Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

IMPROVING SIMULATION OPTIMIZATION RUN TIME WHEN SOLVING FOR PERIODIC REVIEW INVENTORY POLICIES IN A PHARMACY

Lauren L. Czerniak
Mark S. Daskin
Mariel S. Lavieri

Department of Industrial and Operations Engineering
University of Michigan
1891 IOE Building 1205, Beal Ave
Ann Arbor, MI 48109, USA

Burgunda V. Sweet

College of Pharmacy, Michigan Medicine
University of Michigan
428 Church St
Ann Arbor, MI 48109, USA

Jennifer Erley
Matthew A. Tupps

Michigan Medicine, Department of Pharmacy Services
University of Michigan
1500 East Medical Center Drive
Ann Arbor, MI 48109, USA

ABSTRACT

Pharmaceutical drugs are critical to patient care, but demand and supply uncertainties in this inventory system make decision-making a challenging task. In this paper, we present a simulation-optimization model that determines near-optimal (s,S) periodic review inventory policies that minimize the expected cost per day. The model accounts for perishability, positive lead time, stochastic demand, and supply disruptions. We implement a Binary Grid-Search algorithm which uses the structure of the objective function to quickly solve the simulation-optimization model. The numerical results illustrate how the Binary Grid-Search algorithm performs 21 times faster (when performing 10,000 replications) in terms of run time when compared to an Exhaustive Grid-Search, without sacrificing solution accuracy. This paper provides an efficient method to solve for the near-optimal (s,S) periodic review inventory policies which is essential in the pharmacy inventory system that handles thousands of different drugs.

1 INTRODUCTION

It is estimated the United States spends about $329 billion annually on drugs (Hartman et al. 2018). However, demand and supply uncertainties in a pharmacy inventory system make decision-making a challenging task. Having insufficient inventory can lead to shortages which increase cost for care, increase medication errors, and lead to inferior treatments (Phuong et al. 2019). In contrast, drugs are a perishable product, so holding too much inventory can lead to waste. Accounting for these different sources of uncertainty, perishability, and system characteristics (like positive lead time) make it computationally difficult to determine how much inventory to keep on hand and when orders should be placed.

In this paper, we develop a simulation-optimization model to determine near-optimal (s,S) periodic review inventory policies where s denotes the reorder point and S denotes the order-up-to level. The objective is to minimize the expected cost per day and the model represents a drug at one hospital

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

pharmacy. Simulation-optimization is often used for optimization problems where the objective function cannot be computed exactly, usually due to stochastic parameters in the problem of interest. Instead, replications of a simulation model with realized random variables are used to estimate the objective function (Fu 2015). In the pharmacy system of interest, an inventory manager can be responsible for over 2,500 drugs. Hence, quickly solving the simulation-optimization model is crucial. To do this, we implement a Binary Grid-Search (GS) algorithm to solve for the near-optimal periodic review policies and compare its run time to an Exhaustive GS and a Convergent Optimization via Most-Promising-Area Stochastic Search (COMPASS). We refer to these solutions as near-optimal because we discretize the solution space. Also, throughout this paper, we refer to this discretized solution space as a square grid where the rows correspond to the values of s and the columns correspond to the values of S.

To illustrate our solution framework, we apply our methodology to solve for the (s,S) parameters in a periodic review inventory system for a drug test case. While this paper looks at the problem from a pharmacy perspective, the models presented can be applied to other first-in-first-out perishable inventory systems such as chemicals and food warehouses. Also, the model places no restrictions on the probability distributions for the uncertain parameters, which provides greater flexibility for products that may be of interest. Below, we present related research from inventory management to motivate how the simulation-optimization model expands upon current literature. We also explicitly highlight the contributions of this paper.

Past literature includes a variety of methodologies to solve for periodic review inventory policies: stochastic programming, reinforcement learning, simulation, closed-form solutions. With respect to (s,S) periodic review policies, it is important to recognize the valuable research that studies (s,S) policies from a gradient estimation perspective (e.g., Bashyam and Fu 1991; Fu and Hu 1997) and early research that does not consider a perishable and supply disrupted system like the one presented in this paper (e.g., Sahin 1982; Zheng and Federgruen 1991). Recent perishable inventory research uses models such as stochastic programming and reinforcement learning to determine optimal periodic review policies in the presence of stochastic demand (Dillon et al. 2017; Rajendran and Ravindran 2019; Rajendran and Srinivas 2020; Kara and Dogan 2018). Further, Nguyen and Chen (2019) use stochastic programming to model stochastic demand and stochastic yield supply. When looking at the supply disruption side, Atan and Rousseau (2016) consider supply disruptions in a perishable inventory system, but they do not account for uncertain demand. For supply disruptions in non-perishable inventory systems, some researchers include stochastic demand with supply disruptions (Schmitt et al. 2010) and some only account for supply disruptions (Skouri et al. 2014; Konstantaras et al. 2019), but again, these models do not account for perishability.

For research that solves for optimal inventory policies in a pharmacy system, Zhang et al. (2014) use a deterministic demand and supply model whereas Saedi et al. (2016) account for uncertain demand and supply disruptions, but they assume there is no lead time. Other researchers look into the lead time question like Li et al. (2018) who consider stochastic lead time, but no uncertainty in terms of supply and demand, while Franco and Alfonso-Lizarazo (2020) include stochastic demand and lead time, but they do not consider supply disruptions. In summary, there is a gap in periodic review policies for perishable products that face stochastic demand, supply disruptions, and positive lead time.

