Songklanakarin J. Sci. Technol.
35 (1), 115-121, Jan. - Feb. 2013

SJST SONGKLANAKARIN JOURNAL OF SCIENCE AND TECHNOLOGY
http://www.sjst.psu.ac.th

Original Article

Computing nonstationary (s, S) inventory policies via genetic algorithm

Kannapha Amaruchkul* and Surapong Auwatanamongkol

Graduate School of Applied Statistics,
National Institute of Development Administration (NIDA), Bangkapi, Bangkok, 10240 Thailand.

Received 14 February 2012; Accepted 29 October 2012

Abstract

A periodic-review inventory model with nonstationary stochastic demand under an (*s*, *S*) policy is considered. We apply a genetic algorithm to solve for reorder points and order-up-to levels which minimize an expected total cost. A closed-form exact expression for the expected cost is obtained from a nonstationary discrete-time Markov chain. In our numerical experiments, our approach performs very well.

Keywords: stochastic model applications; genetic algorithm; nonstationary inventory management

1. Introduction

We consider a periodic-review single-item inventory problem. Demand process is nonstationary; specifically, demand in each time period is independent but not necessarily identically distributed. Demand distribution depends on the time period. Nonstationary demand process is commonly found in practice; e.g., repair parts of a machine in which the demand rate varies with time, parts for preventive maintenance where the maintenance schedule is pre-specified, and a seasonal retail product.

In this article, we address an (*s*, *S*) inventory policy: if the inventory position is less than or equal to the reorder point, denoted by *s*, then an order is placed so that the inventory position is brought back to the order-up-to level, denoted by *S*. The policy is easy to implement in practice. The control parameters, *s* and *S*, are chosen such that an expected total inventory cost over a finite planning horizon is minimized. We investigate the use of a genetic algorithm (GA) to heuristically solve the optimization problem, in which the objective function is the expected cost and the decision variables are reorder points and order-up-to levels for all time periods. From numerical experiments, our method is very close to optimal for most problem instances. For a real-world problem instance in which the optimal policy could not be computed, our heuristic outperforms other heuristics of which variants may be found in practice.

There are many papers concerning the (*s*, *S*) policy under the assumption of the stationary demand process. For instance, Sobel & Zhang (2001) show that the (*s*, *S*) policy is optimal. Zheng & Federgruen (1991) propose an algorithm to find the optimal *s* and *S*. Nevertheless, there are fewer papers on the (*s*, *S*) policy under the assumption of the nonstationary demand; e.g., Bollapragada & Morton (1999) propose a heuristic procedure to find the control parameters. They first transform the nonstationary problem into a stationary problem, in which a demand distribution is identical throughout time period, and a parameter of the demand distribution is averaged over a pre-specified length of time. Then, they apply the algorithm in Zheng & Federgruen (1991) to find the control parameters.

Another related problem in which demand is nonstationary (time-varying) but deterministic, known as the dynamic lot sizing problem, has been solved by several solution procedures, including meta-heuristics such as tabu search, GA and simulated annealing; see Jans & Degraeve (2007) for a review. To the best of our knowledge, this article

* Corresponding author.
Email address: kamaruchkul@gmail.com

is among the first to apply the GA to the inventory model that explicitly incorporates demand uncertainty. To a certain extent, the nonstationary stochastic inventory is more complex than a deterministic dynamic lot sizing model. This article exemplifies that GA is powerful enough to deal with such a complex model.

The rest of this article is organized as follows. We describe the nonstationary (s, S) inventory model in Section 2 and some solution procedures, including the GA, for solving the optimization problem in Section 3. Section 4 contains our numerical experiments to evaluate the performance of the proposed GA. Finally, the article is concluded in Section 5.

2. Nonstationary (*s*, *S*) Inventory Model

In this section, we present a mathematical programming model for choosing the control parameters, which minimize the expected total cost. Throughout this article, $1\{A\}$ denotes an indicator function: if condition holds, the function is equal to one and zero otherwise. Notation $(y)^+ = \max\{y, 0\}$ denotes the positive part of a real number y.

