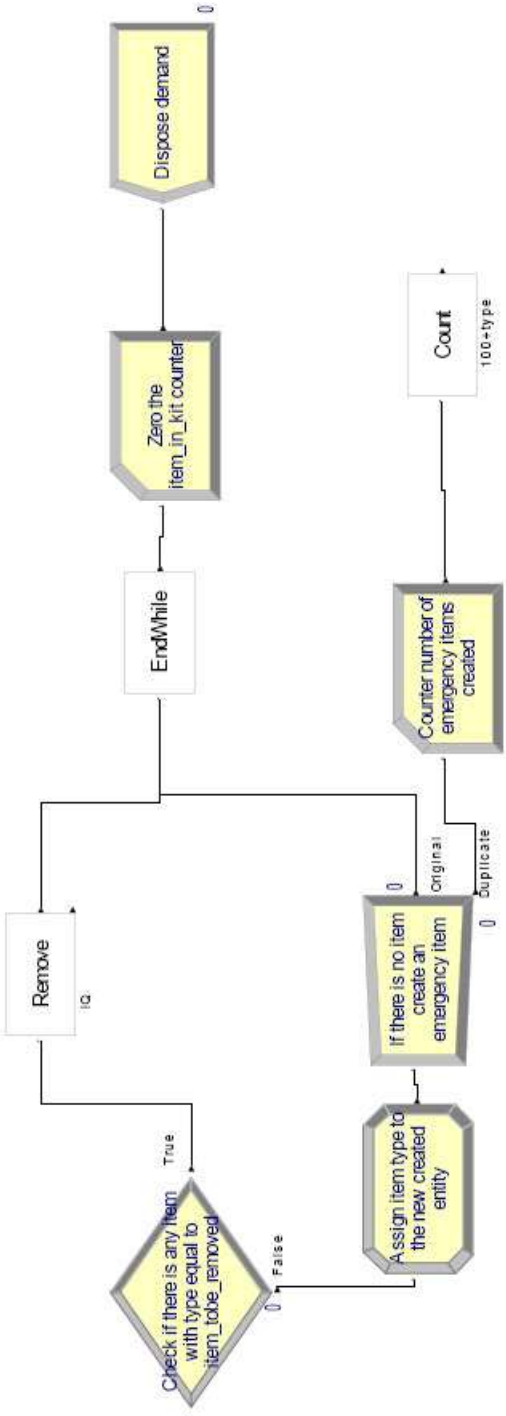

Figure C.4. Detailed Arena flow - 4

When an item is selected to be used, the kit is splitted to its items and unused items are sent back to the stock queue. The used item can be replenished independently through an infinite server queue as displayed in Figure C.5 or can be produced in a single or multiple server queue as shown in Figure C.6. For dependent replenishment, a resource (e.g. a machine) is seized to produce the item used according to a specified distribution. For example, for an M/M/1 system, the resource is delayed according to an expression like:

$$(type == 1) * expo(1) + (type == 2) * expo(1,2) + (type == 3) * expo(1,5)$$

depending on the item type to be processed. When the item is produced, the machine is released and the replenished item sent to stock to be kept until a kit containing that item is requested.