The two main contributions of this paper are:

1. We develop a simulation-optimization model to determine near-optimal (s,S) periodic review inventory policies for a perishable system that faces stochastic demand, supply disruptions, and positive lead time. Further, there are no probability distribution restrictions for the uncertain parameters.
2. We implement a Binary GS algorithm which uses the structure of the objective function to solve for the (s,S) policies rather than taking an Exhaustive GS approach.

The remainder of the paper is organized as follows: Section 2 describes a (s,S) inventory system and how to model this system using simulation. Section 3 presents three methods for solving the simulation-optimization model: Exhaustive GS, COMPASS, and Binary GS. Section 4 provides numerical results using all three methods. Finally, Section 5 closes the paper with concluding thoughts and future research.

2 (s,S) PERIODIC REVIEW INVENTORY SYSTEM

2.1 (s,S) Inventory Process

There exists a variety of inventory policies that any given system can follow. Consistent with the system of interest, this research focuses on (s,S) periodic review inventory policies. We define inventory position as the amount of inventory on hand plus the amount of inventory en route. When the inventory position falls below s, an order is placed for S minus the inventory position. Here, the review period is 1 day. Figure 1 illustrates (s,S) inventory policies in a perishable lost-sales inventory system where H is the total inventory on hand, H' is the total inventory on hand that expires at the end of day t, P is the inventory position, d is the demand, s is the reorder point, and S is the order-up-to level.

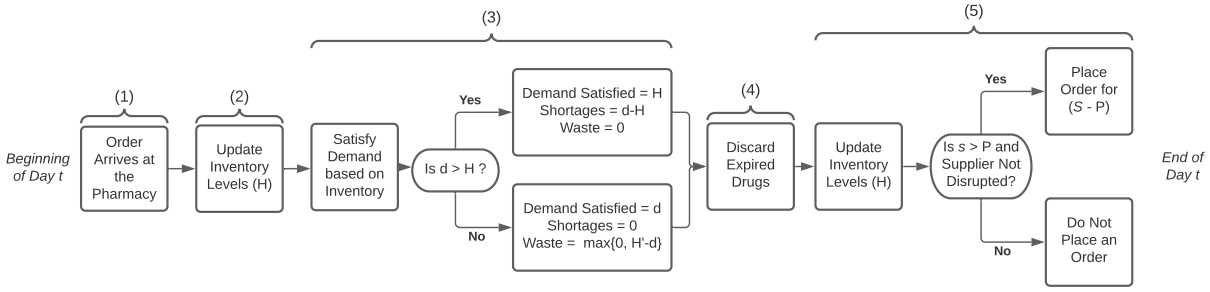


Figure 1: Process Flow of (s,S) Perishable Inventory System ((#) correspond to model in Section 2.2.3).

2.2 Simulation Model

To develop a simulation-optimization model, the first step is to create a simulation model that represents a (s,S) periodic review inventory system and outputs the objective of the model given (s,S) as inputs. The objective of the model is to minimize the expected cost per day where cost is measured relative to purchasing cost. The costs of interest are shortage, waste, holding, and ordering.

2.2.1 Assumptions

The simulation model represents a drug at one hospital pharmacy and drug substitutes are not considered. Our modeling framework makes this no substitute assumption because in practice, there is often a preferred choice and even when substitutes are available, these substitutes often come with additional logistics like ensuring the patient does not have a certain comorbidity. Drugs typically have an expiration date that is with respect to the end of the month (e.g., July 31). Therefore, our model assumes all drugs that arrive in the same month come from the same production batch and have the same end of the month expiration date. Hence, drugs are only discarded at the end of the month. We assume the inventory on-hand and estimated days until inventory expires is known each day t. Also, the lead time is deterministic and non-negative ($l \in \mathbb{N}_0$) where $l = 0$ implies an order placed at the end of day $(t-1)$ arrives at the beginning of day t. Demand is stochastic and supply uncertainty is due to disruptions (i.e., no supply yield uncertainty). We assume these two sources of uncertainty are independent from one another. Last, we assume all demand not met is lost, first-in-first-out protocols are always followed, and there is no seasonal demand.

2.2.2 Input Parameters

For input parameters, we consider a planning horizon of length $|\mathbf{T}|$ days $t = 1,2,...,|\mathbf{T}|$. We denote l as the deterministic lead time ($l \in \mathbb{N}_0$), e as the shelf-life of the drug (months), $|\mathbf{R}|$ as the number of simulation replications (r denotes a particular replication), and b, z, h, and o as the shortage, waste, holding, and ordering cost relative to purchasing cost, respectively. For the discretized square grid, we denote $s_l = S_l$

as the minimum value for s and S, $s_u = S_u$ as the maximum value for s and S, and Δ as the grid increment for s and S. For example, $s_l = 25$, $s_u = 100$, $\Delta = 25 \implies s \in [25, 50, 75, 100]$. For the uncertain parameters in the model, we define d_t as the demand on day t and y_t as a binary variable for the supply disruption status on day t. For the binary variable, $y_t = 0$ denotes supply is disrupted and $y_t = 1$ denotes supply is not disrupted. There are no restrictions on the probability distributions of these sources of uncertainty.

2.2.3 Simulation Model Description

Given these assumptions and input parameters, we describe the simulation model at a high level. Pseudocode for the simulation model can be found in the Appendix and the steps in the simulation model are identified on Figure 1. Also, it is important to note the decision variables (s, S) are inputs to the simulation model and we discuss how to optimize these decision variables in Section 3.