2.1 Problem formulation

The planning horizon is divided into n periods, such that in each period demand materializes once. In time period t, let D_t denote the random demand. Assume that unsatisfied demand is lost. Let K_t be a setup cost, h_t a holding cost per unit of left-over inventory, and b_t a penalty cost per unit of lost sale. We denote the reorder point and the order-up-to level by s_t and S_t respectively. For each period $t=1,2,\ldots,n$, the sequence of events is as follows:

1. The inventory level is reviewed. Based on the current inventory level X_t, we decide how much additional stock to order, a_t. The order is received immediately; i.e., the replenishment lead time is zero. (This is not restrictive, because the requirement in each period can be shifted to take account of positive lead time, say l; we would interpret D_t as the shifted demand D_{t+l}.)

Under a nonstationary (s, S) policy, if the inventory level at the beginning of period t is X_t, then we order

$$a_t = \begin{cases} S_t - X_t & \text{if } X_t \le s_t \\ 0 & \text{if } X_t > s_t \end{cases}$$

If the inventory level is less than or equal to s_t, we place an order to bring the inventory level to S_t; otherwise, we do not order.

2. Demand D_t materializes. If demand is greater than the inventory level, then the excess demand is lost. The amount of lost sale is $\left[D_t - (X_t + a_t)\right]^+$. The inventory at the end of period t is $(X_t + a_t - D_t)^+$.

3. The following cost is incurred

$$c_t(X_t, a_t) = E[K_t 1\{a_t > 0\} + h_t (X_t + a_t - D_t)^+ + b_t \left[D_t - (X_t + a_t)\right]^+]$$

At the end of the planning horizon, if the ending inventory is X_{n+1}, a terminal cost of $r_{n+1}(X_{n+1})$ is incurred.

Let i_0 be the initial inventory at the beginning of the planning horizon, and κ the maximum inventory level (e.g., warehouse capacity). We want to determine the control parameter $((s_1, S_1), \ldots, (s_n, S_n))$ that minimizes the expected total cost; i.e.,

$$\min E\left[\sum_{t=1}^{n} c_t(X_t, a_t) + r_{n+1}(X_{n+1}) \mid X_0 = i_0\right] \quad (1)$$

$$\text{s.t. } 0 \le s_t \le S_t \le \kappa \quad \text{for each } t=1,2,\ldots,n \quad (2)$$

For short-hand, the expected total cost in (1) is denoted by $\gamma((s_1, S_1), \ldots, (s_n, S_n); i_0)$.

2.2 Procedure to calculate expected cost

We propose a procedure to evaluate the expected total cost $\gamma((s_1, S_1), \ldots, (s_n, S_n); i_0)$, based on some properties of a discrete-time MC. Assume that demands $D_1, D_2, \ldots, D_n$ are independent but not necessarily identically distributed. For practical purpose, let D_t be a nonnegative integer-valued random variable.

Let Z_t be the inventory level at the beginning of period t after order a_t is received but before demand D_t materializes. After the first order is received, the inventory level Z_1 becomes

$$z_1 = \begin{cases} S_1 & \text{if } i_0 \le s_1 \\ i_0 & \text{otherwise} \end{cases} \quad (3)$$

For each $t=1,2,\ldots,n-1$, we have the following recursive equation:

$$Z_{t+1} = \begin{cases} S_{t+1} & \text{if } Z_t - D_t \le s_{t+1} \\ Z_t - D_t & \text{if } Z_t - D_t > s_{t+1} \end{cases} \quad (4)$$

Equation (4) can be explained below. Note that $(Z_t - D_t)$ is the inventory level after demand in period materializes. If it is less than or equal to the reorder point s_{t+1}, then the inventory level is brought back to the order-up-to level S_{t+1}.