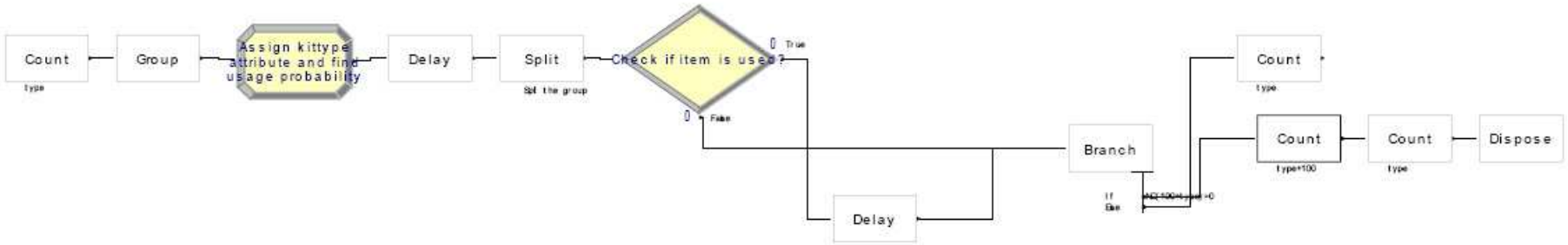

Figure C.5. Detailed Arena flow - 5, Independent replenishment

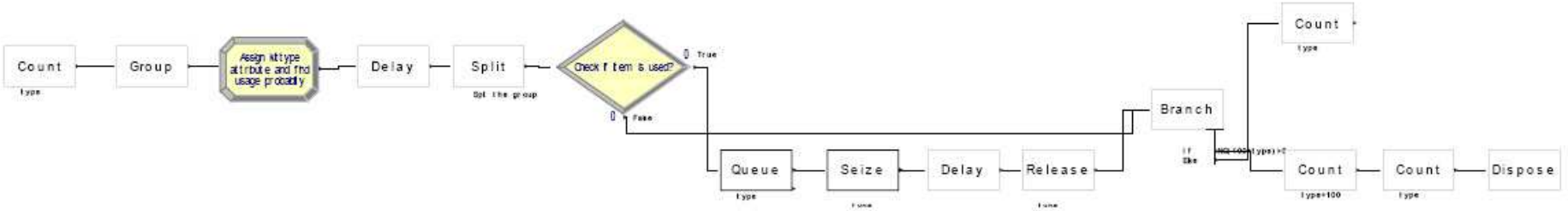

Figure C.6. Detailed Arena flow - 5, Dependent replenishment

REFERENCES

1. Hsu, V.N., C.Y. Lee, and R.C. So, "Optimal component stocking policy for assemble-to-order systems with leadtime-dependent component and product pricing", *Management Science*, Vol. 52, pp. 337-351, 1996.

2. Vardar, B., *Multi-item inventory control with joint set-up cost*, Graduation Project, Bogazici University, 2003.

3. Song, J., "On the order fill rate in a multi item, base-stock inventory system", *Operations Research*, Vol. 46, pp. 831-903, 1998.

4. Lu, Y., J. Song and D. D. Yao, "Order fill rate, lead time variability, and advance demand information in an assemble-to-order system", *Operations Research*, Vol. 51, pp. 292-308, 2003.

5. "Inventory Management", http://www.sba.gov/library/pubs/mp-22.pdf#search=%22inventory%20management%22, accessed 15 July 2006.

6. Axsäter, S., *Inventory Control*, Boston: Kluwer, 2000.

7. Lambert, D. M. and J. R. Stock, *Strategic Logistics Management*, Homewood: Irwin, 1993.

8. Glasserman, P. and Y. Wang, "Leadtime-inventory trade-offs in assemble-to-order systems", *Operations Research*, Vol. 46, pp. 858-871, 1998.

9. Cerda, C. B. and A. J. Monteros, "Evaluation of a (R,s,Q,c) multi-item inventory replenishment policy through simulation", *Proceedings of the 1997 Winter Simulation Conference*, Vol. 6, pp. 289-320, 1999.

10. "Fundamentals of Inventory Management and Control", http://www.amanet.org/selfstudy/b13503.htm, accessed 15 July 2006.

11. "Economic Order Quantity and Economic Production Quantity Models for Inventory Management", http://home.ubalt.edu/ntsbarsh/Business-

stat/otherapplets/Inventory.htm, accessed 10 July 2006.

12. Tersine, R. J., "Principles of Inventory and Materials Mangement", Englewood Cliffs, N.J. : PTR Prentice Hall, 1994.

13. Hausman, W. H., H. L. Lee and A. X. Zhang, "Joint demand fulfillment probability in a multi-item inventory system with independent order-up-to policies", *European Journal of Operational Research*, Vol. 109, pp. 646-659, 1998.

14. Mohebbi, E. and F. Choobineh, "The impact of component commonality in an assemble-to-order environment under supply and demand uncertainity", *Omega*, Vol. 33, pp. 472-482, 2005.

15. Benfaajar, S. and M. Elhafsi, "Production and inventory control of a single product assemble-to-order system with multiple customer classes", Unpublished Paper, Available from Author, 2006.

16. Song, J. and D. D. Yao, "Performance analysis and optimization of assemble-to-order systems with random lead times", *Operations Research*, Vol. 50, pp. 889-903, 2002.

17. Song, J. S. and P. Zipkin, "Supply chain operations assemble-to-order sytems", http://www.gsm.uci.edu/ song/Working%20Paper/Atocto28Jan03.pdf, accessed 5 August 2006.

18. Song, J., "Order-based backorders and their implications in multi-item inventory systems", *Management Science*, Vol. 48, pp. 499-516, 2002.

19. Zhang, A. X., "Demand fulfillment rates in an assemble-to-order system with multiple products and dependent demands", *Production and Operations Management*, Vol. 6, pp. 309-324, 1997.

20. Song, J., S. H. Xu, and B. Liu, "Order-fulfillment performance measures in an assemble-to-order system with stochastic leadtimes", *Operations Research*, Vol. 47, pp. 131-149, 1999.

21. Lu, Y. and J. Song, "Order-based cost optimization in assemble-to-order systems",

Operations Research, Vol. 53, pp. 151-169, 2005.

22. DeCroix, D., J. Song, and P. Zipkin, "A series system with returns: stationary analysis", *Operations Research*, Vol. 53, pp. 350-362, 2005.