The simulation model begins with the function StaticSim where for each replication r, the function generates a daily demand pattern and supply disruption pattern using the associated input parameters and probability distributions. Then, the InventoryProcess function is called to calculate the number of shortages, drugs wasted, orders placed, and drugs held during the planning horizon for this replication r. The function InventoryProcess calculates these values by taking the input parameters, demand pattern for replication r, supply pattern for replication r, s, and S and simulating how the system would operate with these parameters. For each day in the planning horizon, (1) orders arrive to the pharmacy at the beginning of the day, (2) inventory levels are updated, (3) inventory levels are updated based on the demand observed on day t, (4) if day t corresponds to the end of the month, expired drugs are discarded, and (5) an order is placed according to the (s, S) inputs if supply is not disrupted. This process continues until the end of the planning horizon is achieved. At this point, the objective value is calculated and InventoryProcess returns this output to the StaticSim function. All replications r follow the same process (computed in parallel to reduce computation time) and at the end, StaticSim calculates the average objective value over all replications r.

3 SIMULATION-OPTIMIZATION

When using the simulation model described in Section 2.2 to solve for the near-optimal (s, S) periodic review policies, we consider three approaches: Exhaustive GS, COMPASS, and Binary GS. Also, we use common random numbers regardless of the search procedure.

3.1 Near-Optimal Policies Using Exhaustive Grid-Search

Exhaustive GS takes a discretized grid for the parameters s and S and enumerates the objective value for all feasible solutions. In this inventory policy setting, a solution is only feasible if $s \leq S$. Since one is enumerating all solutions, the computation time can become excessive, especially for fine grid spaces.

3.2 Near-Optimal Policies Using COMPASS

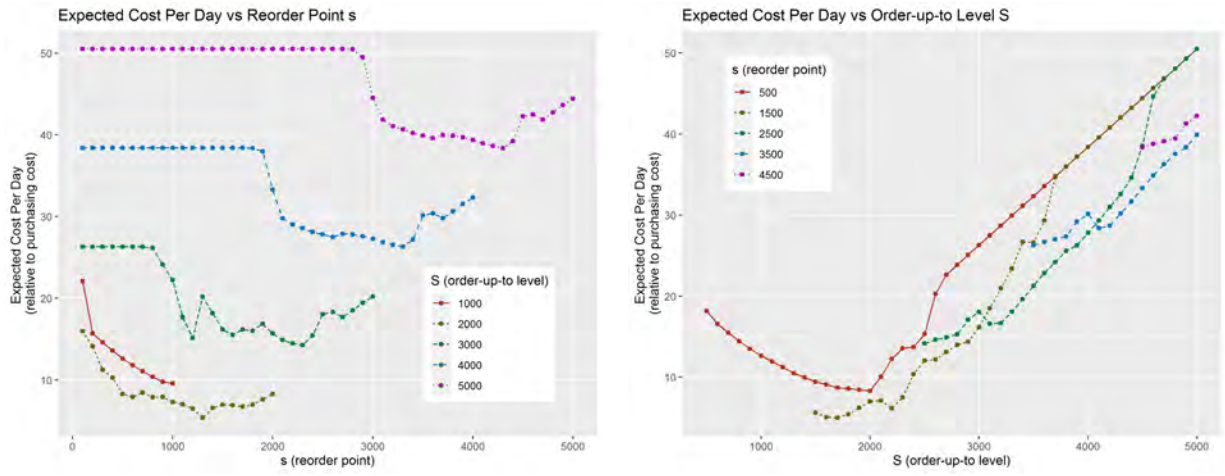
COMPASS is a discrete simulation-optimization algorithm where at each iteration, a most promising area is defined which depends on the distance between a feasible solution and the best solution found thus far (Hong and Nelson 2006). This most promising area is adaptive and to keep consistency with the other search procedures, we allocate the same number of simulation replications to all enumerated solutions. Also, the algorithm requires a uniform search procedure over the discretized grid, so we implement the RMD algorithm (Hong and Nelson 2006).

3.3 Near-Optimal Policies Using Binary Grid-Search

Binary GS takes a discretized grid for the parameters s and S, but instead of enumerating all solutions, binary searches are completed successively on the rows and columns until no improvement to the objective value is made. For initialization, the algorithm enumerates all diagonal solutions on the discretized grid.

3.3.1 Structural Properties of the Objective Function

Our empirical observations support that we cannot assume the objective function is always monotonically decreasing and then monotonically increasing with respect to s for fixed S and S for fixed s (i.e., the negation of the objective function is unimodal). Consider the drug test case (described in Section 4) with 5,000 replications and solved using an Exhaustive GS (Section 3.1). Figure 2 displays (a) Expected Cost Per Day vs Reorder Point s and (b) Expected Cost Per Day vs Order-up-to Level S for five different values of S and s, respectively. For (a), one may consider why the objective function is flat up to some reorder point s. This occurs in cases where the reorder point, s, is too small and the order-up-to level, S, is too large resulting in there (almost surely) being a significant number of drugs that expire before there is sufficient demand to use these drugs (i.e., inventory position drops to 0). Therefore, an order is placed to bring the inventory level up to S regardless of the value of s and it is important to consider that all of these drugs have the same expiration date. Figure 2 illustrates the objective function is not always monotonically decreasing and then monotonically increasing, but that it does exhibit characteristics of this structure.



(a) Expected Cost Per Day vs Reorder Point s.

(b) Expected Cost Per Day vs Order-up-to Level S.

Figure 2: Objective Function (Expected Cost Per Day) With Respect to s for Fixed S and S for Fixed s.

We also consider if the monotonically decreasing and then increasing property does not hold for this test case due to simulation error. Regardless of the number of simulation replications completed, there will be some error with respect to the objective value since this output is a result of a stochastic process. When considering solutions that are significantly different from one another at a confidence level of 95%, 5.7% of the pairwise comparisons with respect to the columns of the grid and 3.9% of the pairwise comparisons with respect to the rows of the grid fail to meet the monotonically decreasing then increasing structure. These results demonstrate the proportion of time the monotone property does not hold is relatively small.