Let $s_t^u = \max\{i_0, S_1, S_2, \ldots, S_t\}$ and $\tilde{S}_t = \{s_t + 1, \ldots, s_t^u\}$. Note that $\tilde{S}_t$ is not empty, since $S_t > s_t \ge 0$ and $i_0 \ge 0$. Under the policy with the control parameter $((s_1, S_1), \ldots, (s_n, S_n))$, we have that $s_t + 1 \le Z_t \le s_t^u$; i.e., for each $t=1,2,\ldots,n$

$$Z_t \in \tilde{S}_t \quad (5)$$

We explain (5) as follows. In period t, clearly $Z_t > s_t$, since s_t is a reorder point. From our assumption that $D_1, D_2, \ldots$ are integer-valued random variables, Z_t takes on integers: $Z_t > s_t$ is equivalent to $Z_t \ge s_t + 1$. From (3), the largest possible value of Z_1 is $\max\{S_1, i_0\}$. From (4), the largest possible value of Z_2 is the maximum of S_2 and $\max\{S_1, i_0\}$, which is $\max\{S_2, S_1, i_0\}$. Continuing in this fashion, we have that the largest possible value of Z_t is $\max\{i_0, S_1, S_2, \ldots, S_{t-1}, S_t\}$.

Denote the (one-step) transition probability $p_{ij}(t) = P(Z_{t+1}=j|Z_t=i)$ for each $t=1,2,\ldots,n-1$. The transition probability function $p_{ij}(t)$ is given as follows:

1. For each $i \le s_{t+1}$,

$$p_{ij}(t) = 1\{S_{t+1} = j\} \tag{6}$$

2. For each $s_{t+1} < i < S_{t+1}$,

$$p_{ij}(t) = \begin{cases} P(D_t = i-j) & \text{if } s_{t+1} < j < S_{t+1} \\ P(D_t \ge i - s_{t+1}) & \text{if } j = S_{t+1} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

3. For $i = S_{t+1}$,

$$p_{ij}(t) = \begin{cases} P(D_t = i-j) & \text{if } s_{t+1} < j < S_{t+1} \\ P(D_t \ge i - s_{t+1}) + P(D_t = 0) & \text{if } j = S_{t+1} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

4. For each $i > S_{t+1}$,

$$p_{ij}(t) = \begin{cases} P(D_t = i-j) & \text{if } s_{t+1} < j < S_{t+1} \\ P(D_t \ge i - s_{t+1}) + P(D_t = i-j) & \text{if } j = S_{t+1} \\ P(D_t = i-j) & \text{if } j > S_{t+1} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Below, we explain how to obtain formulae (6)—(9). Suppose that $Z_t = i$: At the beginning of period t after the order is received but before demand materializes, the inventory level $Z_t = i$.

1. If $i \le s_{t+1}$, in the next period we must place an order so that $Z_{t+1} = S_{t+1}$ with probability one. Thus, the transition probability becomes

$$P(Z_{t+1} = j|Z_t = i) = \begin{cases} 1, & \text{if } j = S_{t+1} \\ 0, & \text{if } j \ne S_{t+1} \end{cases}$$

which is equivalent to (6).

2. If $s_{t+1} < i < S_{t+1}$. Event $Z_{t+1} = j < S_{t+1}$ occurs when we do not place an order at the beginning of period $(t+1)$. It can be implied from (4) that $Z_{t+1} = Z_t - D_t$; i.e., $D_t = i - j$. Event $Z_{t+1} = j = S_{t+1}$ occurs when we place an order at the beginning of period $(t+1)$. It can be implied from (4) that $Z_t - D_t \le s_{t+1}$; i.e., $D_t \ge i - s_{t+1}$. Transition probability (7) is obtained.

3. If $i = S_{t+1}$. Logic for (8) is similar to that for (7) except that $Z_{t+1} = j = i = S_{t+1}$ implies that $D_t \ge i - s_{t+1}$ or that $D_t = 0$.

4. If $i > S_{t+1}$. Logic for (9) is similar to that for (7) except that now there is a possibility that $Z_{t+1} = j > S_{t+1}$.

Define a t-step transition probability as $p_{ij}^{(t)} = P(Z_{t+1} = j|Z_1 = i)$ for each $j \in \tilde{S}_t$. From the one-step transition probabilities found in (6)-(9), we can compute the t-step transition probabilities by conditioning on the previous state:

$$p_{z_1,j}^{(t)} = \sum_{z \in \tilde{S}_t} P(Z_{t+1} = j \mid Z_t = z) P(Z_t = z \mid Z_1 = z_1) = \sum_{z \in \tilde{S}_t} p_{zj}(t)\, p_{z_1,z}^{(t-1)}$$

After obtaining these probabilities, we are ready to evaluate the objective function.

