Multi-Item (s,S) Inventory Systems with a Service Objective

J. Christopher Mitchell

University of North Carolina at Chapel Hill
Chapel Hill, N.C.

Mathematical and Information Sciences Division
Office of Naval Research, Code 434
Arlington, VA 22217

November 1983

157 and 43 (Appendix)

Approved for public release; distribution unlimited.

This report considers a Multi-item (s,S) Inventory System in which shortage costs are replaced by a stockout probability constraint. Necessary and sufficient conditions for a policy to be optimal are derived, and a computational efficient algorithm is developed. Computation experience indicates the operating costs can be reduced significantly.
MULTI-ITEM \((s,S)\) INVENTORY SYSTEMS
WITH A SERVICE OBJECTIVE

Technical Report #26
J. Christopher Mitchell

November 1983

Work Sponsored By
Office of Naval Research (N00014-78-C-0467)
Decision Control Models in Operations Research

Harvey M. Wagner
Principal Investigator
School of Business Administration
University of North Carolina at Chapel Hill

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED
FOREWARD

As part of the ongoing program in "Decision Control Models in Operations Research," Mr. J. Christopher Mitchell has studied an inventory model that uses a system-wide stockout constraint rather than individual shortage costs. He shows how to easily compute policies. The computational experience demonstrates that inventory investment can be reduced more than 20% using this approach. The report also suggests how to implement the algorithm in a real-life system that may contain thousands of stocked items. Other related reports dealing with the program are given on the following pages.


This paper considers a multi-item \((s,S)\) inventory system. The model differs from standard treatments in that shortage costs are replaced by stockout probability constraints to be satisfied in every period. The value of such a model is that it is often easier to express service objectives in terms of stockout probability constraints than it is to specify shortage costs. Specifically, system service is defined in terms of a weighted average of single-item stockout probabilities. An optimal policy minimizes system cost while satisfying a constraint on system service. Using standard single-item approximations, necessary and sufficient conditions for a policy to be optimal (for the approximate model) are derived, and a computationally efficient algorithm, the Generalized Knapsack Duality (GKD) Algorithm, is developed to find such a policy. Computational experience on inventory systems typical of many found in the real world indicates that operating costs can be reduced significantly when this model is used rather than the simpler uniform service model often used by managers.

Sensitivity experience on inventory systems with a structure typical of many real-world inventory systems is reported.
Specifically, the sensitivity of the GKD Algorithm to changes in the reorder quantities and lower bounds on feasible policies is reported. This experience suggests that, as in the single-item case, very accurate specifications of the reorder quantities are unnecessary. Recommendations are made to specify the lower bounds as high as possible while achieving significant cost savings below that of the Identical Service Approach. This recommendation is consistent with the objectives of many managers, and in certain cases improves algorithm performance.

This paper also reports computational experience with sampling schemes of various sizes for large-scale inventory aggregation. This experience indicates that inventory systems typical of many found in the real world can be well-managed based on decisions made from a relatively small number of items from the systems. In particular, this experience suggests that a central uniform sample of about 32 items is sufficient to make accurate decisions for the entire inventory system. The paper concludes with a detailed implementation procedure using the GKD Algorithm with sampling to manage a large-scale inventory system.
CONTENTS

I. INTRODUCTION
   1. Inventory Theory with Shortage Costs  3
   2. Inventory Theory with Service-Level Constraints  10
   3. Multi-Item Inventory System Aggregation  15

II. THE GENERALIZED KNAPSACK DUALITY (GKD) ALGORITHM
   1. Introduction  18
   2. Analysis  21
   3. GKD Algorithm  40
   4. Policy Performance  46
   5. GKD Algorithm Modification when $D_i$ is Small  50
   6. Conclusions  52
      - Notation  53
      - Frequently Used Relations  55
      - Theorems and Lemmas  55
      - Appendix  56

III. COMPUTATIONAL EXPERIENCE WITH THE GKD ALGORITHM  57
    1. Introduction  57
    2. Total Cost Savings  57
    3. Single-Item Service-Levels  62
    4. Items with $D_i$ Too Small  63
    5. Conclusions  65

IV. SENSITIVITY EXPERIENCE WITH THE GKD ALGORITHM  80
    1. Introduction  80
    2. Sensitivity Experience for $D_i$  81
    3. Sensitivity Experience for Lower Bounds on $S_i$  85
    4. Sensitivity Experience for Sampling Schemes for Inventory Aggregation  92
    5. Sensitivity Experience for the Sample Size for Inventory Aggregation  99
    6. Conclusions  104
       - Appendix  134
V. CONCLUSION: IMPLEMENTATION OF THE GKD ALGORITHM WITH SAMPLING FOR A LARGE-SCALE INVENTORY SYSTEM

1. Introduction 146
2. Implementation Procedure 148
3. Example 151
4. Conclusions, and Future Research Directions 152
   - Appendix 158

REFERENCES 195
ACKNOWLEDGEMENTS

I want to thank above all my dissertation advisor, Professor Harvey Wagner. His extensive experience and sharp insights have been invaluable in determining both the content and direction of this research. To Professor Wagner I also offer special thanks for always helping me to see "the forest for the trees."

I want to thank Professor Richard Ehrhardt for his generous time and interest in my work, as well as his technical expertise in inventory theory. Many of our conversations helped shape the theoretical development of this work.

I want to thank Professor Jon Tolle for the considerable and valuable time he invested in my work. His expertise in nonlinear programming as well as his careful reading of earlier manuscripts resulted in significant improvements in this work. The mathematical clarity of Chapter II was greatly enhanced by his suggestions.

I want to thank my typist, Miss Kristina Hanan. She has always done a careful and excellent job, even in the most tedious parts of the paper.

Special thanks are given to the Office of Naval Research, who partially supported this research under Grant #N00014-78-C-0467.
I. INTRODUCTION

This paper examines the problem of specifying single-item service objectives in a multi-item inventory system subject to an overall (or system-wide) service-level constraint. We develop a computationally efficient algorithm for doing so when the number of items in the inventory system is of moderate size. We also investigate methods of aggregating very large inventory systems so that they are computationally more manageable. For both of these problems, we show by extensive numerical investigations that our methods can result in a substantial total cost savings over methods which specify uniform service-level objectives for all items in the inventory system.

This chapter is a non-technical survey of the literature on both exact and approximate methods for inventory management. Section 1 describes the theoretical and computational difficulties involved in single-item inventory management as well as methods that have been used to deal with these difficulties. Certainly a multi-item inventory system will inherit these difficulties, and so we use such single-item methods in our model when appropriate.

Section 2 discusses service-level constraints in single-item inventory systems in the same spirit as Section 1. We also survey the methods that have been used in multi-item inventory systems with service-level constraints (or similar models) in order to motivate the direction of our research.
Section 3 surveys the literature on the problem of large-scale inventory aggregation. In particular, we discuss an empirically-observed structure that is typical for many real-world inventory systems.

Chapter II contains the mathematical derivation of our algorithm to specify single-item service objectives in multi-item inventory systems with a system-wide service-level constraint. There are certain properties the items in the inventory system must satisfy in order to guarantee algorithm convergence. We illustrate algorithm performance on some two-item inventory systems which contain only items satisfying these properties. We also discuss our recommendations when there are items in the inventory system that do not satisfy all these properties.

Chapter III contains a numerical investigation of some 32-item inventory systems (with some items that do not satisfy the above-mentioned properties) which reflect a structure often observed in practice. We show that there is a significant cost decrease when using our algorithm to specify operating policies rather than using the popular method of specifying operating policies which give uniform service.

Chapter IV contains a numerical sensitivity investigation of 32-item and 128-item inventory systems (with the same structure mentioned), and a numerical investigation of sampling schemes for inventory aggregation.

Chapter V concludes this paper with a detailed implementation procedure of the GKD Algorithm with sampling to manage a large-
scale inventory system. We illustrate this procedure with a 512-item inventory system, and offer conclusions and directions for future research.

1. INVENTORY THEORY WITH SHORTAGE COSTS

1.1 Model Formulation and Optimal Policies

We consider the periodic-review dynamic inventory model. We give a non-technical survey of the literature on this model, leaving relevant mathematical descriptions for the next chapter.

At the beginning of each period \( n, n=1,2, \ldots \), the inventory position (stock on hand plus stock on order) is reviewed, at which time a positive order may be placed. An order placed in period \( n \) is received and paid for in period \( n+k \), where \( k \), the leadtime, is a fixed positive integer. There is charged a fixed ordering cost \( K \) plus a linear ordering cost \( c \). After the inventory position is reviewed there is a random demand \( \xi_n \), where \( \xi_1, \xi_2, \ldots \) are independent and identically distributed with cumulative distribution function (cdf) \( \Phi \), density \( \phi \), mean \( \mu \), and finite standard deviation \( \sigma \). After demand realization, a linear holding cost \( h \) is assessed for each unit of inventory on hand. If demand exceeds the inventory on hand, the excess demand is completely backlogged, and a linear shortage cost \( p \) is assessed for each unit of backlogged demand. Then period \( n+1 \) is entered, repeating the process. Future costs are discounted at the single-period rate \( \beta \) with \( 0 \leq \beta \leq 1 \) (\( \beta=0 \) corresponds to a single-period model and \( \beta=1 \) corresponds to an undiscounted model), and the objective is to minimize the total expected cost of operating the system over a prescribed horizon.
This model can be formulated as a dynamic program. Doing so and using induction, Scarf [1960] showed that under the assumptions of finite horizon length $T$ and convex differentiable expected single-period holding costs, there is an optimal policy of the form $(s_n, S_n)$, $n = 1, \ldots, T$. This policy requires that if the inventory position at the beginning of period $n$ is less than $s_n$, an order is placed to raise it to $S_n$. Otherwise, no order is placed. Zabel [1962] extended this result to the case when the single-period holding and shortage costs are not differentiable. Earlier Karlin [1958a] showed the optimality of $(s_n, S_n)$ policies under much stronger assumptions than Scarf; and Veinott [1966a] and Schäl [1976] extended Scarf's result to models with more general cost functions. If, however, holding and shortage costs are linear, as in our model, Scarf's result is sufficient to guarantee the optimality of $(s_n, S_n)$ policies.

A basic difficulty in implementing these policies is that they usually vary from period to period and hence require tremendous computational effort. The situation becomes much simpler in the infinite horizon case. In this case, to minimize total expected cost we must impose $\beta < 1$ or else for almost any policy this cost is infinite. One can consider an undiscounted model ($\beta = 1$) if the objective criterion is changed to the average expected cost per period. If this is the criterion, Iglehart [1963a] and [1963b] proved that given the cost structure we have assumed, in both the discounted and undiscounted case, respectively, there is an optimal policy that is stationary $(s, S)$. Such a policy requires that when the inventory position in any period falls below $s$, an order is placed to bring it up to $S$. Thus, there are
only two numbers to be computed. One must, of course, assume that the true horizon length is of sufficient length to be reasonably approximated by an infinite horizon model.

The computational procedures to find exact stationary \((s,S)\) policies generally involve deriving a closed form expression for the expected infinite horizon average cost per period given that the policy \((s,S)\) is being followed in every period, and then minimizing this cost over all \((s,S)\) policies. There have been several methods used to derive this steady-state cost. Karlin [1958a] derived it using linear operator theory and in [1958b] using renewal theory, which was simplified by Sahin [1982]. We mention that in both references cited, Karlin also derived a closed form expression for the optimal policy when the demand distribution is exponential. Morse [1959] derived the expected average cost per period using Markov processes theory, and Leneman and Beutler [1969] used a stationary point process approach. The first general and exact computational procedure to find an optimal stationary \((s,S)\) policy is found in Veinott and Wagner [1965]. Assuming that the demand distribution is discrete, they derived the steady-state cost and then minimized it using finite difference calculus. This algorithm has been programmed in PL-1, and the documentation can be found in Kaufman [1976]. Bell [1970] has suggested an improvement to the Veinott-Wagner algorithm using optimal stopping rule theory, and recently Federgruen and Zipkin [1981] have described a completely different approach using a policy iteration technique for Markov decision processes. A method similar to this was suggested earlier by Johnson [1968]. In a somewhat different spirit, Sivazlian [1971] uses Gaussian quadrature methods to create graphs to compute optimal \((s,S)\) policies.
There are three significant difficulties involved in finding optimal \((s,S)\) policies. We discuss these in order to introduce the various approximations to optimal \((s,S)\) policies that have been suggested in the literature.

First, the steady-state costs always involve the renewal function of the demand process. If the demand distribution is discrete, Veinott and Wagner [1965] showed that the renewal function can be evaluated recursively, although this can be computationally expensive when \(S\) is very large. If the demand distribution is continuous, there is no general computationally efficient method of even approximating the renewal function (the exception being the exponential distribution).

Second, although the steady-state cost is, in general, convex in \(S\), it is not even unimodal in \(D=S-s\). Thus, local minima may not be global minima. Veinott and Wagner [1965] dealt with this by establishing bounds on the optimal \((s,S)\) policy and examining all values of \(D\) among these policies to find a global minimum. Recently Sahin [1982] has shown that for a class of distributions the steady-state cost is pseudo-convex in \(D\) and \(S\), so local minima are global minima. It is not clear, however, that this class contains distributions useful for inventory models.

Third, exact algorithms require that the demand distribution be fully known. Many times in practice only a couple of moments will be known, and even these values may be statistical estimates.
1.2 Approximations to Optimal Policies

In light of the difficulties involved in finding optimal policies, numerous approximation procedures have been suggested. We describe the best of these.

For an undiscounted infinite horizon inventory model the ordering cost $c$ does not appear in the cost functions. Essentially this is because all demand must eventually be satisfied, and the cost of doing so is undiscounted. Thus, there are three components of total expected cost, one associated with each of the parameters $K$, $h$ and $p$. They can be denoted, respectively, as the expected replenishment cost, expected holding cost and expected shortage cost. Another quantity of interest is the service-level, the frequency of periods without any backorder (roughly, the steady-state probability of meeting all demand in any given period).

One of the earliest and best approximations is the Normal Approximation of Wagner [1975, pp.831-836]. It is based on asymptotic renewal theory results of Roberts [1962], the empirically-based heuristic when $D=S-s$ is small discussed in Wagner et al [1965], and the assumption that demand is well-enough approximated by a Normal distribution. There were extensive numerical investigations of this approximation done in MacCormick [1974], Estey and Kaufman [1975], MacCormick et al [1977], and MacCormick [1977] in which the actual underlying demand distribution was assumed to be Negative Binomial or Poisson. The Normal Approximation in general performed very well even when statistical estimates of the demand distribution mean and variance were used (although degradation was greater in this case). The greatest degrad-
ation in performance was observed when the coefficient of variation \( \frac{\sigma}{\mu} \) was large (in which case there is a non-negligible probability of negative demand for a Normal random variable), and when both \( \frac{\sigma^2}{\mu} \) and \( \frac{p}{h} \) were large (the latter corresponds to a high service level). When \( \frac{\sigma^2}{\mu} \) and \( \frac{p}{h} \) were large, the expected shortage cost and service-level degraded much more than the expected replenishment and holding costs.

Using the general approach of Norman and White [1968] of computing approximately optimal policies in Markov Decision processes by replacing the probability distributions by their moments, Porteus [1979] developed an algorithm to compute approximately optimal \((s,S)\) policies. However, the empirical examination done in Freeland and Porteus [1980] showed that it was not much better than the Normal Approximation, and was a lot more work to compute.

Using regression models suggested by both asymptotic analysis and empirical observations, Ehrhardt [1976] developed the Power Approximation. Extensive numerical investigations were performed by Ehrhardt [1976] (compare Ehrhardt [1978]) and Klincewicz [1976a] and [1976b]. These investigations showed that not only are the approximations especially easy to compute, require only the first two moments of the demand distribution and very useful for sensitivity analysis, but they are extremely accurate over a wide range of parameter settings. The Power Method’s performance was always superior to the performance of the Normal Approximation. The only cases when the Power Approximation did not perform well occurred in the expected shortage cost and service-level when statistical demand moments were used and \( \frac{p}{h} \) was large (which implies a high service-level).
Some technical improvements were made in the Power Approximation by Mosier [1981], and the regression approach was used by Ehrhardt [1977] (compare Ehrhardt [1981]) to approximate the expected costs and other quantities of interest when the (s,S) policy is given.

There are other policies of a suboptimal form that have been used to control inventory systems, primarily because these policies are considerably easier to compute and analyze than are (s,S) policies. Two of the most popular are the (t,S) and (s,Q) policies (see Hadley and Whitin [1963, pp.235-295]). A (t,S) policy requires that every t periods an order be placed to raise the inventory position to S. An (s,Q) policy requires that whenever the inventory position falls below s, an order of size Q is to be placed. Using these policies Naddor [1975] developed rules for approximating the optimal (s,S) policy. These approximations were numerically compared in Kastner and Ehrhardt [1979] and in Ehrhardt and Kastner [1980]. The Power Approximation was always superior, although the Naddor Approximation did quite well for a large number of parameter settings.

We conclude by summarizing how these approximately optimal policies deal with the three difficulties described in Section 1.1. Typically, the renewal function is approximated using standard results from asymptotic renewal theory. Although we did not discuss it in detail in this section, the complicated total expected cost dependence on D is also dealt with using asymptotic renewal theory, with an empirically-based heuristic being implemented when D is small (in other words, the asymptotic assumptions are unreasonable). In order to have approximations that depend upon only a few moments of the demand distribution rather than the entire density, the form of the demand distribution is
assumed to be given (typically Normal, Gamma, Poisson or Negative Binomial). The Power Approximation utilized a very general distribution form in which parameters were determined by a regression fit.

2. INVENTORY THEORY WITH SERVICE-LEVEL CONSTRAINTS

2.1 Model Formulation and Optimal Policies

A significant practical difficulty inherent in all these approximations (and exact algorithms) is the specification of shortage costs. These penalize backorders, but it can be very difficult to measure the cost impact of a backorder. It entails such things as "loss of goodwill" or "customer dissatisfaction", or equally difficult quantities to measure. One way to deal with this difficulty is to replace the shortage cost with a service-level constraint. For example, one can require that the frequency of backorders be no greater than 15%. Such measures can often be easier to specify than shortage costs.

A question of some interest is whether or not there is an optimal policy of the \((s,S)\) form when a service-level constraint is used rather than a shortage cost. In general, there is not. Using a fairly general service-level constraint, Fromovitz [1965] showed that even in a single-period model the optimal policy may be randomized. He does show that if the service-level constraint is convex there is an optimal policy that is non-randomized, but most service-level constraints considered in the literature are not convex. Beesack [1967] has shown for the finite-horizon model (with the constraint that the expected number of stockouts be at most a prescribed fraction of expected demand) that \((s_n,S_n)\) policies are still optimal. However, as most managers are unwilling to implement a policy more complicated than an \((s,S)\) policy,
most of the research is aimed at finding the best \((s,S)\) policy that meets a given service-level.

2.2 Single-Item Models

Roberts [1962] showed that asymptotically the optimal value for \(D=S-s\) is independent of the shortage cost \(p\), so typically it is assumed that \(D\) is given, and then approximations for \(s\) are derived such that the resulting \((s,S)\) policy satisfies a prescribed service-level. The two service-levels most often considered are the one we use (frequency of backorders) and the one Beesack considered (fraction of expected demand met). In the spirit of Robert's work, Greenberg [1964] and Schneider [1978] derived closed form expressions for these two service-levels as functions of \(s\) and \(D\). Using asymptotic renewal theory and both a Normal and Gamma demand distribution, Schneider [1978] derived approximations to find an approximately optimal \(s\) given \(D\) for both service-level constraints. Only the approximation to the constraint used by Beesack was investigated numerically. The Gamma Approximation always performed well, and the Normal Approximation performed well as long as the coefficient of variation was not too large. Tijms and Groenevelt [1982] extended these results to the case when the leadtime is a random variable.

2.3 Multi-Item Models

We now discuss the model whose study is the object of this paper. We consider an \(N\)-item inventory system in which each item \(i\) is an inventory model described in Section 1.1 without shortage costs assigned. Thus, for each item \(i, i=1,\ldots,N\), we have a fixed leadtime
k_i, fixed ordering cost K_i plus linear ordering cost c_i, and linear holding cost h_i. The demand realized for item i in any period has continuous cdf \( \phi_i \), density \( \phi_i \), mean \( \mu_i \) and finite standard deviation \( \sigma_i \). All demands for all items in all periods are assumed to be independent. The criterion of optimality is the minimum expected undiscounted cost per period over an infinite horizon, and the service-level constraint to be satisfied in every period is

\[
\sum_{i=1}^{N} W_i [\text{frequency of periods item } i \text{ is not backordered}] \geq \alpha,
\]

where \( W_1, \ldots, W_N > 0 \) and \( 0 < \alpha < 1 \) are specified with \( \sum_{i=1}^{N} W_i = 1 \).

A feasible policy is a pair \((s, S)\), where \( s=(s_1, \ldots, s_N) \) and \( S=(S_1, \ldots, S_N) \), and the stationary policy \((s_i, S_i)\) is followed for item i in every period.

This model has not been examined in the literature, but several related (but much simpler) models have. We briefly survey the results.

For a single-item inventory model with a shortage cost, if \( K=0 \) there is an optimal policy of the form \((S, S)\), a base-stock policy (see Veinott [1966b]). Iglehart and Jaquette [1969] showed that if the shortage cost is replaced by a service-level constraint, a base-stock policy is still optimal. Mitchell [1982] considered this model with all leadtimes \( k_i=0 \) and exponential demands, and developed a computationally efficient algorithm to find the optimal base-stock policy. Evans [1967] considered a finite-horizon base-stock model in which total costs were minimized subject to a linear resource constraint. Using induction on the dynamic programming formulation, he derived the
(extremely complicated) optimal policy.

In Section 1.2 we described suboptimal policy forms which, because of the relative ease in analyzing them, have been used in inventory control when $K_i > 0$. Recall we described the $(s,Q)$ policy (often called a $(Q,r)$ policy in the literature), which requires that whenever the inventory position drops below $s$, an order of size $Q$ is placed. There are several papers that consider our multi-item model (with perhaps a different constraint) governed by stationary $(s,Q)$ rather than $(s,S)$ policies.

There are two approaches to solving such problems that have been used, both of which are discussed in Hadley and Whitin [1963, pp.213-219, 304-307, 323-226]. One is to treat the problem as a constrained nonlinear program and then solve the first-order Kuhn-Tucker conditions. The second is to formulate the model as a dynamic program (a generalized Knapsack Problem) and then solve it stage by stage.

The method using the Kuhn-Tucker conditions is used in Winters [1962], Parker [1964], Gerson and Brown [1970], Presutti and Trepp [1970], Schrady and Choe [1971], and Schroeder [1974]. The method using dynamic programming is used in Kaplan [1978].

as did Schroeder [1974]. Kaplan [1972] considered a somewhat different model, a finite-horizon model with budget constraints that penalize dollars spent before the end of the horizon. It is a single-item model with $s=0$. We note that the models in Parker [1964] and Schroeder [1974] assume a continuous rather than periodic review of the inventory position. Such models tend to produce policies that give rather poor performance in a periodic-review setting (see Wagner et al [1965]).

For two reasons, all of these studies are somewhat unsatisfactory. First, as mentioned, $(s,Q)$ policies are not optimal. Second, in none of them are the policies produced compared with any other method, making it difficult to evaluate the performance of such policies in a multi-item inventory system. In our paper we use $(s,S)$ policies and derive an algorithm, the GKD (Generalized Knapsack Duality) Algorithm to approximately solve our model, and then compare the $(s,S)$ policies produced with an approach frequently used in practice by managers, the Identical Service Approach (compare Mitchell [1982]). This approach sets shortage costs for each item in an inventory system so that the service-level for each item is the same, say $\alpha$. Then each item is treated individually, with policies being computed by some existing method. We will show that using our model, which in effect varies the individual item service-levels while still maintaining an overall service-level of $\alpha$ (via our service-level constraint), total expected costs can be significantly reduced for many multi-item inventory systems. This is the first issue with which we deal.

The second issue involves aggregating very large inventory systems in such a way that a representative fraction of them can be used to determine the policies for all of them, which we discuss next.
3. MULTI-ITEM INVENTORY SYSTEM AGGREGATION

Many real-world inventory systems contain thousands of items. It would be cost prohibitive to try to include every one of them in any model which is to be used to set inventory policies. Inventory aggregation is any method of partitioning such a system into blocks, each of which will be dealt with essentially the same. For example, a representative item could be chosen from each block to determine the policy that will be used for all items in the block. Thus the problem is reduced to more manageable proportions. The question is, of course, how to aggregate so that the resulting policies are as close as possible (in a total expected cost sense) to policies that would have been produced had every item in the inventory system been included in the model.

Very little is known about this problem (see Bitran and Hax [1982], which deals with a finite horizon, deterministic demand inventory system). However, extensive empirical investigations of many real-world inventory systems have suggested some remarkable similarities in the overall distribution of items in inventory systems. These results are discussed in Brown [1959], [1963], [1967] and [1977], and Peterson and Silver [1979, pp. 30-37, 71-80]. We summarize the observations in Peterson and Silver.

The annual dollar demand of an item is defined to be its annual expected demand times its value per unit. Of course, the "value" of an item may be difficult to determine exactly, but can often be taken to be its ordering or holding cost. Typically, five to ten percent of the items in an inventory system account for about fifty percent of the total annual dollar demand of the entire inventory.
These items tend to be very important, and are denoted Type A items. On the other hand, typically about thirty to fifty percent of the items account for only five to ten percent of the total annual dollar demand. These are the relatively unimportant items, and are denoted Type C items. The rest of the items in the inventory system, the moderately important ones, are denoted Type B items.

If the items in such an inventory system are ordered in descending order by their annual dollar demand, and then the cumulative percent of the items is plotted against the corresponding cumulative percent of annual dollar demand, the graph would look something like Exhibit 1. Notice also that about twenty percent of the items account for about eighty percent of the total annual dollar demand. This also is typical for many real-world inventory systems.
Empirical observations have shown that this graph can very often be well-approximated by a Lognormal cumulative distribution function.

We address the second issue of this paper, large-scale inventory aggregation, by showing that sampling techniques which stratify on the item types and which maintain the structure illustrated in Figure 1 can be used to accurately predict the (s,S) policies for all the items in the inventory system. We exploit the special structure of our inventory problem by using the sample to estimate the Lagrange constraint multiplier of the inventory problem, from which we compute all (s,S) policies.
II. THE GENERALIZED KNAPSACK DUALITY (GKD) ALGORITHM

1. INTRODUCTION

We consider a multi-item inventory system and study the problem of specifying both system-wide and individual service objectives. Most theoretical models considered in the literature assume that holding or shortage costs are applied to any excess inventory or unsatisfied demand, respectively. A major difficulty in applying such models in practice is the specification of shortage costs. Frequently, the manager sets shortage cost parameters based on an objective of satisfying demand with at least some minimum probability. Such an approach may be preferred because subjective factors can more easily be expressed as a probability of satisfying demand than as a cost for each shortage incurred.

A significant shortcoming of this approach is that it usually entails the setting of a probability of demand satisfaction that is applied uniformly to all items in the system. In this paper we raise the issue of specifying different service objectives for individual items while still satisfying some given system-wide objective. The value of this approach is that total system costs can be reduced below those of the method that requires identical service for all items.

We are interested in the following multi-item inventory system. For each item $i$ there is a set-up ordering cost $K_i$, a unit ordering cost $c_i$, and a fixed leadtime $k_i$ between placement and delivery of orders. These costs are assessed upon order delivery. There is a unit holding cost $h_i$ assessed at the end of each period, and the demand
realized for item $i$ in any period has absolutely continuous cumulative distribution function (cdf) $F_i$, density $\phi_i$, mean $\mu_i$, and finite standard deviation $\sigma_i$. All demands for all items in all periods are mutually independent. There are $N$ items, and the criterion of optimality is the minimum expected undiscounted cost per period over an infinite horizon. We do not include shortage costs to penalize back-orders. Rather we introduce the following service-level constraint to be satisfied in every period:

$$\sum_{i=1}^{N} W_i \geq \alpha,$$

where $0 < \alpha < 1$, $W_1, \ldots, W_N > 0$ and $\sum_{i=1}^{N} W_i = 1$.

For such an inventory model, an $(s, S)$ policy requires that whenever the inventory position (inventory on hand plus on order) drops below $s$, an order is placed bringing the inventory position up to $S$. An $(s, S)$ policy is called stationary if it is applied in every period. We only consider stationary $(s, S)$ policies because when shortage costs are included in the model, there is an optimal policy which is stationary and $(s, S)$ (Iglehart [1963]). Any stationary $(s, S)$ policy for our model gives rise to a unique and well-defined specification of shortage costs. If we denote by $P_i(s_i, S_i)$ the frequency of periods that item $i$ is not backordered when following the policy $(s_i, S_i)$, then, when $z_i$ is continuous,

$$P_i(s_i, S_i) = \frac{p_i}{p_i + h_i}, \quad (1)$$

where $p_i$ is the shortage cost under which the policy $(s_i, S_i)$ is optimal.
(Veinott and Wagner [1965]). We can solve this equation for $p_i$ and thus determine the shortage costs associated with any stationary $(s,S)$ policy.

For our model a feasible policy is a set $\{(s,S)\}$, where $s=(s_1,\ldots,s_N)$ and $S=(S_1,\ldots,S_N)$. For item $i$ we follow the policy $(s_i,S_i)$ in every period. We define $D_i$ by $D_i=S_i-s_i$, and denote by $\phi_i^*$ the $(k_i+1)$-fold convolution of $\mathcal{N}_i$ with itself, and by $\phi_i^*$ its density. Also let $\mu_i^*$ and $\sigma_i^*$ be the mean and standard deviation of a random variable with cdf $\mathcal{N}_i^*$. In Mitchell [1982] we considered this problem when $K_i=0$ for all items $i$, and developed a computationally efficient algorithm for the case when all $k_i=0$ and all demands are exponentially distributed. We extend those results in this paper, dropping all three of these restrictions.

Section 2 describes the formulation of our inventory problem as a non linear program. Using the Kuhn-Tucker conditions, we prove necessary and sufficient conditions for (possibly local) optimality. The main results are found in Theorems 1, 2, 6, 7 and 8 (the reader may choose to skip the proofs of these results). There is at the end of this chapter a list of the notation, a summary of frequently used relations, and a summary of the theorems and lemmas.

Section 3 is a description of the algorithm. It is based on local duality theory of nonlinear programming, and the proof of convergence uses results from Section 2. Section 4 reports a numerical investigation. In Section 5 we discuss our recommendations when the sufficient conditions may not hold for some items in the inventory system. Finally, Section 6 contains some conclusions.
2. ANALYSIS

If we denote by $C_i(S_i, D_i)$ the expected average undiscounted cost per period over an infinite horizon, and by $P_i(S_i, D_i)$ the frequency of periods that item $i$ is not backordered, we have the following result.

Theorem 1. For $S_i > D_i$,

(i) $C_i(S_i, D_i) = \frac{1}{1 + M_i(D_i)} \left\{ h_i \left[ \int_0^{D_i} \int_{S_i-y}^{D_i} (S_i-y-\phi_i(x)m_i(y)) dx dy 
+ \int_{-\infty}^{S_i-x} \phi_i*(x) dx \right] + K_i \right\}$

(ii) $P_i(S_i, D_i) = \frac{1}{1 + M_i(D_i)} \left\{ \int_0^{D_i} \xi_i*(S_i-y)m_i(y) dy + \xi_i*(S_i) \right\}$,

where $M_i$ is the renewal function of $\xi_i$ and $m_i = M_i'$.  


For two reasons we restrict attention to $(s_i, S_i)$ policies with $s_i \geq 0$ (equivalently, $S_i > D_i$). First, the expressions given in Theorem 1 do not hold when $s_i < 0$. One must replace with $(0, S_i)$ the limits of integration in the first integral of $C_i$ and the limits of integration in $P_i$. Second, many managers consider policies with $s_i < 0$ to be unsatisfactory for practical implementation.

Thus we want to solve the following constrained nonlinear program (NLP) to obtain an optimal policy for our inventory system:
\begin{aligned}
\text{minimize } & \sum_{i=1}^{N} C_i(S_i, D_i) \\
\text{subject to } & \sum_{i=1}^{N} W_i \left( P_i(S_i, D_i) - \alpha \right) \geq 0 \\
& S_i \geq D_i, \ i = 1, \ldots, N ,
\end{aligned}

where \( 0 < \alpha < 1 \), \( W_1, \ldots, W_N > 0 \) and \( \sum_{i=1}^{N} W_i = 1 \).

There are three serious difficulties in finding a solution. First, the renewal function \( M_i \) generally does not have a closed-form expression, and usually is complicated when it does. Second, the functional dependence of \( C_i \) and \( P_i \) on the variable \( D_i \) is complicated, as is easily seen if one works out the partial derivatives. Third, the behavior of the functions depends heavily on the demand distributions \( F_i \).

We deal with the first difficulty by using asymptotic approximations for the renewal functions. In particular, Smith [1954] has shown

\[ M_i(y) = \frac{y}{\mu_i} + \frac{\sigma_i^2}{2\mu_i^2} - \frac{1}{2} + o(1) \text{ as } y \to \infty. \quad (2) \]

We use the following approximations.

\[ M_i(D_i) \approx \frac{D_i}{\mu_i} + \frac{\sigma_i^2}{2\mu_i^2} - \frac{1}{2}. \]

\[ \frac{1}{1 + M_i(D_i)} \approx \frac{\mu_i}{D_i + (\mu_i + \sigma_i^2/\mu_i)/2} \approx r_i. \quad (3) \]

We approximate \( m_i(y) = M_i'(y) \) by a constant function, since \( M_i(y) \) is asymptotically affine. One could differentiate (2) and approxi-
mate \( m_1(y) \) by \( \frac{1}{\mu} \), but we use the somewhat better approximation discussed in Ehrhardt [1981], namely,

\[
m_1(y) \approx \frac{1-p_i}{\mu D_i}, \quad y \in (s_i, S_i).
\] (4)

(This approximation approaches \( \frac{1}{\mu} \) as \( D_i \to \infty \)).