These findings motivate a regular binary search coupled with a forced binary search to ensure one is not caught in a local optimum. Forced binary search means we complete a binary search above, to the right, below, and to the left of the current solution regardless of the objective value of the neighboring solutions. The algorithm can be seen as a metaheuristic where we exploit when neighboring solutions improve the objective value. However, before terminating, we also accept worse neighbors with the aim of avoiding local optimality. We simply refer to the algorithm as Binary GS and details follow in the subsequent section.

3.3.2 Description of Binary Grid-Search

Throughout the explanation of the Binary GS algorithm, we will refer to the discretized grid described earlier (square grid matrix where the rows correspond to s values in increasing magnitude and the columns correspond to S values in increasing magnitude). A binary search, also known as a half-interval search, is when one successively divides a unimodal search interval in half to find the global maximum (i.e., negation finds global min). For example, consider the objective value, V, for a fixed s across nine increasing S values: $V = (100, 99, 75, 72, 84, 88, 94, 100, 106)$. One first starts at the midpoint of the full interval (i.e., 84). Then, one considers the neighboring solutions (i.e., 72 and 88). Since $72 < 84$, one considers the midpoint (i.e., 75) of the halved interval $(100, 99, 75, 72, 84)$ and the same procedure continues.

To initialize the algorithm, (a) we enumerate all feasible solutions on the diagonal of the discretized grid with the number of replications desired by the user. From these initial enumerations, (b) we select the column that minimizes the objective function and perform a binary search on this column. (c) The feasible solution minimizing the objective function is chosen as the starting point.

Then, we begin an iterative process where (1) we enumerate the objective value to the left and right of our current solution. If both of these objective values are greater than the current solution, no change is made to the current solution. Otherwise, we carry out a binary search on this row of the grid and update our current solution with the new minimum found. (2) We enumerate the objective value above and below our current solution. If both of these objective values are greater than the current solution, no change is made to the current solution. Otherwise, we carry out a binary search on this column of the grid and update our current solution with the new minimum found. (3) We check if the solution found in the previous iteration is the same solution as this iteration. If not, the iterative process starts back at (1) with the current solution. If yes, then (4) a binary search is completed on the interval above, to the right, below, and to the left of the current solution to avoid local optimality (Section 3.3.1). As an example, the above interval means we consider the $S_{current}$ column and search the interval $s \in [s_1, s_{current}]$. These four interval searches are completed regardless of the value of the neighboring solutions of the current solution (i.e., the neighboring solutions may be worse). The key idea is that if the current solution is truly optimal, then performing a binary search on these four intervals should find a solution no better than the current solution. If the current solution remains optimal, the algorithm terminates. If not, then the iterative procedure starts back at (1) with the best solution found thus far as the current solution. Figure 3 below demonstrates the process where the light shaded circle with (*) denotes the solution and the fully shaded black circles denote the search area for that step. Pseudocode for the algorithm can be found in the Appendix.

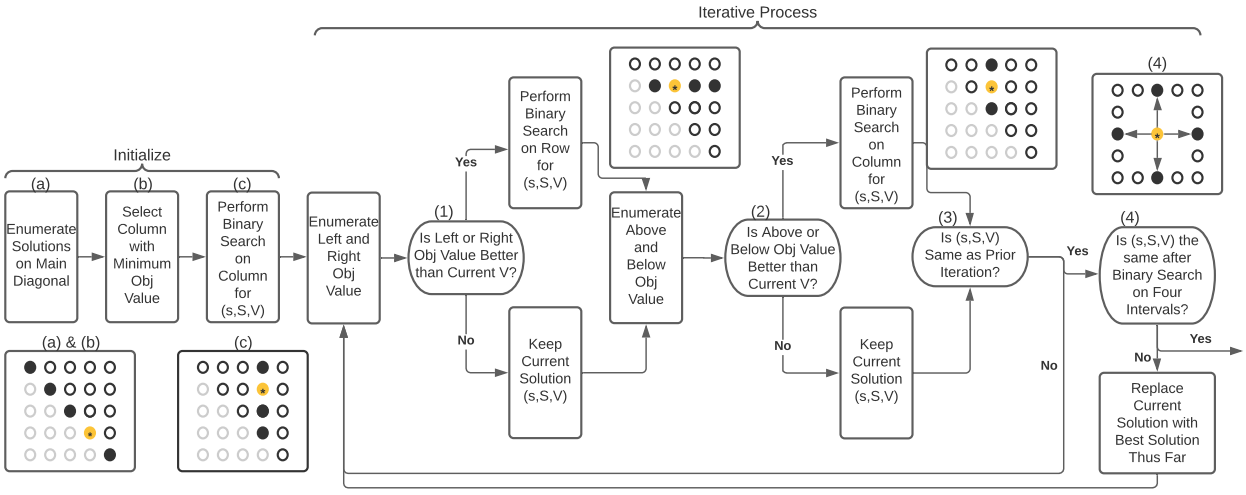


Figure 3: Process Flow of Binary Grid-Search (s: reorder point, S: order-up-to level, V: objective value).