Steps in calculating the expected cost $\gamma((s_1,S_1),\ldots,(s_n,S_n);i_0)$ are as follows:

1. Compute the one-step transition probabilities from (6)-(9)
   a. $\{p_{z_1,j}(1) : j \in \tilde{S}_2\}$.
   b. For each $t = 2,3,\ldots,n-1$, $\{p_{ij}(t) : i \in \tilde{S}_t, j \in \tilde{S}_{t+1}\}$

2. Compute t-step transition probabilities
   a. $p_{z_1,j}^{(1)} = p_{z_1,j}(1)$ for all $j \in \tilde{S}_2$
   b. For each $t = 2,3,\ldots,n-1$,

$$p_{z_1,j}^{(t)} = \sum_{z \in \tilde{S}_t} p_{zj}(t)\, p_{z_1,z}^{(t-1)} \quad \text{for all} \quad j \in \tilde{S}_{t+1}$$

3. Compute expected costs
   a. Expected total setup cost

$$c_K((s_1,S_1),\ldots,(s_n,S_n);i_0) = K_1 1\{i_0 \le s_1\} + K_2 P(D_1 \ge z_1 - s_2) + \sum_{t=3}^{n} \sum_{z \in \tilde{S}_t} P(D_{t-1} \ge z - s_t)\, p_{z_1,z}^{(t-2)}$$

   b. Expected total holding cost

$$c_h((s_1,S_1),\ldots,(s_n,S_n);i_0) = h_1 E[(z_1 - D_1)^+] + \sum_{t=2}^{n} \sum_{z \in \tilde{S}_t} h_t E\left[(z - D_t)^+\right] p_{z_1,z}^{(t-1)}$$

   c. Expected total penalty cost

$$c_g((s_1,S_1),\ldots,(s_n,S_n);i_0) = b_1 E[(D_1 - z_1)^+] + \sum_{t=2}^{n} \sum_{z \in \tilde{S}_t} b_t E\left[(D_t - z)^+\right] p_{z_1,z}^{(t-1)}$$

   d. Expected total cost

$$\gamma((s_1,S_1),\ldots,(s_n,S_n);i_0) = \sum_{z \in \tilde{S}_n} E\left[r_{n+1}(z - D_n)\right] p_{z_1,z}^{(t-1)} + c_K((s_1,S_1),\ldots,(s_n,S_n);i_0) + c_h((s_1,S_1),\ldots,(s_n,S_n);i_0) + c_g((s_1,S_1),\ldots,(s_n,S_n);i_0)$$

3. Solution Procedures

The periodic-review nonstationary inventory problem considered in this article can be formulated as a Markov decision process (MDP); see Section 3.1 below. However, directly

using this approach may become computationally infeasible for large problem instances. We consider heuristics, which can deal with a large problem instance by applying the steps previously developed in Section 2.2. We are interested in two heuristics; the first in Section3.2 is simple and easy to implement, and the latter in Section 3.3 is based on a genetic algorithm.

3.1 Optimal policy

An optimal policy can be constructed from an MDP; see, e.g., Yang *et al.* (2008) and Section 3.2 in Puterman (2005). The corresponding MDP has n decision periods t. In period, the state is the inventory level at the beginning, X_t, and the action is the amount of stock to order, a_t.

For each $i+a\le\kappa$, let $g_t(j|i,a)=P(X_{t+1}=j|X_t=i, a_t=a)$ the probability that the inventory level at the beginning of period $t+1$ is j given that in the previous period it is i and the order of size a is placed.

$$g_t(j|i,a)=\begin{cases}P(D_t\ge i+a) & \text{if } j=0\\ P(D_t=i+a-j) & \text{if } j=1,2,\ldots,i+a\\ 0 & \text{if } j>i+a\end{cases} \quad (10)$$