For the complicated functional dependence on \( D_i \), we approximate \( D_i \) by a constant, and regard \( C_i \) and \( P_i \) as functions only of \( S_i \).

Roberts [1962] proves that in the single-item case where a shortage cost \( p \) is specified, the optimal value \( D^* \) of \( D \) is given by

\[
D^* = \sqrt{\frac{2K\mu}{h}} + \text{constant} + o(1)
\]

as \( K \to \infty, p \to \infty \) such that \( \frac{K}{p} \) remains constant.

Since \( D^* \) is asymptotically independent of the shortage cost \( p \), and hence by (1) of the service-level \( \alpha \), it would seem reasonable to approximate \( D^* \) by the constant \( \sqrt{\frac{2K\mu}{h}} \), the so-called "economic order quantity."

An extensive empirical comparison of approximately optimal inventory policies by Wagner, et al [1965], showed this to be a remarkably good approximation, even for \( K \) and \( p \) fairly small, although it seems to give better results when \( \sqrt{\frac{2K\mu}{h}} \geq 1.5\mu \). Moreover, the total expected cost is typically quite flat near \( D^* \), so if \( S^* \) is well-approximated, the approximation for \( D^* \) does not need to be especially accurate to have a total cost close to the cost associated with the optimal policy.
Actually, we use the Power Approximation discussed in Mosier [1981] for the optimal $D$. He reports empirical investigations that indicate that this approximation is superior to the economic order quantity. As in the case of the economic order quantity, this approximation tends to be poor when it is smaller than $1.5\mu$. For the derivation of our algorithm, we assume $D_i$ is at least $\mu_i$, and thus use the approximation

$$D_i = \max \left\{ \mu_i, 1.3\mu_i \cdot 0.494 \left( \frac{K_i}{h_i} \right) \cdot 0.506 \left[ 1 + \left( \frac{\sigma_i^*}{\mu_i} \right)^2 \right] \cdot 116 \right\}. \tag{5}$$

We note that asymptotically (as $\frac{K_i}{h_i} \to \infty$) this approximation differs very little from the economic order quantity.

To deal with the third difficulty, we intended to assume that the demand distribution in Normal, since Wagner, et al [1965], Wagner [1975, ch. 19], MacCormick [1974], Estey and Kauffman [1975], MacCormick et al [1977], and MacCormick [1977] all suggest that a Normal distribution gives good approximations to the optimal policies even when the true distribution is skewed (By a good approximation we mean that the expected total cost of the approximation policy is nearly the same as for an optimal policy). But because the Normal cdf cannot be expressed in a closed form, we use a Logistic distribution to approximate the Normal distribution. It is an especially simple and accurate approximation to the Normal distribution, as discussed in Johnson and Kotz [1970, ch. 22].
They show that if $\Phi^*$ is the cdf of a Normal random variable with mean $\mu^*$ and standard deviation $\sigma^*$, then

$$
\Phi^*(x) = \frac{1}{1 + \exp(-\gamma(x - \mu^*))},
$$

where

$$
\gamma = \frac{15\pi}{16\sqrt{3}\sigma^*}.
$$

This is the standard Logistic distribution with parameters 0 and $\frac{1}{\gamma}$.

We introduce the change of variables

$$
\theta_i = e^{-\gamma_i(S_i - \mu^*)},
$$

where

$$
\gamma_i = \frac{15\pi}{16\sqrt{3}\sigma_i^*},
$$

and reformulate the NLP using the approximations discussed. The result is given in Theorem 2 and (17). Since the transformation

$$
\Theta = (\theta_1, \ldots, \theta_N)
$$

is one-to-one, infinitely differentiable, and invertible everywhere, the problem as formulated in $\Theta$ is equivalent to the problem as formulated in $S$. We note that $\theta_i > 0$ for any finite $S_i$. It is clear from the formulation of the problem that the optimal $S$ is finite, and so the optimal $\Theta$ is positive. Thus we do not need to add the constraint $\Theta_i \geq 0$.

**Theorem 2.** If the approximations (2) through (6) and the change of variables (7) are used in the expressions for $C_i(S_i, D_i)$ and $P_i(S_i, D_i)$, and if we denote by $C_i(\theta_i)$ and $P_i(\theta_i)$ the resulting approximations (without policy-independent terms and a function in $C_i$ that is $O(1)$ as
S.\to \infty), we have

(i) \[ C_i(e_i) = \frac{h_i}{\gamma_i} \log \left[ \frac{(1+\theta_i)\rho_i}{\theta_i} \right] \]

(ii) \[ p_i(\theta_i) = (1-\rho_i) \left( 1 + \frac{1}{\gamma_i D_i} \log \left( \frac{1+\theta_i}{1+\theta_i^D} \right) \right) + \frac{\rho_i}{1+\theta_i} , \]

where \( \delta_i = e^{\gamma_i D_i} \).

Proof. We suppress the subscript \( i \).

(i) If the approximations (2) through (5) are used in the expression for \( C(S,D) \), and \( C(S) \) denotes the resulting approximation without policy-independent terms, then we have

\[ C(S) = \rho \left[ \int_D \int_0^{S-y} (S-y-x)\phi^*(x) \frac{1-\rho}{DD} \, dx \, dy + h \int_{-\infty}^S (S-x)\phi^*(x) \, dx \right] - (\text{policy-independent terms}). \]

Now

\[ \int_0^D \int_{-\infty}^{S-y} (S-y-x)\phi^*(x) \, dx \, dy = \int_0^D \int_{-\infty}^{\infty} (S-y-x)\phi^*(x) \, dx \, dy - \int_0^D \int_{S-y}^\infty (S-y-x)\phi^*(x) \, dx \, dy, \quad (8) \]

provided these integrals exist.

We show that they exist by evaluating them. First,

\[ \int_0^D \int_{-\infty}^{\infty} (S-y-x)\phi^*(x) \, dx \, dy = \int_0^D (S-y) \left[ \int_{-\infty}^\infty \phi^*(x) \, dx \right] \, dy - \int_0^D \left[ \int_0^\infty x\phi^*(x) \, dx \right] \, dy \]

\[ = \int_0^D (S-y) \, dy - \int_0^D \mu \, dy \]

\[ = DS - D \left( \mu^* + \frac{D}{2} \right). \]
To evaluate the other integral in (8) we define

$$
e(S) = - \int \int_{D} (S-y-x) \phi^*(x) dx dy
$$

$$= \int \int_{D} (x+y-S) \phi^*(x) dx dy.$$

It is straightforward to verify that on the domain of integration

$$0 < (x+y-S) \phi^*(x) < x \phi^*(x).$$

Since \( \mu^* < \infty \) and \( D < \infty \), we have

$$\int \int_{D} \int_{-\infty}^{\infty} |x \phi^*(x)| dx dy < \infty,$$

and so the Dominated Convergence Theorem (Hoffman [1975, p. 331]) justifies taking the derivative of \( \epsilon \) under the integral. This gives

$$\epsilon'(S) = \int \int_{D} \phi^*(x) dx dy
$$

$$= -\int \int_{0}^{D} [1-\Xi^*(S-y)] dy
$$

$$\geq -D \max_{0 \leq y \leq D} [1-\Xi^*(S-y)]
$$

$$= -D [1-\Xi^*(S-D)].$$

Obviously \( \epsilon'(S) \leq 0 \), so

$$0 > \lim_{S \to \infty} \epsilon'(S) \geq \lim_{S \to \infty} [-D(1-\Xi^*(S-D))] = 0,$$

which implies \( \lim_{S \to \infty} \epsilon'(S) = 0 \). Thus we have \( \epsilon(S) = a + o(1) \) as \( S \to \infty \), for some constant \( a \).

By assumption we do not include the \( o(1) \) function in the approximate cost function, and so we have that
\[ C(S) = h(1-p)S + h \rho \int_{-\infty}^{S} (S-x) \phi^*(x) \, dx, \quad (9) \]

where we have dropped all policy-independent terms. Using the approximation (6), we have that

\[
\int_{-\infty}^{S} (S-x) \phi^*(x) \, dx = \int_{-\infty}^{S} \frac{S-x}{1+e^{-\gamma(x-\mu^*)}} \, dx.
\]

After the change of variables \( x = e^{-\gamma(x-\mu^*)} \), this becomes

\[
\int_{\theta}^{\infty} \frac{(S-\mu^*) + \frac{1}{Y} \log X}{(1+X)^2} \, dX, \quad (10)
\]

where \( \theta \) is given by (7). Then

\[
\int_{\theta}^{\infty} \frac{dX}{(1+X)^2} = \frac{1}{1+\theta}, \quad (11)
\]

and

\[
\int_{\theta}^{\infty} \frac{\log X}{(1+X)^2} \, dX = -\frac{\log \theta}{1+\theta} + \log \theta - \log(1+\theta), \quad (12)
\]

and so (7), (9), (10), (11), (12) and the definition of \( C(\theta) \) give

\[
C(\theta) = h(1-p)(\mu^* - \frac{1}{Y} \log \theta) - \frac{h \rho}{Y} \log + \frac{h \rho}{Y} \log(1+\theta) - (\text{policy-independent terms})
\]

\[
= \frac{h \rho}{Y} \log(1+\theta) - \frac{h}{Y} \log \theta
\]

\[
= \frac{h}{Y} \log \left( \frac{(1+\theta)^{\rho}}{\theta} \right). \]

(ii) If the approximations (2) through (5) are used in the expression for \( P(S,D) \), and \( P(S) \) denotes the resulting approximation, then we have
\[ P(S) = \rho \left\{ \int_0^D \xi^*(S-y) \frac{1-p}{DD} \, dy + \xi^*(S) \right\} \]
\[ = \frac{1-p}{D} \int_0^D \xi^*(S-y) \, dy + \rho \xi^*(S). \]  \hspace{1cm} (13)

Using approximation (6) we have

\[ \int_0^D \xi^*(S-y) \, dy = \int_0^D \frac{dy}{1-e^{-\gamma(S-y-\mu^*)}}. \]

Making the change of variables \( Y = e^{-\gamma(S-y-\mu^*)} \), this becomes

\[ \frac{1}{\gamma} \int_{-\gamma(S-\mu^*)}^{1+e^{-\gamma}(S-y-\mu^*)} \frac{dY}{1+e^{-\gamma}(S-y-\mu^*)}. \]

Then

\[ \int \frac{dY}{1+e^{-\gamma}(S-y-\mu^*)} = Y - \log(1+e^{-\gamma}(S-y-\mu^*)), \]  \hspace{1cm} (15)

so (7), (13), (14), (15) and the definition of \( P(\theta) \) give

\[ P(\theta) = \frac{1-p}{\gamma D} \left[ \gamma D + \log\left( \frac{1+\theta}{1+\delta\theta} \right) \right] + \frac{\rho}{1+\theta} \]
\[ = (1-p) \left[ 1 + \frac{1}{\gamma D} \log\left( \frac{1+\theta}{1+\delta\theta} \right) \right] + \frac{\rho}{1+\theta} \square. \]

Let

\[ \theta_i^u = e^{-\gamma_i(D_i-\mu_i^*)}, \]  \hspace{1cm} (16)

so \( S_i \geq D_i \) is equivalent to \( \theta_i \leq \theta_i^u \). Then we have formulated the following NLP, the solution to which gives an approximately optimal policy for our multi-item inventory system via the change of variables (7):
\[
\text{minimize } C(\theta) = \sum_{i=1}^{N} C_i(\theta_i) \\
\text{subject to } P(\theta) = \sum_{i=1}^{N} W_i[P_i(\theta_i) - \alpha] \geq 0,
\]
\[
\theta_i \in \Theta_i, i=1,\ldots,N.
\]

We use nonlinear programming techniques to solve (17). Specifically, we derive first-order necessary conditions for optimality using the familiar Kuhn-Tucker conditions. Although the objective function is convex, the feasible region in general is not. Therefore the necessary conditions may not be sufficient to guarantee even local optimality (Luenberger [1973, Sections 6.4, 10.6]). We show that for \(D_i\) sufficiently large, however, the Kuhn-Tucker conditions are sufficient for (at least) local optimality. We also describe a computationally efficient algorithm for finding an optimal policy when it exists, and compare the performance of these policies with policies generated by an existing method.

The following formulas are needed.

Lemma 3. For \(i=1,\ldots,N,\)

(a) \(C_i'(\theta_i) = \frac{h_i}{\gamma_i} \left[ \frac{1+(1-\rho_i)\theta_i}{(1+\theta_i)\theta_i} \right] \)

(b) \(C_i''(\theta_i) = \frac{h_i}{\gamma_i} \left[ \frac{(1-\rho_i)\theta_i^2 + 2\theta_i + 1}{(1+\theta_i)^2\theta_i^2} \right] \)

(c) \(P_i'(\theta_i) = -\frac{[D_i\rho_i\gamma_i + (1-\rho_i)(\delta_i-1)] + [D_i\rho_i\gamma_i\delta_i + (1-\rho_i)(\delta_i-1)]\theta_i}{\gamma_iD_i(1+\theta_i)^2(1+\delta_i\theta_i)} \)
Proof (a) Differentiating \( C_i'(\theta_i) \) and suppressing \( i \) gives,

\[
C'(\theta) = \frac{h}{Y} \left[ \frac{\rho}{1+\theta} - \frac{1}{\theta} \right] - h \frac{1+(1-\phi)\theta}{(1+\theta)^2} .
\]  

\[(18)\]

(b) Differentiating \( C''(\theta) \) gives

\[
C''(\theta) = \frac{h}{Y} \left[ \frac{\phi}{-(1+\theta)^2} + \frac{1}{\theta^2} \right]
= \frac{h}{Y} \left[ \frac{(1-\rho)^2+2\theta+1}{(1+\theta)^2} \right] .
\]

\[(19)\]

(c) Differentiating \( P_i(\theta_i) \) and suppressing \( i \) gives,

\[
p'(\theta) = \frac{(1-\phi)}{\gamma D} \left[ \frac{1}{1+\theta} - \frac{\delta}{1+\delta \theta} \right] - \frac{\rho}{(1+\theta)^2}
= \frac{(1-\phi)(1-\delta)}{\gamma D(1+\theta)(1+\delta \theta)} - \frac{\rho}{(1+\theta)^2} .
\]

\[\frac{(1-\phi)(\delta-1)(1+\theta) + D\phi(1+\delta \theta)}{\gamma D(1+\theta)^2(1+\delta \theta)}
= \frac{[D\phi + (1-\phi)(\delta-1)] + [D\phi \delta + (1-\phi)(\delta-1)]\theta}{\gamma D(1+\theta)^2(1+\delta \theta)} .
\]

\[
\theta^* = \theta^u \text{ is globally optimal for (17)}.
\]

\[\square\]

### Theorem 4

Suppose

\[
P(\theta^u) \geq 0 .
\]

(20)

Then \( \theta^* = \theta^u \) is globally optimal for (17).

**Proof.** It is straightforward to verify (since \( \phi_i \) is decreasing as a function of \( D_i \)) that (making use of (5))

\[
D_i \geq \mu_i \text{ implies } 0 < \phi_i < \frac{2}{3}.
\]

(21)

This with Lemma 3(a) implies that

\[
C_i'(\theta_i) < 0 \text{ whenever } \theta_i \leq \theta_i^u.
\]

(22)
and so $C(\theta)$ is decreasing in each argument for $\theta_i < \theta_i^U$. But by hypothesis $\theta = \theta^U$ is feasible, so $\theta = \theta^U$ is a global minimum of (17). \]

In order to investigate the case when $P(\theta^U) < 0$, we make use of the Kuhn-Tucker necessary and sufficient conditions for optimality. To use these, however, one must verify that some constraint qualification holds at the optimal point. We show that if $P(\theta^U) < 0$, then every feasible point is a regular point, that is, the gradients of the active constraints at the point are linearly-independent. Luenberger [1973, 223-227] shows that this constraint qualification is sufficient to guarantee that if the Kuhn-Tucker necessary and sufficient conditions hold at a point, then that point is an (at least local) optimum.

Lemma 5. If $P(\theta^U) < 0$, then any $\theta$ feasible for (17) is a regular point of (17), that is, the gradients (at $\theta$) of the constraints active at $\theta$ are all linearly-independent.

Proof. Let $\theta$ be feasible for (17). Without loss of generality, suppose that of the constraints $\theta_i \leq \theta_i^U$, $i=1,\ldots,N$, the first $j$ are active and the last $N-j$ are inactive. Since by assumption $\theta = \theta^U$ is not feasible, we have $N-j \geq 1$. Let $T$ denote the matrix which has as its columns the gradients (at $\theta$) of the constraints active at $\theta$. If the constraint $P(\theta) \geq 0$ is inactive, then

$$T = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \\ 0 & \cdots & 0 \end{bmatrix}.$$ 

which certainly has linearly-independent columns. If the constraint $P(\theta) \geq 0$ is active, then
Certainly the first $j$ columns of $T$ are linearly-independent. If the last column were a linear combination of the first $j$, then (since $N-j \leq 1$) $W_N \cdot P_N' \cdot (\theta_N) = 0$. Now it is straightforward to verify that $D_i \geq 0$ implies $\delta_i > 1$.

so this with Lemma 3(c) and (21) implies that $P_N'(\theta_N) > 0$.

And since $W_N > 0$, we have a contradiction, and so the columns of $T$ are linearly-independent, which implies that $\theta$ is a regular point of (17) $\Box$.

**Theorem 6.** Suppose $P(e^u) < 0$ and for $i = 1, \ldots, N$, let

$$
\begin{align*}
A_i &= A_i(D_i) = D_i h_i \\
B_i &= B_i(D_i) = D_i h_i [2 + \delta_i - \rho_i] \\
E_i &= E_i(D_i) = D_i h_i [\delta_i + (1 - \phi_i)(\delta_i + 1)] \\
F_i &= F_i(D_i) = D_i h_i (1 - \phi_i) \delta_i \\
G_i &= G_i(D_i) = W_i [D_i \phi_i \gamma_i + (1 - \phi_i)(\delta_i - 1)] \\
H_i &= H_i(D_i) = W_i [D_i \phi_i \gamma_i \delta_i + (1 - \phi_i)(\delta_i - 1)]
\end{align*}
$$

Then the first-order Kuhn-Tucker necessary condition that $\theta^*$ be a minimum of (17) is the following:

$$
\lambda_i(\theta_i) = \frac{A_i + B_i \theta_i + E_i \theta_i^2 + F_i \theta_i^3}{G_i \theta_i^2 + H_i \theta_i^2}.
$$
There exists a real number $\lambda^* \geq 0$ such that

\begin{align}
(a) & \quad \lambda_i'(\theta_i^*) = \lambda^* \quad \text{when } \theta_i^* < \theta_i^u \\
& \quad \lambda_i'(\theta_i^*) \geq \lambda^* \quad \text{when } \theta_i^* = \theta_i^u
\end{align}

(b) $P(\theta^*) = 0$.

**Proof.** As proved in Luenberger [1973, pp.232-234], given the constraint qualification guaranteed in Lemma 5, a first-order necessary condition for $\theta^*$ to be a minimum of (17) is that $\theta^*$ be feasible for (17), and that there exists $\lambda^* \geq 0$ and $\xi^* = (\xi_1^*, \ldots, \xi_N^*) \geq 0$ such that

\begin{align}
\nabla L(\theta^*, \lambda^*, \xi^*) & = \nabla C(\theta^*) - \lambda^* \nabla P(\theta^*) - \xi^* I = 0 \\
\lambda^* P(\theta^*) & = 0 \\
\xi^* (\theta^* - \theta^u) & = 0,
\end{align}

where $I$ is the $N\times N$ identity matrix, and a superscript $T$ denotes transpose.

By assumption $P(\theta^u) < 0$, and clearly

\begin{equation}
P(0) = \sum_{i=1}^{N} W_i (1-\alpha) > 0;
\end{equation}

since it cannot cost less to give a higher service-level, we can assume that at optimality $P(\theta^*) = 0$. Now Lemma 3(c), (21) and (23) imply that for all $i$,

\begin{equation}
P_i'(\theta_i) = 0 \quad \text{whenever } \theta_i = \theta_i^u,
\end{equation}

and so by (22) we have that (25) is equivalent to the following condition:

There exists $\lambda^* \geq 0$ such that for all $i$,
We note that $\zeta_i^* = C_i'(\theta_i^*) - \lambda^* P_i'(\theta_i^*)$, for each $i$. To complete the proof, we need only to show

$$\lambda_i(e_i^*) = \frac{C_i'(\theta_i^*)}{P_i'\theta_i^*)}.$$  

We suppress the subscript $i$. By Lemma 3(a) and (c),

$$\frac{C_i'(\theta)}{WP_i'(\theta)} = \left(-\frac{h}{\gamma}\right)\left[\frac{1+(1-\rho)\theta}{(1+\theta)\theta}\right](-\gamma D)\left[\frac{(1+\theta)^2(1+5\theta)}{G+H\theta}\right].$$

After factoring $(1+\theta)$ out of the numerator and denominator, the numerator becomes $Dh[1+(1-\rho)\theta][1+(\sigma+1)\theta+6\theta^2]$, which, when multiplied out, is $A+B\theta+E\theta^2+F\theta^3$, and the denominator is $G+H\theta^2$. Thus we have

$$\frac{C_i'(\theta_i)}{P_i'\theta_i^*)} = \lambda_i(\theta_i),$$

as was to be shown $\square$.

Although the first-order Kuhn-Tucker condition is necessary for (at least local) optimality, in general it is not sufficient. In Theorem 7 we give a condition sufficient to guarantee that the Hessian of the Lagrangian of (17) is positive-definite, a strong second-order sufficient condition (see Luenberger [1973, p.235]). In Theorem 8 we give conditions in terms of $D_i$ which guarantee that the hypotheses of Theorem 7 are satisfied.

**Theorem 7.** Suppose $P(e^U)<0$ and that there exists $\lambda^* > 0$ so that the necessary conditions (24) hold at $\theta^*$. If also $\lambda_i(\theta_i)$ is decreasing
for $\theta_i < \theta_i^u$, then the Hessian matrix $\nabla_{\theta} L(\theta^*, \lambda^*, \xi^*)$ is positive-definite, which implies that (24) is sufficient for (at least local) optimality.

Proof. We first note that Lemma 3(b) and (21) imply that

$$C_i''(\theta_i) > 0.$$  \hfill (30)

We have

$$\nabla_{\theta} L(\theta^*, \lambda^*, \xi^*) = \nabla^2 C(\theta^*) - \lambda^* \nabla^2 P(\theta^*)$$

$$= \begin{bmatrix}
C_1''(\theta_1^*) - \lambda^* W_1 P_1''(\theta_1^*) & 0 \\
0 & \ddots \\
0 & 0 & C_N''(\theta_N^*) - \lambda^* W_N P_N''(\theta_N^*)
\end{bmatrix}.$$  \hfill (31)

Let $Q_i$ denote the $i^{th}$ diagonal entry of $\nabla_{\theta} L$. We show that $Q_i > 0$ for each $i$, as this implies that $\nabla_{\theta} L$ is positive-definite. If $P_i''(\theta_i^*) < 0$, (30) implies $Q_i > 0$, since $\lambda^*, W_i > 0$. If $P_i''(\theta_i^*) > 0$, (24) implies $Q_i \geq C_i''(\theta_i^*) - [\lambda^*_i(\theta_i^*)] W_i P_i''(\theta_i^*)$, which by (29) is

$$Q_i \geq C_i''(\theta_i^*) - \frac{C_i'(\theta_i^*)}{P_i''(\theta_i^*)} P_i''(\theta_i^*),$$

or

$$Q_i \geq \frac{P_i''(\theta_i^*) C_i''(\theta_i^*) - C_i'(\theta_i^*) P_i''(\theta_i^*)}{P_i''(\theta_i^*)}.$$  \hfill (31)

By (27) the denominator is negative, so we can conclude the proof by showing that the numerator is negative. From (29) we have

$$\lambda_i'(\theta_i^*) = \frac{P_i''(\theta_i^*) C_i''(\theta_i^*) - C_i'(\theta_i^*) P_i''(\theta_i^*)}{W_i [P_i''(\theta_i^*)]^2}.$$  \hfill (31)

By hypothesis $0 > \lambda_i'(\theta_i^*)$, so the numerator of (31) is negative. Since $W_i > 0$, we have $Q_i > 0$, as was to be shown.
Luenberger [1973, p.235] proves that given the constraint qualifications guaranteed in Lemma 5, the first-order necessary conditions (24) are sufficient for (at least local) optimality provided \( \nabla_{\theta} L(\theta^*, \lambda^*, \varsigma^*) \) is positive-definite on the subspace tangent to the constraints tight at \( \theta^* \). We have shown that \( \nabla_{\theta} L(\theta^*, \lambda^*, \varsigma^*) \) is positive-definite, which concludes the proof. We note that to prove this theorem for more general functions \( C_i \) and \( P_i \) it is sufficient that \( C_i \) be convex, and \( P_i \) and \( \lambda_i \) be decreasing.

**Theorem 8.** For \( i=1, \ldots, N \), if \( D_i \) is sufficiently large, then \( \lambda_i(\theta_i) \) is convex and decreasing when \( \theta_i \leq \theta_i^u \).

**Proof.** As usual we suppress \( i \). We write \( \lambda(\theta) \) in partial fraction form making differentiation easier. We then show that for \( D \) large, the coefficients of \( \lambda(\theta) \) are such that \( \lambda'(\theta) < 0 \) for \( \theta \leq \theta^u \). By long division,

\[
\lambda(\theta) = \frac{F\theta}{H} + \frac{EH-FG}{H^2} + \frac{\left( BH^2 - G(EH-FG) \right)}{H^2 \theta(G+H\theta)} A.
\]

For any \( X \),

\[
\frac{X\theta + A}{\theta(G+H\theta)} = \frac{A}{G\theta} + \frac{XG-AH}{G(G+H\theta)},
\]

and so

\[
\lambda(\theta) = \frac{F\theta}{H} + \frac{EH-FG}{H^2} + \frac{A}{G\theta} + \frac{n}{GH^2(G+H\theta)},
\]

where

\[
n = GH^2 - G^2(EH-FG) - AH^3.
\]

By (21) and (23) we have

\[
A, B, E, F, G, H > 0,
\]

so certainly \( \lambda(\theta) > 0 \) whenever \( \theta > 0 \) (the change of variables (7) is such that \( \theta > 0 \)). Differentiating (32) gives
\[ \lambda'(\Theta) = \left( \frac{F}{H} - \frac{A}{G\Theta^2} \right) - \frac{\eta}{GH(G+H\Theta)^2} \]

and

\[ \lambda''(\Theta) = \frac{2A}{G\Theta^3} + \frac{2\eta}{G(G+H\Theta)^3}. \]

We first show that \( \lambda''(\Theta) > 0 \) for \( D \) sufficiently large by showing \( \eta > 0 \) for \( D \) sufficiently large (since (33) implies all other quantities are positive). We use the following notation.

\[ f(x) \sim g(x) \text{ if } \lim_{x \to \infty} \frac{f(x)}{g(x)} = 1. \]

To examine the behavior of \( \eta \) for large \( D \), we examine the behavior of \( A, B, E, F, G \) and \( H \) for large \( D \). The following are easily inferred from their definitions:

\[ \begin{align*}
\lim_{D \to \infty} \delta &= \infty, \\
\lim_{D \to \infty} \Theta &= 0, \\
\lim_{D \to \infty} D\eta &= \mu.
\end{align*} \]  

(34)
They imply

\begin{align*}
A & \sim Dh \\
B & \sim Dh\delta \\
E & \sim 2Dh\delta \\
F & \sim Dh\delta \\
G & \sim W\delta \\
H & \sim W(\mu + 1)\delta
\end{align*}

and so \( n \sim (W\delta)(Dh\delta) W^2(\mu + 1)^2 \delta^2 - W^2 \delta^2 [(2Dh\delta)(W(\mu + 1)\delta) - (Dh\delta)(W\delta)] - (Dh)(W^3(\mu + 1)^3 \delta^3) \).

The limiting behavior of \( n \) is determined by its highest order term, which is \( D\delta^4 \), so

\[
n \sim \left[ W^3 h(\mu + 1)^2 - 2W^3 h(\mu + 1) + W^3 h \right] D\delta^4 = W^3 h(\mu)^2 D\delta^4.
\]

Since \( W^3 h(\mu)^2 > 0 \), \( (34) \) implies \( \lim_{D \to \infty} W^3 h(\mu)^2 D\delta^4 = \infty \), so \( \lim n = \infty \) also.

Thus, for \( D \) sufficiently large, \( n > 0 \), and so \( \lambda''(0) > 0 \), as was to be shown. Hence \( \lambda(0) \) is convex. To show that \( \lambda'(0) < 0 \) for \( \theta \leq \theta^u \), it is sufficient to show \( \lambda'(\theta^u) < 0 \). We have

\[
\lambda'(\theta^u) = \xi - \frac{\eta}{GH(\theta^u + H\theta)^2},
\]

where

\[
\xi = \frac{F}{H} - \frac{A}{G(\theta^u)^2}.
\]

Since for \( \eta > 0 \) large \( D \), we need only show that \( \xi < 0 \) for large \( D \) to conclude \( \lambda'(\theta) < 0 \) for \( \theta \leq \theta^u \). Relations (35) imply that

\[
\xi \sim \frac{Dh\delta}{W(\mu + 1)\delta} - \frac{Dh}{W\delta e^{2\mu^*}}.
\]
Relations (34) imply the quantity in parentheses tends to \(-\infty\) as \(D \to \infty\), and so

\[
\lim_{D \to \infty} \xi = \lim_{D \to \infty} \frac{Dh}{w} \left( \frac{1}{\sqrt{\gamma + 1}} - \frac{\delta}{e^{\gamma \mu^*}} \right) = -\infty.
\]

Thus, \(\xi < 0\) for \(D\) sufficiently large, and so the proof is complete \(\square\).

Thus, Theorems 7 and 8 imply that if \(D_i \) is large enough that \(\lambda_i\) is decreasing, then the first-order conditions (24) are sufficient for (at least local) optimality. We exploit this in our algorithm, which is described in the next section.

3. GKD ALGORITHM

Our algorithm is based upon local duality theory of nonlinear programming (see Luenberger [1973, pp.312-320]). Suppose \(x^*\) is a local minimum (at which the constraint qualification described in Lemma 5 holds) of the nonlinear program

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad g(x) = 0,
\end{align*}
\]

with associated Kuhn-Tucker constraint multiplier \(\lambda^*\). Writing the Lagrangian \(L(x, \lambda) = f(x) - \lambda g(x)\), the NLP dual to (36) is defined as

\[
\begin{align*}
\text{maximize} & \quad \psi(\lambda), \\
\lambda
\end{align*}
\]

where

\[
\psi(\lambda) = \min_{x \text{ near } x^*} L(x, \lambda).
\]

(We mention that (36) and (37) can be written, respectively, as

\[
\min_{x} \max_{\lambda} L(x, \lambda) \quad \text{and} \quad \max_{\lambda} \min_{x} L(x, \lambda),
\]

which illustrates their dual nature).
If $L(x, \lambda^*)$ is twice continuously-differentiable and convex at $x^*$ (and hence convex in a neighborhood of $x^*$), then the following can be shown to be true:

1. $\psi(\lambda)$ is twice continuously-differentiable and convex at $\lambda^*$.
2. $\lambda^*$ uniquely solves (37).
3. If we denote by $x(\lambda)$ the (unique) value of $x$ minimizing the right-hand side of $\psi(\lambda)$ in (37), then $x^* = x(\lambda^*)$.

Thus rather than solving (36) one could solve (37), and do this by solving

$$\nabla \psi(\lambda) = 0,$$

which can be shown to be equivalent to solving

$$g(x(\lambda)) = 0.$$ (38)

Generally, this approach is not useful computationally because the function $x(\lambda)$ is often difficult to compute. This method may be quite efficient, however, if $x(\lambda)$ is reasonably easy to compute and the dimension of $\lambda$ is smaller than that of $x$. One problem with such structure is the "Generalized Knapsack Problem," which is

\[
\begin{align*}
\text{minimize } f(x) &= \sum_{i=1}^{N} f_i(x_i) \\
\text{subject to } g(x) &= \sum_{i=1}^{N} g_i(x_i) = 0.
\end{align*}
\]

Clearly (17) is a Generalized Knapsack Problem with upper-bounded variables. These bounds do not change the essential structure of the problem or its solution by solving its dual, and the details are left to Theorem 9.

For (39), $x(\lambda)$ is found by solving the system of equations

$$\nabla_x L(x, \lambda) = 0.$$
where
\[ \lambda_i^L = \lambda_i^L(\theta_i^U). \]

For convenience, we assume that the items are reordered, if necessary, so that
\[ \lambda_1^L \geq \lambda_2^L \geq \ldots \geq \lambda_N^L. \] (41)

Theorem 9. If either \( P(e^U) \geq 0 \) or all \( D_i \) are sufficiently large so that the hypotheses of Theorem 8 hold, then the following algorithm is well-defined and converges to a (possibly local) minimum of (17).

GKD Algorithm.