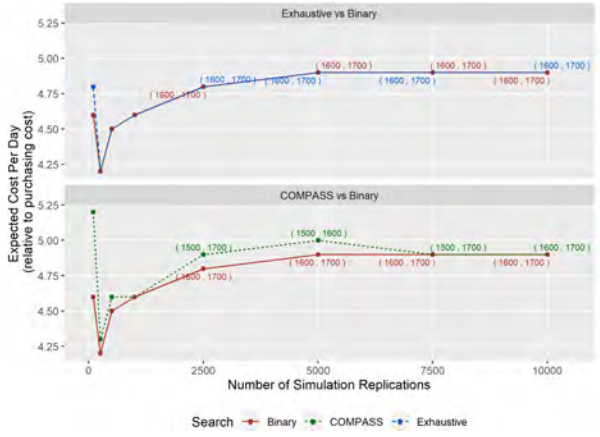
4 EXPERIMENTAL ANALYSIS

To evaluate the performance of the simulation-optimization model solved with an Exhaustive GS, COMPASS, and Binary GS, we present eight different cases where we vary the number of replications ("computational cost") for each method. We vary the replications from 100-10,000. We provide this range because $|\mathbf{R}| = 5,000$ replications ensures a confidence interval width, w, of at most 1. Specifically, one can determine $|\mathbf{R}|$ by ensuring $w > (2cd_{max})/(\sqrt{|\mathbf{R}|})$ where c is a user-specified confidence level (e.g., $c = 1.96$ for 95%), d_{max} is the maximum standard deviation from the simulation model when analyzed on a discretized grid, and w is the desired confidence interval width. For each of these eight cases, we present the near-optimal objective value, near-optimal s, near-optimal S, and run time when using (a) Exhaustive GS, (b) COMPASS, and (c) Binary GS. Objective values are rounded to the nearest tenths place. For COMPASS, we consider $m = \frac{1}{10}(no.\ of\ feasible\ solutions)$ solutions to randomly and uniformly search with replacement each iteration and consider a warm-up of $W = 10$ iterations for the uniform search. We terminate this algorithm when all neighboring solutions have been enumerated and the same solution has been found for 10 iterations in a row. All numerical experiments are completed in R (Version 4.0.3) on a Windows desktop (4.28GHz).

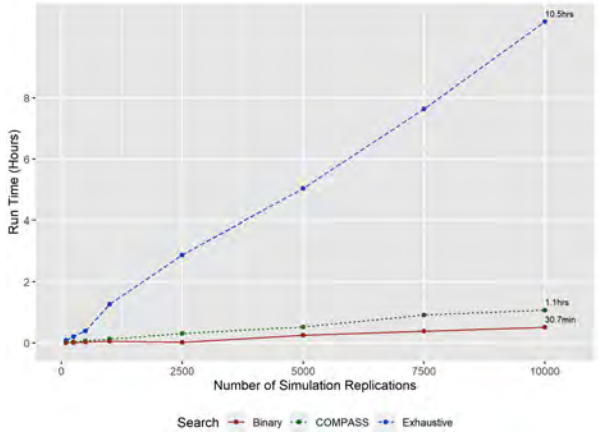
4.1 Pharmacy Model Input Parameters and Numerical Results

We use the input parameters across all eight cases where the cost (relative to purchasing cost) associated with waste, shortages, holding, and ordering are relevant to the pharmacy system of interest. We define $|\mathbf{T}| = 360$ Days (30 Days/Month), $l = 6$ Days, e = 3 Months, b = 5 (Haijema 2013), z = 1 (Haijema 2013), h = 0.001 (Jia and Zhao 2017), o = 0.5 (Haijema 2013), $s_l = S_l = 100$, $s_u = S_u = 5000$, $\Delta = 100$, Demand $\sim$ Poisson($\lambda = 25$/day), Days to Supply Disruption [Recovery] $\sim$ Geometric($p_1 = 0.01$) [Geometric($p_2 = \frac{1}{30}$)].

Figure 4 below displays (a) Expected Cost Per Day vs Number of Simulation Replications and (b) Run Time (hours) vs Number of Simulation Replications. For Figure 4 (a), we present the objective value with replications independent of the replications used to optimize the (s,S) policies to avoid optimization bias. Notice that in all cases, the Binary GS algorithm finds the exact same solution as the Exhaustive GS algorithm or a solution that is better in objective value when accounting for optimization bias. This also holds for the COMPASS algorithm. Furthermore, the Binary GS stays within an hour run time (maximum run time is approximately 30 minutes) whereas the Exhaustive GS increases to a computation time of approximately 10.5 hours at 10,000 replications. Another interesting aspect is the Binary GS algorithm always outperforms the COMPASS algorithm with respect to run time.



(a) Expected Cost Per Day vs Number of Simulation Replications (no. in () represent the (s,S) policies).



(b) Simulation Run Time vs Number of Simulation Replications (run time indicated for 10,000 replications).

Figure 4: Numerical Results with Pharmacy Model Input Parameters.

The Exhaustive GS guarantees the best solution is found (before considering optimization bias). However, it is not only computationally inefficient, but also unrealistic for the pharmacy system of interest. In particular, when considering the demand for all drugs at the University of Michigan's Central Pharmacy, the 20% highest demand drugs (545 drugs) make up about 85% of the total demand. Hence, if one is interested in updating policies every 3 months (2,160 hours) using 10,000 replications, one would need $(545)*(10.5hrs) = 5,722.5hrs$, or about 7.95 months, when using the Exhaustive GS. Hence, even if one desktop computer was solely utilized to run the exhaustive algorithm for the entire 3 months, one could not solve for all policies within this time frame. However, if using the Binary GS, one could solve for the policies in $(545)*(0.5hrs) = 272.5hrs \approx 11.35$ days.

5 CONCLUSION AND FUTURE RESEARCH

In this paper, we created a simulation-optimization (s,S) periodic review inventory model that accounts for stochastic demand, supply disruptions, perishability, and positive lead time. We solve the simulation-optimization model using a Binary GS algorithm which uses the structural properties of the objective function. Based on our experimental results, we notice this algorithm improves computation time without sacrificing accuracy of the solution.