Let $v_t^*(i)$ be the value (cost-to-go) function for period t given the current state i, i.e., the optimal cost from period t until the end of horizon given that at the beginning of period the inventory level is i. The value function satisfies the following optimality equations:

$$v_t^*(i)=\min_{a\in\{0,1,\ldots,\kappa-i\}}\left\{c_t(i,a)+\sum_{j=0}^{i+a}g_t(j|i,a)v_{t+1}^*(j)\right\} \text{ for each } t=1,2,\ldots,n$$

$$v_{n+1}^*(j)=r_{n+1}(j)$$

The optimal cost is $v_1^*(i_0)$. In period t, if the inventory level at the beginning of the period is i, an optimal action is to place an order quantity of

$$a_t^*(i)=\operatorname{argmin}\left\{c_t(i,a)+\sum_{j=0}^{i+a}g_t(j|i,a)v_{t+1}^*(j)\,\middle|\,a=0,1,2,\ldots,\kappa-i\right\}.$$

The optimal solution can be derived from exhaustive enumeration (e.g., the backward induction algorithm). An optimal order quantity $a_t^*(i)$ depends not only on the current inventory level i but also the current period due to nonstationarity.

The MDP can also be solved via linear programming (LP). A randomized policy is specified by

$$\{d_t(i,a)|t=1,2,,\ldots,n,\ i=0,1,\ldots,\kappa,\ a=0,1,\ldots,\kappa-i\}$$

where $d_t(i,a)$ is the probability that an order of size a is placed given that in period t the inventory level at the beginning of the period is i. The convenient decision variable for the LP model is the unconditional probability that an order of size a is placed in period t and the inventory level at the beginning of the period is i, denoted by $z_t(i,a)$.

Once the $z_t(i,a)$ values are obtained, we have

$$d_t(i,a)=\frac{z_t(i,a)}{\sum_{a=0}^{\kappa-i}z_t(i,a)}.$$

By optimality of deterministic Markov policy, only one $z_t(i,a)>0$ for each i and t. Hence, $d_t(i,a)$ is either zero or one; i.e., given that in period t the inventory level at the beginning of the period is i

$$d_t(i,a)=\begin{cases}1 & \text{if an order of size } a \text{ is placed}\\ 0 & \text{otherwise}\end{cases}$$

The LP formulation for our MDP is

$$\min\sum_{t-1}^{n}\sum_{i=0}^{\kappa}\sum_{a=0}^{\kappa-1}c_t(i,a)z_t(i,a)+\sum_{i=0}^{\kappa}\sum_{a=0}^{\kappa-i}g_n(j|i,a)z_n(i,a)r_{n+1}(j)$$

$$\sum_{i=0}^{\kappa}\sum_{a=0}^{\kappa-1}z_t(i,a)=1;\qquad t=1,\ldots,n \quad (11)$$

$$\sum_{a=0}^{\kappa-j}g_t(j,a)-\sum_{i=0}^{\kappa}\sum_{a=0}^{\kappa-i}g_{t-1}(j|i,a)z_{t-1}(i,a)=0;$$

$$t=1,\ldots,n, j=0,\ldots,\kappa \quad (12)$$

$$z_t(i,a)\ge 0;\qquad t=1,2,\ldots,n,i=0,\ldots,\kappa,a=0,\ldots,\kappa-i$$

The number of decision variables

$$\{z_t(i,a)\,|t=1,2,\ldots,n,\ i=0,1,\ldots,\kappa,\ a=0,1,\ldots,\kappa-i\}$$

is $(\kappa+1)(\kappa+2)n/2$, and the number of functional constraints (11)-(12) is $(\kappa+2)n$. The size of the LP problem is not small. For instance, in the real-world problem instance (Experiment 2) in Section 4, the number of planning horizon is $n=12$ months, and the capacity is $\kappa=648$ units, the number of decision variables is 2.531×10^6, and the number of functional constraints is 7800. Some LP optimizers (e.g., MS Excel Solver Add-In, AMPL student edition) cannot handle such a large problem instance.

3.2 Simple heuristic procedure

The simple heuristic procedure that might be implemented in practice is described below:

In period t the reorder point and order-up-to level are

$$s_t=E[D_t]+\Phi^{-1}(b_t/(b_t+h_t))\sqrt{\text{var}(D_t)}$$

$$S_t=s_t+\sqrt{2K_t\,E[D_t]/h_t}$$

rounding to the nearest integer. This approximation is based on the assumption that demand D_t is normally distributed, and it can be found in some operations management textbooks; see, e.g., page 293 in Nahmias (2009), page 45 in Simchi-Levi *et al.* (2008) and Section 8.5 in Silver *et al.* (1998). The reorder point, s_t, is the sum of the average demand and

the safety stock, which is the safety factor $\Phi^{-1}(b_t/(b_t+h_t))$ times the standard deviation of demand. The order-up-to level, S_t, is the sum of s_t and the economic order quantity, $\sqrt{2K_t E[D_t]/h_t}$.