1. If \( P(\theta^U) > 0 \) set \( i = N \), and go to step (6). Otherwise set \( i = 0 \).
2. Set \( \lambda^L = \lambda_{i+1}^L = \max \{ \lambda_i^L, \ldots, \lambda_N^L \} \), by (41).
3. Define the function \( P_i \) on \([\lambda^L, +\infty)\) by
   \[ P_i(\lambda) = P(\theta_1^U, \ldots, \theta_i^U, \lambda_{i+1}(\lambda), \ldots, \lambda_N(\lambda)). \]
4. If \( P_i(\lambda^L) \leq 0 \), go to step (5).
   Otherwise set \( i = i + 1 \), go to step (2).
5. Solve \( P_i(\lambda) = 0 \) for \( \lambda^* \in [\lambda^L, \infty) \).
6. Set \( \theta_j^* = \begin{cases} \theta_j^U & j = 1, \ldots, i \\ \lambda_{i+1}(\lambda^*) & j = i + 1, \ldots, N \end{cases} 
   s_j^* = P_j^* - \frac{1}{\theta_j^*} \log \theta_j^*
   S_j^* = s_j^* - D_j.

Proof. Steps (1) and (2) are certainly well-defined. By definition of \( \lambda^L \), \( \lambda \geq \lambda^L \) is in the domain of \( t_{i+1}(\lambda), \ldots, t_N(\lambda) \), so these are all well-defined. Hence so is \( P(\lambda) \), and so step (3) us well-defined. Step (4) is well-defined, except for the possibility that \( i \) may become larger than \( N \). But if \( i = N \), then we have \( P(\theta_1^U, \ldots, \theta_N^U) > 0 \), which is equivalent to \( P(\theta^U) > 0 \). But step (1) guarantees \( P(\theta^U) < 0 \), so \( i \) will never exceed \( N \).
Step (5) requires justification. We show that $\hat{P}_i'(\lambda)>0$ on $[\lambda^L,+\infty)$ and $\hat{P}_i(+\infty)>0$. These with step (4) imply that there is a unique solution in $[\lambda^L,+\infty)$ to $\hat{P}_i(\lambda)=0$. We have

$$\hat{P}_i'(\lambda) = P(\theta_1^u,\ldots,\theta_i^u, t_{i+1}(\lambda),\ldots,t_N(\lambda)),$$

so

$$\hat{P}_i'(\lambda) = \sum_{j=1}^N W_j \hat{P}_j'(\theta_j) t_j'(\lambda), \text{ where } \theta_j = t_j(\lambda),$$

by the chain rule. Relation (27) implies that $\hat{P}_j'(\theta_j)<0$, and clearly $t_j'(\lambda)<0$, so indeed $\hat{P}_i'(\lambda)>0$. Stage $i$ is the first time that $P_i(\lambda^L)<0$. If $i=0$, then

$$\hat{P}_i(\infty) = P(\theta_1^u,\ldots,\theta_i^u, t_{i+1}(\infty),\ldots,t_N(\infty))$$

$$= P(0,\ldots,0) > 0,$$

by (26). And if $i \geq 1$, then $\hat{P}_{i-1}(\lambda^L) > 0$, so

$$0 < \hat{P}_{i-1}(\lambda^L) = P(\theta_1^u,\ldots,\theta_{i-1}^u, t_i,\ldots,t_N)$$

$$= P(\theta_1^u,\ldots,\theta_i^u, t_{i+1},\ldots,t_N),$$

since at stage $i-1$, $\lambda^L=\lambda_{i-1}^L$, and $t(\lambda_i^L) = \theta_i^u$.

Hence, $0 < P(\theta_1^u,\ldots,\theta_i^u, t_{i+1},\ldots,t_N)$.

We have

$$\hat{P}_i(\infty) = P(\theta_1^u,\ldots,\theta_i^u, t_{i+1}(\infty),\ldots,t_N(\infty))$$

$$= P(\theta_1^u,\ldots,\theta_i^u, 0,\ldots,0).$$

By (27) $P$ is decreasing in each argument, so $\hat{P}_i(\infty) > \hat{P}_{i-1}(\lambda^L) > 0$, as was to be shown.

Step (6) and (7) are well-defined, so the entire algorithm is verified. If $P(\theta^u)>0$ at step (1), then by Theorem 4, $\theta^u=\theta^u$ is globally optimal for (17). If $P(\theta^u)<0$ at step (1), the fact that the functions $\lambda_j$ and $t_j$ are inverses with the fact that $\lambda_j^L=\lambda_j(\theta_j^u)$ imply that the
algorithm terminates with $0 < \theta^* \leq \theta^U$ and $\lambda^* \geq 0$ satisfying

(a) $\lambda_j(\theta^j_*) = \lambda^L_j, \quad j=1,\ldots,i$

(b) $P(\theta^*) = 0$.

But (41) implies $\lambda_j(\theta^j_*) \geq \lambda^L_j$ for $j=1,\ldots,i$, and certainly $\lambda^L_j \geq \lambda^*$. Moreover, since $\theta^j_*=\theta^j_U$, for $j=1,\ldots,N$, $\lambda^*$ satisfies (24). By Theorems 6, 7 and 8, these equations imply the local optimality of $\theta^*$ for (17).

Recall that we reformulated the multi-item inventory problem using the one-to-one change of variables given by (7). Our GKD Algorithm finds an optimal $\theta^*$ for (17), and this gives rise to an optimal $S^*$ via the inverse of the change of variables (17), which is

$$S_i = \mu_i^* - \frac{\log \theta_i}{\gamma_i},$$

as given in step (6) □.

The algorithm is computationally efficient in that it only requires solving $\hat{P}_1(\lambda)=0$, where $\hat{P}_1(\lambda)$ is a real-valued function of one real variable. The computation of the functions $t_j(\lambda)$ must, however, be done numerically (the details are in the Appendix to this chapter).

Moreover, the hypotheses of Theorem 8 are easy to verify and guarantee that the algorithm is well-defined and converges to a minimum of (17).

The proof of Theorem 8 shows that it is sufficient to check that $\lambda_1''(\theta^U_i) < 0$ for each $i$. The extension of the algorithm when $\lambda_1''(\theta^U_i) \geq 0$ is discussed in Section 5.
4. POLICY PERFORMANCE

In order to illustrate the performance of the (s,S) policies generated by our algorithm, we performed the following test. We examined fifteen 2-item inventory systems with the parameter settings

\[ K_1 = K_2 = 32 \]
\[ k_1 = k_2 = 0 \]
\[ h_1 = 0.05, \quad \frac{h_2}{h_1} = 1, 10, 25, 50, 100 \]
\[ \mu_1 = 3, \quad \frac{\mu_2}{\mu_1} = \frac{1}{3}, 1, 3 \]
\[ \frac{\sigma_1^2}{\mu_1} = \frac{\sigma_2^2}{\mu_2} = 9 \]
\[ \sigma_1^2 = \sigma_2^2 \]
\[ \sigma_1^2 = \sigma_2^2 \]
\[ W_1 = W_2 = \frac{1}{2} \]

We chose these parameter settings for three reasons

1. Most of them give rise to large D (much larger than 1.5μ). Since our approximations and theorems generally hold for large D, it should perform well on these systems.

2. These parameters are realistic, in that they reflect costs associated with many actual inventory systems.

3. As in Mitchell [1982], our algorithm should perform effectively when there are both "expensive" and "inexpensive" items in the system. This holds for those systems we considered with \( \frac{h_2}{h_1} \) large.

The comparison we used assumes the underlying distribution is Negative Binomial (the discrete version of a Gamma distribution, which is skewed for our parameter settings), specifies a service-level α, and uses the Veinott-Hagner exact algorithm (Veinott
and Wagner [1965]) to compute optimal policies for both items in each system using the same service-level $\alpha$. We ran our algorithm using $\alpha$. Then we compared the holding plus set-up costs generated by the Veinott-Wagner algorithm and our algorithm. The results are displayed in Table 1.

We make the following observations.

1. The service-level $\alpha$ is not the same for all runs, but varies from .89 to .97. The average service-level for the fifteen runs is .93. There are two reasons for this. First, for some of these items, the policy with the smallest service-level, $(0,D)$, has a very high service-level, so that the system-wide service-level is high. For other items, the smallest service-level is small. It turns out that to specify a uniform service-level is to specify $\alpha=.97$ (since one of the systems has this as the smallest service-level for both items), which is unreasonably large. The second reason is that the Logistic approximation we use in our algorithm does not always generate policies with service-level $\alpha$ when the actual distribution is Negative Binomial. However, preliminary tests on larger systems (32 items) seem to indicate that this problem is less severe for larger systems.

2. Our approximation for $D_i$ is almost exact for every single item. This suggests that the asymptotic approximations we are using should be fairly accurate. For all items $D_i$ is sufficiently large to guarantee that the hypotheses of Theorem 7 are satisfied, so all policies are (possibly local) minima (not saddle points).
3. Over half the systems had an item with $s < 0$ for the policy generated by the Veinott-Wagner algorithm. Our algorithm can only consider policies with $s \geq 0$, but its policies still compared very favorably against those generated by the Veinott-Wagner algorithm.

4. As expected, there was a cost reduction for all systems, except the ones with identical holding costs. Moreover, the greatest cost decreases were for those systems with $\frac{h_2}{h_1}$ large.

This initial investigation demonstrates that inventory-operating costs can be reduced significantly when a system-wide service level is specified, rather than specifying an identical service-level for each item.
Table 1

Percent decrease in holding + set-up costs

Assumptions:

\[ k_1 = k_2 = 32, \quad k_1 = k_2 = 0, \quad h_1 = .05 \]

\[ \frac{\sigma_1^2}{\mu_1} = \frac{\sigma_2^2}{\mu_2} = 9, \quad \mu_1 \cdot \mu_2 = \frac{1}{2}, \quad \nu_1 = 3 \]

All demands are Negative Binomial.
5. GKD ALGORITHM MODIFICATION WHEN $D_i$ IS SMALL

Theorem 8 of Section 2 contains the most general condition under which we were able to prove (see Theorem 7) that the GKD Algorithm converges to a (possibly local) minimum of (17). This condition is that for all items $i=1,...,N$ the functions $\lambda_i(\theta_i)$ are decreasing on $(0, \theta_i^U)$. We give our recommendations when this condition may not hold for some items in the inventory system.

Recall from the proof of Theorem 8 that $\lambda_i(\theta_i)$ is convex on $(0, \theta_i^U)$ when $\eta_i>0$ and is decreasing on $(0, \theta_i^U)$ when both $\eta_i>0$ and $\xi_i<0$. Moreover, $\lim_{D_i \to \infty} \eta_i = \infty$ and $\lim_{D_i \to \infty} \xi_i = -\infty$, so $\lambda_i(\theta_i)$ is decreasing on $(0, \theta_i^U)$ when $D_i$ is sufficiently large. The GKD Algorithm (given in Theorem 9) exploits this property by assuming the existence of the inverse function of $\lambda_i$, which is denoted by $t_i$.

The question is what to do when $D_i$ is not large enough to guarantee $\eta_i>0$ and $\xi_i<0$. In either case it is impossible to guarantee that $\lambda_i$ is decreasing, and hence invertible, so step (3) of the GKD Algorithm may not be well-defined. The algorithm may converge, but there is no guarantee that it will converge to a local minimum.

We recommend the following modifications when $\eta_i<0$ or $\xi_i>0$. If $\xi_i>0$ but $\eta_i>0$, then $\lambda_i$ is still convex but not decreasing on $(0, \theta_i^U)$. It is always true (and easy to verify) that $\lambda_i'(0)=-\infty$, so this case is illustrated in Figure 1.

In this case we recommend that the domain of $\lambda_i(\theta_i)$ be shortened to $(0, \theta_i^U)$, where $\theta_i^U$ is the largest value of $\theta_i$ for which $\lambda_i(\theta_i)$ is decreasing. This has the effect of introducing the additional constraint $S_i \geq S_i^L$, where $S_i^L > D_i$. Since the $S_i$ are lower bounded anyway by $D_i$, ...
this constraint does not change the structure of the problem at all. In particular, the GKD Algorithm will still converge to a (possibly local) minimum.

\[ \hat{\theta}_i^u \]

Figure 1

Computationally, \( \hat{\theta}_i^u \) can be found by increasing the given lower bound \( D_i \) on \( S_i \) a unit at a time until \( \hat{\theta}_i^u \) is found. This procedure is described in detail at the end of this section, and computational experience on typical 32-item inventory systems is reported in the next chapter.

Although we have in this case a recommendation that still assures that the GKD Algorithm will converge, the case when \( \eta_i < 0 \) poses greater difficulty. The \( \lambda_i \) may not be convex, in which case we have been unable to find a modification of the problem that guarantees convergence to a local minimum. We essentially ignore this case and use the algorithm as it is (making the recommended modification when \( \epsilon_i > 0 \)). Our computational experience reported in the next chapter seems to indicate that this is not a problem in many cases. Since \( \eta_i \) is easy to compute prior to using the GKD Algorithm, we recommend doing so in order to be
alerted to possible problems when \( n_i < 0 \).

In light of all this, the only recommended change involves the computation of \( \Theta_i^U \). The GKD Algorithm is just as given in Theorem 9, with the following step inserted between steps (1) and (2):

\begin{enumerate}
\item For \( j = 1, \ldots, N \):
\begin{align*}
\text{Set } S_j^L &= D_j \\
(1.1.1) &\quad \text{Set } \Theta_j^U = e^{-\gamma_j(S_j^L - \mu_j^*)}.
\end{align*}
\item If \( \lambda_j^U(\Theta_j^U) < 0 \), go to step \((1.1.2)\).
\item Otherwise, set \( S_j^L = S_j^L + 1 \).
\item Go to step \((1.1.1)\).
\end{enumerate}

\begin{enumerate}
\item Next \( j \).
\end{enumerate}

6. CONCLUSIONS

We have formulated a constrained NLP to solve an \((s,S)\) inventory model with a service-level constraint rather than shortage costs. This model has the advantage that a manager can more easily specify service objectives than shortage costs. We derived necessary and sufficient conditions for an optimal \((s,S)\) policy. Using asymptotic approximations (where \( D = S - s \) is large) and assuming Logistically distributed demands, we derived a computationally efficient algorithm to find approximately optimal \((s,S)\) policies. We showed that for some inventory systems there is a substantial cost savings when using this algorithm instead of the Independent and Identical Service approach that is often used in practice. We also gave recommendations to modify our algorithm when \( D \) is not sufficiently large to guarantee its convergence.
<table>
<thead>
<tr>
<th>NOTATION</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_i$</td>
<td>18</td>
</tr>
<tr>
<td>$c_i$</td>
<td>18</td>
</tr>
<tr>
<td>$k_i$</td>
<td>18</td>
</tr>
<tr>
<td>$h_i$</td>
<td>18</td>
</tr>
<tr>
<td>$X_i$</td>
<td>19</td>
</tr>
<tr>
<td>$\phi_i$</td>
<td>19</td>
</tr>
<tr>
<td>$\mu_i$</td>
<td>19</td>
</tr>
<tr>
<td>$\sigma_i$</td>
<td>19</td>
</tr>
<tr>
<td>$N$</td>
<td>19</td>
</tr>
<tr>
<td>$W_i$</td>
<td>19</td>
</tr>
<tr>
<td>$a$</td>
<td>19</td>
</tr>
<tr>
<td>$(s,S)$</td>
<td>19</td>
</tr>
<tr>
<td>$D_i$</td>
<td>20</td>
</tr>
<tr>
<td>$X_i^*$</td>
<td>20</td>
</tr>
<tr>
<td>$\phi_i^*$</td>
<td>20</td>
</tr>
<tr>
<td>$\mu_i^*$</td>
<td>20</td>
</tr>
<tr>
<td>$\sigma_i^*$</td>
<td>20</td>
</tr>
<tr>
<td>$C_i(S_i,D_i)$</td>
<td>21</td>
</tr>
<tr>
<td>$P_i(S_i,D_i)$</td>
<td>21</td>
</tr>
<tr>
<td>$M_i$</td>
<td>21</td>
</tr>
<tr>
<td>$m_i$</td>
<td>21</td>
</tr>
<tr>
<td>$C(S,D)$</td>
<td>22</td>
</tr>
<tr>
<td>$P(S,D)$</td>
<td>22</td>
</tr>
<tr>
<td>Symbol</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------</td>
<td>------</td>
</tr>
<tr>
<td>$\rho_i$</td>
<td>22</td>
</tr>
<tr>
<td>$\theta_i$</td>
<td>25</td>
</tr>
<tr>
<td>$\gamma_i$</td>
<td>25</td>
</tr>
<tr>
<td>$C_i(\theta_i)$</td>
<td>25</td>
</tr>
<tr>
<td>$P_i(\theta_i)$</td>
<td>25</td>
</tr>
<tr>
<td>$\delta_i$</td>
<td>26</td>
</tr>
<tr>
<td>$\theta_i^u$</td>
<td>29</td>
</tr>
<tr>
<td>$C(\theta)$</td>
<td>30</td>
</tr>
<tr>
<td>$P(\theta)$</td>
<td>30</td>
</tr>
<tr>
<td>$A_i$</td>
<td>33</td>
</tr>
<tr>
<td>$B_i$</td>
<td>33</td>
</tr>
<tr>
<td>$E_i$</td>
<td>33</td>
</tr>
<tr>
<td>$F_i$</td>
<td>33</td>
</tr>
<tr>
<td>$G_i$</td>
<td>33</td>
</tr>
<tr>
<td>$H_i$</td>
<td>33</td>
</tr>
<tr>
<td>$\lambda_i(\theta_i)$</td>
<td>33</td>
</tr>
<tr>
<td>$L(\theta^<em>,\lambda^</em>,\zeta^*)$</td>
<td>34</td>
</tr>
<tr>
<td>$t_i(\lambda_i)$</td>
<td>42</td>
</tr>
<tr>
<td>$\lambda_i^L$</td>
<td>42</td>
</tr>
<tr>
<td>$\lambda^L$</td>
<td>43</td>
</tr>
<tr>
<td>$\hat{P}_i(\lambda)$</td>
<td>43</td>
</tr>
</tbody>
</table>
its unit holding cost \( h \). This system is described in Figure 2, where the items are ordered in decreasing order by \( h_i \). Essentially the first seven items are high-value items (have a very high annual dollar demand), and the other twenty-five items are low-valued items.

As a base-case, we specified inventory parameters as follows. For \( i = 1, \ldots, N \),

\[
K_i = 24 \\
\mu_i = \frac{1}{32} \\
k_i = 4 \\
\frac{\sigma_i^2}{\mu_i} = 9
\]

and

\[
\mu_1, \mu_5, \mu_9, \ldots, \mu_{29} = 16 \\
\mu_2, \mu_6, \mu_{10}, \ldots, \mu_{30} = 8 \\
\mu_3, \mu_7, \mu_{11}, \ldots, \mu_{31} = 4 \\
\mu_4, \mu_8, \mu_{12}, \ldots, \mu_{32} = 2.
\]

Of course, \( h_i \) is determined by \( \mu_i \) from the value \( (h \mu)_i = h_i \mu_i \).

We are interested primarily in the algorithm performance at a service-level of 85\%, one managers often prefer. Our criterion of comparison is to assume that the demand distribution is Negative Binomial and compare the total expected cost associated with the \((s,S)\) policies that the GKD Algorithm produces with the total expected cost of the policies given by optimally setting all the \((s,S)\) policies so that every item has a service-level of 85\%. We call this the Identical Service Approach, and it is an approach often used by managers.

We intended to use the Veinott-Wagner [1965] Algorithm (also Kaufman
to compute the optimal policies, but the amount of computation required was impractical because the optimal policies often had $S_i$ too large.

Instead we used the Power Approximation of Ehrhardt [1976] and [1979] to compute approximately optimal policies. We chose this approximation because extensive numerical investigations in Ehrhardt [1976], and Klincewicz [1976a] and [1976b] showed this to be both the most accurate and most easily computed approximately optimal policy available. We use the version of the Power Approximation given in Mosier [1981] because of the technical improvements made. The Power Approximation requires the specification of shortage costs rather than the service-level. Letting $\alpha$ denote the service-level specified for the inventory system (using the Identical Service Approach), we specified the shortage costs $p_i$ using the expression (see Ehrhardt [1977, p.8])

$$\alpha = \frac{p_i + 0.0695h_i}{p_i + h_i}.$$  

This expression is an empirical improvement over the well-known relationship (see Veinott and Wagner [1965])

$$\alpha = \frac{p_i}{p_i + h_i}.$$  

Because both the GKD Algorithm and the Power Approximation are approximations, it was impossible to realize the exact service-level specified. We therefore specified various service-levels, and interpolated intermediate service-levels. Since we used the Power Approximation in the GKD Algorithm to compute the values for $D$, we
only compared the expected holding costs for this algorithm and the Power Approximation, the expected replenishment costs being identical (Typically the expected replenishment cost accounts for about half of the total expected cost). The result of this experiment is displayed in Figure 3. There is a significant decrease in expected holding costs at and near a service-level of 85%.

In order to investigate the sensitivity of expected holding costs to the various parameters, we performed four additional experiments in which we modified one parameter (or parameter group) at a time in our base-case model. The same experimental design was used for these experiments as for the base-case.

Figure 4 describes the result when the means were changed to

\[ \mu_1, \mu_3, \ldots, \mu_3 = 16 \]
\[ \mu_2, \mu_4, \ldots, \mu_32 = 8. \]

As can be seen, there was a very slight degradation in the GKD Algorithm performance. The cost savings are still substantial.

Figure 5 describes the result when the base-case was altered so that

\[ k_i = 8, \; i = 1, \ldots, 32. \]

Again there was only a slight degradation in cost.

Figure 6 is the base-case with

\[ \frac{\sigma_i^2}{\mu_i} = 3, \; i = 1, \ldots, 32. \]

The cost degradation is greater than in previous experiments, but the cost savings remains substantial.
Figure 7 describes the change

\[ \nu_1 \cdot \nu_2 \cdots \cdot \nu_{16} = 16 \]
\[ \nu_{17} \cdot \nu_{18} \cdots \cdot \nu_{32} = 8. \]

The cost savings are still substantial, but not as great as in previous experiments.

As a final experiment, we did a worst-case type experiment, in which all of the following changes were made:

\[ k_i = 8 \]
\[ \frac{\sigma_i^2}{\nu_i} = 3 \]

\[ \nu_1, \ldots, \nu_{16} = 16 \]
\[ \nu_{17}, \ldots, \nu_{32} = 8. \]

This experiment is described in Figure 8. The degradation is substantial, but the cost savings are still large. The behavior of the GKD Algorithm policies for service-levels larger than about 87%, where the total cost is greater, seem to be caused by the error in approximating the Negative Binomial demand by a Logistic distribution (Section 2 of the previous chapter). We address this issue at the end of this chapter.

We make the following general observations concerning the performance of the GKD Algorithm for these 32-item inventory systems.

1. The total holding cost savings are substantial at and near a service level of 85%. The algorithm is an improvement over the Identical Service Approach often used by managers, and significantly reduces the overall
cost of maintaining an inventory system subject to a service-level constraint.

2. In most cases the policies produced by the GKD Algorithm degrade sharply around a service-level of 86% or 87%, as is evidenced by the sudden sharp increase of the holding cost. This suggests that the GKD Algorithm may not produce substantial cost savings when a high overall service-level is specified.

3. The GKD Algorithm seems to perform best when there is substantial variation in demands means, when the leadtime is not too large, and when the variance to mean is not too small. The degradation in performance is due to the error in approximating a Negative Binomial demand distribution by a Logistic distribution. This is especially a problem when there are certain patterns in the demand means and large leadtimes. This issue is discussed in detail in the next chapter.

3. SINGLE-ITEM SERVICE-LEVELS

For the six experiments described above, we also examined the single-item service-levels of the policies produced by the GKD Algorithm. For each inventory system we chose that experiment that had the average service-level closest to 85% and plotted the items (in decreasing order of their annual dollar demand, \( h_i \)) against their optimal service-level. The results are displayed in Figures 9, 10, 11, 12, 13 and 14. We make the following observations concerning them.

1. Typically the items in the inventory system are skewed, in
that the seven high-value items \( (i=1, \ldots, 7) \) have a service-level considerably below 85\%, and most of the twenty-five low-value items have a service-level considerably above 85\%. In some cases there are a few items with service-levels near 85\%.

2. There is an obvious negative correlation between the value \( h_i \mu_i \) and the service-level for item \( i \). This indicates that low-value items should be stocked high and high-value items should be stocked much lower, rather than stocking all items the same.

3. There is an obvious cyclical pattern corresponding to the cyclical pattern of the means. For the high-value items, the service-levels are higher for those items with smaller means, and for the low-value items, the service-levels are lower for those items with the smaller means. This suggests that it may be possible to well-approximate the ordering of the items by their service-levels using only \( h_i \mu_i \) and \( \mu_i \).

4. ITEMS WITH \( D_i \) TOO SMALL

The discussion in Section 5 of the previous chapter indicates that a sufficient, but not necessary, condition that the GKD Algorithm converge to a (possibly local) minimum (17) is that \( \xi_i < 0 \) and \( \eta_i > 0 \). These are guaranteed if \( D_i \) is sufficiently large. If \( D_i \) is not large enough to guarantee that \( \xi_i < 0 \), it is sufficient to in effect add the additional constraint \( S_{i-1} \leq S_i \) in order to still guarantee that the algorithm will converge to a local minimum. If \( \eta_i > 0 \), no recommendation could be given to guarantee algorithm convergence to a local minimum. In this chapter we report on those items in the six
inventory systems examined that violate either $\xi_i < 0$ or $\eta_i > 0$.

Interestingly enough, there is not a single item in any inventory system with $\eta_i < 0$, and so the GKD Algorithm converged to a (possibly local) minimum in every experiment. Each system contained items with $\xi_i > 0$, however, necessitating adding constraints $S_i \geq S_i^L$, as discussed in the previous chapter. This is reported in Table 2.
5. CONCLUSIONS

We have found that for inventory systems typical of ones found in the real-world, there is a significant cost savings when the GKD Algorithm is used rather than the Identical Service Approach often used by managers. There are two issues that still need to be addressed.

First, the behavior observed in Figure 8 (the identical service approach performed better than the GKD Algorithm) needs to be better understood. This is especially necessary in order that practical recommendations may be given as to what types of inventory systems one can expect substantial cost reductions when using the GKD Algorithm rather than the identical-service approach. Moreover, insights into the behavior observed in Figure 8 may suggest modifications to the GKD Algorithm that would enable it to perform better in situations like those which prevail in Figure 8. This issue is addressed in the next chapter.

Second, methods of large-scale inventory aggregation need to be explored so that the GKD Algorithm can be used in an inventory system with perhaps many thousands of items. Only then can the algorithm be of practical value. This issue is also addressed in the next chapter.
32 items, $k_i = 24$, $W_i = \frac{7}{32}$, $k_i = 4$,

$\beta_1 = 9$, $\nu_i: 16, 8, 4, 2, ... 16, 8, 4, 2$

$D_i$ using Power Method

**Figure 3**
Figure 4

- 32 items, \( K = 24, W_i = \frac{1}{32}, k_i = 4 \)
- \( \frac{\sigma_i}{\mu_i} = 9, \mu_i: 16, 8 \ldots, 16, 8 \)
- Using Power Method

Expected holding cost vs. service level graph.
Power (identical service)

32 items, \( k_i = 24, \ W_i = \frac{1}{32}, \ k_i = 8 \)

\( \frac{a_k^2}{\mu_1} = 9, \mu: 16, 8, 4, 2, \ldots, 16, 8, 4, 2 \)

D_1 using Power Method

Figure 5
expected holding cost vs. service-level

32 items, \( k_s = 24, W_i = \frac{1}{32}, k_i = 4 \)

\[
\sigma^2_i = 3, \mu_i = 16, 8, 4, 2, \ldots, 16, 8, 4, 2
\]

Di using Power Method

Figure 6
32 items, $K_1 = 24$, $W_1 = \frac{1}{32}$, $k_1 = 4$,

$\sigma_{i}^2 = 9$, $\mu_{i} = 16, \ldots, 16, 8, \ldots, 8$

$D_1$ using Power Method

Figure 7
optimal CKD service-level

$K_i = 24, \; K_j = \frac{1}{32}$, Average Service-Level = .856, $k_i = 4$,

$\frac{O_i^2}{\mu_i} = 9$, $\mu_i$: 16, 8, 4, 2, 16, 8, 4, 2

Diagram Figure 9
optimal GKD service-level

item (decreasing order by \( h_i \))

\[ K_i = 24, \ W_i = \frac{1}{32}, \ \text{Average Service-Level} = 0.854, \ k_i = 4, \]

\[ \frac{\alpha_i}{\mu_i} = 9, \ \mu_i: 16, 8, \ldots, 16, 8 \]

D, using Power Method

Figure 10
optimal SKD service-level

item (decreasing order by hμ)

K₁=24, W₁=1/32, Average Service-Level = .852, k₄=4,

μ₁ = 9, μ₂: 16, 8, 4, 2, ..., 16, 8, 4, 2

D₁ using Power Method
Figure 2: Optimal GKO service-level

- $K_1 = 24, w_1 = \frac{1}{32}$
- Average Service-level = 0.850, $k_1 = 4$

- $\sigma_1^2 = 3, \mu_i = 16, \ldots, 16, 8, \ldots, 8$
- $D_i$ using Power Method

(decreasing order by $h_\mu$)
optimal GKD service-level

item (decreasing order by $\mu_i$)

$K_i = 24, W_i = \frac{1}{32}$, Average Service-level = .851, $k_i = 4$,

$\frac{\sigma_i^2}{\mu_i} = 9, \mu_i: 16, 16, 8, 8$

$D_i$ using Power Method

Figure 13
Figure 14
D. using Power Method

\[ x_1^2 = \frac{1}{3}, x_2 = 3, x_3 = 4, \ldots, 16, 8, \ldots. \]

Item (decreasing order of unit)

Average Service Level = 0.851, \( k_i = 8, \ldots. \)

Optimal GKGD service-level
### Table 2

<table>
<thead>
<tr>
<th>Parameters, $i=1,...,32$</th>
<th>Figure with Cost Function</th>
<th>Figure with Service-levels</th>
<th># items with $c_i &gt; 0$</th>
<th>Items with $c &gt; 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_i$</td>
<td>$\theta_i^2$</td>
<td>$\nu_i, ..., \nu_{32}$</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>16, 8, 4, 2, ...</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>16, 8, ...</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>16, 8, 4, 2, ...</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>16, 8, 4, 2, ...</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>16, ..., 8, ...</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>16, ..., 8, ...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For all systems: $k_i = 24, i=1,...,32$.

$w_i = \frac{1}{32}, i=1,...,32$. 


IV. SENSITIVITY EXPERIENCE WITH THE GKD ALGORITHM

1. INTRODUCTION

In Chapter II we considered the problem of specifying single-item service objectives in a multi-item \((s,S)\) inventory system subject to a system-wide service objective. We formulated this problem as a constrained NLP and developed the Generalized Knapsack Duality (GKD) Algorithm to solve it. In Chapter III we reported our computational experience with several 32-item inventory systems with a structure typical of many real-world inventory systems. In this chapter we report sensitivity experience with the base-case and worst-case inventory systems used in Chapter III. The criterion of comparison is to compare the expected holding cost (computed exactly) associated with the \((s,S)\) policies produced by the GKD Algorithm for a given system-wide service level \(\alpha\) with the expected holding cost of the policies given by the Identical Service Method, in which all the \((s,S)\) policies are set by the Power Approximation given in Mosier [1981] so that every item has a service-level \(\alpha\) (as discussed in Chapter III the exact algorithm of Veinott and Wagner [1965] is computationally unwieldy for our inventory system, and the Power Approximation is the most accurate approximation available). In computing the expected costs we assume the underlying demand distribution is Negative Binomial, and we are primarily interested in the sensitivity of the GKD Algorithm at service-levels of 85\% and 88\%. We are interested in 85\% because this is a service-level managers often
prefer, and we are interested in 88% because of the difficulties reported in Chapter III with the worst-case inventory system at this service-level.

In Section 2 of this chapter we report sensitivity experience for $D_i = S_i - s_i$, which is a constant input parameter to the GKD Algorithm. In Section 3 we report sensitivity experience for the lower bounds on $S_i$, which are input to the GKD Algorithm, but may be increased by the GKD Algorithm (Chapter II Section 5). In Section 4 we consider several sampling schemes for large-scale inventory aggregation, where a sample is used with the GKD Algorithm to specify $(s_i, S_i)$ policies for every item in the inventory system. In Section 5 we investigate the sensitivity of these sampling schemes to the number of items in the scheme, and in Section 6 we offer conclusions and recommendations concerning specifying the $D_i$, the lower bounds on $S_i$ and the size and type of inventory scheme to use for large-scale inventory aggregation. All figures are at the end of the chapter.

2. SENSITIVITY EXPERIENCE FOR $D_i$

Recall from Section 2 of Chapter II that $D_i = S_i - s_i$ is approximated by a constant (the Power Approximation of $D_i$) which is input to the GKD Algorithm. This is justified because in the single-item case the optimal $D_i$ is asymptotically independent of the service-level. For any given item $i$ in the inventory system, however, the policy $(s_i, S_i)$ produced by the GKD Algorithm with associated service-level $a_i$ may not be an optimal policy when the item $i$ is considered as a single-item
inventory system. And for a non-optimal policy \((s_i, S_i)\), \(D_i\) is not necessarily independent of the service-level \(\alpha_i\).