As for limitations, we assume a first-in-first-out system. If this is violated, then one can expect more waste and more shortages. However, through observations at the Central Pharmacy at the University of Michigan, first-in-first-out is a valid assumption with the protocols in place. We also assume demand and supply are independent of one another. When constructing the demand and supply patterns for a replication, r, one can consider the dependency of these uncertainties and we leave that to future work. Other assumptions such as lost-sales, deterministic lead time, and expired drugs being discarded at the end of the month can easily be adjusted in the model to tailor it to the characteristics of the system of interest.

In this research, we do not theoretically exploit the structure of the objective function and we do not provide theoretical guarantees on the Binary GS algorithm. Also, we find the Binary GS algorithm does not sacrifice the accuracy of the solution when defining the discretized grid with respect to s and S, but an alternative approach is defining the discretized grid with respect to s and $\delta = S - s$. Finally, we compare the Binary GS algorithm to an Exhaustive GS and the COMPASS algorithm, but other methods one can consider for comparison are stochastic gradient descent with random restart, random search, and Bayesian optimization. We leave these ideas for future research.

This research provides pharmacy inventory managers with a valuable tool that can aid decision-making in the complex pharmacy system.

ACKNOWLEDGMENTS

The Central Pharmacy at the University of Michigan provided information regarding protocols in place (e.g., first-in-first-out) and the number of drugs the pharmacy is responsible for. We would like to thank Bruce Chaffee, Seema Jetli, and Karl Renius for providing access to the data. We would also like to thank Nicole Mor for the helpful literature searches.

APPENDICES

A PSEUDOCODE FOR (s,S) INVENTORY POLICY SIMULATION MODEL

function StaticSim

for r=1:$|\mathbf{R}|$ *% iterate through all replications*
  Generate Demand d_t for $t = 1,2,...,|\mathbf{T}|$ *% generate demand pattern using appropriate probability distribution*
  Generate Supply y_t for $t = 1,2,...,|\mathbf{T}|$ *% generate supply pattern using appropriate probability distributions*
  ObjValue[r] = InventoryProcess *% calculate objective value using InventoryProcess function*
  ————————
  function InventoryProcess
  for t=1:$|\mathbf{T}|$ *% iterate through each day in planning horizon*
    Step 1: Update inventory based on amount that arrives at the beginning of day t
    Inventory[e]=Inventory[e]+InventoryArrival[1] *% store the amount that arrives as the "newest" inventory*
    Step 2: Update inventory en-route levels
    if $l = 0$ *% lead time is* $l = 0$
      InventoryArrival[1]=0 *% inventory to arrive tomorrow updated when order decision is made below*
    else
      InventoryArrival[i]=InventoryArrival[$i+1$] $\forall i = 1,...,l$ *% shift all inventory en route by one day*
      InventoryArrival[$l+1$]=0 *% inventory to arrive in l days updated when order decision made below*
    end if
    Step 3: Update inventory levels based on the amount of demand observed on day t
    NeedsFilled= d_t *% keep track of amount of demand that needs filled on day t*
    Short[t]=max$\{0, d_t - sum(\text{Inventory})\}$ *% keep track of the amount short on day t*
    $i = 1$ *% keep track of remaining lifetime index to enforce FIFO policy (lifetime is in months)*
    while NeedsFilled $\neq$ 0 *% iterate until all demand filled or all inventory used*
      if Short[t]>0 *% check if one is short on day t*
        NeedsFilled=0 *% all inventory is used*
        Inventory[i]=0 $\forall i = 1,2,...,e$ *% update that all inventory is used*
      else if NeedsFilled $\geq$ Inventory[i] *% drug with lifetime remaining i has less inventory than required*
        NeedsFilled=NeedsFilled-Inventory[i] *% update how much demand needs filled after satisfying demand with "oldest" stock*
        Inventory[i]=0 *% indicate all inventory with remaining lifetime i is used*
      else
        Inventory[i]=Inventory[i]-NeedsFilled *% update inventory for this remaining lifetime i*
        NeedsFilled=0 *% no more demand for day t needs filled*
      end if
      $i = i + 1$ *% increment lifetime remaining index*
    end while
    Step 4: Discard any expired drugs after observing demand on day t (this only occurs at the end of the month)
    if $t \in \{30,60,90,120,150,180,210,240,270,300,330,360\}$ *% last day of the month*
      Waste[t]=max{0,Inventory[1]} *% calculate waste in day t*
      Inventory[1]=0 *% update inventory after discarding expired*
    end if
    Step 5: Place order based on (s,S) inventory policy
    if [sum(Inventory)+sum(InventoryArrival)] $< s$ *% place an order because inventory position falls below reorder point s*
      Arrival=S-[sum(Inventory)+sum(InventoryArrival)] *% place an order for S minus the current inventory position*
      Arrival=(y_t)Arrival *% inventory only arrives if supply is not disrupted*
      Order[t]=1(y_t) *% indicate an order is placed on day t if supply is not disrupted*
    else
      Arrival=0 *% do not place an order*
    end if
    InventoryArrival[$l+1$]=Arrival *% update inventory en route based on most recent order decision*
    Step 6: Update inventory levels by age (this only occurs at the end of the month)
    if $t \in \{30,60,90,120,150,180,210,240,270,300,330,360\}$ *% last day of the month*
      Inventory[i]=Inventory[$i+1$] $\forall i = 1,...,(e-1)$ *% subtract 1 month from remaining lifetime*
      Inventory[e]=0 *% "newest" inventory set to 0 until order arrives next day*
    end if
    Hold[t]=sum(Inventory) *% store how many drugs are held on day t*
  end for
  Output: ObjRep[r]= $\sum_{t=31}^{|\mathbf{T}|} \frac{b(Short[t])+z(Waste[t])+o(Order[t])+h(Hold[t])}{(b+z+o+h)(|\mathbf{T}|-30)}$ *% begin at day 31 to remove intialization bias*
  ————————
end for
Output: Expected Cost Per Day = $\frac{1}{|\mathbf{R}|} \sum_{r=1}^{|\mathbf{R}|}$ ObjValue[r]