3.3 Genetic algorithm

GA has proven to be a powerful tool for solving optimization problems. Although the algorithm does not guarantee to get the optimal solution for a problem, it can find nearly optimal or useful solutions for the problem. GA mimics a biological metaphor of genetic evolution. It evolves a population of candidate solutions or chromosomes, encoded in a form of strings of genes, through genetic operators, i.e. selection, crossover, mutation and replacement; see Reeves (2003). The evolution process is performed repeatedly leading to gradual improvement of fitness (objective function) values of candidate solutions and eventually convergence to optima. The procedure of a simple GA is described in Table 1.

3.3.1 Chromosome encoding

Since the goal is to determine parameters $((s_1,S_1),\ldots,(s_n,S_n))$ that minimize the expected total cost, each candidate solution or chromosome consists of *n* genes. Each gene comprises of a pair of integers representing each pair of the parameters. To be a valid solution, the values of the two integers of each pair must satisfy the constraint $0 \le s_t < S_t \le \kappa$ for $t = 1,2,\ldots,n$. . During the course of the GA, all members of population must represent valid solutions so the condition must be held for all the members.

3.3.2 Initialize population

Each member of population is created by randomly assigning an integer value to each member of a pair (s_t,S_t). First, is assigned with a random integer value between 0 and $\kappa-1$; i.e., s_t is randomly selected from a uniform distribution on a set $\{0,1,\ldots,\kappa-1\}$. Then S_t is assigned with a random integer value between the value of s_t+1 and κ. This initialization scheme guarantees that values of the two integers meet the required constraints.

3.3.3 Selection criteria

Parent chromosomes are selected from the current population based on their fitness. The selection criteria must ensure that chromosomes with good fitness have high chance to be selected as a parent for crossover. Fitness proportionate, tournament or ranking selection schemes are commonly used selection strategy. The tournament selection scheme is used in this case as it is simpler and produces reasonably good results. The scheme randomly picks two chromosomes from the population and selects the one with higher fitness as a parent with a predefined probability (e.g., 0.7). With the probability that is greater than 0.5, the fitter chromosome has higher chance to be selected.

3.3.4 Crossover operator

The crossover operators interchange genes between two parents and produce two chromosomes or offspring that inherit genes from these parents. As mentioned earlier, the population must be maintained to have only valid solutions. The crossover operator must produce only valid offspring. Let $(s_{t,1},S_{t,1})$ and $(s_{t,2},S_{t,2})$ be the corresponding genes at locus/position of the two selected parents, respectively. The crossover is performed on each pair of genes with a probability equal to a predefined crossover rate (e.g., 0.8). The crossover is performed on a pair as follows:

1) If $(s_{t,1}<S_{t,2})$ and $(s_{t,2}<S_{t,1})$, then one of the following three ways of interchange is performed with equal probability $(=1/3)$

1. interchange $s_{t,1}$ and $s_{t,2}$
2. interchange $s_{t,1}$ and $s_{t,2}$
3. do both 1 and 2

2) Otherwise, no interchange is performed as it will create invalid solutions.

Notice that the interchange between the two parent genes can occur only when the condition in the above procedure is held.

Table 1. Procedure of GA

- Initialize the population
- While (termination condition is not met) do
  Begin
  - Evaluate the fitness value of each member of the population
  - Select members of the population based on fitness
  - Perform crossover on pairs of selected members to produce offspring
  - Perform mutation on the offspring
  - Replace members of the population with the offspring
  End

3.3.5 Mutation operator

The mutation operator changes the values of a gene in a given offspring with a predefined mutation rate (e.g., 0.05). The mutation could help the search of the GA progress toward alternative optima avoiding a premature convergence to local optima. In our problem, each gene of an offspring is subjected to the mutation with the same probability of the mutation rate. To ensure that the result gene from a mutation is a valid one, the mutation can be performed on a pair (s_t, S_t) in either of the two ways as follows:

1. change the value of s_t to a random integer value on $\{0,1,\ldots,S_t-1\}$, or
2. change the value of S_t to a random integer value on $\{S_t+1, S_t+2,\ldots,\kappa\}$

Either of the two ways is chosen with no bias (i.e., equal probability = 0.5).