Thus we investigated the impact of varying the input values of the \(D_i\) to determine if the GKD Algorithm performance degraded (or perhaps even improved). As stated in the Introduction, we used the 32-item base-case inventory system of Chapter III. This system has the structure that twenty percent of the items constitute eighty percent of the total value (where value is taken to be \(h_i\)) of the inventory system. The parameters are as follows for \(i=1,\ldots,N\):

\[
\begin{align*}
K_i &= 24 \\
W_i &= \frac{1}{32} \\
k_i &= 4 \\
\sigma_i &= 9 \\
\frac{2}{\nu_i} & \text{and} \\
\nu_1, \nu_5, \nu_9, \ldots, \nu_{29} &= 16 \\
\nu_2, \nu_6, \nu_{10}, \ldots, \nu_{30} &= 8 \\
\nu_3, \nu_7, \nu_{11}, \ldots, \nu_{31} &= 4 \\
\nu_4, \nu_8, \nu_{12}, \ldots, \nu_{32} &= 2.
\end{align*}
\]

The \(h_i\) are determined through the value \(h_i \nu_i\). Additional details are given in Chapter III.

Because of the structure of the inventory system, the items 1-25 are low-value items and items 26-32 are high-value items. Typically, as discussed in Section 3 of Chapter III, the GKD Algorithm produces policies so that the service-levels of the high-value items are very low and the
service-levels of the low-value items are very high. These observations suggest two experiments.

In the first, we increased the value of $D_i$ (given by the Power Approximation) 25% for the low-value items and decreased the value of $D_i$ 25% for the high-value items. In the second experiment, we decreased the value of $D_i$ by 25% for the low-value items and increased it 25% for the high-value items.

These are severe tests of robustness. In the experiments in Chapter III we compared the expected holding costs of the GKD Algorithm policies and the Power Approximation policies (which give identical service to all items) because, since we used the Power Approximation $D_i$ in the GKD Algorithm, the expected replenishment costs are the same. Such is not the case in the experiments described in this section, so we compare the total expected costs, namely, the expected holding plus replenishment costs. These experiments and their outcomes are summarized in Table 3 and Figures 15, 16 and 17. In Experiment 2, for example, in which the Power Approximation $D_i$ is increased 25% for the low-value items and decreased 25% for the high-value items, the expected total cost for the GKD policies at a service-level of 85% is approximately 198, and the expected total costs for the Identical Service Method at that same service-level is approximately 276. This represents a 28% decrease in expected cost when using the GKD policies rather than the Identical Service policies.
Table 3. Sensitivity Experience for $D_i$: Comparison of GKD Algorithm Policies with Identical Service Policies.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Low-Value Items (1-25)</th>
<th>High-Value Items (26-32)</th>
<th>% + in cost at Service-Level 85%</th>
<th>% + in cost at Service-Level 88%</th>
<th>See Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No change</td>
<td>No change</td>
<td>29</td>
<td>26</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>$D_i + 25%$</td>
<td>$D_i + 25%$</td>
<td>26</td>
<td>22</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>$D_i + 25%$</td>
<td>$D_i + 25%$</td>
<td>28</td>
<td>27</td>
<td>17</td>
</tr>
</tbody>
</table>

We make the following observations and conclusions:

(1) The change in the performance of the GKD Algorithm in Experiment 2 and 3 is slight, especially considering the severity of the test. This suggests that not only is an especially accurate specification of $D$ unnecessary for a single-item inventory system, it is also unnecessary for a multi-item inventory system of the general structure of our base-case. Of course, additional experiments on variations of this base-case would help confirm or refute this conjecture.

(2) Although it may be possible to specify the $D_i$ so that greater cost savings are realized than when using the Power Approximation $D_i$, we recommend using the Power Approximation, since obvious ways of changing $D_i$ do not improve the GKD Algorithm performance.
3. SENSITIVITY EXPERIENCE FOR LOWER BOUNDS ON $S_i$

Recall from Chapter II that we required $S_i \geq D_i$ in the GKD Algorithm with that lower bound on $S_i$ possibly being increased by the algorithm to guarantee convergence to a (possibly local) minimum (see Section 5). The main reason given to restrict attention to policies with $S_i \geq D_i$ is that (aside from the mathematical difficulties involved) many managers consider policies with $S_i < D_i$ (equivalently, $s_i < 0$) unsatisfactory for practical implementation. There are two reasons, however, that even higher lower bounds on $S_i$ might be imposed.

First, many managers prefer not only that $s_i \geq 0$, but that $s_i \geq \mu_i^* = (k_i + 1)\mu_i$, so the safety stock ($s_i$) is always at least large enough to cover the expected leadtime demand.

Second, in Chapter III the Identical Service Approach performed better than the GKD Algorithm in the worst-case experiment at a service-level at and near 88%.

We offer a conjecture to explain this behavior and show that by modifying the lower bounds on $S_i$ the GKD Algorithm will in the case just described perform considerably better than the Identical Service Approach at both service levels 85% and 88%.

The worst-case inventory system has the same 80/20 structure (twenty percent of the items represent eighty percent of the total system value) with the following parameters for $i=1,\ldots,N$:

\begin{align*}
K_i &= 24 \\
W_i &= \frac{1}{32} \\
k_i &= 8
\end{align*}
There is a notable difference in the single-item service levels determined by the GKD Algorithm at a service-level of only 85% for both the base-case and the worst-case inventory systems, as reported in Figures 9 and 14 of Chapter III, respectively. For the high-value items (items 1-7) in both cases the (s,S) policies given by the GKD Algorithm have all the S_i at their lower bounds. The service-levels associated with the high-value items for the worst-case are considerably lower than those for the base-case, and so the service-levels for the low-value items for the worst-case are considerably higher than those for the base-case. The GKD Algorithm assumes the demand distribution is Logistic, whereas in these experiments we assume the demand distribution is Negative Binomial. The approximation of a Negative Binomial distribution by a Logistic distribution is poorest in the tail, which corresponds to very high individual item service-levels. Thus we conjecture that the poor performance of the GKD Algorithm in the worst-case is due to the poorness of the Logistic demand approximation. We test this conjecture by imposing higher values of the lower bounds on the S_i. This forces the high-value items to be stocked at a higher service-level, so the low-value items are stocked at a lower service level, as in the base-case. We note that Figures 9 and 14 indicate that the higher the demand mean, the lower the service-level of the (s,S) policy with S at its lower bound.
perform as well at a service-level of 88% as it does at 91% with \( r = 1 \).

These experiments and their outcomes for the base-case are summarized in Table 6. The results are similar to those in Experiments 2, 3, and 4, and so we do not report the expected cost graphs.

Table 6. Sensitivity Experience for Lower Bounds on \( S_i \): Base-Case.
Comparison of GKD Algorithm Policies with Identical Service Policies.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>( r ), where ( S_i \geq D_i + r w_i )</th>
<th>Minimum Overall Service-Level</th>
<th>% + in cost at Service-Level 88%</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>.8</td>
<td>.860</td>
<td>33</td>
</tr>
<tr>
<td>7</td>
<td>.9</td>
<td>.885</td>
<td>--</td>
</tr>
</tbody>
</table>

These experiments and their outcomes for the worst-case are summarized in Table 7 and Figure 22. Since the results of these experiments are similar to each other we report the expected cost graph only for Experiment 6.
Table 7. Sensitivity Experience for Lower Bounds on $S_j$: Worst-Case
Comparison of GKD Algorithm Policies with Identical Service Policies.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Minimum $r$, where $S_j \geq D_i + r_i u_i^*$</th>
<th>Overall Service-Level</th>
<th>% in cost at Service-Level 85%</th>
<th>% in cost at Service-Level 88%</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>.8</td>
<td>.811</td>
<td>26</td>
<td>23</td>
<td>22</td>
</tr>
<tr>
<td>7</td>
<td>.9</td>
<td>.878</td>
<td>--</td>
<td>20</td>
<td>--</td>
</tr>
</tbody>
</table>

We make two additional observations:

(5) The previous remarks for the base-case also hold for $r = .8, .9$.

(6) For the worst-case system there is a dramatic improvement in algorithm performance when $r = .8$. In this case there is a substantial cost savings at both service-levels 85% and 88%.

We offer the following conclusions.

(1) The behavior at a service-level of 88% for the worst-case inventory system does seem to be caused (at least in part) by the poorness of approximating a Negative Binomial distribution by a Logistic distribution.

(2) The algorithm performs best when there is a lower bound on $S$ just large enough to prevent too many of the low-value items to be stocked extremely high. We recommend that a user experiment with
various lower bounds (in terms of the single input value $r$) to find the best one.

(3) As most managers prefer as high a lower bound as possible on $S$ (as long as average system service is not too large), we recommend specifying $r$ as large as possible to include the desired targeted service-level. Our experience with both the best and worst-case inventory systems suggests that this approach will yield a significant cost reduction over the identical service approach.

4. SENSITIVITY EXPERIENCE FOR SAMPLING SCHEMES FOR INVENTORY AGGREGATION

Many real-world inventory systems contain tens of thousands of items. Although the GKD Algorithm is quite efficient, it could not handle inventory systems of this size. This is not a problem unique to our algorithm, and most managers deal with it using inventory aggregation. This is any method of partitioning an inventory system into groups of items so that every item in the group is controlled essentially the same. A representative item is chosen from each group and used to specify $(s,S)$ policies for all items in its group.

Our method of aggregation exploits the special structure of our problem (as formulated as an NLP) and of the GKD Algorithm, and certain empirical observations in Chapter III. Recall from Section 3 of Chapter II that the GKD Algorithm exploits the "Knapsack" structure of the NLP by solving its dual NLP. The dual has only one variable, $\lambda$, and to find the optimal $S^*$ of the original NLP it is sufficient to find the optimal dual variable $\lambda^*$. This is reflected in the GKD Algorithm (Chapter II
Section 3) by solving \( \hat{P}(\lambda) = 0 \) for \( \lambda^* \), from which \( S^* \) is computed. Our method of aggregation is to use a subset of the inventory system to estimate \( \lambda^* \), and then compute \( S^* \) for all the items in the inventory system from this estimate.

The inventory system we constructed to test various aggregation schemes is a 126-item system with the same essential structure as the 32-item base-case system described in Section 2. In particular, twenty percent of the items (items 101-128) constitute eighty percent of the total value of the inventory system, and the parameters are as follows, for \( i = 1, 2, \ldots, 128 \):

\[
\begin{align*}
K_i &= 24 \\
W_i &= \frac{1}{128} \\
k_i &= 4 \\
\sigma_i^2 &= 9 \\
\mu_i &= 2
\end{align*}
\]

and

\[
\begin{align*}
\mu_1, \mu_5, \ldots, \mu_{125} &= 16 \\
\mu_2, \mu_6, \ldots, \mu_{126} &= 8 \\
\mu_3, \mu_7, \ldots, \mu_{127} &= 4 \\
\mu_4, \mu_8, \ldots, \mu_{128} &= 2.
\end{align*}
\]

To examine the cost savings in using the GKD Algorithm rather than the Identical Service Method (via the Power Approximation) described in Section 1, we ran the GKD Algorithm with the entire system. As described in Section 1, we compared the expected holding costs with those of the Identical Service Method, and the result is displayed in Figure 23.
divided by the total value of the sample from that strata. This was done for both strata, so each maintained its total value, thus preserving the 80/20 structure. And to maintain the given constraint weights \( W_i \), they were each recomputed as the total weight of the strata divided by the total weight of the sample from that strata.

Initially we tested three central nonrandom stratified uniform sampling schemes. If one wants to choose \( M \) items from a stratum consisting of the \( N \) items \( 1, 2, \ldots, N \) by such a scheme, then the items

\[
\frac{N}{2^M}, \frac{3N}{2^M}, \frac{5N}{2^M}, \ldots, \frac{(2^M-1)N}{2^M}
\]

are taken, where the fractions \( \frac{jN}{2^M} \) are rounded to the nearest integer. For this sample, the 80/20 value structure and the constraint-weight structure is maintained by multiplying \( h_i \) and \( K_i \) by \( \frac{\sum h_i \mu_i}{\sum h_i \mu_i} \) and \( W_i \) by \( \frac{N}{M} \times \frac{1}{j} \) for each item in the sample. The three schemes differed in the number of items taken from each of the two strata. They are described in Table 8. Notice that Scheme 1 is a nonstratified scheme that roughly takes every 4th item from the inventory system (since the means repeat themselves every four items, to take every 4th item would create an inventory system with every mean the same, which seemed objectionable). Scheme 2 creates an inventory subsystem of half high-value and half low-value items, and that Scheme 3 takes all the high-value items (it seems reasonable that \( \lambda^* \) is more sensitive to them than to the very low value items, and so this scheme seemed plausible).
Table 8. Central Nonrandom Stratified Uniform Sampling.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Number Chosen from Low-Value (1-100)</th>
<th>Number Chosen from High-Value (101-128)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>28</td>
</tr>
</tbody>
</table>

The experiment we performed to test these schemes is as follows:

For a variety of specified service-levels we ran the GKD Algorithm using the inventory subsystem (determined by the sampling scheme) to compute \( \hat{\lambda}^* \), the optimal multiplier for the subsystem. Each value of \( \hat{\lambda}^* \) gives rise to a service-level for the subsystem (\( \alpha_{32} \)) and for the entire system (\( \alpha_{128} \)), and to the expected holding cost at these service-levels (\( H_{32} \) and \( H_{128} \), respectively). The criterion of comparison is how close \( \alpha_{32} \) and \( \alpha_{128} \) and \( H_{32} \) and \( H_{128} \) (for a given \( \lambda \)) are. A "perfect" subsystem would give rise to \( \alpha_{32} = \alpha_{128}, H_{32} = H_{128} \), that is, its optimal multiplier \( \hat{\lambda}^* = \lambda^* \), the optimal multiplier for the entire system. The service-level graphs for the three schemes are given in Figures 24, 25 and 26, and the expected holding cost graphs are given in Figures 27, 28 and 29.

We make the following observations concerning these sampling schemes:

1. They are all good. Although each one gives rise to a misspecification of \( \alpha_{128} \) and \( H_{128} \), the total cost savings will still be
substantial over the Identical Service Approach (compare Figure 23). (2) Scheme 2 seems to be superior to Schemes 1 and 3. Scheme 1 has a preponderance of low-value items, Scheme 3 has a preponderance of high-value items, and Scheme 2 has the same number of each. It seems that the proportion of items that are high-value/low-value at least in part affects the performance of the scheme.

The efficiency of Scheme 2 may have something to do with the fact that the demand means of our base-case inventory system follow the pattern 2, 4, 8, 16, ..., 2, 4, 8, 16. To investigate this, we used a central random stratified uniform sampling scheme with 16 expensive and 16 inexpensive items (like Scheme 2). In other words, we chose 16 items uniformly randomly (using the IMSL Routine GGUD) from both the high-value and low-value items. We performed six such tests, one with a different initial seed. The details of the subsystems chosen can be found in the Appendix to this chapter, and the scheme itself, Scheme 4, is described in Table 9.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Number Chosen from Low-Value (1-100)</th>
<th>Number Chosen from High-Value (101-128)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

We performed the same experiment as we did for the first three schemes, and the results of this experiment for the first seed is
reported in Figures 30 and 31 (results for all of the seeds were similar). We make the following observations.

(1) As for the three nonrandom schemes, the random scheme is good.
(2) The random scheme is not as good as Scheme 2, the nonrandom scheme with the same proportion of high-value/low-value items as this scheme. It would appear that the sequence of demand means is important, and that further research in stratifying not only on the value but on the demand means could give even better sampling schemes.
(3) The method of sampling seems relatively unimportant as long as the 80/20 value structure of the inventory system is maintained.

We recommend, therefore, that the scheme be chosen on the basis of its ease in implementation. Perhaps the nonrandom Scheme 1 is easiest, as it requires only one stratum (it represents a central sampling of every 4th item).

As a final experiment we tested whether Scheme 4 with the seed of the first experiment was accurate when lower bounds are imposed on the $S_i$. We chose Scheme 4 because it gave rise to neither the best nor the worst performance among the schemes. We chose lower bounds corresponding to $r=.75$ (see Section 3) because this was the largest value of $r$ which allowed an overall service-level of at least 85%. This experiment is described in Table 10 and the outcome is reported in Figures 32, 33, and 34, where Figure 32 reports the expected cost when using all 128 items in the GKD Algorithm.
Table 10. Central Random Stratified Uniform Sampling: $S_i \geq D_i + 0.75\mu_i^*.$

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Number Chosen from Low-Value (1-100)</th>
<th>Number Chosen from High-Value (101-128)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

We make the following observations:

(1) There is a significant cost savings at service-levels 85% and 88%.

(2) The sampling scheme is very accurate. It seems that our previous conclusions about the accuracy of these sampling schemes also hold true when lower bounds are imposed on the $S_i$.

5. SENSITIVITY EXPERIENCE FOR THE SAMPLE SIZE FOR INVENTORY AGGREGATION

In order to investigate how the reliability of inventory sampling schemes depends on the size of the scheme, we constructed a 512-item base-case inventory system with the 80/20 value structure described in Section 2. Not only is this structure typical of many real-world inventory systems, but if the items in the system are ordered in descending order by their value ($h_i\mu_i$) and the cumulative percent of the items is plotted against the corresponding cumulative value, then the resulting graph is frequently well-approximated by a Lognormal cumulative distribution function (Peterson and Silver [1979, pp. 30-37]). We constructed
(Nonrandom) Uniform Sampling Scheme (compare Scheme 1 of the previous section). The last experiment (Scheme 4) of Section 4 suggests that the approach of using a sample to investigate the sensitivity of the system to $r$ is a reliable procedure. We considered $r=0, .6, .7, .8, .9, 1$, with a targeted service-level of 85% (a service-level popular with many managers), and we want to include for consideration service-levels 83% to 88%.

This experiment and its result is reported in Table 11 and Figure 35. Since we want to include for consideration a service-level of 83% we chose $r=.6$. The cost savings for $r=.6$ at a service-level of 85% is about 51%, which is very significant.

Table 11. 128-Item Central Nonrandom Uniform Sample from the 512-Item Base-Case System: Comparison of GKD Algorithm Policies with Identical Service Policies.

<table>
<thead>
<tr>
<th>$r$, where $S_i = D_i + r u_i^*$</th>
<th>Minimum Overall Service-Level</th>
<th>$% +$ in cost at Service-Level 85%</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&lt; .8100</td>
<td>50</td>
</tr>
<tr>
<td>.6</td>
<td>.8140</td>
<td>51</td>
</tr>
<tr>
<td>.7</td>
<td>.8439</td>
<td>47</td>
</tr>
<tr>
<td>.8</td>
<td>.8719</td>
<td>--</td>
</tr>
<tr>
<td>.9</td>
<td>.8951</td>
<td>--</td>
</tr>
<tr>
<td>1</td>
<td>.9163</td>
<td>--</td>
</tr>
</tbody>
</table>
As an additional check of the reliability of using this sampling method to choose r, we repeated the experiment on the entire 512-item inventory system. The results are summarized in Table 12 and Figure 36. The sampling method of choosing r is seen to be very reliable, as was the case in Section 4.

<table>
<thead>
<tr>
<th>r, where $S_i \geq D_i + r\mu_i^*$</th>
<th>Minimum Overall Service-Level</th>
<th>% in cost at Service-Level 85%</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&lt; .8100</td>
<td>49</td>
</tr>
<tr>
<td>.6</td>
<td>.8138</td>
<td>50</td>
</tr>
<tr>
<td>.7</td>
<td>.8441</td>
<td>46</td>
</tr>
<tr>
<td>.8</td>
<td>.8720</td>
<td>--</td>
</tr>
<tr>
<td>.9</td>
<td>.8961</td>
<td>--</td>
</tr>
<tr>
<td>1</td>
<td>.9165</td>
<td>--</td>
</tr>
</tbody>
</table>

There were two sampling schemes considered in Section 4 which seemed to perform the best. The first is the recommended scheme, the Central Nonrandom Uniform Sampling Scheme. The second is the Central Nonrandom
Stratified Uniform Sampling Scheme (compare Scheme 2) in which half the sample is taken from the high-value items and the other half is taken from the low-value items. Because of the 80/20 value structure of our 512-item base-case system, items 1-102 are high-value and items 103-512 are low-value. The sensitivity experimental design is identical to that described in Section 4, here with \( r = .6 \). We chose sample sizes of 32, 64 and 128 for both schemes described, and the six experiments are summarized in Table 13. For each sample size, the results of these experiments were similar for both the uniform and stratified uniform schemes, and so the comparative graphs are reported only for the Central Nonrandom Uniform Sampling Scheme (Figures 37 through 42).

**Table 13. Comparison of GKD Algorithm Policies with Identical Service Policies.**

Central Nonrandom Uniform Sampling from 512-Item Base-Case System, \( S_i \geq D_i + .6\mu_i \).

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Sample Size</th>
<th>Figure Reporting Service-Level</th>
<th>Figure Reporting Expected Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>37</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>38</td>
<td>41</td>
</tr>
<tr>
<td>3</td>
<td>128</td>
<td>39</td>
<td>42</td>
</tr>
</tbody>
</table>

Central Nonrandom Stratified Uniform Sampling from 512-Item Base-Case System, \( S_i \geq D_i + .6\mu_i \).

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Number from High-Value Items</th>
<th>Number from Low-Value Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>64</td>
</tr>
</tbody>
</table>
We make the following observations:

(1) **All six schemes are reliable enough for practical implementation.** The 128-item schemes are superior to the 64-item schemes which are superior to the 32-item schemes.

(2) There does not seem to be any real difference between the uniform and stratified schemes.

(3) An examination of the experiments in Section 5, in which 32-item samples were taken from a 128-item inventory system, indicates that the 32-item samples are just as reliable in the 512-item system as in the 128-item system. This is extremely significant because it strongly suggests that 32-item schemes are sufficiently reliable for practical implementation no matter how large the inventory system is, as long as it has the 80/20 value structure described.

(4) We recommend using the Central Uniform Sampling Scheme with around 32 items for any inventory system with an 80/20 value structure. Although larger samples may be more reliable, this scheme is reliable enough. Moreover, the computational work necessary to use the GKD Algorithm on such a sample is very small, making this scheme suitable for frequent use in a real-world setting.

6. CONCLUSIONS

In this chapter we performed sensitivity tests on the GKD Algorithm with inventory systems with a structure typical of real-world systems, that twenty percent of the items represent 80 percent of the value of the system. We concluded the following:
(1) The total cost savings are not affected very much by changes in $D_i$.

(2) Increasing the lower bounds on $S_i$ may improve the algorithm performance, especially when the high-value items also tend to have high expected demands. Even when increasing these lower bounds degrades the performance, however, the cost savings are substantial. We recommend increasing these lower bounds as high as possible (while still maintaining a cost savings below that of the Identical Service Method) as this approach is most consistent with managerial goals.

(3) Uniform sampling with about 32 items is a reliable and computationally efficient method of inventory aggregation, giving accurate predictions via the GKD Algorithm of overall inventory service and expected operating costs.
Figure 15. 32-Item Base - Case

$D_j$ using Power Method
Figure 16. 32-Item Base-Case

* $D_i + 25\%$ for Low-Value Items
* $D_i + 25\%$ for High-Value Items
Figure 17: 32-Item Base-Case

- $D_i + 25\%$ for Low-Value Items
- $D_i + 25\%$ for High-Value Items
Figure 18  32-Item Base-Case

\[ S_i \geq D_i + .25 \mu_i^* \]
Figure 19: 32-Item Base Case Service-Level

- Service-Level: .90, .91, .92, .93, .94
- Expected Holding Cost:
  - 160, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300

- Power (identical service) - 27%
Figure 20  32-Item Worst-Case
\[ S_1 > D_1 + 0.25 \mu_1 \]
Figure 21. 32-Item Worst-Case

\[ S_i \leq D_i + u_i^* \]
Figure 22. 32-Item Worst Case

\[ S_1 \geq D_1 + 0.80 \mu_1^* \]
Figure 23. 128-Item Base-Case

\[ S_i \geq D_i \]
Figure 24. 128-Item Base-Case with Central Nonrandom Stratified Uniform Sampling Scheme 1:

\[ S_1 = D_1 \]
Figure 25. 128-Item Base-Case with Central Nonrandom Stratified Uniform Sampling Scheme 2:
\[ S_i \leq D_i \]
Figure 26. 128-Item Base-Case with Central Nonrandom Stratified Uniform Sampling Scheme 3

\[ S_i \geq D_i \]
Figure 27: 128-Item Base-Case with Central Nonrandom Stratified Uniform Sampling Scheme 1:

\[ S_i = D_i \]
Figure 28. 128-Item Base-Case with Central Nonrandom Stratified Uniform Sampling Scheme 2

$$S_i \geq D_i$$
Figure 29. 128-Item Base-Case with Central Nonrandom Stratified Uniform Sampling Scheme 3

\[ S_i \geq D_i \]
Figure 30. 128-Item Base-Case with Central Random Stratified Uniform Sampling Scheme 4

$S_i \geq D_i$. 
Figure 31. 128-Item Base-Case with Central Random Stratified Uniform Sampling Scheme 4

$S_i \geq D_i$
Figure 33. 128-Item Base Case with Central Random Stratified Uniform Sampling Scheme 4

\[ S_1 \geq D_1 + 0.75u_1^* \]
Figure 35. 128-Item Central Nonrandom Uniform Sampling from 512-Item Base-Case System

$S_i \geq D_i + ru_j$
Figure 36. 512-Item Base-Case System

\[ S_i \geq D_i + r \mu_i \]

Power (identical service)
Figure 37. 512-Item Base-Case with Central Nonrandom Uniform Sampling Scheme (32-Item Sample)

\[ S_i = \theta_i + 0.6\bar{\theta}_i \]
Figure 38. 512-Item Base-Case with Central Nonrandom Uniform Sampling Scheme (64-Item Sample)

\[ S_i \geq D_i + 6\mu_i^* \]
Figure 39. 512-Item Base-Case with Central Nonrandom Uniform Sampling Scheme (128-Item Sample)

\[ S_i \geq D_i + 0.6 \mu_i \]
Figure 40. 512 - Item Base - Case with Central Nonrandom Uniform Sampling Scheme (32-Item Sample)

\[ S_i \geq D_i + 0.6\mu_i^{*} \]
Figure 41. 512 - Item Base - Case with Central Nonrandom Uniform Sampling Scheme (64 - Item Sample)

\[ S_i \geq D_i + 0.6 \mu_i^* \]
Figure 42. 512 - Item Base-Case with Central Nonrandom Uniform Sampling Scheme (128-Item Sample)

\[ S_i \geq D_i + 0.6 \mu_i^* \]
APPENDIX TO CHAPTER IV

This appendix reports the seeds used in the IMSL Routine GGUD to create the six random samples of Scheme 4 of Section 4, and a listing of the 512-item inventory system of Section 5. The seed used in the IMSL Routine GGUD to generate the $\mu_i$ for this system is 872245376.


<table>
<thead>
<tr>
<th>Test</th>
<th>Seed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>96665107</td>
</tr>
<tr>
<td>2</td>
<td>227164307</td>
</tr>
<tr>
<td>3</td>
<td>929811759</td>
</tr>
<tr>
<td>4</td>
<td>2058317624</td>
</tr>
<tr>
<td>5</td>
<td>75813808</td>
</tr>
<tr>
<td>6</td>
<td>1743717576</td>
</tr>
</tbody>
</table>

NOTE: Experiment for Test 1 reported in Figures 29 and 30.
512-Item Inventory System of Chapter IV

For all items \( i = 1, 2, \ldots, 512 \):

\[
K_i = 24 \\
k_i = 4 \\
\frac{\sigma_i^2}{\mu_i} = 9
\]