B PSEUDOCODE FOR BINARY GRID-SEARCH

function BinaryOptGrid

Step 1: Initialize the Grid for s and S
sGrid=$[s_l, s_u]$; by=Δ *% initialize grid for s*
SGrid=$[S_l, S_u]$; by=Δ *% initialize grid for S*
Step 2: Form Matrix and Denote Infeasible Solutions
Matrix is |sGrid| by |SGrid|, but referred to by (s,S) for clarity
ObjMatrix=matrix($|sGrid|, |SGrid|$) *% initialize matrix to store objective values*
ObjMatrix[s,S]=$Infeasible \quad \forall s,S$ such that $s > S$ *% denote infeasible solutions*
Step 3: Determine Objective Value for Main Diagonal
ObjMatrix[s,S]=StaticSim($s = c,\ S = c$) $\quad \forall c \in$ sGrid *% store objective value from StaticSim function given the s and S inputs*
CurrentOpt=[minimum sol_c, $s_{min\,c}$, $S_{min\,c}$] *% store the current minimum objective value from the main diagonal solution and the associated (s,S) in a vector*
CurrentOpt=[$s = \max\{s_l, \lfloor \frac{S_{current}}{2} \rfloor\},\ S = S_{current}$] *% update the current solution using the midpoint value of s for this column*
Step 4: Perform a Binary Search on the Column With the Current Solution
CurrentOpt=BinarySearch(column, CurrentOpt) *% Perform binary search on the column; store new objective value and associated (s,S)*
OptFound=FALSE *% initialize an indicator variable for iterative process*
MaxIter *% define the maximum number of iterations*
while OptFound==FALSE
    Step 5: Check Solutions to the Left and Right
    % Left of Current Solution
    ObjMatrix[s,S]=StaticSim(s'=$s_{current},\ S = \max\{s', \max\{S_l, S_{current} - \Delta\}\}$)
    % Right of Current Solution
    ObjMatrix[s,S]=StaticSim(s'=$s_{current},\ S = \max\{s', \min\{S_u, S_{current} + \Delta\}\}$)
    if CurrentOpt $\leq$ LeftSolution & CurrentOpt $<$ RightSolution *% no improvement to objective value*
        continue algorithm
    else
        % moving improves the objective value
        CurrentOpt=BinarySearch(row, CurrentOpt) *% Perform binary search on the row; store new objective value and associated (s,S)*
    end if
    Step 6: Check Solutions Above and Below
    % Above Current Solution
    ObjMatrix[s,S]=StaticSim(s'=$\max\{s_l, s_{current} - \Delta\},\ S = S_{current}$)
    % Below Current Solution
    ObjMatrix[s,S]=StaticSim(s'=$\min\{s_u, s_{current} + \Delta\},\ S = S_{current}$)
    if CurrentOpt $\leq$ AboveSolution & CurrentOpt $<$ BelowSolution *% no improvement to objective value*
        continue algorithm
    else
        % moving improves the objective value
        CurrentOpt=BinarySearch(column, CurrentOpt) *%Perform binary search on column; store new objective value and associated (s,S)*
    end if
    Step 7: Check If Solution Remains Unchanged from Prior Iteration
    if $(sol_{current}, s_{current}, S_{current})$==$(sol_{prior\ iteration}, s_{prior\ iteration}, S_{prior\ iteration})$
        % Perform Forced Binary Search When True
        Perform binary search on the search interval above, to the right, below, and to the left of $(s_{current}, S_{current})$ regardless of the value of the neighboring solutions
        if All solutions are inferior to $(s_{current}, S_{current})$
            OptFound==TRUE *% near-optimal solution found and algorithm completed*
        else
            % a new near-optimal solution is found
            Replace $(s_{current}, S_{current})$ with the new optimal found and continue algorithm
        end if
    else
        continue algorithm
    end if
    if MaxIter is exceeded
        terminate algorithm early
        % if the algorithm is not terminating, increase the number of simulation replications
    end if
end while
Output: Near-Optimal Solution = $(sol_{current}, s_{current}, S_{current})$

REFERENCES

Atan, Z., and M. Rousseau. 2016. "Inventory optimization for perishables subject to supply disruptions". *Optimization Letters*:89–108.

Bashyam, S., and M. C. Fu. 1991. "Application of Perturbation Analysis to (s,S) Inventory Systems". *Proceedings of the 1991 Winter Simulation Conference*.

Dillon, M., F. Oliveira, and B. Abbasi. 2017. "A two-stage stochastic programming model for inventory management in the blood supply chain". *International Journal of Production Economics*:27–41.

Franco, C., and E. Alfonso-Lizarazo. 2020. "Optimization under uncertainty of the pharmaceutical supply chain in hospitals". *Computers and Chemical Engineering*:1–13.

Fu, M. 2015. *Handbook of simulation optimization*. New York: Springer.

Fu, M., and J. Q. Hu. 1997. *Conditional Monte Carlo*. Boston, MA: Springer US.

Haijema, R. 2013. "A new class of stock-level dependent ordering policies for perishables with a short maximum shelf life". *International Journal of Production Economics*:434–439.