3.3.6 Replacement strategy

The GA uses a generational approach for replacement strategy. The number of created offspring for each round of the algorithm is equal to the number of the population and so all of the offspring replace all the current population and become the new population for the next round.

3.3.7 Terminating condition

The iteration process of the GA is stopped when the number of rounds performed reaches a maximum value. The best solution found so far during the process is kept for each trial. Several trials can be performed to increase the chance that the algorithm finds the global optima. The final solution is the best solution from all trials.

4. Numerical Experiments

Two sets of experiments are conducted. The planning horizon is $n=12$. In all experiments, cost parameters are stationary; i.e., $K_t = K$, $h_t = h$ and $b_t = b$ for all $t = 1,2,\ldots,n$, and the terminal cost is zero. Assume that demand in period follows a Poisson distribution with mean μ_t. The initial inventory is assumed to be zero; i.e., $i_0 = 0$.

The parameter setting of the GA is as follows: The number of population is 100, the tournament selection probability is 0.7, the crossover rate is 0.8, the mutation rate is 0.05, the number of rounds is 400, and the number of trials is 10.

Experiment 1: Generated problem instances

We systematically vary the setup cost, the holding cost, the penalty cost, and the mean demand vector according to a $2\times2\times2\times10$ factorial experiment. (The total number of problem instances is 80.) The first factor—the setup cost—has two levels $K \in \{650,1950\}$. The second factor—the holding cost—has two levels $h \in \{41,123\}$. The third factor—the penalty cost—has two levels $b \in \{205,615\}$. Finally, the fourth factor—the mean demand vector $(\mu_1, \mu_2, \ldots, \mu_{12})$—has ten levels shown in Table 2. The capacity is $\kappa = 75$.

Figure 1 shows the optimal expected cost and that from the GA. The optimal policy is constructed from the MDP mentioned in Section 3. From Figure 1, the gaps between the two costs are small. This suggests that the GA performs very

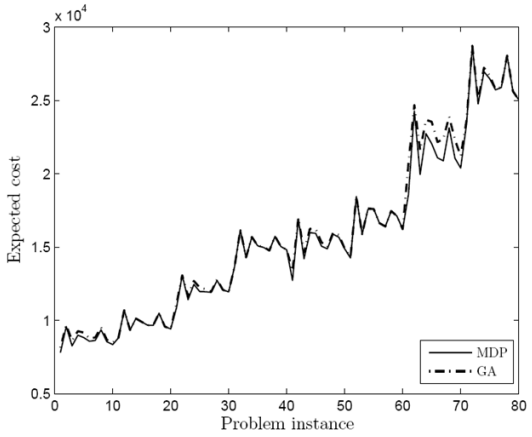

Figure 1. Total expected costs from MDP and GA

Table 2. Mean demands for Experiment 1

Level	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8	μ_9	μ_{10}	μ_{11}	μ_{12}
1	22.26	15.40	6.02	4.17	14.68	16.40	11.57	2.52	6.77	11.10	11.04	2.10
2	14.60	23.76	16.78	16.06	22.61	16.43	8.37	16.65	9.50	10.74	23.01	5.46
3	11.22	13.57	15.33	20.90	12.60	18.70	24.98	4.89	6.89	1.21	6.40	4.62
4	21.74	17.90	8.15	21.40	1.40	17.98	23.58	6.76	23.29	20.00	9.64	9.57
5	9.05	18.26	19.70	21.97	5.68	13.95	20.55	2.40	24.11	2.65	14.78	10.20
6	20.79	24.46	12.69	14.93	6.52	19.74	22.25	13.13	17.26	6.66	1.04	5.29
7	7.36	21.56	2.47	5.39	13.82	20.92	24.03	10.51	5.02	11.95	10.82	11.94
8	17.74	14.12	7.75	3.29	13.29	16.60	14.72	9.80	23.04	14.42	17.79	14.71
9	15.30	5.72	23.49	7.03	21.41	11.24	4.20	4.88	22.55	9.80	20.26	22.61
10	21.68	9.12	9.16	13.47	7.02	11.64	9.49	8.37	15.33	4.84	6.38	24.95