<table>
<thead>
<tr>
<th>( i )</th>
<th>( h_i )</th>
<th>( \mu_i )</th>
<th>( \sigma_i )</th>
<th>( D_i )</th>
<th>( D_i/\mu_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.8145</td>
<td>9.0000</td>
<td>9.0000</td>
<td>17.5010</td>
<td>1.9446</td>
</tr>
<tr>
<td>2</td>
<td>2.0363</td>
<td>6.0000</td>
<td>8.4853</td>
<td>15.7557</td>
<td>1.9695</td>
</tr>
<tr>
<td>3</td>
<td>2.7002</td>
<td>6.0000</td>
<td>7.3485</td>
<td>12.1972</td>
<td>2.0329</td>
</tr>
<tr>
<td>4</td>
<td>1.2363</td>
<td>13.0000</td>
<td>10.8167</td>
<td>24.6235</td>
<td>1.8941</td>
</tr>
<tr>
<td>5</td>
<td>2.6514</td>
<td>6.0000</td>
<td>7.3485</td>
<td>12.3104</td>
<td>2.0517</td>
</tr>
<tr>
<td>6</td>
<td>1.2090</td>
<td>13.0000</td>
<td>10.8167</td>
<td>24.9025</td>
<td>1.9156</td>
</tr>
<tr>
<td>7</td>
<td>0.9691</td>
<td>16.0000</td>
<td>12.0000</td>
<td>30.3035</td>
<td>1.8940</td>
</tr>
<tr>
<td>8</td>
<td>3.8186</td>
<td>4.0000</td>
<td>6.0000</td>
<td>8.7403</td>
<td>2.1851</td>
</tr>
<tr>
<td>9</td>
<td>1.0020</td>
<td>15.0000</td>
<td>11.6190</td>
<td>29.0211</td>
<td>1.9347</td>
</tr>
<tr>
<td>10</td>
<td>1.8469</td>
<td>8.0000</td>
<td>8.4853</td>
<td>16.5535</td>
<td>2.0692</td>
</tr>
<tr>
<td>11</td>
<td>1.1163</td>
<td>13.0000</td>
<td>10.8167</td>
<td>25.9282</td>
<td>1.9945</td>
</tr>
<tr>
<td>12</td>
<td>1.0957</td>
<td>13.0000</td>
<td>10.8167</td>
<td>26.1746</td>
<td>2.0134</td>
</tr>
<tr>
<td>13</td>
<td>1.0747</td>
<td>13.0000</td>
<td>10.8167</td>
<td>26.4318</td>
<td>2.0332</td>
</tr>
<tr>
<td>14</td>
<td>1.5218</td>
<td>9.0000</td>
<td>9.0000</td>
<td>19.1301</td>
<td>2.1256</td>
</tr>
<tr>
<td>15</td>
<td>0.8388</td>
<td>16.0000</td>
<td>12.0000</td>
<td>32.6002</td>
<td>2.0375</td>
</tr>
<tr>
<td>16</td>
<td>2.6290</td>
<td>5.0000</td>
<td>6.7082</td>
<td>11.5135</td>
<td>2.3027</td>
</tr>
<tr>
<td>17</td>
<td>0.9193</td>
<td>14.0000</td>
<td>11.2250</td>
<td>29.4759</td>
<td>2.1054</td>
</tr>
<tr>
<td>18</td>
<td>2.0997</td>
<td>6.0000</td>
<td>7.3485</td>
<td>13.8526</td>
<td>2.3088</td>
</tr>
<tr>
<td>19</td>
<td>1.5411</td>
<td>8.0000</td>
<td>8.4853</td>
<td>18.1413</td>
<td>2.2677</td>
</tr>
<tr>
<td>20</td>
<td>1.2063</td>
<td>10.0000</td>
<td>9.4868</td>
<td>22.4390</td>
<td>2.2439</td>
</tr>
<tr>
<td>21</td>
<td>0.9077</td>
<td>13.0000</td>
<td>10.8167</td>
<td>28.7900</td>
<td>2.1216</td>
</tr>
<tr>
<td>22</td>
<td>0.9618</td>
<td>12.0000</td>
<td>10.3923</td>
<td>27.0711</td>
<td>2.2559</td>
</tr>
<tr>
<td>23</td>
<td>2.2574</td>
<td>5.0000</td>
<td>6.7082</td>
<td>12.4364</td>
<td>2.4873</td>
</tr>
<tr>
<td>24</td>
<td>2.7593</td>
<td>4.0000</td>
<td>6.0000</td>
<td>10.3021</td>
<td>2.5755</td>
</tr>
<tr>
<td>25</td>
<td>1.0792</td>
<td>10.0000</td>
<td>9.4868</td>
<td>23.7387</td>
<td>2.3739</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>26</td>
<td>0.8793</td>
<td>12.0000</td>
<td>10.3923</td>
<td>28.3268</td>
<td>2.3606</td>
</tr>
<tr>
<td>27</td>
<td>0.8597</td>
<td>12.0000</td>
<td>10.3923</td>
<td>28.6517</td>
<td>2.3876</td>
</tr>
<tr>
<td>28</td>
<td>3.3623</td>
<td>3.0000</td>
<td>5.1962</td>
<td>8.3412</td>
<td>2.7804</td>
</tr>
<tr>
<td>29</td>
<td>1.4088</td>
<td>7.0000</td>
<td>7.9373</td>
<td>18.0101</td>
<td>2.5729</td>
</tr>
<tr>
<td>30</td>
<td>0.8035</td>
<td>12.0000</td>
<td>10.3923</td>
<td>29.6504</td>
<td>2.4709</td>
</tr>
<tr>
<td>31</td>
<td>4.7132</td>
<td>2.0000</td>
<td>4.2426</td>
<td>6.0171</td>
<td>3.0085</td>
</tr>
<tr>
<td>32</td>
<td>0.7680</td>
<td>12.0000</td>
<td>10.3923</td>
<td>30.3348</td>
<td>2.5279</td>
</tr>
<tr>
<td>33</td>
<td>3.0037</td>
<td>3.0000</td>
<td>5.1962</td>
<td>8.8310</td>
<td>2.9437</td>
</tr>
<tr>
<td>34</td>
<td>0.8811</td>
<td>10.0000</td>
<td>9.4868</td>
<td>26.3044</td>
<td>2.6304</td>
</tr>
<tr>
<td>35</td>
<td>2.8719</td>
<td>3.0000</td>
<td>5.1962</td>
<td>9.0338</td>
<td>3.0113</td>
</tr>
<tr>
<td>36</td>
<td>0.5266</td>
<td>16.0000</td>
<td>12.0000</td>
<td>41.2601</td>
<td>2.5788</td>
</tr>
<tr>
<td>37</td>
<td>0.9155</td>
<td>9.0000</td>
<td>9.0000</td>
<td>24.7401</td>
<td>2.7489</td>
</tr>
<tr>
<td>38</td>
<td>0.6715</td>
<td>12.0000</td>
<td>12.0000</td>
<td>32.4688</td>
<td>2.7057</td>
</tr>
<tr>
<td>39</td>
<td>0.4926</td>
<td>16.0000</td>
<td>12.0000</td>
<td>42.6775</td>
<td>2.6673</td>
</tr>
<tr>
<td>40</td>
<td>1.9272</td>
<td>4.0000</td>
<td>6.0000</td>
<td>12.3539</td>
<td>3.0885</td>
</tr>
<tr>
<td>41</td>
<td>0.6855</td>
<td>11.0000</td>
<td>9.9499</td>
<td>31.0273</td>
<td>2.8207</td>
</tr>
<tr>
<td>42</td>
<td>0.5269</td>
<td>14.0000</td>
<td>11.2250</td>
<td>39.0649</td>
<td>2.7904</td>
</tr>
<tr>
<td>43</td>
<td>0.4511</td>
<td>16.0000</td>
<td>12.0000</td>
<td>44.6212</td>
<td>2.7888</td>
</tr>
<tr>
<td>44</td>
<td>0.7846</td>
<td>9.0000</td>
<td>9.0000</td>
<td>26.7486</td>
<td>2.9721</td>
</tr>
<tr>
<td>45</td>
<td>1.7274</td>
<td>4.0000</td>
<td>6.0000</td>
<td>13.0572</td>
<td>3.2643</td>
</tr>
<tr>
<td>46</td>
<td>1.3523</td>
<td>5.0000</td>
<td>6.7082</td>
<td>16.1173</td>
<td>3.2235</td>
</tr>
<tr>
<td>47</td>
<td>2.2058</td>
<td>3.0000</td>
<td>5.1962</td>
<td>10.3242</td>
<td>3.4414</td>
</tr>
<tr>
<td>48</td>
<td>6.4770</td>
<td>1.0000</td>
<td>3.0000</td>
<td>3.9324</td>
<td>3.9324</td>
</tr>
<tr>
<td>49</td>
<td>0.7925</td>
<td>8.0000</td>
<td>8.4853</td>
<td>25.3993</td>
<td>3.1749</td>
</tr>
<tr>
<td>50</td>
<td>0.6206</td>
<td>10.0000</td>
<td>9.4868</td>
<td>31.4075</td>
<td>3.1407</td>
</tr>
<tr>
<td>51</td>
<td>0.5064</td>
<td>12.0000</td>
<td>10.3923</td>
<td>37.4529</td>
<td>3.1211</td>
</tr>
<tr>
<td>52</td>
<td>0.3718</td>
<td>16.0000</td>
<td>12.0000</td>
<td>49.2023</td>
<td>3.0751</td>
</tr>
<tr>
<td>53</td>
<td>1.1651</td>
<td>5.0000</td>
<td>6.7082</td>
<td>17.3793</td>
<td>3.4759</td>
</tr>
<tr>
<td>54</td>
<td>1.1410</td>
<td>5.0000</td>
<td>6.7082</td>
<td>17.5642</td>
<td>3.5128</td>
</tr>
<tr>
<td>55</td>
<td>0.3492</td>
<td>16.0000</td>
<td>12.0000</td>
<td>50.7896</td>
<td>3.1744</td>
</tr>
<tr>
<td>56</td>
<td>0.4975</td>
<td>11.0000</td>
<td>9.9499</td>
<td>36.4898</td>
<td>3.3173</td>
</tr>
<tr>
<td>57</td>
<td>0.4874</td>
<td>11.0000</td>
<td>9.9499</td>
<td>36.8728</td>
<td>3.3521</td>
</tr>
<tr>
<td>58</td>
<td>1.7507</td>
<td>3.0000</td>
<td>5.1962</td>
<td>11.6048</td>
<td>3.8683</td>
</tr>
<tr>
<td>59</td>
<td>0.6432</td>
<td>8.0000</td>
<td>8.4853</td>
<td>28.2280</td>
<td>3.5285</td>
</tr>
<tr>
<td>60</td>
<td>0.4584</td>
<td>11.0000</td>
<td>9.9499</td>
<td>38.0353</td>
<td>3.4578</td>
</tr>
<tr>
<td>61</td>
<td>0.4492</td>
<td>11.0000</td>
<td>9.9499</td>
<td>38.4273</td>
<td>3.4934</td>
</tr>
<tr>
<td>62</td>
<td>4.8424</td>
<td>1.0000</td>
<td>3.0000</td>
<td>4.5559</td>
<td>4.5559</td>
</tr>
<tr>
<td>63</td>
<td>0.2966</td>
<td>16.0000</td>
<td>12.0000</td>
<td>55.1623</td>
<td>3.4476</td>
</tr>
<tr>
<td>64</td>
<td>0.3877</td>
<td>12.0000</td>
<td>10.3923</td>
<td>42.8717</td>
<td>3.5726</td>
</tr>
<tr>
<td>65</td>
<td>0.2850</td>
<td>16.0000</td>
<td>12.0000</td>
<td>56.2865</td>
<td>3.5179</td>
</tr>
<tr>
<td>66</td>
<td>0.7452</td>
<td>6.0000</td>
<td>7.3485</td>
<td>23.3980</td>
<td>3.8997</td>
</tr>
<tr>
<td>67</td>
<td>0.3372</td>
<td>13.0000</td>
<td>10.8167</td>
<td>47.5148</td>
<td>3.6550</td>
</tr>
<tr>
<td>68</td>
<td>4.2988</td>
<td>1.0000</td>
<td>3.0000</td>
<td>4.8388</td>
<td>4.8388</td>
</tr>
<tr>
<td>69</td>
<td>0.4216</td>
<td>10.0000</td>
<td>9.4868</td>
<td>38.1969</td>
<td>3.8197</td>
</tr>
<tr>
<td>70</td>
<td>0.2953</td>
<td>14.0000</td>
<td>11.2250</td>
<td>52.3615</td>
<td>3.7401</td>
</tr>
<tr>
<td>71</td>
<td>1.3518</td>
<td>3.0000</td>
<td>5.1962</td>
<td>13.2271</td>
<td>4.4090</td>
</tr>
<tr>
<td>72</td>
<td>0.6630</td>
<td>6.0000</td>
<td>7.3485</td>
<td>24.8234</td>
<td>4.1372</td>
</tr>
<tr>
<td>73</td>
<td>1.9513</td>
<td>2.0000</td>
<td>4.2426</td>
<td>9.4012</td>
<td>4.7006</td>
</tr>
<tr>
<td>74</td>
<td>0.7658</td>
<td>5.0000</td>
<td>6.7082</td>
<td>21.4917</td>
<td>4.2983</td>
</tr>
<tr>
<td>75</td>
<td>0.3757</td>
<td>10.0000</td>
<td>9.4868</td>
<td>40.4908</td>
<td>4.0491</td>
</tr>
</tbody>
</table>
76  0.9216  4.0000  6.0000  17.9437  4.4859
77  0.3289  11.0000  9.9499  44.9928  4.0903
78  0.8876  4.0000  6.0000  18.2878  4.5719
79  0.2178  16.0000  12.0000  64.4933  4.0308
80  0.4276  8.0000  8.4853  34.7056  4.3382
81  1.0990  3.0000  7.3485  27.0443  4.5074
82  1.6187  2.0000  4.2426  10.3336  4.8959
83  3.1788  1.0000  3.0000  5.6372  5.1668
84  0.2081  15.0000  11.6190  53.9564  5.6372
85  0.7665  4.0000  6.0000  19.6980  4.2856
86  0.6022  5.0000  6.7082  24.2694  4.9245
87  0.4930  6.0000  7.3485  28.8395  4.8509
88  0.3632  8.0000  8.4853  37.6980  4.7119
89  0.2178  16.0000  12.0000  75.2350  4.6529
90  0.8876  4.0000  6.0000  19.6980  4.6667
91  0.3289  11.0000  9.9499  44.9928  4.1138
92  0.2178  16.0000  12.0000  64.4933  4.9922
93  0.9216  4.0000  6.0000  19.6980  6.0215
94  0.3289  11.0000  9.9499  44.9928  6.6102
95  0.8876  4.0000  6.0000  19.6980  4.9939
96  0.2178  16.0000  12.0000  64.4933  5.9480
97  0.3289  11.0000  9.9499  44.9928  5.9084
98  0.2178  16.0000  12.0000  64.4933  5.2144
99  0.9216  4.0000  6.0000  19.6980  5.0338
100 0.3289  11.0000  9.9499  44.9928  5.6215
101 0.8876  4.0000  6.0000  19.6980  6.6102
102 0.2178  16.0000  12.0000  64.4933  4.9939
103 0.3289  11.0000  9.9499  44.9928  6.6102
104 0.8876  4.0000  6.0000  19.6980  4.9939
105 0.2178  16.0000  12.0000  64.4933  5.9480
106 0.3289  11.0000  9.9499  44.9928  5.9084
107 0.2178  16.0000  12.0000  64.4933  5.5337
108 0.3289  11.0000  9.9499  44.9928  6.8944
109 0.8876  4.0000  6.0000  19.6980  6.6102
110 0.2178  16.0000  12.0000  64.4933  5.9480
111 0.3289  11.0000  9.9499  44.9928  5.9084
112 0.8876  4.0000  6.0000  19.6980  6.6102
113 0.2178  16.0000  12.0000  64.4933  5.5337
114 0.3289  11.0000  9.9499  44.9928  6.2128
115 0.8876  4.0000  6.0000  19.6980  5.6215
116 0.2178  16.0000  12.0000  64.4933  6.6102
117 0.3289  11.0000  9.9499  44.9928  6.6035
118 0.8876  4.0000  6.0000  19.6980  6.8099
119 0.2178  16.0000  12.0000  64.4933  6.0065
120 0.3289  11.0000  9.9499  44.9928  6.7811
121 0.8876  4.0000  6.0000  19.6980  6.4488
122 0.2178  16.0000  12.0000  64.4933  6.0065
123 0.3289  11.0000  9.9499  44.9928  6.7811
124 0.8876  4.0000  6.0000  19.6980  6.4488
125 0.2178  16.0000  12.0000  64.4933  6.0065
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>126</td>
<td>0.1436</td>
<td>11.0000</td>
<td>9.9499</td>
<td>68.4225</td>
<td>6.2202</td>
</tr>
<tr>
<td>127</td>
<td>0.1729</td>
<td>9.0000</td>
<td>9.0000</td>
<td>57.5011</td>
<td>6.3890</td>
</tr>
<tr>
<td>128</td>
<td>0.7663</td>
<td>2.0000</td>
<td>4.2426</td>
<td>15.0862</td>
<td>7.5431</td>
</tr>
<tr>
<td>129</td>
<td>1.5096</td>
<td>1.0000</td>
<td>3.0000</td>
<td>8.2170</td>
<td>8.2170</td>
</tr>
<tr>
<td>130</td>
<td>0.1859</td>
<td>8.0000</td>
<td>8.4853</td>
<td>52.9084</td>
<td>6.6130</td>
</tr>
<tr>
<td>131</td>
<td>0.1465</td>
<td>10.0000</td>
<td>9.4868</td>
<td>65.2118</td>
<td>6.5212</td>
</tr>
<tr>
<td>132</td>
<td>0.1203</td>
<td>12.0000</td>
<td>10.3923</td>
<td>77.5194</td>
<td>6.8599</td>
</tr>
<tr>
<td>133</td>
<td>0.1580</td>
<td>9.0000</td>
<td>9.0000</td>
<td>60.1889</td>
<td>6.6877</td>
</tr>
<tr>
<td>134</td>
<td>0.2001</td>
<td>7.0000</td>
<td>7.9373</td>
<td>48.3499</td>
<td>6.9071</td>
</tr>
<tr>
<td>135</td>
<td>0.2760</td>
<td>5.0000</td>
<td>6.7082</td>
<td>36.0159</td>
<td>7.2032</td>
</tr>
<tr>
<td>136</td>
<td>0.1700</td>
<td>8.0000</td>
<td>8.4853</td>
<td>55.3493</td>
<td>6.9187</td>
</tr>
<tr>
<td>137</td>
<td>0.1031</td>
<td>13.0000</td>
<td>10.8167</td>
<td>86.5519</td>
<td>6.6578</td>
</tr>
<tr>
<td>138</td>
<td>0.0825</td>
<td>16.0000</td>
<td>12.0000</td>
<td>105.3780</td>
<td>6.5861</td>
</tr>
<tr>
<td>139</td>
<td>0.0868</td>
<td>15.0000</td>
<td>11.6190</td>
<td>100.0795</td>
<td>6.6720</td>
</tr>
<tr>
<td>140</td>
<td>1.2828</td>
<td>1.0000</td>
<td>3.0000</td>
<td>8.9225</td>
<td>8.9225</td>
</tr>
<tr>
<td>141</td>
<td>0.1580</td>
<td>6.0000</td>
<td>8.4853</td>
<td>57.4299</td>
<td>7.1877</td>
</tr>
<tr>
<td>142</td>
<td>1.2462</td>
<td>1.0000</td>
<td>3.0000</td>
<td>9.0540</td>
<td>9.0540</td>
</tr>
<tr>
<td>143</td>
<td>0.0945</td>
<td>13.0000</td>
<td>10.8167</td>
<td>90.4492</td>
<td>6.9576</td>
</tr>
<tr>
<td>144</td>
<td>1.2109</td>
<td>1.0000</td>
<td>3.0000</td>
<td>9.1865</td>
<td>9.1865</td>
</tr>
<tr>
<td>145</td>
<td>0.3979</td>
<td>3.0000</td>
<td>5.1962</td>
<td>24.5588</td>
<td>8.1863</td>
</tr>
<tr>
<td>146</td>
<td>0.1681</td>
<td>8.0000</td>
<td>7.9373</td>
<td>52.8028</td>
<td>7.5433</td>
</tr>
<tr>
<td>147</td>
<td>0.0829</td>
<td>14.0000</td>
<td>11.2250</td>
<td>99.5967</td>
<td>7.1141</td>
</tr>
<tr>
<td>148</td>
<td>0.3814</td>
<td>3.0000</td>
<td>5.1962</td>
<td>25.0930</td>
<td>8.3643</td>
</tr>
<tr>
<td>149</td>
<td>0.2256</td>
<td>5.0000</td>
<td>6.7082</td>
<td>39.8857</td>
<td>7.9771</td>
</tr>
<tr>
<td>150</td>
<td>0.0856</td>
<td>13.0000</td>
<td>10.8167</td>
<td>95.1075</td>
<td>7.3160</td>
</tr>
<tr>
<td>151</td>
<td>0.0686</td>
<td>16.0000</td>
<td>12.0000</td>
<td>115.7561</td>
<td>7.2348</td>
</tr>
<tr>
<td>152</td>
<td>0.1545</td>
<td>7.0000</td>
<td>7.9373</td>
<td>55.1047</td>
<td>7.8721</td>
</tr>
<tr>
<td>153</td>
<td>0.0821</td>
<td>13.0000</td>
<td>10.8167</td>
<td>97.1407</td>
<td>7.4724</td>
</tr>
<tr>
<td>154</td>
<td>0.1315</td>
<td>8.0000</td>
<td>8.4853</td>
<td>63.0241</td>
<td>7.8780</td>
</tr>
<tr>
<td>155</td>
<td>0.0741</td>
<td>14.0000</td>
<td>11.2250</td>
<td>105.3869</td>
<td>7.5276</td>
</tr>
<tr>
<td>156</td>
<td>0.0640</td>
<td>16.0000</td>
<td>12.0000</td>
<td>119.8814</td>
<td>7.4926</td>
</tr>
<tr>
<td>157</td>
<td>0.0918</td>
<td>11.0000</td>
<td>9.9499</td>
<td>85.8229</td>
<td>7.8021</td>
</tr>
<tr>
<td>158</td>
<td>0.0905</td>
<td>11.0000</td>
<td>9.9499</td>
<td>86.4180</td>
<td>7.8562</td>
</tr>
<tr>
<td>159</td>
<td>0.0893</td>
<td>11.0000</td>
<td>9.9499</td>
<td>87.0152</td>
<td>7.9105</td>
</tr>
<tr>
<td>160</td>
<td>0.2423</td>
<td>4.0000</td>
<td>6.0000</td>
<td>35.2760</td>
<td>8.8190</td>
</tr>
<tr>
<td>161</td>
<td>0.0869</td>
<td>11.0000</td>
<td>9.9499</td>
<td>88.2160</td>
<td>8.0196</td>
</tr>
<tr>
<td>162</td>
<td>0.2359</td>
<td>4.0000</td>
<td>6.0000</td>
<td>35.7612</td>
<td>8.9403</td>
</tr>
<tr>
<td>163</td>
<td>0.1551</td>
<td>6.0000</td>
<td>7.3485</td>
<td>51.7659</td>
<td>8.6276</td>
</tr>
<tr>
<td>164</td>
<td>0.0918</td>
<td>10.0000</td>
<td>9.4868</td>
<td>82.5855</td>
<td>8.2586</td>
</tr>
<tr>
<td>165</td>
<td>0.0697</td>
<td>13.0000</td>
<td>10.8167</td>
<td>105.4965</td>
<td>8.1511</td>
</tr>
<tr>
<td>166</td>
<td>0.2981</td>
<td>3.0000</td>
<td>5.1962</td>
<td>28.4234</td>
<td>9.4745</td>
</tr>
<tr>
<td>167</td>
<td>0.1261</td>
<td>7.0000</td>
<td>7.9373</td>
<td>61.0821</td>
<td>8.7260</td>
</tr>
<tr>
<td>168</td>
<td>0.0968</td>
<td>9.0000</td>
<td>9.0000</td>
<td>77.1265</td>
<td>8.5696</td>
</tr>
<tr>
<td>169</td>
<td>0.0661</td>
<td>13.0000</td>
<td>10.8167</td>
<td>108.3613</td>
<td>8.3355</td>
</tr>
<tr>
<td>170</td>
<td>0.0530</td>
<td>16.0000</td>
<td>12.0000</td>
<td>131.8308</td>
<td>8.2394</td>
</tr>
<tr>
<td>171</td>
<td>0.0523</td>
<td>16.0000</td>
<td>12.0000</td>
<td>132.7069</td>
<td>8.2942</td>
</tr>
<tr>
<td>172</td>
<td>0.1033</td>
<td>8.0000</td>
<td>8.4853</td>
<td>71.2145</td>
<td>8.9018</td>
</tr>
<tr>
<td>173</td>
<td>0.0906</td>
<td>9.0000</td>
<td>9.0000</td>
<td>79.7237</td>
<td>8.8582</td>
</tr>
<tr>
<td>174</td>
<td>0.0503</td>
<td>16.0000</td>
<td>12.0000</td>
<td>135.3535</td>
<td>8.4596</td>
</tr>
<tr>
<td>175</td>
<td>0.3975</td>
<td>2.0000</td>
<td>4.2426</td>
<td>21.0302</td>
<td>10.5151</td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>0.0872</td>
<td>0.01937</td>
<td>0.05588</td>
<td>0.2517</td>
<td>0.1864</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>326</td>
<td>0.0145</td>
<td>11.0000</td>
<td>9.9499</td>
<td>218.2586</td>
<td>19.8417</td>
</tr>
<tr>
<td>327</td>
<td>0.0105</td>
<td>15.0000</td>
<td>11.6190</td>
<td>290.7210</td>
<td>19.3814</td>
</tr>
<tr>
<td>328</td>
<td>0.0131</td>
<td>12.0000</td>
<td>10.3923</td>
<td>238.3285</td>
<td>19.8607</td>
</tr>
<tr>
<td>329</td>
<td>0.0389</td>
<td>4.0000</td>
<td>6.0000</td>
<td>89.8069</td>
<td>22.2674</td>
</tr>
<tr>
<td>330</td>
<td>0.0193</td>
<td>8.0000</td>
<td>8.4853</td>
<td>166.6244</td>
<td>20.8280</td>
</tr>
<tr>
<td>331</td>
<td>0.0218</td>
<td>7.0000</td>
<td>7.9373</td>
<td>148.4071</td>
<td>21.2010</td>
</tr>
<tr>
<td>332</td>
<td>0.0504</td>
<td>3.0000</td>
<td>5.1962</td>
<td>69.8341</td>
<td>23.2780</td>
</tr>
<tr>
<td>333</td>
<td>0.0167</td>
<td>9.0000</td>
<td>9.0000</td>
<td>187.8061</td>
<td>20.8673</td>
</tr>
<tr>
<td>334</td>
<td>0.0744</td>
<td>2.0000</td>
<td>4.2426</td>
<td>49.1140</td>
<td>24.5570</td>
</tr>
<tr>
<td>335</td>
<td>0.0164</td>
<td>9.0000</td>
<td>9.0000</td>
<td>189.4807</td>
<td>21.0534</td>
</tr>
<tr>
<td>336</td>
<td>0.0365</td>
<td>4.0000</td>
<td>6.0000</td>
<td>91.8888</td>
<td>22.9722</td>
</tr>
<tr>
<td>337</td>
<td>0.0091</td>
<td>16.0000</td>
<td>12.0000</td>
<td>322.4315</td>
<td>20.1520</td>
</tr>
<tr>
<td>338</td>
<td>0.0160</td>
<td>9.0000</td>
<td>9.0000</td>
<td>192.0082</td>
<td>21.3342</td>
</tr>
<tr>
<td>339</td>
<td>0.0178</td>
<td>8.0000</td>
<td>8.4853</td>
<td>173.4085</td>
<td>21.6761</td>
</tr>
<tr>
<td>340</td>
<td>0.0101</td>
<td>14.0000</td>
<td>11.2250</td>
<td>289.2186</td>
<td>20.6585</td>
</tr>
<tr>
<td>341</td>
<td>0.0087</td>
<td>16.0000</td>
<td>12.0000</td>
<td>328.1514</td>
<td>20.5095</td>
</tr>
<tr>
<td>342</td>
<td>0.0116</td>
<td>12.0000</td>
<td>10.3923</td>
<td>253.5781</td>
<td>21.1315</td>
</tr>
<tr>
<td>343</td>
<td>0.0275</td>
<td>5.0000</td>
<td>6.7082</td>
<td>115.6913</td>
<td>23.1383</td>
</tr>
<tr>
<td>344</td>
<td>0.0682</td>
<td>2.0000</td>
<td>4.2426</td>
<td>51.3213</td>
<td>25.6606</td>
</tr>
<tr>
<td>345</td>
<td>0.0270</td>
<td>5.0000</td>
<td>6.7082</td>
<td>116.7033</td>
<td>23.3407</td>
</tr>
<tr>
<td>346</td>
<td>0.0191</td>
<td>7.0000</td>
<td>7.9373</td>
<td>158.5338</td>
<td>22.6477</td>
</tr>
<tr>
<td>347</td>
<td>0.0083</td>
<td>16.0000</td>
<td>12.0000</td>
<td>336.8383</td>
<td>21.0524</td>
</tr>
<tr>
<td>348</td>
<td>0.0329</td>
<td>4.0000</td>
<td>6.0000</td>
<td>96.8375</td>
<td>24.2094</td>
</tr>
<tr>
<td>349</td>
<td>0.0109</td>
<td>12.0000</td>
<td>10.3923</td>
<td>261.4045</td>
<td>21.7837</td>
</tr>
<tr>
<td>350</td>
<td>0.0216</td>
<td>6.0000</td>
<td>7.3485</td>
<td>140.4336</td>
<td>23.4056</td>
</tr>
<tr>
<td>351</td>
<td>0.0428</td>
<td>3.0000</td>
<td>5.1962</td>
<td>75.8883</td>
<td>25.2961</td>
</tr>
<tr>
<td>352</td>
<td>0.0085</td>
<td>15.0000</td>
<td>11.6190</td>
<td>324.4671</td>
<td>21.6311</td>
</tr>
<tr>
<td>353</td>
<td>0.0126</td>
<td>10.0000</td>
<td>9.4868</td>
<td>225.4156</td>
<td>22.5416</td>
</tr>
<tr>
<td>354</td>
<td>0.0313</td>
<td>4.0000</td>
<td>6.0000</td>
<td>99.3673</td>
<td>24.8418</td>
</tr>
<tr>
<td>355</td>
<td>0.0207</td>
<td>6.0000</td>
<td>7.3485</td>
<td>143.4757</td>
<td>23.9126</td>
</tr>
<tr>
<td>356</td>
<td>0.0123</td>
<td>10.0000</td>
<td>9.4868</td>
<td>228.3220</td>
<td>22.8322</td>
</tr>
<tr>
<td>357</td>
<td>0.0203</td>
<td>6.0000</td>
<td>7.3485</td>
<td>144.7030</td>
<td>24.1112</td>
</tr>
<tr>
<td>358</td>
<td>0.0086</td>
<td>14.0000</td>
<td>11.2250</td>
<td>312.5748</td>
<td>22.3268</td>
</tr>
<tr>
<td>359</td>
<td>0.0080</td>
<td>15.0000</td>
<td>11.6190</td>
<td>334.2946</td>
<td>22.2863</td>
</tr>
<tr>
<td>360</td>
<td>0.0132</td>
<td>9.0000</td>
<td>9.0000</td>
<td>211.1276</td>
<td>23.4586</td>
</tr>
<tr>
<td>361</td>
<td>0.0079</td>
<td>15.0000</td>
<td>11.6190</td>
<td>337.1332</td>
<td>22.4755</td>
</tr>
<tr>
<td>362</td>
<td>0.0146</td>
<td>8.0000</td>
<td>8.4853</td>
<td>191.4476</td>
<td>23.9310</td>
</tr>
<tr>
<td>363</td>
<td>0.0145</td>
<td>8.0000</td>
<td>8.4853</td>
<td>192.2550</td>
<td>24.0319</td>
</tr>
<tr>
<td>364</td>
<td>0.0384</td>
<td>3.0000</td>
<td>5.1962</td>
<td>80.1962</td>
<td>26.7321</td>
</tr>
<tr>
<td>365</td>
<td>0.0114</td>
<td>10.0000</td>
<td>9.4868</td>
<td>237.1690</td>
<td>23.7169</td>
</tr>
<tr>
<td>366</td>
<td>0.0087</td>
<td>13.0000</td>
<td>10.8167</td>
<td>302.1882</td>
<td>23.2452</td>
</tr>
<tr>
<td>367</td>
<td>0.0140</td>
<td>8.0000</td>
<td>8.4853</td>
<td>195.5042</td>
<td>24.4380</td>
</tr>
<tr>
<td>368</td>
<td>0.0074</td>
<td>15.0000</td>
<td>11.6190</td>
<td>347.1765</td>
<td>23.1451</td>
</tr>
<tr>
<td>369</td>
<td>0.0276</td>
<td>4.0000</td>
<td>6.0000</td>
<td>105.8552</td>
<td>26.4638</td>
</tr>
<tr>
<td>370</td>
<td>0.0073</td>
<td>15.0000</td>
<td>11.6190</td>
<td>350.0770</td>
<td>23.3385</td>
</tr>
<tr>
<td>371</td>
<td>0.0181</td>
<td>6.0000</td>
<td>7.3485</td>
<td>153.4614</td>
<td>25.5769</td>
</tr>
<tr>
<td>372</td>
<td>0.0098</td>
<td>11.0000</td>
<td>9.9499</td>
<td>266.2042</td>
<td>24.2004</td>
</tr>
<tr>
<td>373</td>
<td>0.0153</td>
<td>7.0000</td>
<td>7.9373</td>
<td>177.7266</td>
<td>25.3895</td>
</tr>
<tr>
<td>374</td>
<td>0.0177</td>
<td>6.0000</td>
<td>7.3485</td>
<td>155.3766</td>
<td>25.8961</td>
</tr>
<tr>
<td>375</td>
<td>0.0131</td>
<td>8.0000</td>
<td>8.4853</td>
<td>202.0963</td>
<td>25.2620</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>376</td>
<td>0.0075</td>
<td>14.0000</td>
<td>11.2250</td>
<td>336.9713</td>
<td>24.0694</td>
</tr>
<tr>
<td>377</td>
<td>0.0069</td>
<td>15.0000</td>
<td>11.6190</td>
<td>360.3380</td>
<td>24.0225</td>
</tr>
<tr>
<td>378</td>
<td>0.0103</td>
<td>10.0000</td>
<td>9.4986</td>
<td>250.2880</td>
<td>25.0289</td>
</tr>
<tr>
<td>379</td>
<td>0.1018</td>
<td>1.0000</td>
<td>3.0000</td>
<td>32.1526</td>
<td>32.1526</td>
</tr>
<tr>
<td>380</td>
<td>0.0084</td>
<td>12.0000</td>
<td>10.3923</td>
<td>297.7060</td>
<td>24.8088</td>
</tr>
<tr>
<td>381</td>
<td>0.0072</td>
<td>14.0000</td>
<td>11.2250</td>
<td>343.9351</td>
<td>25.5668</td>
</tr>
<tr>
<td>382</td>
<td>0.0083</td>
<td>12.0000</td>
<td>10.3923</td>
<td>300.1412</td>
<td>25.0118</td>
</tr>
<tr>
<td>383</td>
<td>0.0197</td>
<td>5.0000</td>
<td>6.7082</td>
<td>136.8934</td>
<td>27.3787</td>
</tr>
<tr>
<td>384</td>
<td>0.0082</td>
<td>12.0000</td>
<td>10.3923</td>
<td>302.5878</td>
<td>25.2156</td>
</tr>
<tr>
<td>385</td>
<td>0.0061</td>
<td>16.0000</td>
<td>12.0000</td>
<td>394.8866</td>
<td>24.6904</td>
</tr>
<tr>
<td>386</td>
<td>0.0963</td>
<td>1.0000</td>
<td>3.0000</td>
<td>33.0801</td>
<td>33.0801</td>
</tr>
<tr>
<td>387</td>
<td>0.0478</td>
<td>2.0000</td>
<td>4.2426</td>
<td>61.4488</td>
<td>30.7244</td>
</tr>
<tr>
<td>388</td>
<td>0.0316</td>
<td>3.0000</td>
<td>5.1962</td>
<td>88.5092</td>
<td>29.5031</td>
</tr>
<tr>
<td>389</td>
<td>0.0085</td>
<td>11.0000</td>
<td>9.9499</td>
<td>285.3096</td>
<td>25.9372</td>
</tr>
<tr>
<td>390</td>
<td>0.0117</td>
<td>8.0000</td>
<td>8.4853</td>
<td>214.7961</td>
<td>26.8495</td>
</tr>
<tr>
<td>391</td>
<td>0.0116</td>
<td>8.0000</td>
<td>8.4853</td>
<td>215.6586</td>
<td>26.9573</td>
</tr>
<tr>
<td>392</td>
<td>0.0131</td>
<td>7.0000</td>
<td>7.9373</td>
<td>191.9905</td>
<td>27.4272</td>
</tr>
<tr>
<td>393</td>
<td>0.0304</td>
<td>3.0000</td>
<td>5.1962</td>
<td>90.3006</td>
<td>30.1002</td>
</tr>
<tr>
<td>394</td>
<td>0.0056</td>
<td>16.0000</td>
<td>12.0000</td>
<td>409.4194</td>
<td>25.5884</td>
</tr>
<tr>
<td>395</td>
<td>0.0100</td>
<td>9.0000</td>
<td>9.0000</td>
<td>243.7023</td>
<td>27.0780</td>
</tr>
<tr>
<td>396</td>
<td>0.0074</td>
<td>12.0000</td>
<td>10.3923</td>
<td>317.5083</td>
<td>26.4590</td>
</tr>
<tr>
<td>397</td>
<td>0.0088</td>
<td>10.0000</td>
<td>9.4868</td>
<td>270.1981</td>
<td>27.0198</td>
</tr>
<tr>
<td>398</td>
<td>0.0097</td>
<td>9.0000</td>
<td>9.0000</td>
<td>246.6200</td>
<td>27.4022</td>
</tr>
<tr>
<td>399</td>
<td>0.0217</td>
<td>4.0000</td>
<td>6.0000</td>
<td>119.5422</td>
<td>29.8855</td>
</tr>
<tr>
<td>400</td>
<td>0.0108</td>
<td>8.0000</td>
<td>8.4853</td>
<td>223.5112</td>
<td>27.9389</td>
</tr>
<tr>
<td>401</td>
<td>0.0285</td>
<td>3.0000</td>
<td>5.1962</td>
<td>93.2100</td>
<td>31.0700</td>
</tr>
<tr>
<td>402</td>
<td>0.0065</td>
<td>13.0000</td>
<td>10.8167</td>
<td>349.6675</td>
<td>26.8975</td>
</tr>
<tr>
<td>403</td>
<td>0.0065</td>
<td>13.0000</td>
<td>10.8167</td>
<td>351.0436</td>
<td>27.0034</td>
</tr>
<tr>
<td>404</td>
<td>0.0167</td>
<td>5.0000</td>
<td>6.7082</td>
<td>148.8506</td>
<td>29.7701</td>
</tr>
<tr>
<td>405</td>
<td>0.0064</td>
<td>13.0000</td>
<td>10.8167</td>
<td>353.8052</td>
<td>27.2158</td>
</tr>
<tr>
<td>406</td>
<td>0.0206</td>
<td>4.0000</td>
<td>6.0000</td>
<td>122.8745</td>
<td>30.7186</td>
</tr>
<tr>
<td>407</td>
<td>0.0068</td>
<td>12.0000</td>
<td>10.3923</td>
<td>331.5509</td>
<td>27.6292</td>
</tr>
<tr>
<td>408</td>
<td>0.0135</td>
<td>6.0000</td>
<td>7.3485</td>
<td>178.0443</td>
<td>29.6741</td>
</tr>
<tr>
<td>409</td>
<td>0.0804</td>
<td>1.0000</td>
<td>3.0000</td>
<td>36.2354</td>
<td>36.2354</td>
</tr>
<tr>
<td>410</td>
<td>0.0160</td>
<td>5.0000</td>
<td>6.7082</td>
<td>152.3736</td>
<td>30.4747</td>
</tr>
<tr>
<td>411</td>
<td>0.0079</td>
<td>10.0000</td>
<td>9.4868</td>
<td>285.4337</td>
<td>28.5434</td>
</tr>
<tr>
<td>412</td>
<td>0.0112</td>
<td>7.0000</td>
<td>7.9373</td>
<td>207.6954</td>
<td>29.6708</td>
</tr>
<tr>
<td>413</td>
<td>0.0156</td>
<td>5.0000</td>
<td>6.7082</td>
<td>154.1530</td>
<td>30.8306</td>
</tr>
<tr>
<td>414</td>
<td>0.0387</td>
<td>2.0000</td>
<td>4.2426</td>
<td>68.3488</td>
<td>39.1744</td>
</tr>
<tr>
<td>415</td>
<td>0.0064</td>
<td>12.0000</td>
<td>10.3923</td>
<td>341.9855</td>
<td>28.4988</td>
</tr>
<tr>
<td>416</td>
<td>0.0085</td>
<td>9.0000</td>
<td>9.0000</td>
<td>264.5501</td>
<td>29.3945</td>
</tr>
<tr>
<td>417</td>
<td>0.0095</td>
<td>8.0000</td>
<td>8.4853</td>
<td>238.7892</td>
<td>29.8486</td>
</tr>
<tr>
<td>418</td>
<td>0.0375</td>
<td>2.0000</td>
<td>4.2426</td>
<td>69.4074</td>
<td>34.7037</td>
</tr>
<tr>
<td>419</td>
<td>0.0106</td>
<td>7.0000</td>
<td>7.9373</td>
<td>213.3618</td>
<td>30.4803</td>
</tr>
<tr>
<td>420</td>
<td>0.0739</td>
<td>1.0000</td>
<td>3.0000</td>
<td>37.8036</td>
<td>37.8036</td>
</tr>
<tr>
<td>421</td>
<td>0.0046</td>
<td>16.0000</td>
<td>12.0000</td>
<td>454.8307</td>
<td>28.4269</td>
</tr>
<tr>
<td>422</td>
<td>0.0091</td>
<td>8.0000</td>
<td>8.4853</td>
<td>243.3946</td>
<td>30.4243</td>
</tr>
<tr>
<td>423</td>
<td>0.0045</td>
<td>16.0000</td>
<td>12.0000</td>
<td>458.3055</td>
<td>28.6441</td>
</tr>
<tr>
<td>424</td>
<td>0.0055</td>
<td>13.0000</td>
<td>10.8167</td>
<td>380.6686</td>
<td>29.2822</td>
</tr>
<tr>
<td>425</td>
<td>0.0089</td>
<td>8.0000</td>
<td>8.4853</td>
<td>246.1825</td>
<td>30.7728</td>
</tr>
<tr>
<td>426</td>
<td>0.0044</td>
<td>16.0000</td>
<td>12.0000</td>
<td>463.5466</td>
<td>28.9717</td>
</tr>
<tr>
<td>427</td>
<td>0.0702</td>
<td>1.0000</td>
<td>3.0000</td>
<td>38.8216</td>
<td>38.8216</td>
</tr>
<tr>
<td>428</td>
<td>0.0696</td>
<td>1.0000</td>
<td>3.0000</td>
<td>38.9684</td>
<td>38.9684</td>
</tr>
<tr>
<td>429</td>
<td>0.0173</td>
<td>4.0000</td>
<td>6.0000</td>
<td>134.1998</td>
<td>33.5499</td>
</tr>
<tr>
<td>430</td>
<td>0.0049</td>
<td>14.0000</td>
<td>11.2250</td>
<td>416.5789</td>
<td>29.7556</td>
</tr>
<tr>
<td>431</td>
<td>0.0085</td>
<td>8.0000</td>
<td>8.4853</td>
<td>251.8137</td>
<td>31.4767</td>
</tr>
<tr>
<td>432</td>
<td>0.0085</td>
<td>8.0000</td>
<td>8.4853</td>
<td>252.7595</td>
<td>31.5949</td>
</tr>
<tr>
<td>433</td>
<td>0.0075</td>
<td>9.0000</td>
<td>9.0000</td>
<td>282.1588</td>
<td>31.3510</td>
</tr>
<tr>
<td>434</td>
<td>0.0061</td>
<td>11.0000</td>
<td>9.9499</td>
<td>339.6170</td>
<td>30.8743</td>
</tr>
<tr>
<td>435</td>
<td>0.0132</td>
<td>5.0000</td>
<td>6.7082</td>
<td>167.5711</td>
<td>33.5142</td>
</tr>
<tr>
<td>436</td>
<td>0.0073</td>
<td>9.0000</td>
<td>9.0000</td>
<td>285.3350</td>
<td>31.7039</td>
</tr>
<tr>
<td>437</td>
<td>0.0130</td>
<td>5.0000</td>
<td>6.7082</td>
<td>168.8233</td>
<td>33.7647</td>
</tr>
<tr>
<td>438</td>
<td>0.0065</td>
<td>10.0000</td>
<td>9.4868</td>
<td>316.1968</td>
<td>31.6197</td>
</tr>
<tr>
<td>439</td>
<td>0.0054</td>
<td>12.0000</td>
<td>10.3923</td>
<td>374.4240</td>
<td>31.2020</td>
</tr>
<tr>
<td>440</td>
<td>0.0046</td>
<td>14.0000</td>
<td>11.2250</td>
<td>432.4043</td>
<td>30.8860</td>
</tr>
<tr>
<td>441</td>
<td>0.0211</td>
<td>3.0000</td>
<td>5.1962</td>
<td>108.5672</td>
<td>36.1891</td>
</tr>
<tr>
<td>442</td>
<td>0.0314</td>
<td>2.0000</td>
<td>4.2426</td>
<td>75.9585</td>
<td>37.9793</td>
</tr>
<tr>
<td>443</td>
<td>0.0125</td>
<td>5.0000</td>
<td>6.7082</td>
<td>172.6129</td>
<td>34.5226</td>
</tr>
<tr>
<td>444</td>
<td>0.0062</td>
<td>10.0000</td>
<td>9.4868</td>
<td>323.2835</td>
<td>32.3284</td>
</tr>
<tr>
<td>445</td>
<td>0.0077</td>
<td>8.0000</td>
<td>8.4853</td>
<td>265.2435</td>
<td>33.1554</td>
</tr>
<tr>
<td>446</td>
<td>0.0153</td>
<td>4.0000</td>
<td>6.0000</td>
<td>142.9468</td>
<td>35.7367</td>
</tr>
<tr>
<td>447</td>
<td>0.0038</td>
<td>16.0000</td>
<td>12.0000</td>
<td>501.2124</td>
<td>31.3258</td>
</tr>
<tr>
<td>448</td>
<td>0.0120</td>
<td>5.0000</td>
<td>6.7082</td>
<td>175.8086</td>
<td>35.1617</td>
</tr>
<tr>
<td>449</td>
<td>0.0050</td>
<td>12.0000</td>
<td>10.3923</td>
<td>388.4489</td>
<td>32.3707</td>
</tr>
<tr>
<td>450</td>
<td>0.0037</td>
<td>16.0000</td>
<td>12.0000</td>
<td>506.7341</td>
<td>31.6709</td>
</tr>
<tr>
<td>451</td>
<td>0.0042</td>
<td>14.0000</td>
<td>11.2250</td>
<td>450.2121</td>
<td>32.1580</td>
</tr>
<tr>
<td>452</td>
<td>0.0053</td>
<td>11.0000</td>
<td>9.9499</td>
<td>362.8951</td>
<td>32.9905</td>
</tr>
<tr>
<td>453</td>
<td>0.0097</td>
<td>6.0000</td>
<td>7.3485</td>
<td>210.8340</td>
<td>35.1390</td>
</tr>
<tr>
<td>454</td>
<td>0.0041</td>
<td>14.0000</td>
<td>11.2250</td>
<td>455.1421</td>
<td>32.5101</td>
</tr>
<tr>
<td>455</td>
<td>0.0071</td>
<td>8.0000</td>
<td>8.4853</td>
<td>275.0872</td>
<td>34.3859</td>
</tr>
<tr>
<td>456</td>
<td>0.0063</td>
<td>9.0000</td>
<td>9.0000</td>
<td>307.0439</td>
<td>34.4160</td>
</tr>
<tr>
<td>457</td>
<td>0.0113</td>
<td>5.0000</td>
<td>6.7082</td>
<td>181.6476</td>
<td>36.3295</td>
</tr>
<tr>
<td>458</td>
<td>0.0080</td>
<td>7.0000</td>
<td>7.9373</td>
<td>246.5741</td>
<td>35.2249</td>
</tr>
<tr>
<td>459</td>
<td>0.0278</td>
<td>2.0000</td>
<td>4.2426</td>
<td>80.8092</td>
<td>40.4046</td>
</tr>
<tr>
<td>460</td>
<td>0.0050</td>
<td>11.0000</td>
<td>9.9499</td>
<td>373.5322</td>
<td>33.9575</td>
</tr>
<tr>
<td>461</td>
<td>0.0046</td>
<td>12.0000</td>
<td>10.3923</td>
<td>405.6792</td>
<td>33.8066</td>
</tr>
<tr>
<td>462</td>
<td>0.0060</td>
<td>9.0000</td>
<td>9.0000</td>
<td>313.7392</td>
<td>34.8599</td>
</tr>
<tr>
<td>463</td>
<td>0.0090</td>
<td>6.0000</td>
<td>7.3485</td>
<td>218.5636</td>
<td>36.4273</td>
</tr>
<tr>
<td>464</td>
<td>0.0077</td>
<td>7.0000</td>
<td>7.9373</td>
<td>251.9348</td>
<td>35.9907</td>
</tr>
<tr>
<td>465</td>
<td>0.0089</td>
<td>6.0000</td>
<td>7.3485</td>
<td>220.1292</td>
<td>36.6882</td>
</tr>
<tr>
<td>466</td>
<td>0.0035</td>
<td>15.0000</td>
<td>11.6190</td>
<td>506.0466</td>
<td>33.7364</td>
</tr>
<tr>
<td>467</td>
<td>0.0263</td>
<td>2.0000</td>
<td>4.2426</td>
<td>83.1531</td>
<td>41.5765</td>
</tr>
<tr>
<td>468</td>
<td>0.0065</td>
<td>8.0000</td>
<td>8.4853</td>
<td>288.2001</td>
<td>36.0250</td>
</tr>
<tr>
<td>469</td>
<td>0.0065</td>
<td>8.0000</td>
<td>8.4853</td>
<td>289.2236</td>
<td>36.1530</td>
</tr>
<tr>
<td>470</td>
<td>0.0086</td>
<td>6.0000</td>
<td>7.3485</td>
<td>224.0720</td>
<td>37.3453</td>
</tr>
<tr>
<td>471</td>
<td>0.0032</td>
<td>16.0000</td>
<td>12.0000</td>
<td>546.3857</td>
<td>34.1491</td>
</tr>
<tr>
<td>472</td>
<td>0.0507</td>
<td>1.0000</td>
<td>3.0000</td>
<td>45.7480</td>
<td>45.7480</td>
</tr>
<tr>
<td>473</td>
<td>0.0056</td>
<td>9.0000</td>
<td>9.0000</td>
<td>326.2352</td>
<td>36.2484</td>
</tr>
<tr>
<td>474</td>
<td>0.0250</td>
<td>2.0000</td>
<td>4.2426</td>
<td>85.2364</td>
<td>42.6182</td>
</tr>
<tr>
<td>475</td>
<td>0.0033</td>
<td>15.0000</td>
<td>11.6190</td>
<td>527.4059</td>
<td>34.8271</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>476</td>
<td>0.0070</td>
<td>7.0000</td>
<td>7.9373</td>
<td>262.8604</td>
<td>37.5515</td>
</tr>
<tr>
<td>477</td>
<td>0.0035</td>
<td>14.0000</td>
<td>11.2250</td>
<td>483.9922</td>
<td>39.2852</td>
</tr>
<tr>
<td>478</td>
<td>0.0243</td>
<td>2.0000</td>
<td>4.2426</td>
<td>56.4062</td>
<td>43.2203</td>
</tr>
<tr>
<td>479</td>
<td>0.0035</td>
<td>14.0000</td>
<td>11.2250</td>
<td>497.4593</td>
<td>35.5328</td>
</tr>
<tr>
<td>480</td>
<td>0.0060</td>
<td>8.0000</td>
<td>8.4853</td>
<td>300.6247</td>
<td>37.5781</td>
</tr>
<tr>
<td>481</td>
<td>0.0034</td>
<td>14.0000</td>
<td>11.2250</td>
<td>500.9408</td>
<td>35.7815</td>
</tr>
<tr>
<td>482</td>
<td>0.0032</td>
<td>15.0000</td>
<td>11.6190</td>
<td>535.3425</td>
<td>35.6895</td>
</tr>
<tr>
<td>483</td>
<td>0.0067</td>
<td>7.0000</td>
<td>7.9373</td>
<td>269.3603</td>
<td>38.4800</td>
</tr>
<tr>
<td>484</td>
<td>0.0029</td>
<td>16.0000</td>
<td>12.0000</td>
<td>571.8179</td>
<td>35.7386</td>
</tr>
<tr>
<td>485</td>
<td>0.0232</td>
<td>2.0000</td>
<td>4.2426</td>
<td>88.5719</td>
<td>44.2860</td>
</tr>
<tr>
<td>486</td>
<td>0.0461</td>
<td>1.0000</td>
<td>3.0000</td>
<td>48.0402</td>
<td>48.0402</td>
</tr>
<tr>
<td>487</td>
<td>0.0030</td>
<td>15.0000</td>
<td>11.6190</td>
<td>544.6977</td>
<td>36.3132</td>
</tr>
<tr>
<td>488</td>
<td>0.0076</td>
<td>6.0000</td>
<td>7.3485</td>
<td>238.6094</td>
<td>39.7682</td>
</tr>
<tr>
<td>489</td>
<td>0.0075</td>
<td>6.0000</td>
<td>7.3485</td>
<td>239.4329</td>
<td>39.9055</td>
</tr>
<tr>
<td>490</td>
<td>0.0034</td>
<td>13.0000</td>
<td>10.8167</td>
<td>483.0563</td>
<td>37.1582</td>
</tr>
<tr>
<td>491</td>
<td>0.0032</td>
<td>14.0000</td>
<td>11.2250</td>
<td>518.5639</td>
<td>37.0803</td>
</tr>
<tr>
<td>492</td>
<td>0.0147</td>
<td>3.0000</td>
<td>5.1962</td>
<td>130.1654</td>
<td>43.3885</td>
</tr>
<tr>
<td>493</td>
<td>0.0146</td>
<td>3.0000</td>
<td>5.1962</td>
<td>130.6121</td>
<td>43.5374</td>
</tr>
<tr>
<td>494</td>
<td>0.0029</td>
<td>15.0000</td>
<td>11.6190</td>
<td>557.9564</td>
<td>37.1971</td>
</tr>
<tr>
<td>495</td>
<td>0.0036</td>
<td>12.0000</td>
<td>10.3923</td>
<td>456.9109</td>
<td>38.0759</td>
</tr>
<tr>
<td>496</td>
<td>0.0054</td>
<td>8.0000</td>
<td>8.4853</td>
<td>317.6751</td>
<td>39.7094</td>
</tr>
<tr>
<td>497</td>
<td>0.0071</td>
<td>6.0000</td>
<td>7.3485</td>
<td>246.0816</td>
<td>41.0136</td>
</tr>
<tr>
<td>498</td>
<td>0.0047</td>
<td>9.0000</td>
<td>9.0000</td>
<td>355.7140</td>
<td>39.5238</td>
</tr>
<tr>
<td>499</td>
<td>0.0026</td>
<td>16.0000</td>
<td>12.0000</td>
<td>602.0170</td>
<td>37.6261</td>
</tr>
<tr>
<td>500</td>
<td>0.0019</td>
<td>1.0000</td>
<td>3.0000</td>
<td>50.3990</td>
<td>50.3990</td>
</tr>
<tr>
<td>501</td>
<td>0.0035</td>
<td>12.0000</td>
<td>10.3923</td>
<td>466.3276</td>
<td>38.8606</td>
</tr>
<tr>
<td>502</td>
<td>0.0034</td>
<td>12.0000</td>
<td>10.3923</td>
<td>467.9082</td>
<td>38.9923</td>
</tr>
<tr>
<td>503</td>
<td>0.0103</td>
<td>4.0000</td>
<td>6.0000</td>
<td>174.6764</td>
<td>43.6691</td>
</tr>
<tr>
<td>504</td>
<td>0.0031</td>
<td>13.0000</td>
<td>10.8167</td>
<td>506.6400</td>
<td>38.9723</td>
</tr>
<tr>
<td>505</td>
<td>0.0029</td>
<td>14.0000</td>
<td>11.2250</td>
<td>543.8452</td>
<td>38.8461</td>
</tr>
<tr>
<td>506</td>
<td>0.0067</td>
<td>6.0000</td>
<td>7.3485</td>
<td>253.6908</td>
<td>42.2818</td>
</tr>
<tr>
<td>507</td>
<td>0.0040</td>
<td>1.0000</td>
<td>3.0000</td>
<td>51.6036</td>
<td>51.6036</td>
</tr>
<tr>
<td>508</td>
<td>0.0031</td>
<td>13.0000</td>
<td>10.8167</td>
<td>513.5010</td>
<td>39.5001</td>
</tr>
<tr>
<td>509</td>
<td>0.0028</td>
<td>14.0000</td>
<td>11.2250</td>
<td>551.2000</td>
<td>39.3714</td>
</tr>
<tr>
<td>510</td>
<td>0.0098</td>
<td>4.0000</td>
<td>6.0000</td>
<td>178.8342</td>
<td>44.7085</td>
</tr>
<tr>
<td>511</td>
<td>0.0028</td>
<td>14.0000</td>
<td>11.2250</td>
<td>554.8994</td>
<td>39.6357</td>
</tr>
<tr>
<td>512</td>
<td>0.0129</td>
<td>3.0000</td>
<td>5.1962</td>
<td>139.2730</td>
<td>46.4243</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