Hartman, M., A. B. Martin, N. Espinosa, A. Catlin, and The National Health Expenditure Acc. 2018. "National Health Care Spending In 2016: Spending And Enrollment Growth Slow After Initial Coverage Expansions". *Health Affairs*:150–160.

Hong, L. J., and B. L. Nelson. 2006. "Discrete Optimization via Simulation Using COMPASS". *Operations Research*.

Jia, J., and H. Zhao. 2017. "Mitigating the U.S. Drug Shortages Through Pareto-Improving Contracts". *Production and Operations Management*:1463–1480.

Kara, A., and I. Dogan. 2018. "Reinforcement learning approaches for specifying ordering policies of perishable inventory systems". *Expert Systems with Applications*:150–158.

Konstantaras, I., K. Skouri, and A. Lagodimos. 2019. "EOQ with independent endogenous supply disruptions". *Omega*:96–106.

Li, J., L. Liu, H. Hu, Q. Zhao, and L. Guo. 2018. "An Inventory Model for Deteriorating Drugs with Stochastic Lead Time". *International Journal of Environmental Research and Public Health*:1–20.

Nguyen, D. H., and H. Chen. 2019. "Optimization of a perishable inventory system with both stochastic demand and supply: comparison of two scenario approaches". *Croatian Operational Research Review*:175–185.

Phuong, J. M., J. Penm, B. Chaar, L. D. Oldfield, and R. Moles. 2019. "The impacts of medication shortages on patient outcomes: A scoping review". *PLOS ONE*.

Rajendran, S., and R. A. Ravindran. 2019. "Inventory management of platelets along blood supply chain to minimize wastage and shortage". *Computers & Industrial Engineering*:714–730.

Rajendran, S., and S. Srinivas. 2020. "Hybrid ordering policies for platelet inventory management under demand uncertainty". *IISE Transactions on Healthcare Systems Engineering*:113–126.

Saedi, S., O. E. Kundakcioglu, and A. C. Henry. 2016. "Mitigating the impact of drug shortages for a healthcare facility: An inventory management approach". *European Journal of Operational Research*:107–123.

Sahin, I. 1982. "On the Objective Function Behavior in (s, S) Inventory Models". *Operations Research*:709–724.

Schmitt, A. J., L. V. Snyder, and Z.-J. M. Shen. 2010. "Inventory systems with stochastic demand and supply: Properties and approximations". *European Journal of Operational Research*:313–328.

Skouri, K., I. Konstantaras, A. Lagodimos, and S. Papachristos. 2014. "An EOQ model with backorders and rejection of defective supply batches". *International Journal of Production Economics*:148–154.

Zhang, X., D. Meiser, Y. Liu, B. Bonner, and L. Lin. 2014. "Kroger Uses Simulation-Optimization to Improve Pharmacy Inventory Management". *Interfaces*:70–84.

Zheng, Y.-S., and A. Federgruen. 1991. "Finding Optimal (s,S) Policies is About as Simple as Evaluating a Single Policy". *Operations Research*:654–665.

AUTHOR BIOGRAPHIES

LAUREN L. CZERNIAK is a Ph.D. student in the Department of Industrial and Operations Engineering at the University of Michigan. She received her BS in Industrial Engineering from the University of Pittsburgh and received her MSE in Industrial and Operations Engineering from the University of Michigan. Her research focuses on developing and applying stochastic models to address current challenges in healthcare with applications in pharmaceutical drugs, glaucoma, and concussion management. Her email address is czernL@umich.edu. Her website is https://sites.google.com/umich.edu/lauren-czerniak/home.

MARK S. DASKIN holds the Clyde W. Johnson Collegiate Professorship in the Department of Industrial and Operations Engineering at the University of Michigan. He received his BS and PhD in Civil Engineering from the Massachusetts Institute of Technology. His research focuses on supply chain network design in general and in particular, facility location models. His email address is msdaskin@umich.edu. His website is https://daskin.engin.umich.edu.

MARIEL S. LAVIERI is an Associate Professor in the Department of Industrial and Operations Engineering at the University of Michigan. She received her BS in both Industrial and Systems Engineering and Statistics and a minor in String Bass Performance from the University of Florida. She received her MS and PhD in Management Science from the University of British Columbia. Her research applies operations research to healthcare topics with methods such as dynamic programming, stochastic control, and continuous, partially observable state space models. Her email address is lavieri@umich.edu. Her website is http://www-personal.umich.edu/~lavieri/.

BURGUNDA V. SWEET is the Assistant Dean for Curriculum and Assessment and a Clinical Professor in the Department of Clinical Pharmacy at the University of Michigan College of Pharmacy, and a Clinical Pharmacist for the Michigan Medicine Health System. She received her PharmD from University of the Pacific. Her email address is gsweet@med.umich.edu. Her website is https://pharmacy.umich.edu/people/gsweet.

JENNIFER ERLEY is a Central Pharmacy Manager at Michigan Medicine and Adjunct Clinical Faculty at the University of Michigan College of Pharmacy. She received her BS, PharmD, and MBA from the University of Michigan, and is a Board Certified Pharmacotherapy Specialist. Her email address is jerley@med.umich.edu. Her website is https://medicine.umich.edu/dept/pharmacy/jenn-erley-pharmd-mba-bcps.

MATTHEW A. TUPPS is a Pharmacy Manager over inventory and automation at the University of Michigan Health System as well as Adjunct Clinical Instructor at the University of Michigan College of Pharmacy. He received his PharmD from Ohio Northern University and MHA from Ohio University. His email address is mtupps@med.umich.edu. His website is https://medicine.umich.edu/dept/pharmacy/matthew-tupps-pharmd-mha-0.