This chapter concludes the studies in Chapter II, III, and IV, in which we examined the problem of specifying single-item service objectives in a multi-item inventory system subject to an overall service objective. A list of the notation we use may be found at the end of Chapter II.

In that chapter we formulated this inventory problem as a constrained nonlinear program (NLP) and developed the Generalized Knapsack Duality (GKD) Algorithm to solve it. This algorithm finds the Lagrange constraint multiplier of the NLP and from it computes the (s,S) policies for all items in the inventory system. In Chapter III we reported computational experience with several 32-item inventory systems which have a structure typical of many real-world inventory systems. This structure is called an 80/20 value structure, and is one in which twenty percent of the items represent eighty percent of the value of the systems, where value is taken to be \( h_u \). These items are the high-value items, and the other eighty percent are the low-value items. In order to investigate the performance of the GKD Algorithm, we assumed that the demand distribution is Negative Binomial and compared the (exact) expected holding cost of the (s,S)
policies produced by the GKD Algorithm with the (s,S) policies produced by a method popular with managers, the Identical Service Approach (The replenishment costs for these methods are the same, and represent about half of the total expected cost). Given a targeted overall service-level $\alpha$ (we targeted 85%), the Identical Service Approach sets (s,S) policies by the Power Approximation of Ehrhardt (Mosier [1981]) so that every item has a service-level $\alpha$. (The GKD Algorithm varies individual service-levels while still maintaining an average overall service-level $\alpha$.) We showed that there is a significant cost savings when using the GKD Algorithm rather than the Identical Service Approach to manage these inventory systems.

In Chapter IV we reported several sensitivity tests performed on the base-case and worst-case inventory systems of Chapter III. In particular, we performed sensitivity tests on the $D_i$ and on the lower bounds for $S_i$. We also examined several sampling schemes and sizes for inventory aggregation for larger (128- and 512-item) systems. Since most real-world inventory systems contain thousands of items, typically managers sample from the system and make decisions about policies for all items in the system based on the sample. We examined sampling schemes (both random and nonrandom) of several sizes which maintained the 80/20 value structure and the overall service constraint weights. We summarize the conclusions and recommendations given in that chapter as follows:
1. Specify all $D_i$ using Ehrhardt's Power Method (Mosier [1981]).

2. Specify the lower bounds on $S_i$ as $D_i + r\mu_i^*$, where $r$ is taken as large as possible while still including as feasible the targeted service-level and while the expected holding cost is significantly smaller than that of the Identical Service Approach.

3. Sample about 32 items using a central nonrandom uniform sampling scheme to create the inventory subsystem for aggregation (sample items from a stratum containing items 1, 2, ..., $N$ by taking items $N/2M$, $3N/2M$, $5N/2M$, ..., $(2M-1)N/2M$, where the fractions are rounded to the nearest integer). Recompute the holding and replenishment costs to maintain the value structure of the system, and recompute the constraint weights to maintain the constraint-weight structure of the system (multiply the costs by $\frac{\sum_i h_i \mu_i}{\sum_i h_i \mu_i}$ and the system constraint weights by $N/M$ for each of the $M$ items sampled from 1, 2, ..., $N$).

In Section 2 of this chapter we prescribe a detailed implementation procedure of the GKD Algorithm with sampling for a large-scale inventory system with an overall service objective. In Section 3 we report the results of implementing this procedure for the 512-item system described in Chapter IV. Section 4 contains conclusions and directions for future research, and the Appendix contains a listing of the FORTRAN code of the GKD Algorithm and related programs.

2. IMPLEMENTATION PROCEDURE

Based on the conclusions and recommendations of Chapters II, III, and IV, we offer the following procedure to specify $(s, S)$
policies in large-scale inventory systems. Our empirical evidence suggests that this procedure is a significant improvement over the Identical Service Approach often used by managers, capable of substantially reducing operating costs in an inventory system with an overall service objective.

The procedure is as follows:

1. Determine the desired range of service-levels and the targeted service-level. This is a managerial decision, but typically the range is 80% to 90% with a target of 85%.

2. Determine r as follows. For \( r = 0, 0.5, 0.6, 0.7, 0.8, 0.9 \) and 1, input the following values to the program LSMISS (see the Appendix to this chapter):

   - \( \text{TITLE} = \) user-supplied title
   - \( \text{LBAL} = \) smallest service-level to consider
   - \( \text{UBAL} = \) largest service-level to consider
   - \( \text{INCRAL} = \) service-level increment between \( \text{LBAL} \) and \( \text{UBAL} \)
   - \( \text{NSIG} = 8 \)
   - \( \text{MAXFN} = 100 \)
   - \( \text{NI} = \) number of items in inventory system
   - \( \text{SCHEME} = 1 \)
   - \( \text{ENTIRE} = 0 \)
   - \( \text{STRA}(1) = \text{NI} \)
   - \( \text{NUMB}(1) = \) about 32.

Using the output from LSMISS, graph the expected holding costs for the various service-levels for each value of \( r \) and for the
Identical Service Approach (compare Figure 34). Take as $r$ the largest value of $r$ such that LBAL is achievable and such that there is an acceptable cost savings below that of the Identical Service Approach.

**NOTE 1:**

The GKD Algorithm assumes that the demand distribution is Logistic, an approximation to the Normal distribution (Section 2 of Chapter II). If the actual distribution is not well-approximated by a Normal distribution, then the service-level of the $(s,S)$ policies produced by the GKD Algorithm may be different from the specified service-level. The user may therefore have to try various LBAL, UBAL and INCRAL. If the service-level specifications are as described in Step (1) and the demand distribution is well-approximated by a Gamma or Negative Binomial distribution, our experience suggests using

- LBAL = .75
- UBAL = .85
- INCRAL = .01.

**NOTE 2:**

We recommend a sample size (NUMB(1)) of about 32 because in our experience larger samples do not significantly improve the performance of the sampling scheme.

**NOTE 3:**

The parameters NSIG and MAXFN are used to terminate IMSL subroutines in the GKD Algorithm. We have found the recommended values to be satisfactory in every experiment performed in this paper.
3. Using the output from LSMISS and linear interpolation, determine \( \lambda^* \) for a targeted service-level. Input the following values to the program MISSNLSI (see the Appendix to this chapter):

- **TITLE** = user-supplied title
- **LS** = \( \lambda^* \)
- **NSIG** = 8
- **MAXFN** = 100
- **NI** = number of items in system
- **OUT** =
  - 0 write output only to printer
  - 1 write output to both printer and an external file

The output of MISSNLSI contains all single-item policy specifications.

**NOTE:**

The parameter OUT allows the user to save the output of MISSNLSI in an external file. See the Appendix to this chapter for details.

3. EXAMPLE

In the previous chapter we performed Steps (1) and (2) of this procedure for the 512-item base-case inventory system, and decided to use \( r = 0.6 \). We now perform Step (3) for this system. We used the output from LSMISS for the scheme described in Step (2) of the procedure, which corresponds to Experiment 1 of Table 3. For this experiment, Figure 43 reports the values of \( \lambda^* \) for the service-levels near 85%, our targeted service-level. Linear interpolation gives \( \lambda^* = 4207 \). The output from the program MISSNLSI is summarized
in Figures 44, 45, and 46. The operating characteristics of the individual items follow the same pattern noted in Chapter III, in that the high-value items are stocked at low service-levels and the low-value items are stocked at high service-levels.

4. CONCLUSIONS, FUTURE RESEARCH DIRECTIONS

Using the GKD Algorithm of Chapter II and the extensive computational and sensitivity experience reported in Chapters III and IV, we have described a detailed implementation procedure of the GKD Algorithm using sampling to manage a large-scale inventory system with an overall service objective. We have shown that for inventory systems with a structure typical of that found in many real-world inventory systems, this procedure is practical and results in a significant cost savings below that of the Identical Service Approach often used by managers.

We recommend the following future research directions:

1. This procedure should be tested in a statistical environment in which the moments of the demand distribution are estimated from recent demand history. In light of the results reported in Ehrhardt [1976], Estey and Kaufman [1975], Klincewicz [1976a], MacCormick [1974, 1977], and MacCormick et al. [1977], the performance of the procedure will probably degrade somewhat, and this should be investigated. Considering the cost savings, however, we conjecture that in a statistical environment our procedure will still result in significant cost savings below that of the Identical Service Approach.
2. The variable \( r \) should be included in the GKD optimization routine. This would improve accuracy and make an implementation procedure simpler in that Step (2) of the procedure would be eliminated.

3. The GKD Algorithm assumes the demand distribution is Logistic, while demand distributions are typically skewed like a Gamma or Negative Binomial distribution. As reported in Section 3 of Chapter III, this can lead to significant degradation in algorithm performance (although we showed it is possible to improve such performance). Moreover, as discussed in Note 1 under Step (2) of our implementation procedure, the actual service-level of the policies produced by the GKD Algorithm may differ from that specified. We recommend that the Logistic distribution be replaced by a distribution which better approximates a Gamma (or Negative Binomial) distribution. The distribution to use is not obvious; an examination of the proof of Theorem 8 of Chapter II suggests that the distribution function must have a fairly simple form in order to prove algorithm convergence in the way we did \( \lambda(\theta) \) needs to be rational in \( \theta \). We have been unable to find a distribution that better approximates a Gamma distribution and yet is simple enough to allow an extension of the proof of Theorem 8.

4. Inventory systems with a value structure different than 80/20 should be studied. Based on previous but unreported experiments, we conjecture that the algorithm performance will degrade for \( X/20 \) systems as \( X \) decreases.
Figure 43. 32-Item Central Nonrandom Uniform Sample of 512-Item Inventory System.
Figure 44. 512-Item Inventory System
Figure 45: 512-Item Inventory System
Figure 46. 512-Item Inventory System
APPENDIX TO CHAPTER V

This appendix contains a listing of the FORTRAN computer program for the programs LSMISS and MISSNLSI. The subroutines EXACT and POWAPP are the same for both of these programs, and the subroutine COMPAR is slightly different for the two programs.
PROGRAM LSNMISS(NL) : LARGE - SCALE
MULTI - ITEM (S,S)
NORMAL DEMAND DISTRIBUTION APPROXIMATION
LOWER BOUNDS ON BIG S

VERSION DATED C7/20/83

THE INPUT IS FROM CARDS

TITLE = USER - SUPPLIED TITLE (LE 36 ALPHA CHARACTERS)
NI = NUMBER OF ITEMS (NI LE 512)
K = SET-UP ORDERING COST
L = LEADTIME + 1
HO = UNIT HOLDING COST
MU = MEAN DEMAND
SD = DEMAND STANDARD DEVIATION
WT = CONSTRAINT WEIGHT

NSIG = NUMBER OF SIGNIFICANT DIGITS DESIRED FROM THE IMSL
ROUTINES ZBRENT AND ZFALSE
EPS = 1/(10 ** NSIG)
MAXPN = MAXIMUM NUMBER OF ITERATIONS ALLOWABLE BY USER FOR
THE IMSL ROUTINES ZBRENT AND ZFALSE
INR = (OUTPUT) ERROR PARAMETER FROM IMSL ROUTINES AND EXACT

THE FOLLOWING ARE INPUT TO BE COMPUTED BY GKD

LITS = LITTLE S
BIGS = BIG S
D = REORDER QUANTITY BIG S - LITTLE S
BSL = R * L * MU
    = LOWER BOUND FOR BIG S (MAY BE INCREASED BY GKD)
AL = OVERALL SERVICE-LEVEL (0 < AL < 1)

THIS PROGRAM COMPUTES POLICIES FOR
AL = LBAL, LBAL+INCRAL, LBAL+2*INCRAL, . . . ,UBAL

THE FOLLOWING ARE COMPUTED BY EHHRARDT'S POWER METHOD

PLITS = LITTLE S
PBIGS = BIG S

THE FOLLOWING ARE COMPUTED EXACTLY

HGD = EXPECTED HOLDING COST FOR GKD POLICIES
TGKD = EXPECTED TOTAL COST FOR GKD POLICIES
AVSL = WEIGHTED-AVERAGE SERVICE-LEVEL FOR GKD POLICIES
HPOW = EXPECTED HOLDING COST FOR POWER POLICIES
TPow  =  Expected total cost for power policies
PAviL  =  Weighted-average service-level for power policies

The following parameters are used in the stratified sampling
subroutines STSAMP and RNDSMS

SCHEME =  0  Do not use any sampling scheme
         1  Use stratified sampling scheme
         2  Use random stratified sampling scheme

SAMP  =  Number of items in the sample
NUMST  =  Number of strata to sample from
STA(I)  =  Last item in strata I
NUMB(I)  =  Number of items from strata I in the sample

The following variable is needed on input if scheme = 2
SEED  =  Seed for Psudo-random number generator GGUD (IMSL)
       Require seed in (1,2147483674)

ITEM(1), ..., ITEM(SAMP) are the items in the sample

ENTIRE  =  0  Otherwise
         1  Evaluate expected costs, etc., for entire inventory
             system (this is for testing)

******************************************************************************

Input the system parameters as follows:

<table>
<thead>
<tr>
<th>TITLE</th>
<th>LBAL</th>
<th>UBAL</th>
<th>INCRA1</th>
<th>NSIG</th>
<th>MAXPW</th>
<th>WI</th>
</tr>
</thead>
<tbody>
<tr>
<td>WT(1)</td>
<td>D(1)</td>
<td>K(1)</td>
<td>L(1)</td>
<td>HO(1)</td>
<td>MU(1)</td>
<td>SD(1)</td>
</tr>
<tr>
<td>WT(2)</td>
<td>D(2)</td>
<td>K(2)</td>
<td>L(2)</td>
<td>HO(2)</td>
<td>MU(2)</td>
<td>SD(2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SCHEME   ENTIRE
SAMP  NUMST
STRA(1)  STRA(2)  ...  STRA(NUMST)  -  16 to A
NUMB(1)  NUMB(2)  ...  NUMB(NUMST)  -  Line
SEED  ] Only if scheme = 2

The input format is:

XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX X
ALL DATA STARTS IN COLUMN 1 (EXCEPT TITLE STARTS IN COLUMN 2), AND THERE IS A SINGLE SPACE BETWEEN EACH DATA ENTRY.

REAL*8 LIIS(512),PIGS(512),BSL(512),D(512)  
REAL*8 BSL1(512),D1(512),BSLB(512)  
REAL*8 K(512),I(512),HO(512),MU(512),SD(512),WT(512)  
REAL*8 K1(512),L1(512),HO1(512),MU1(512),SD1(512),WT1(512)  
REAL*8 AL, EPS, UBAL, UBAL, INCRAL, LS(20), LS1, SLI, SON, SEED, R

INTEGER NI, N, NSIG, MAXFN, TITLE(8), J, NUM, ISEM(512), ENTRI  
INTEGER SCHEME, SAMP, NUMST, STREX(512), NUMB(512), ISEED

READ(1,3) (TITLE(J),J=1,8)  
3 FORMAT(8A4)  
READ(1,5) LBA, UBA, INCR, NSIG, MAXFN, NI  
5 FORMAT(3(F7.5,1X),3(I6,1X))  
EPS = 10.**(-NSIG)  

DO 10 I = 1,NI  
READ(1,8) WT(I), D(I), K(I), L(I), HO(I), MU(I), SD(I), R  
BSL(I) = D(I)*K*L(I)*MU(I)  
8 FORMAT(8(F8.4,1X))  
10 CONTINUE

READ(1,11) SCHEME, ENTRI
11 FORMAT(12,1X,12)  
IF (SCHEME.EQ.0) GO TO 19  
READ(1,15) SAMP, NUMST

NUMST, STRA AND NUMB MUST BE CHANGED AS FOLLOWS FOR INPUT TO  
THE SUBROUTINES STSAMP AND ENDST---SEE THE DOCUMENTATION FOR  
THOSE SUBROUTINES

NUMST = NUMST+1  
STRA(1) = 0  
NUMB(1) = 0  
READ(1,15) (STRA(I),I=2,NUMST)  
READ(1,15) (NUMB(I),I=2,NUMST)
15 FORMAT(16(I4,1X))
C
ISEED = 0
IF (SCHEME.EQ.2) READ(1,16) ISEED
16 FORMAT(I10)
SEED = ISEED
C
MAKE THE WEIGHTS WT INTO EQUIVALENT CONVEX WEIGHTS WT
C
19 SUM = 0.
DO 20 I = 1,NI
20 SUM = SUM+WT(I)
SUM = 1.00D/SUM
DO 25 I = 1,NI
25 WT(I) = SUM*WT(I)
C
M = 2*(UBAL-LBAL)/INCRAL
IF (SCHEME.EQ.0) GO TO 300
C
CREATE THE INVENTORY SUBSYSTEM VIA STRATIFIED SAMPLING AND PUT
THE SUBSYSTEM IN N, K1, L1, . . . , BSL1
C
IF (SCHEME.EQ.1) CALL STSSAMP(NI,K,L,H0,MU,SD,D,WT,BSL,N,K1,L1,
& H01,MU1,SD1,D1,WT1,BSL1,SAMP,
& NUMST,STRA,NUMB,ITEM)
C
IF (SCHEME.EQ.2) CALL BNDSTS(NI,K,L,H0,MU,SD,D,WT,BSL,N,K1,L1,
& H01,MU1,SD1,D1,WT1,BSL1,SAMP,
& NUMST,STRA,NUMB,ITEM,SEED)
C
DO 200 II = 1,M
AL = LBAL+(II-1.00)*INCRAL
SLI = AL
NUM = N
DO 30 III = 1,H
30 BSLB(III) = ESL1(III)
C
COMPUTE LAMBDASTAR (LSI) USING THE GKD ALGORITHM WITH THE
INVENTORY SUBSYSTEM
C
CALL GKD(NUM,I1,HC1,MU1,SL1,WT1,D1,SL1,LITS,BIGS,BSLB,EP1,
& NSIG,MAIPN,LSI)
L3(II) = LSI
C
WRITE(3,40) SCHEME,(STRA(I),I=2,NUMST)
40 FORMAT(' RESULTS FOR INVENTORY SUBSYSTEM'//SCHEME ',12//'
& ' STRATA ',15(815//))
WRITE(3,41) (NUM(I),I=2,NUMST)
41 FORMAT(* NUMBER ',15(815//))
WRITE(3,42) (ITEM(I),I=1,SAMP)
42 Format(' ITEM ',15(815//))
42 FORMAT( 'ITEM ', 15(B15/) )
WRITE(3,43) ISEED
43 FORMAT( 'SEED = ',I20/
C
C COMPUTE THE (UNIFORM SERVICE-LEVEL) POWER APPROXIMATION POLICIES
C AND COMPARE THEIR ASSOCIATED COSTS WITH THE COSTS ASSOCIATED WITH
C THE GKD POLICIES
C
CALL COMPAR(N,K1,L1,H01,MU1,SD1,WT1,BIGS,LITS,TITLE,LSI,AL
IF (ENTIRE.EQ.0) GO TO 200
C
C COMPUTE THE OPTIMAL POLICIES ASSOCIATED WITH THE MULTIPLIER
C LAMBDASON (LSI)
C
CALL BIGSLS(NI,L,H0,MU,SD,D,WT,BSL,LSI,EPS,NSIG,MAIFN,BIGS,LITS)
C
WRITE(3,45)
45 FORMAT( 'I' )
WRITE(3,50) SCHEME,(STBA(I),I=2,NUMST)
50 FORMAT( 'RESULTS FOR INVENTORY SYSTEM/'SCHEME ',I2//'
& 'STBA ',15(B15/) )
WRITE(3,51) (NUMB(I),I=2,NUMST)
51 FORMAT( 'NUMBER ',15(B15/) )
WRITE(3,52) (ITEM(I),I=1,SAFP)
52 FORMAT( 'ITEM ',15(B15/) )
WRITE(3,53) ISEED
53 FORMAT( 'SEED = ',I20/
C
C COMPUTE THE (UNIFORM SERVICE-LEVEL) POWER APPROXIMATION POLICIES
C AND COMPARE THEIR ASSOCIATED COSTS WITH THE COSTS ASSOCIATED WITH
C THE BIGSLS POLICIES
C
CALL COMPAR(NI,K,L,H0,MU,SD,D,WT,BIGS,LITS,TITLE,LSI,AL
200 CONTINUE
STOP
C
300 DO 350 II = 1,M
AL = LBAL+(II-1.DO)*INCRAL
SLI = AL
NUM = NI
DO 320 III = 1,NI
320 BSLB(III) = BSL(III)
C
C COMPUTE LAMBDASON (LSI) USING THE GKD ALGORITHM WITH THE
C ENTIRE INVENTORY SYSTEM
C
CALL GKD(NUM,1,HC,MU,SD,WT,D,SLI,LITS,BIGS,BSLB,EPS,
& NSIG,MAIFN,LSI)
SUBROUTINE STSAMP(NI,K,L,H0,MU,SD,D,WT,BSL,N,K1,L1,H01,MU1,
&
SD1,D1,WT1,BSL1,SAMP,NUMST,STRA,NUME,ITEM)

STSAMP : STRATIFIED SAMPLING

THIS SUBROUTINE TAKES A STRATIFIED SAMPLE FROM THE INVENTORY
SYSTEM ACCORDING TO THE FOLLOWING PARAMETERS:

SAMP = NUMBER OF ITEMS IN THE SAMPLE
NUMST = NUMBER OF STRATA + 1
STRA(1) = 0
STRA(I+1) = LAST ITEM IN STRATA I, I = 1, 2, . . . , NUMST-1
NUMB(1) = 0
NUMB(I+1) = NUMBER OF ITEMS IN THE SAMPLE FROM STRATA I,
I = 1, 2, . . . , NUMST-1

FOR A GIVEN STRATA WITH NN ITEMS FROM WHICH M ARE TO BE
SAMPLED, THE ITEMS NN/2M, 3NN/2M, 5NN/2M, . . . , (2M-1)NN/2M
ARE TAKEN, WHERE THESE FRACTIONS ARE BOUNDED-OFF TO THE NEAREST
INTEGER

THESE ITEMS ARE DENOTED
ITEM(1), ITEM(2), . . . , ITEM(M)

THE CONSTRAINT WEIGHTS (WT), THE UNIT HOLDING COSTS (HO),
AND THE REPLENISHMENT SET-UP COSTS (K)
ARE RECOMPUTED TO REFLECT THE IDEA OF REPLACING ITEMS NOT IN
THE SAMPLE BY THOSE ITEMS NEAR THEM IN THE SAMPLE (IE, THOSE
IN THE SAME STRATA)
IN PARTICULAR, THESE COSTS ARE MULTIPLIED BY THE TOTAL
VALUE OF THE STRATA (VALUE = H*MU) DIVIDED BY THE TOTAL VALUE
OF THE SAMPLE FROM THAT STRATA
THE WEIGHT FOR EACH ITEM IN A GIVEN STRATA IS THE SAME, BEING
THE TOTAL WEIGHT OF THE STRATA DIVIDED BY THE NUMBER OF ITEMS
SAMPLED FROM THE STRATA

REAL*8 K(512), L(512), H0(512), MU(512), SD(512), WT(512),
&
D(512), BSL(512)
REAL*8 K1(512), L1(512), HO1(512), MU1(512), SD1(512),
&
WT1(512), D1(512), BSL1(512)
REAL*8 WEIGHT, RATIO, XH, XNSTR, HALP, ZERO, HWT, HSAMWT
C INTEGER SAMP, NUMST, STRA(512), NUMB(512), ITEM(512), FIRST, LAST
C INTEGER N, 0, I, J, IJ, IJJ, JJ, NUMST, N, STRA, NSTR
C DATA HALP, ZERO/0.5D0, 0.0D0/
C
N = SAMP
IJ = 0
DO 60 I = 2, NUMST
STRAI = STRA(I-1)
M = NUMB(I)
XM = M
C IF M = 0, THEN NO ITEM IN THE SAMPLE IS TO BE TAKEN FROM THIS
C STRATA
C IF (M.EQ.0) GO TO 60
C NSTR = STRA(I) - STRAI
NSTB = NSTR
BATIO = NSTB/XM
HWT = ZERO
WEIGHT = ZEBC
DO 20 J = 1, NSTR
JJ = STRAI+J
HWT = HWT+HO(JJ)*MU(JJ)
20 WEIGHT = WEIGHT+W1(JJ)
WEIGHT = WEIGHT/XM
HSAMWT = ZEBC
FIRST = IJ+1
DO 40 J = 1, M
IJ = IJ+1
IJJ = STRAI+(J-HALF)*BATIO+HALF
ITEM(IJ) = IJJ
K1(IJ) = K(IJJ)
L1(IJ) = L(IJJ)
H01(IJ) = HO(IJJ)
MU1(IJ) = MU(IJJ)
SD1(IJ) = SD(IJJ)
D1(IJ) = D(IJJ)
BSL1(IJ) = BSL(IJJ)
HSAMWT = HSAMWT+HO1(IJ)*MU1(IJ)
WT1(IJ) = WEIGHT
40 CONTINUE
LAST = FIRST+M-1
HWT = HWT/HSAMWT
DO 50 J = FIRST, LAST
HO1(J) = HWT*HO1(J)
50 K1(J) = HWT*K1(J)
60 CONTINUE
RETURN
END

SUBROUTINE RNDSTS (H1, K, L, HO, MU, SD, D, WT, BSL, N, K1, L1, HO1, MU1,
SD1, D1, WT1, BSL1, SAMPL, NUMST, STRA, NUMB, ITEM, SEED)

BNDSTS : RANDOM STRATIFIED SAMPLING

THIS SUBROUTINE TAKES A RANDOM STRATIFIED SAMPLE FROM THE
INVENTORY SYSTEM ACCORDING TO THE FOLLOWING PARAMETERS:

SAMP = NUMBER OF ITEMS IN THE SAMPLE
NUMST = NUMBER OF STRATA + 1
STRA(1) = 0
STRA(I+1) = LAST ITEM IN STRATA I, I = 1, 2, . . . , NUMST-1
Numb(I) = 0
Numb(I+1) = NUMBER OF ITEMS IN THE SAMPLE FROM STRATA I,
I = 1, 2, . . . , NUMST-1
SEED = SEED FOR PSEUDO-RANDOM NUMBER GENERATOR GGUD (IMSL)
REQUIRE SEED IN (1.0D0, 2.0D0, 3.0D0)

FOR A GIVEN STRATA WITH N ITEMs FROM WHICH M ARE TO BE
SAMPLED, THE ITEMS ARE SAMPLED RANDOMLY ACCORDING TO A DISCRETE
UNIFORM DISTRIBUTION

THESE ITEMS ARE DENOTED

ITEM(1), ITEM(2), . . . . , ITEM(M)

THE CONSTRAINT WEIGHTS (WT), THE UNIT HOLDING COSTS (HO),
AND THE REPLENISHMENT SET-UP COSTS (K)
ARE RECOMPUTED TO REFLECT THE IDEA OF REPLACING ITEMS NOT IN
THE SAMPLE BY THOSE ITEMS NEAR THEM IN THE SAMPLE (I.E., THOSE
IN THE SAME STRATA)
IN PARTICULAR, THESE COSTS ARE MULTIPLIED BY THE TOTAL
VALUE OF THE STRATA (VALUE = H*MU) DIVIDED BY THE TOTAL VALUE
OF THE SAMPLE FROM THAT STRATA

THE WEIGHT FOR EACH ITEM IN A GIVEN STRATA IS THE SAME, BEING
THE TOTAL WEIGHT OF THE STRATA DIVIDED BY THE NUMBER OF ITEMS
SAMPLED FROM THE STRATA

REAL*8 K (512), L (512), H0 (512), MU (512), SD (512), WT (512),
D (512), BSL (512)
REAL*8 K1 (512), L1 (512), H01 (512), MU1 (512), SD1 (512),
WT1 (512), D1 (512), BSL1 (512)
REAL*8 WEIGHT, PATIO, IN, INSTR, ZERO, HWT, HSAMWT, SEED, DSEED

INTEGER SAMP, NUMST, STRA (512), NUMB (512), ITEM (512), FIRST, LAST
INTEGER HI, W, I, J, IJ, JJ, STRAI, NSTR, INDEX, ONE,
ITEM (1)
DATA ZERO,ONE/0.0D0,1/

DSEED = SEED
N = SAMP
IJ = 0
DO 60 I = 2,NUMB
STPAI = STRA(I-ONE)
M = NUMB(I)
IN = M

IF M = 0, THEN MC ITEM IN THE SAMPLE IS TO BE TAKEN FROM THIS STRATA

IF (M.EQ.0) GC TO 60

MSTR = STRA(I)-STRAI
XMSTR = MSTR
RATIO = XMSTR/M
HWT = ZERO
WEIGHT = ZEP0
DO 20 J = CNE,MSTR
JJ = STRAI+J
HWT = HWT+HC(JJ)*MU(JJ)

20 WEIGHT = WEIGHT+WT(JJ)
WEIGHT = WEIGHT/XM
HSAMWT = ZEP0
FIRST = IJ+ONE
DO 40 J = CNE,M

25 CALL GGUD(DSEED,MSTR,ONE,IR)
IJIIJ = STRAI+IB(CNE)
DO 30 INDEX = CNE,IJ
IF (IJIIJ.EQ.ITEM(INDEX)) GO TO 25

30 CONTINUE
IJ = IJ+ONE
ITEM(IJ) = IJIIJ
K1(IJ) = K(IJIIJ)
L1(IJ) = L(IJIIJ)
HO1(IJ) = HO(IJIIJ)
MU1(IJ) = MU(IJIIJ)
SD1(IJ) = SD(IJIIJ)
D1(IJ) = D(IJIIJ)
BSL1(IJ) = BSL(IJIIJ)
HSAMWT = HSAMWT+HC1(IJ)*MU1(IJ)
WT1(IJ) = WEIGHT

40 CONTINUE
LAST = FIRST+M-CNE
HWT = HWT/HSAMWT
DO 50 J = FIRST,LAST
HO1(J) = HWT*HC1(J)
50 K1(J) = HWT*K1(J)
60 CONTINUE
   RETURN
END

C
SUBROUTINE BIGSLS(NI,L,HO,MU,SD,D,WT,BSL,LSI,EPS,NSIG,MAXFN,
   &
   BIGS, LITS)
C
THIS SUBROUTINE COMPUTES THE OPTIMAL (S,S) POLICIES ASSOCIATED
WITH THE KUHN-TUCKER CONSTRAINT MULTIPLIER LAMBDA-STAR (LS1)
C
C
COMMON /FINDTS/AI,BI,PI,GI,HI,LAM
C
REAL*8 LITS(512),BIGS(512)
REAL*8 L(512),HO(512),MU(512),SD(512),WT(512),
   &
D(512),EPS(512)
REAL*8 EPS,LSI,LI,HOI,MUI,SDI,DI,MSI,WTI,TSI,LAM,
   &
   RHI,QI,GAI,DEI,AI,BI,PI,GI,HI,VAR1,VAR2,ETI,TUI,
   &
   KSI,LPTUI,BSLI,X,TUI,UB,ZERO,XONE,HALF,CONST
C
INTEGER NI,NSIG,MAXFN,MAX,ONE,CNTR
C
REAL*8 LAMTH
EXTERNAL LAMTH
C
ZERO = 0.
ONE = 1
XONE = 1.
HALF = .5
CONST = 1.700436904
C
DO 60 I = ONE,NI
   HOI = HO(I)
   MUI = MU(I)
   LI = L(I)
   MSI = LI*MUI
   SDI = SD(I)
   WTI = WT(I)
   DI = D(I)
   RHI = MUI/DI+((MUI+SDI/SDI/MUI)*HALF)
   QI = XONE-RHI
   GAI = CONST/(LI*SDI)
   DEI = DEXP(GAI*DI)
   AI = DI*HOI
   VAR1 = XGNE+DEI
   BI = AI*(QI+VAR1)
   EI = AI*(DEI+QI+VAR1)
   FI = AI*QI*DEI
   VAR1 = DI*RHI*GAI
   60 CONTINUE
\[ \text{VAR2} = Q_i \cdot (DEI - XCN) \]
\[ GI = WTI \cdot (VAB1 + \text{VAR2}) \]
\[ HI = WTI \cdot (\text{VAR1} \cdot DEI + \text{VAR2}) \]

**Determine BSL by the largest value of \( \theta \), \( Tu \), and hence the smallest value of \( Big S \), so that \( D(\lambda) / D(\theta) \leq 0 \), guaranteeing that \( \lambda(\theta) \) is invertible on \((0, Tu)\). If the user-supplied BSL is greater than this, it becomes the BSL used.**

\[ \text{ETI} = GI \cdot (BI \cdot HI - GI \cdot (EI \cdot HI - FI)) - AI \cdot HI \cdot HI \]

If \( \text{ETI} \) is less than 0, \( \lambda(\theta) \) may not be convex, and there is no guarantee that the GkD algorithm will converge to a local minimum—-the user should exercise care.

If \( \text{ETI} < 0 \), write (3, 30) ETI

30 FORMAT(' ETA(\',16,') = ''E20.6,'', LT 0')

\[ \text{CMTR} = -1 \]
35 \[ \text{CMTR} = \text{CMTR} + CN \]

\[ \text{TUI} = (\text{DEXP} \cdot (G\theta \cdot (\text{MSI} - \text{CNTR}))) / DEI \]
\[ KSI = (FI / HI) - (AI / (GI \cdot TUI + TUI)) \]
\[ \text{VAR1} = GI \cdot HI \cdot TUI \]
\[ \text{LPTUI} = KSI - (\text{ETI} / (GI \cdot HI \cdot \text{VAR1} \cdot \text{VAR1})) \]

If \( \text{LPTUI} \geq \text{ZERO} \) GO TO 35

\[ \text{BSL} = \text{CMTR} + \text{DI} \]
\[ I = \text{BSL} \]
\[ \text{BSL} = \text{BSL} \]

IF \( X \), GT \( \text{BSL} \) \( \text{BSL} \) \( X \)

\[ \text{TUI} = \big{\text{DEXP} \cdot (G\theta \cdot (\text{MSI} - (\text{BSL} - \text{DI}))) / DEI \}

\[ \text{LAM} = LSI \]
\[ \text{MAX} = \text{MAXPN} \]
\[ \text{LB} = \text{ZERO} \]
\[ \text{UB} = \text{TUI} \]

IF \( \text{LAM} \), GT \( \text{ZERO} \) GO TO 50

CALL ZBHENT \( \text{LAM}, \text{EPS}, \text{NSIG}, \text{LB}, \text{UB}, \text{MAX}, \text{IER} \)

\[ \text{TSI} = \text{UB} \]

\[ \text{BIGS} = \text{MSI} - \big{\text{DLOG} \cdot (\text{TSI})} \big{/ G\theta \}

Because of round-off error, it may be that \( \text{Big S} \) LT \( \text{BSL} \).

If so, set \( \text{Big S} = \text{BSL} \).
IF (BIGS(I) .LT. BSL(I)) BIGS(I) = BSL(I)
LITS(I) = BIGS(I) - DI
GO TO 60
50 BIGS(I) = BSL(I)
LITS(I) = BIGS(I) - DI
60 CONTINUE
RETURN
END
REAL FUNCTION LAMTH*8 (THETA)
COMMON /FINDTS/AI, BI, PI, GI, HI, LAM
REAL*8 AI, BI, PI, GI, HI, LAM, THETA
LAMTH = AI*THETA*(BI-LAM*GI*THETA*(EI-LAM*HI+THETA*FI))
RETURN
END
SUBROUTINE CCMBAB(KI, K, L, HO, MU, SD, WT, BIGS, LITS, TITLE, LSI, AL)
THIS SUBROUTINE DOES THE FOLLOWING:
(1) EVALUATES EXACTLY THE HOLDING AND REPLENISHMENT COST, AND
THE SERVICE-LEVEL FOR THE POLICIES (LITS, BIGS) UNDER THE
ASSUMPTION OF NEGATIVE-BINOMIAL DEMAND (IN EXACT)
(2) COMPUTES APPROXIMATELY OPTIMAL (S, S) POLICIES WITH UNIFORM
SERVICE-LEVELS (IN POMAPP)
(3) EVALUATES THE POLICIES DESCRIBED IN (2) EXACTLY AS IN (1)
(4) PRINTS THE RESULTS OF STEPS (1) AND (3)
REAL*8 LITS(512), BIGS(512)
REAL*8 K(512), LI(512), HO(512), MU(512), SD(512), WT(512)
REAL*8 AL, HGKD, HP0W, TGKD, TPOW, LSI, LITSI, BIGSI
REAL*8 KI, LI, HOI, MUI, SDI, SLI, HCI, RCI, AVSL, PAVSL
INTEGER NI, TITLE(8)
AVSL = 0.
HGKD = 0.
TGKD = 0.
DO 70 I = 1, NI
KI = K(I)
LI = LI(I)
HOI = HO(I)
MUI = MU(I)
SDI = SD(I)
BIGSI = BIGS(I)
LITSI = LITS(I)
CALL EXACT(KI, LI, HOI, MUI, SDI, LITSI, BIGSI, SLI, HCI, RCI, IER)
IF (IER.EQ.1) GO TO 300
AVSL = AVSL+SL*WT(I)
HGKD = HGKD+HCI
TGKD = TGKD+HCI+BCI
70 CONTINUE
C
PAVSL = 0.
HPOW = 0.
TPOW = 0.
DO 75 I = 1,NI
KI = K(I)
LI = L(I)
HOI = HO(I)
MU1 = MU(I)
SDI = SD(I)
CALL POWAPP(KI,LI,HCI,MU,SDI,AVSL,LITSI,BIGSI)
CALL EXACT(KI,LI,HCI,MU,SDI,LITSI,BIGSI,SLI,HCI,BCI,IER)
IF (IER.EQ.1) GO TO 300
PAVSL = PAVSL+SL*WT(I)
HPOW = HPow+HCI
TPCW = TPow+HCI+BCI
75 CONTINUE
C
C PRINT SUMMARY
C
WRITE(3,98) (TITLE(J),J=1,8),LSI
98 FORMAT(/T2,8A4,* LAMBDA STAR = ',F12.4,/*)
WRITE(3,100) A1,AVSL,PAVSL,HGKD,HPow,TGKD,TPOW
100 FORMAT(T3,'SPECIFIED ALPHA = ',T27,F11.4,/*)
 & T3,'ACTUAL ALPHA (GKD) = ',T27,F11.4,/*)
 & T3,'ACTUAL ALPHA (POW) = ',T27,F11.4,/*)
 & T3,'HOLDING COST (GKD) = ',T27,F11.4,/*)
 & T3,'HOLDING COST (POW) = ',T27,F11.4,/*)
 & T3,'TOTAL COST (GKD) = ',T27,F11.4,/*)
 & T3,'TOTAL COST (POW) = ',T27,F11.4)
RETURN
C
300 WRITE(3,310) I
310 FORMAT(* BIG S OR D > 2000 FOR ITEM*,I6,/
 & * EXACT CANNOT EVALUATE THE (S,S) POLICY*,/)
C
RETURN
END
C
SUBROUTINE EXACT(K,L,H,MU,SD,LS,BS,SL,HC,RC,IER)
C
THIS SUBROUTINE COMPUTES THE SERVICE-LEVEL, HOLDING COST AND
REPLENISHMENT COST EXACTLY FOR THE POLICY (LS,BS) UNDER THE
ASSUMPTION OF NEGATIVE BINOMIAL DEMAND.
C K = SET-UP COST
C L = LEADTIME + 1
C H = UNIT HOLDING COST
C MU = MEAN DEMAND
C SD = DEMAND STANDARD DEVIATION
C LS = LITTLE S
C BS = BIG S
C SL = STEADY-STATE SERVICE-LEVEL (FREQUENCY OF PERIODS THAT
C NO BACKORDER IS PLACED)
C HC = EXPECTED HOLDING COST
C FC = EXPECTED REPLENISHMENT COST
C
REAL*8 K,L,H,MU,SD,LS,BS,SL,HC,RC,P,Q,R,BS,PROB(2000),HALF,
& NB(2000),M(2000),MI,RHO,VAR,SUM,PROBI,MD,ZERO,XONE,XTWO
INTEGER S,D,EB1,UB2,ONE,TWO,IER
DATA ONE,TWO,ZERO,XONE,XTWO,HALF/1,2,0,DO,1,DO,2,DO,..5DO/
C S AND D ARE BOUNDED TO THE NEAREST INTEGER. WE USE THE REORDER
C QUANTITY D = BIG S - LITTLE S - 1 BECAUSE WE USE THE VERSION OF
C AN (S,S) POLICY THAT REQUIRES THAT AN ORDER BE PLACED WHEN THE
C INVENTORY POSITION DROPS STRICTLY BELOW LITTLE S.
C
PROB(I) = PROB(NB RV = I-1)
C NB(I) = PROB(NB = I-1) + PROB(NB RV = I-1)
C M(I) = PROB(RENEWAL PMF OF NB RV = I-1)
C
IER = 0 EXACT WAS ABLE TO EVALUATE (S,S) POLICY
C 1 OTHERWISE
C
IER = 0
S = BS+HALF
D = (BS-LS-ONE)+HALF
Q = MU/(SD*SD)
P = XONE-Q
R = MU*Q/P
BS = L*R
VAR = Q*R
PROB(ONE) = VAR
MB(ONE) = Q*RS
RHO = XONE/(XONE-VAR)
M(ONE) = VAR*BHO
UB1 = S+ONE
IF (D.GT.S) UB1 = D+ONE
IF (UB1.GT.2000) GO TO 100
DO 20 I = TWO,UB1
VAR = XONE/(I-XONE)
PROBI = PROB(I-ONE)*P*(R+I-XTWO)*VAR
PROB(I) = PROBI
C
C
172
WB(I) = WB(I-ONE) * P * (BS*I-ITWO) * VAR
SUM = ZERO
UB2 = I-ONE
DO 10 J = CWE, UB2
10 SUM = SUM + PROB(I-J+ONE) * M(J)
M(I) = (PROBI+SUM) * RHO
20 CONTINUE
UB1 = D+ONE
IF (UB1.GT.2000) GO TO 100
MD = ZERO
DO 30 I = CWE, UB1
30 MD = MD + M(I)
RHO = XONE/(ICNE+MD)
C
C COMPUTE EXPECTED REPLENISHMENT COST
C
BC = K*RHO
C
C COMPUTE EXPECTED HOLDING COST
C
IF (D.GT.5) UE1 = S+ONE
IF (UB1.GT.2000) GO TO 100
HC = ZERO
DO 40 I = CWE, UB1
UB2 = S-I+TWC
MI = M(I)
DO 40 J = CWE, UB2
HC = HC + (UB2-J) * NB(J) * MI
40 CONTINUE
SUM = ZERO
UB2 = S+ONE
DO 45 I = CWE, UB2
45 SUM = SUM + (UB2-I) * NB(I)
HC = H*(HC+SUN)*RHO
C
C COMPUTE (STEADY-STATE) SERVICE-LEVEL
C
SL = ZERO
DO 60 I = CWE, UB1
SUM = ZERO
UB2 = S-I+TWC
DO 50 J = CWE, UB2
50 SUM = SUM + NB(J)
60 SL = SL + SUM * M(I)
SUM = ZERO
UB1 = S+CNE
IF (UB1.GT.2000) GO TO 100
DO 70 I = CWE, UB1
70 SUM = SUM + NB(I)
SL = (SL+SUM)*RHO
RETURN
C
100 IER = 1
RETURN
C
END
C
SUBROUTINE PCWAPP(K,L,H,MU,SD,ALPHA,LITS,BIGS)
C
THIS SUBROUTINE COMPUTES LITTLE S AND BIG S USING THE REVISITED
POWER APPROXIMATION METHOD (MOSIER(1981), TECH REPORT #18).
THE SHORTAGE COST IS COMPUTED USING THE POWER APPROXIMATION
METHOD (EHRLHARDT(1977, PP 18, 45), TECH REPORT #12).
C
K = SET-UP COST
L = LEADTIME + 1
H = UNIT HOLDING COST
MU = MEAN DEMAND
SD = MEAN STANDARD DEVIATION
ALPHA = (STEADY-STATE) SERVICE-LEVEL, THE FREQUENCY OF PERIODS
WIThOUT BACKLOGS
LITS = LITTLE S
BIGS = BIG S
C
REAL*8 P,H,K,MU,D,L,VL,SDL,Z,V,SD,LITS,BIGS,ONE,
C
& C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,ALPHA
C
DATA C1/1.3/
DATA C2/.494/
DATA C3/.506/
DATA C4/.116/
DATA C5/.973/
DATA C6/.183/
DATA C7/1.063/
DATA C8/2.192/
DATA C9/0.0695/
DATA ONE/1./
C
V = SD*SD
VL = V*L
SDL = DSQRT(VL)
C
P = H*(ALPHA-C10)/(C9-ALPHA)
Z = CNE*(VL/(MU*MU))
D = C1*(MU*C2)*(K/H)**C3*(Z**C4)
Z = DSQRT(D**H/(P*SDL))
C9 = (C6/Z)+C7-C8*Z
LITS = C5*L+MU*SDL*C9
BIGS = LITS+D
C
RETURN
END


GKD (GENERALIZED KNAPSACK DUALITY) ALGORITHM

NOTE: THIS SUBROUTINE CHANGES THE VALUES OF THE INPUT
PARAMETER BSL

WT = WEIGHTS (MUST BE CONVEX)
RH = RHO
GA = GAMMA
DE = DELTA
A = A
B = B
E = E
F = F
G = G
H = H
MS = MU-STAR = I * MU
ET = ETA
KS = KSI
TU = THETA-UPPER
LPDO = LAMBDA-PRIME ( THETA-UPPER )
LL = LAMBDA-LOWER
MLL = MAX ( LAMBDA-LOWER )
FBLL = P ( MLL )
EU = LAMBDA-UPPER
MLU = MAX ( LAMBDA-UPPER )
TS = THETA-STAR
LS = LAMBDA-STAR
BSL = LOWER BOUND CN BIG S
MLS(I) = SERVICE-LEVEL FOR ITEM I
CONST = 15 * PI / (16 * SQRT(3))
NI = NUMBER OF ITEMS IN THE INVENTORY SYSTEM
N (AT PROGRAM BEGINNING) = NI
N (AT PROGRAM TERMINATION) = NUMBER OF ITEMS WITH BIG S GREATER
THAN BSL
PERM : PERM(1), ..., PERM(N) ARE THE ITEMS WITH BIG S
GREATER THAN BSL. WHEN NI IS GREATER THAN N,
PERM(N+1), ..., PERM(NI) ARE THE ITEMS WITH
BIG S = BSL

COMMON /INPUT/ WT,ALPHA, EPS, NSIG, MAXFN,N
COMMON /LAM/ A,B,E,F,G,H
COMMON /PARAMS/ TU,TSE,DE,GA, D, XONE, ZERO, PERM, ONE
COMMON /SUBB/ LAM,I

REAL*8 LITS(512),BIGS(512),BSL(512),W(512),DD(512)
REAL*8 L(512), S(512), W(H1,512), Alpha
REAL*8 D(512), R(512), G(512), E(512), B(512), E(512), P(512),
& G(512), B(512), TO(512), NS(512), LL(512), LO(512), MSL(512),
& TS(512), BL, EML, MLU, LS, ZERO, HALF, EPS, EP, AL
REAL*8 Const, HOI, MUI, LI, SDI, WTI, ALI, BI, EI, WTAIVSL,
& DI, PHI, QI, QA, DEL, AI, BI, EI, PI, GI, HI, VAR1, VAR2, TOU, TSI, LU,
& LPTUI, MSI, LAM, BISI, THS, TEST, SUM, I,
& LE, UB, K(512), BSL, T(512), LLI, LLI, LL, LLI, IONE

INTEGER N, ONE, NI, CNTR, PERN(512), ILE, J, INDEX, M, II
INTEGER WS, WSIG, MAXN, MAX

REAL*8 LAMBDA, P, LTML
EXTERNAL P, LTML

ZERO = 0.
ONE = 1.
IONE = 1.
HALF = .5
CONST = 1.700436904

ALPHA = AL
EPS = EP
WSIG = WS
MAXN = MX
N = NI

DO 3 I = CNE, HI
3 WT(I) = W(I)

WRITE(3, 5)
5 FORMAT('1')

INITIALIZE THE PERMUTATION MATRIX PERN

DO 6 I = CNE, HI
6 PERN(I) = I

INITIALLY SET LAMBDA-STAR = 0, SINCE THIS IS HOW TO INTERPRET
LAMBDA-STAR IF ALL POLICIES ARE SUCH THAT BISI = BSL

LS = ZERO

COMPUTE THE CONSTANTS

DO 20 I = CNE, HI
20 HOI = HO(I)
MUI = MU(I)
LI = L(I)
MSI = LI*MUI
MS(I) = MSI

...
SDI = SD(I)
WTI = W(T(I)
DI = DD(I)
D(I) = DI
RHI = MUI/(DI4((G(dI4(SDI*SDI/MUI))*HALF))
RHI(I) = RHI
QI = XONE-RHI
GAI = CONST/(LI*SDI)
GAI(I) = GAI
DEI = DEEP(GAI*DI)
DE(I) = DEI
AI = DI*HOI
A(I) = AI
VAR1 = XCNE+DEI
BI = AI*(G14VAR1)
B(I) = BI
EI = AI*(DEI+CI*VAR1)
E(I) = EI
FI = AI*QI*DEI
F(I) = FI
VAR1 = DI*RHI*GAI
VAR2 = QI*(DEI-XCNE)
GI = WTI*(VAR1+VAR2)
G(I) = GI
HI = WTI*(VAR1*DEI+VAR2)
H(I) = HI

C DETERMINE BSL BY THE LARGEST VALUE OF THETA, TU, AND HENCE THE
C SMALLEST VALUE OF BIG S, SO THAT D(LAMBDA)/D(THETA) < 0,
C GUARANTEING THAT LAMBDA(THETA) IS INVERTIBLE ON (0, TU)
C IF THE USER-SUPPLIED BSL IS GREATER THAN THIS, IT BECOMES THE
C BSL USED
C
ETI = G1*(BI*HI*HI-GI*(EI*HI-FI*G1))-AI*HI*HI*HI

C IF ETA(I) IS LESS THAN 0, LAMBDA(THETA) MAY NOT BE CONVEX, AND
C THERE IS NO GUARANTEE THAT THE GKD ALGORITHM WILL CONVERGE TO
C A LOCAL MINIMUM--THE USER SHOULD EXERCISE CARE
C
IF (ETI.LT.ZERO) WRITE(3,10) I,ETI
10 FORMAT(' ETA(*,16,*) = ',E20.6,' LT 0')

C CNTR = -1
7 CNTR = CNTR+ONE
TUI = (DEEP(GAI*(MSI-CNTR)))/DEI
KSI = (FI/HI)-(AI/(G1*TUI*TUI))
VAR1 = GI*HI*TUI
LPTUI = KSI-(ETI/(GI*HI*VAR1*VAR1))
IF (LPTUI.GE.ZERO) GO TO 7
BSLI = CNTR4DI
X = BSL(I)
BSL(I) = BSLI
IF (X.GT.BSLI) BSL(I) = X
TDI = (DEXP(GAI*(MSI-(BSL(I)-DI))) DEI
TD(I) = TUI

LLI = LAMBDA(TUI,I)
LL(I) = LLI
20 CONTINUE

COMPUTE MLL = MAX (LAMBDA-LOWER) = MAX (LAMBDA(TU)), SO ALL
FUNCTIONS LAMBDA(THETA) ARE INVERTIBLE ON [MLL,INFINITY)

22 MLL = ZER0
DO 30 II = ONE,N
I = PERM(I)
LLI = LL(I)
IF (MLL.LT.LLI) MLL = LLI
30 CONTINUE

PHLL = P(MLL)

P(LAMBDA) IS INCREASING ON [MLL,INFINITY) WITH P(INFINITY)
GREATER THAN 0. CHECK IF P(MLI) IS LESS THAN ON EQUAL TO
0, "TO SET P(INFI0) = 0 FOR LBL = 10 ON [MLL,INFINITY);

IF (PHLL.LE.ZERO) GO TO 50

P(MLL) IS GREATER THAN 0. LOCATE THAT ITEM ILB = PERM(INDEX)
WITH THE LARGEST VALUE OF LL = LAMBDA LOWER, AND SET
BIG S = BSL, IE. SET I = (= THETA-STAR) = TU (= THETA-UPPER)

INDEX = ONE
ILB = PERM(INDEX)
DO 42 I = ONE,N
J = PERM(I)
IF (LL(J).GT.LL(ILB)) INDEX = I
ILB = PERM(INDEX)
42 CONTINUE

CHANGE N TO N - 1, AND REMOVE ITEM ILB FROM THE FIRST N ENTRIES
OF PERM AND PUT IT IN PERM(N+1)

N = N-ONE

IF N = 0, ALL ITEMS ARE SUCH THAT BIG S = BSL

IF (N.EQ.0) GO TO 85
DO 43 I = INDFI,N

43 FERM(I) = FERM(I+1)
FERM(N+1) = ILB
SET TS = TU FOR ITEM ILB
TS(ILB) = TU(ILB)

DETERMINE THE SERVICE-LEVEL ALI FOR ITEM ILB, AND RECOMPUTE THE
AVERAGE SERVICE-LEVEL (WHICH IS AGAIN DENOTED ALPHA) FOR THE
OTHER ITEMS SO THAT THE OVERALL AVERAGE SERVICE-LEVEL IS STILL
THE ORIGINAL SPECIFIED ALPHA

NOTE: WHEN BIG S IS SET TO BSL FOR THE ITEM ILB, THE ITEM
IS NO LONGER USED IN THE COMPUTATION OF THE SOLUTION LS
TO P(LAMBDA) = 0. RATHER, THIS COMPUTATION IS DONE ONLY
FOR THOSE SUMMANS OF P THAT CORRESPOND TO ITEMS WITH
BIG S GREATER THAN BSL. HENCE WE NEED ONLY DETERMINE
THE AVERAGE SERVICE-LEVEL FOR THESE ITEMS SO THAT THE
OVERALL SERVICE-LEVEL IS THE SPECIFIED ALPHA

DI1 = DE(ILB)
GAI = GA(ILB)
TSI = TS(ILB)
BHI = BH(ILB)
QI = XONE-BHI
DI = D(ILB)
WTI = WT(ILB)
ALI = QI*(XONE+DLOG((XONE+TSI)/(XONE+
&
DI*TSI)))/(GAI*DI)+(BHI/(XONE+TSI))
ALPHA = (ALPHA-WTI*ALI)/(XONE-WTI)

IF (ALPHA.LT.XONE) GO TO 22
WRITE(3, U7)

FOR THE NEW AVERAGE SERVICE-LEVEL IS NOT LESS THAN
ONE, SO THE PROBLEM IS INFEASIBLE

GO TO 82

50 VAR1 = -PHLL

SINCE P(MLL) LE 0, SOLVE P(LAMBDA) = 0 FOR LAMBDA = LS
ON [MLL, INFINITY]. FIRST CHECK IF P(MLL) = 0 WITHIN EPSILON;
IF NOT, DETERMINE BLU LESS THAN INFINITY SO THAT P(MLU) IS
GREATER THAN OR EQUAL TO 0. THEN SOLVE P(LAMBDA) = 0 FOR
LAMBDA = LS ON [MLL, MLU]

LS = MLL
IF (VAR1.LE.EPS) GO TO 81

MLU = MLL
DO 80 II = ONE, N
I = PERM(II)
TLI = HALF*TU(I)
RHI = RH(I)
QI = XONE-BHI
GAI = GA(I)
DI = D(I)
DEI = DE(I)

C FIND LU SUFFICIENTLY LARGE SO THAT THE I-TH COMPONENT OF THE
C FUNCTION P IS NON-NEGATIVE (HERE TEST IS THE I-TH COMPONENT
C OF THE FUNCTION P), THEN SET MLU = MAX( LU )

C 60 TEST = ((XONE-BHI)*XONE+DLOG((XONE+TLI)/(XONE+
& DEI*TLI))/(GAI*DI))*RHI/(XONE+TLI)) - ALPHA
IF (TEST.GT.ZEBO) GO TO 70
TLI = HALF*TLI
GO TO 60

70 LUI = LAMBDA(TLI,I)
LU(I) = LUI
IF (LUI.GT.GHI) MLU = LUI

80 CONTINUE

C SOLVE P(LAMBDA) = 0 FOR LAMBDA = LS IN [MLL,MLU]

C MAX = MAIPN
LB = MLL
UB = MLU
CALL ZBENT (P, EPS, NSIG, LB, UB, MAX, IER)

C LS = UB
81 CONTINUE

C COMPUTE THE (S,S) POLICIES FOR THOSE ITEMS WITH BIG S GREATER
C THAN BSL

C 82 DO 84 II = ONE, N
I = PERM(II)
GAI = GA(I)
TSI = TS(I)
MSI = MS(I)
DI = D(I)
BIGS(I) = MSI - ((DLOG(TSI))/GAI)

C BECAUSE OF BOUND-OFF ERROR, IT MAY BE THAT BIG S LT BSL.
C IF SO, SET BIG S = ESL.
C IF (BIGS(I).LT.BSL(I)) BIGS(I) = BSL(I)
84 LITS(I) = BIGS(I) - DI
COMPUTE THE (S, S) POLICIES FOR THOSE ITEMS WITH BIG S EQUAL TO BSL

IF (HI .EQ. WI) GO TO 88

M = M +ONE
DO 86 II = M, WI
I = PERM (II)
BIGS (I) = BSL (I)
86 LITS (I) = BIGS (I) - D (I)

WTAVSL = ZERO

COMPUTE THE SERVICE-LEVEL AND SHORTAGE COST FOR ALL THE ITEMS IN THE INVENTORY SYSTEM, AND WTAVSL = WEIGHTED AVERAGE SERVICE-LEVEL (HOPPERS FULLY THIS EQUALS APPROXIMATE THE ORIGINAL ALPHA)

DO 90 II = ONE, NI
I = PERM (II)
DBI = DB (I)
GAI = GA (I)
TSI = TS (I)
BHI = BH (I)
QI = XONE - BHI
MSI = MS (I)
DI = D (I)
BIGSI = BIGS (I)
THS = DEXP (-GAI * (BIGSI - MSI))
ALI = QI * (XONE + DLOG ((XONE + TSI) / (XONE +
6 * (DEI * TSI) / (GAI * DI) + (BHI / (XONE + TSI))
MSI (I) = ALI
WTAVSL = WTAVSL + ALI * WT (I)
90 CONTINUE

RETURN
END

REAL FUNCTION LAMBDA*8 (THETA, I)

COMMON /LAM/ A, B, E, F, G, H

REAL*8 A (512), B (512), E (512), F (512), G (512), H (512)
REAL*8 THETA

INTEGER I

LAMBDA = (A (I) + THETA * (B (I) + THETA * (E (I) + THETA * F (I)))) / 6 * (THETA * (G (I) + THETA * H (I))))

RETURN
END
REAL FUNCTION P*B (LAM)

COMMON /INBU/I, W-, ALPHA, EPS, NSIG, MAXFN, N
COMMON /PARAMS/ TU, TS, RH, DE, GA, D, XONE, ZERO, PERM, ONE
COMMON /SUBR/ XLAM, I

REAL*8 WT(512), ALPHA
REAL*8 D(512), RH(512), GA(512), DE(512), TU(512), TS(512), ZERO, EPS
REAL*8 LAM, TSI, TUI, RHI, QE, LTML, TEST, LB, UB, XLAM, XONE

INTEGER ONE, I, II, MAX, PERM(512), MAXFN

EXTERNAL LTML

XLAM = LAM
P = ZERO

TO EVALUATE P(LAM), EVALUATE THETA, THE SOLUTION IN [0, TU] TO
LTML(THETA) = LAMBDA(THETA) - LAM = 0
FIRST CHECK IF LTML(TU) = 0

DO 20 II = ONE, N
I = PERM(I)
TUI = TU(I)

SINCE LTML(TUI) IS LESS THAN OR EQUAL TO 0, IF IN THE COMPUTER
LTML(TUI) IS GREATER THAN OR EQUAL TO 0, THIS IMPLIES THAT
INDEED LTML(TUI) = 0

TEST = LTML(TUI)
IF (TEST, LT, ZERO) GO TO 5
TSI = TUI
TS(I) = TSI
GO TO 10

5 LB = ZERO
UB = TUI
MAX = MAXFN
CALL ZFALSE(LTML, EPS, NSIG, LB, UB, TSI, MAX, IER)
TS(I) = TSI

10 RHI = RH(I)
QE = XONE-RHI
P = P+WT(I)*(QE*XONE+DLOG((XONE+TSI)/(XONE+
& DF(I)*TSI))/(GA(I)*D(I)))/(RHI/(XONE+TSI))-ALPHA)

20 CONTINUE

RETURN
END
REAL FUNCTION LTML*8 (THETA)

COMMON /LAM/ A,B,E,F,G,H
COMMON /SUBB/ LAM,I

REAL*8 A(512), B(512), E(512), F(512), G(512), H(512)
REAL*8 THETA, LAM

INTEGER I

LTML = A(I) +
& THETA*(E(I) - LAM*G(I) + THETA*(E(I) - LAM*H(I) + THETA*F(I)))
RETURN
END
PROGRAM MISSWLSI : MULTI - ITEM (S,S)
NORMAL DEMAND DISTRIBUTION APPROXIMATION
LOWER BOUNDS ON BIG S
SINGLE-ITEM POLICY OUTPUT

THE INPUT IS FROM CARDS

TITLE = USER - SUPPLIED TITLE (LE 32 ALPHA CHARACTERS)
NI = NUMBER OF ITEMS (NI LE 512)
K = SET-UP ORDERING COST
L = LEADTIME + 1
H0 = UNIT HOLDING COST
MU = MEAN DEMAND
SD = DEMAND STANDARD DEVIATION
WT = CONSTRAINT WEIGHT

NSIG = NUMBER OF SIGNIFICANT DIGITS DESIRED FROM THE IMSL
ROUTINES ZBBBENT AND ZFALSE
EPS = 1/(10 ** NSIG)
MAXPN = MAXIMUM NUMBER OF ITERATIONS ALLOWABLE BY USER FOR
THE IMSL ROUTINES ZBBBENT AND ZFALSE
IER = (OUTPUT) ERROR PARAMETER FROM IMSL ROUTINES AND XIACT

THE FOLLOWING ARE INPUT TO OR COMPUTED BY GKD

LITS = LITTLE S
BIGS = BIG S
D = REORDER QUANTITY BIG S - LITTLE S
BSL = D + R*60*H0
LS = LAMBDA-STAB

OUT = 0 OTHERWISE
1 WRITE OUTPUT TO THE EXTERNAL FILE
FORTRAN UNIT NUMBER 10
WITH THE FOLLOWING DCB SPECIFICATIONS:

RECFH = FE
LRECL = 160
BLKSIZE = 1600

THE FOLLOWING ARE COMPUTED BY ERHARDT'S POWER METHOD

PLITS = LITTLE S
PBIGS = BIG S

THE FOLLOWING ARE COMPUTED EXACTLY

HC(I) = EXPECTED HOLDING COST FOR GKD POLICY FOR ITEM I
PC(I) = 

SL(I) = 

KVL = 

PHC(I) = 

PRL(I) = 

PSL(I) = 

PAVSL = 

INPUT THE SYSTEM PARAMETERS AS FOLLOWS:

THE INPUT FORMAT IS:

**ALL DATA STABS IN COLUMN 1 (EXCEPT TITLE STABS IN COLUMN 2), AND THERE IS A SINGLE SPACE BETWEEN EACH DATA ENTRY**

**REAL**

**INTEGER**

**DOUBLE**
3 FORMAT(8,4)
   READ(1,5) LSI, NSIG, MAXFM, WI, OUT
5 FORMAT(F15.6,1X,3(I6,1X),1X)
   EPS = 10.**(-NSIG)

C DO 10 I = 1, WI
   READ(1,6) WT(I), D(I), K(I), L(I), HO(I), MU(I), SD(I), R
   BSL(I) = D(I)+R*L(I)*MU(I)
8 FORMAT(8(F8.4,1X))
10 CONTINUE

C MAKE THE WEIGHTS WT INTO EQUIVALENT CONVEX WEIGHTS WT
C
   SUM = 0.
   DO 15 I = 1, WI
15   SUM = SUM+WT(I)
15 SUM = 1.000/SUM
   DO 16 I = 1, WI
16   WT(I) = SUM*WT(I)

C COMPUTE THE OPTIMAL POLICIES ASSOCIATED WITH THE MULTIPLIER
C LAMBDA-STAR (LSI)
C
   CALL BIGSLS(NI, L, HO, MU, SD, D, WT, BSL, LSI, EPS, NSIG, MAXFM, BIGS, LITS)
   WRITE(3,45)
45 FORMAT('I')
C
C COMPUTE THE (UNIFORM SERVICE-LEVEL) POWER APPROXIMATION POLICIES
C AND COMPARE THEIR ASSOCIATED COSTS WITH THE COSTS ASSOCIATED WITH
C THE BIGSLS POLICIES
C
   CALL COMPAR(NI, K, L, HO, MU, SD, WT, BIGS, LITS, TITLE, LSI, OUT)
   STOP
C
END
C
SUBROUTINE BIGSLS(NI, L, HO, MU, SD, D, WT, BSL, LSI, EPS, NSIG, MAXFM, BIGS, LITS)
C
THIS SUBROUTINE COMPUTES THE OPTIMAL (S,S) POLICIES ASSOCIATED
C WITH THE KUHN-TUCKER CONSTRAINT MULTIPLIER LAMBDA-STAR (LSI)
C
COMMON /FINDTS/AI, BI, EI, FI, GI, HI, LAM
C
REAL*8 LITS(512), BIGS(512)
REAL*8 L(512), HO(512), MU(512), SD(512), WT(512),
   D(512), BSL(512)
REAL*8 EPS, LSI, LI, HOI, MUI, SDI, DI, MSI, WTI, TSI, LAM,
   RHI, Q1, GAI, DEI, AI, BI, EI, FI, GI, HI, VAR1, VAR2, ET1, TUI,
C DETERMINE BSL BY THE LARGEST VALUE OF Theta, Tu, AND HENCE THE
C SMALLEST VALUE OF BIG S, SO THAT D(LAMBDA)/D(Theta) LE 0,
C GUARANTEING THAT LAMBDA(Theta) IS INVERTIBLE ON (0, Tu)
C IF THE USER-SUPPLIED BSL IS GREATER THAN THIS, IT BECOMES THE
C BSL USED
C
C ETI = GI*(BI*HI-HI-GI*(EI*HI-FI*GI)-AI*HI*H1*HI
C
C IF ETA(I) IS LESS THAN 0, LAMBDA(THETA) MAY NOT BE CONVEX, AND
C THERE IS NO GUARANTEE THAT THE GAD ALGORITHM WILL CONVERGE TO
C A LOCAL MINIMUM---THE USER SHOULD EXERCISE CARE
C
C IF (ETI.LT.ZERO) WRITE(3,30) I,ETI
30 FORMAT(*, ETA(*,16,*), =*,E20.6,* LT 0*)

C
CNTR = -1
35 CNTR = CNTR+CNTR
TUI = (DEXP(GAI*(MSI-CNTR)))/DEI
MSI = (PI/HI)-(AI/(GI*TUI*TUI))
VAR1 = GI+HI*TUI
LPTUI = MSI-(EIT/(GI*HI*VAR1*VAR1))
IF (LPTUI.GE.ZERO) GO TO 35

C
BSLI = CNTR+DI
I = BSL(I)
BSL(I) = BSL(I)
IF (I.GT.BSLI) BSL(I) = I
TUI = (DEXP(GAI*(MSI-(BSL(I)-DI))))/DEI

C
LAM = LSI
MAX = MAXPN
LB = ZERO
UB = TUI

C IF LAMTH(UB) IS GREATER THAN ZERO, THEN THE FUNCTION
C LAMBDA(THETA) FOR THIS ITEM IS ALWAYS GREATER THAN LAMBDA-STAR,
C AND SO BIGS FOR THIS ITEM IS SET AT ITS LOWER-BOUND BSL
C
IF (LAMTH(UB).GT.ZERO) GO TO 50
CALL ZBRENT(LAMTH, EPS, WSIG, LB, UB, MAX, IER)
TSI = UB
BIGS(I) = MSI-((DLOG(TSI))/GAI)

C BECAUSE OF ROUND-OFF ERROR, IT MAY BE THAT BIG S LT BSL.
C IF SO, SET BIG S = ESL.
C
IF (BIGS(I).LT.BSL(I)) BIGS(I) = BSL(I)
LITS(I) = BIGS(I)-DI
GO TO 60
50 BIGS(I) = BSL(I)
LITS(I) = BIGS(I)-DI
60 CONTINUE
RETURN
END
REAL FUNCTION LAMTH*8 (THETA)
C
COMMON /PINDTS/AI,BI,EL,FI,GI,HI,LAM
REAL*8 AI,BI,EL,FI,GI,HI,LAM,THETA
C
LAMTH = AI+THETA*(BI-LAM*GI+THETA*(EI-LAM*HI+THETA*PI))
RETURN
END

C
SUBROUTINE CCPABC(N1,K,L,H0,MU,SD,WT,BIGS,LITS,TITLE,LSI,OUT)

THIS SUBROUTINE DOES THE FOLLOWING:

(1) EVALUATES EXACTLY THE HOLDING AND REPLENISHMENT COST, AND
THE SERVICE-LEVEL FOR THE POLICIES (LITS,BIGS) UNDER THE
ASSUMPTION OF NEGATIVE-BINOMIAL DEMAND (IN EXACT)

(2) COMPUTES APPROXIMATELY OPTIMAL (S,S) POLICIES WITH UNIFORM
SERVICE-LEVELS (IN POWAPP)

(3) EVALUATES THE POLICIES DESCRIBED IN (2) EXACTLY AS IN (1)

(4) PRINTS THE RESULTS OF STEPS (1) AND (3)

REAL*8 LITS(512),BIGS(512)
REAL*8 K(512),L(512),H0(512),MU(512),SD(512),WT(512)
REAL*8 HGKD,HPOW,TGKD,TPOW,LSI,LITSI,BIGSI
REAL*8 K1,L1,H01,MU1,SD1,SLI,HCI,RCI,AVSL,PAVSL
REAL*8 HC(512),RC(512),SL(512),PLITS(512),PBIGS(512),
& PHC(512),PPC(512),PSL(512)

INTEGER NI,TITLE(8),OUT

AVSL = 0.
HGKD = 0.
TGKD = 0.
DO 70 I = 1,NI
  K1 = K(I)
  L1 = L(I)
  H01 = H0(I)
  MU1 = MU(I)
  SD1 = SD(I)
  BIGSI = BIGS(I)
  LITSI = LITS(I)
  CALL EXACT(K1,L1,H01,MU1,SD1,LITSI,BIGSI,SLI,HCI,RCI,IEE)
  IF (IEE.EQ.1) GO TO 300
  AVSL = AVSL+SLI*WT(I)
  HGKD = HGKD+HCI
  TGKD = TGKD+HCI+RCI
  HC(I) = HCI
  RC(I) = RCI
  SL(I) = SLI
70  CONTINUE

PAVSL = 0.
HPOW = 0.
TPOW = 0.
DO 75 I = 1,NI
  K1 = K(I)
  L1 = L(I)
HOI = HO(I)
MUI = MU(I)
SDI = SD(I)
CALL POWAPP(KI,LI,HCI,MUI,SDI,AVSL,LITSI,BIGSI)
CALL EXACT(KI,LI,HOI,MUI,SDI,LITSI,BIGSI,SLI,HCI,BCI,IEB)
IF (IER.EQ.1) GO TO 300
PAVSL = PAVSL+SLI*WT(I)
HPOW = HPOW+HCI
TPOW = TPOW+HCI+RCI
PLITS(I) = LITSI
PBIGS(I) = BIGSI
PHC(I) = HCI
PRC(I) = BCI
PSL(I) = SLI
75 CONTINUE

PRINT SUMMARY

WRITE(3,98) (TITLE(J),J=1,8),LSI
98 FORMAT(T2,8A4,':',T27,F11.4,'//')
WRITE(3,100) AVSL,PAVSL,HGKD,HPOW,TGKD,TPCW
100 FORMAT(T3,'ACTUAL ALPHA (GKD) =',T27,F11.4,'//')
 & T3,'ACTUAL ALPHA (POW) =',T27,F11.4,'//')
 & T3,'HOLDING COST (GKD) =',T27,F11.4,'//')
 & T3,'HOLDING COST (POW) =',T27,F11.4,'//')
 & T3,'TOTAL COST (GKD) =',T27,F11.4,'//')
 & T3,'TOTAL COST (POW) =',T27,F11.4,'//')
 & T4,'I',T8,'K',T12,'L+1',T19,'H',T25,'MU',
 & T32,'S',T40,'GKD',T50,'POWER',T65,'HOLD',
 & T85,'EEP',T99,'SER LEV',T116,'N*WT',
 & T39,'LS',T44,'BS',T50,'LS',T55,'BS',
 & T62,'GKD',T70,'POWER',T80,'GKD',T88,'POWER',
 & T98,'GKD',T105,'POWER',
DO 130 I = 1,NI
SUM = NI*WT(I)
WRITE(3,120) I,K(I),L(I),HO(I),MU(I),SD(I),LITSI,BIGS(I),
 & PLITS(I),PBIGS(I),HC(I),PHC(I),EC(I),PRC(I),
 & SL(I),PSL(I),SUM
WRITE(10,120) I,K(I),L(I),HO(I),MU(I),SD(I),LITSI,BIGS(I),
 & PLITS(I),PBIGS(I),HC(I),PHC(I),EC(I),PRC(I),
 & SL(I),PSL(I),SUM
120 FORMAT(I4,F5.0,F4.0,F9.4,F4.0,F9.4,F9.4,F9.2,
 & 2F8.4,F12.4,//)
130 CONTINUE
RETURN

300 WRITE(3,310) I
310 FORMAT(* BIG S OR D > 2000 FOR ITEM*,I6,/
 & * EXACT CANNOT EVALUATE THE (S,S) POLICY*,//)
RETURN
END

SUBROUTINE EXACT(K,L,H,MU,SD,LS,BS,SL,HC,RC,R,S,PROB,HALF,NB,M,RHO,VAR, SUM,PROBI,MD,ZERO,XONE,ITWO)

THIS SUBROUTINE COMPUTES THE SERVICE-LEVEL, HOLDING COST AND
REPLENISHMENT COST EXACTLY FOR THE POLICY (LS,BS) UNDER THE
ASSUMPTION OF NEGATIVE BINOMIAL DEMAND.

K = SET-UP COST
L = LEADTIME + 1
H = UNIT HOLDING COST
MU = MEAN DEMAND
SD = DEMAND STANDARD DEVIATION
LS = LITTLE S
BS = BIG S
SL = STEADY-STATE SERVICE-LEVEL (FREQUENCY OF PERIODS THAT
NO BACKORDER IS PLACED)
BC = EXPECTED HOLDING COST
EC = EXPECTED REPLENISHMENT COST

REAL*8 K,L,H,MU,SD,LS,BS,SL,HC,RC,P,Q,R,RS,PROB,HALF,NB,M,RHO,VAR,SUM,PROBI,MD,ZERO,XONE,ITWO
INTEGER S,D,LS,BS,SL,HC,RC,P,Q,R,RS,PROB,HALF,NB,M,RHO,VAR,SUM,PROBI,MD,ZERO,XONE,ITWO
DATA ONE,TWO,ZERO,XONE,ITWO/HALF/1.2,0,D0,1.D0,2.D0,.5D0/

S AND D ARE BOUNDED TO THE NEAREST INTEGER. WE USE THE REORDER
QUANTITY D = BIG S - LITTLE S - 1 BECAUSE WE USE THE VERSION OF
AN (S,S) POLICY THAT REQUIRES THAT AN ORDER BE PLACED WHEN THE
INVENTORY POSITION DROPS STRICTLY BELOW LITTLE S.

PROB(I) = PROB(NB RV = I-1)
NB(I) = PROB(NB = I-1)
L
M(I) = PROB(RENEWAL RV OF NB RV = I-1)

IER = 0 EXACT WAS ABLE TO EVALUATE (S,S) POLICY
1 OTHERWISE

IER = 0
S = BS+HALF
D = (BS-LS-ONE)+HALF
Q = MU/(SD*SD)
P = XONE-Q
R = MU*Q/P
BS = L*R
VAR = Q**R
PROB(ONE) = VAR
NB(ONE) = Q**RS
\[
\begin{align*}
\text{RHO} & = \text{ONE}/(\text{ONE}-\text{VAR}) \\
\text{M(ONE)} & = \text{VAR} \times \text{RHO} \\
\text{UB1} & = \text{S4ONE} \\
\text{IF (D.GT.5)} & \quad \text{UB1} = \text{D4ONE} \\
\text{IF (UB1.GT.2000)} & \quad \text{GO TO 100} \\
\text{DO 20} & \quad \text{I} = \text{TWO}, \text{UB1} \\
\text{VAR} & = \text{ONE}/(\text{I}-\text{ONE}) \\
\text{PROBI} & = \text{PROBI}(\text{I}-\text{ONE}) \times \text{P*}(\text{B}+\text{I}-\text{TW0}) \times \text{VAR} \\
\text{PROB(I)} & = \text{PROBI} \\
\text{NB(I)} & = \text{NB}(\text{I}-\text{ONE}) \times \text{P*}(\text{RS}+\text{I}-\text{TW0}) \times \text{VAR} \\
\text{SUM} & = \text{ZERO} \\
\text{UB2} & = \text{I}-\text{ONE} \\
\text{DO 10} & \quad \text{J} = \text{ONE}, \text{UB2} \\
10 & \quad \text{SUM} = \text{SUM}+\text{PROB}(\text{I}-\text{J}+\text{ONE}) \times \text{M(J)} \\
\text{M(I)} & = (\text{PROB}+\text{SUM}) \times \text{RHO} \\
\text{CONTINUE} \\
\text{UB1} & = \text{D4ONE} \\
\text{IF (UB1.GT.2000)} & \quad \text{GO TO 100} \\
\text{MD} & = \text{ZERO} \\
\text{DO 30} & \quad \text{I} = \text{ONE}, \text{UB1} \\
30 & \quad \text{MD} = \text{MD}+\text{M(I)} \\
\text{RHO} & = \text{ONE}/(\text{ONE}+\text{MD}) \\
\text{C} \\
\text{C COMPUTE EXPECTED REPLENISHMENT COST} \\
\text{C} \\
\text{RC} & = \text{K} \times \text{RHO} \\
\text{C} \\
\text{C COMPUTE EXPECTED HOLDING COST} \\
\text{C} \\
\text{IF (D.GT.5)} & \quad \text{UB1} = \text{S4ONE} \\
\text{IF (UB1.GT.2000)} & \quad \text{GO TO 100} \\
\text{HC} & = \text{ZERO} \\
\text{DO 40} & \quad \text{I} = \text{ONE}, \text{UB1} \\
\text{UB2} & = \text{S4TW0} \\
\text{MI} & = \text{M(I)} \\
\text{DO 40} & \quad \text{J} = \text{ONE}, \text{UB2} \\
40 & \quad \text{SUM} = \text{SUM}+(\text{UB2}-\text{J}) \times \text{NB(J)} \times \text{MI} \\
\text{CONTINUE} \\
\text{SUM} & = \text{ZERO} \\
\text{UB2} & = \text{S4ONE} \\
\text{DO 45} & \quad \text{I} = \text{ONE}, \text{UB2} \\
45 & \quad \text{SUM} = \text{SUM}+(\text{UB2}-\text{I}) \times \text{NB(I)} \\
\text{HC} & = \text{HC} \times (\text{HC}+\text{SUM}) \times \text{RHO} \\
\text{C} \\
\text{C COMPUTE (STEADY-STATE) SERVICE-LEVEL} \\
\text{C} \\
\text{SL} & = \text{ZERO} \\
\text{DO 60} & \quad \text{I} = \text{ONE}, \text{UB1} \\
\text{SUM} & = \text{ZERO} \\
\text{UB2} & = \text{S4TW0} \\
\end{align*}
\]
DO 50 J = ONE, UB2
50 SUM = SUM + NB(J)
60 SL = SL + SUM * M(I)
SUM = ZERO
UB1 = S + ONE
IF (UB1.GT.2000) GO TO 100
DO 70 I = ONE, UB1
70 SUM = SUM + NB(I)
SL = (SL + SUM) * RHO
RETURN
C
100 ITER = 1
RETURN
C
END
C
SUBROUTINE POWERF(K, L, H, MU, SD, ALPHA, LITS, BIGS)
C
THIS SUBROUTINE COMPUTES LITTLE S AND BIG S USING THE REVISED
POWER APPROXIMATION METHOD (MOSIER(1981), TECH REPORT #18).
The SHORTAGE COST IS COMPUTED USING THE POWER APPROXIMATION
METHOD (EHBHAEBD(1977, PP 18, 45), TECH REPORT #12).
C
K = SET-UP COST
L = LEADTIME + 1
H = UNIT HOLDING COST
MU = MEAN DEMAND
SD = MEAN STANDARD DEVIATION
ALPHA = (STEADY-STATE) SERVICE-LEVEL, THE FREQUENCY OF PERIODS
WITHOUT BACKLOGS
LITS = LITTLE S
BIGS = BIG S
C
REAL*8 P, H, K, MU, D, L, VL, SDL, Z, V, SD, LITS, BIGS, ONE,
& C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, ALPHA
C
DATA C1/1.3/
DATA C2/1.494/
DATA C3/1.506/
DATA C4/1.216/
DATA C5/1.973/
DATA C6/1.893/
DATA C7/1.063/
DATA C8/2.192/
DATA C10/1.0695/
DATA ONE/1./
C
VL = SD * SD
VL = V * L
\[ SDL = \sqrt{VL} \]

\[ P = \frac{\alpha(C - C_0)}{(C - \alpha)} \]

\[ Z = 0 \cdot \left( \frac{VL}{R \cdot D} \right) \]

\[ D = C_1 \cdot (\mu^* \cdot C_2) \cdot \left( \frac{(K/H) \cdot C_3}{(Z^* \cdot C_4)} \right) \]

\[ Z = \sqrt{D \cdot B / (P \cdot SDL)} \]

\[ C_9 = (C_6 / Z) + C_7 - C_8 \cdot Z \]

\[ LITS = C_5 \cdot L \cdot \mu + SDL \cdot C_9 \]

\[ BIGS = LITS + D \]

RETURN

END
REFERENCES