Table 3. Mean demands for Experiment 2

μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8	μ_9	μ_{10}	μ_{11}	μ_{12}
159.4	207.5	254.8	167.8	193.3	262.6	229.8	246.8	193.4	201.1	213.1	195.7

well. Over the 80 problems, the average, minimum, and maximum percent differences between the expected cost from GA and the optimal values are 1.48, 0.00, and 11.44, respectively.

Experiment 2: Real-world problem instance

The mean demands in Table 3 are based on a real-world problem. These are monthly average demands of a 15-meter flexible copper pipe with 3/8" diameter at a particular company. Time periods $1,2,\ldots,12$ correspond to January, February, …, December. The company supplies appliance parts to one of the largest modern trade/retail chains in Thailand. In this example, one unit equals 10 pipes; for instance, $\mu_4 = 167.8$ means that the average demand in April is 1678 pipes. Breakdowns of the appliances occur more often in summer (months 3 through 8 in Thailand) than in winter, so the mean demands in summer, except months 4 and 5, appear to be higher than the rest. In months 4 and 5, the number of working days are fewer than those in the other summer months due to long holidays (Thai New Year festival in April and some Buddhist holidays in May); so, we see average demands drop. The setup cost is $K = 1300$, the holding cost $h = 5$, the penalty cost $b = 25$, and the capacity $\kappa = 648$.

The expected cost under the simple procedure is 22068.95, whereas that under GA is 15445.20; the cost reduction, if GA is used instead of the simple procedure, is about 30 percent.

5. Concluding Remarks

Below, we summarize this article and discuss its extension. A periodic-review min-max inventory policy is considered. Demand in each period is independent but needs not be identically distributed. Our objective is to minimize the expected total inventory cost over a planning horizon of length *n*. We propose a nonstationary discrete-time Markov chain to evaluate the total expected cost for given pairs of reorder point and order-up-to level. The optimization problem of finding the control parameter, $((s_1,S_1),\ldots,(s_n,S_n))$, is solved via the GA. Our numerical results reveal that the GA performs very well.

We can extend this model to situations when a company wants to maintain some service levels (e.g., a stock-out probability or a fill rate). The GA can be modified to capture such constraints; for instance, a penalty function may be added to the fitness function if a service level constraint is violated.

References

Bollapragada, S., and Morton, T. E. 1999. A simple heuristic for computing nonstationary policies. Operations research, 47(4), 576–584.

Jans, R., and Degraeve, Z. 2007. Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. European journal of operational research, 177(3), 1855–1875.

Nahmias, S. 2009. Production and operations research. New York: McGraw-Hill, Inc.

Puterman, M. L. 2005. Markov decision processes: Discrete stochastic dynamic programming. Hoboken, NJ: John Wiley & Sons, Inc.

Reeves, C. 2003. Genetic algorithms. In: Glover, F., & Kochenberger, G. A. (eds), Handbook of metaheuristics. Norwell, Massachusetts: Kluwer Academic Publishers.

Silver, E. A., Pyke, D. F., and Peterson, R. 1998. Inventory management and production planning and scheduling. New York: John Wiley & Sons, Inc.

Simchi-Levi, D., Kaminsky, P., and Simchi-Levi, E. 2008. Designing and managing the supply chain: Concepts, strategies and case studies. New York, NY: McGraw-Hill, Inc.

Sobel, M. J., and Zhang, R. Q. 2001. Inventory policies for systems with stochastic and deterministic demand. Operations research, 49(1), 157–162.

Yang, J., Ding, H., Wang, W., and Dong, J. 2008. A new optimal policy for inventory control problem with non-stationary stochastic demand. IEEE International Conference on Service Operations and Logistics and Informatics, Beijing.

Zheng, Y.-S., and Federgruen, A. 1991. Finding optimal policies is about as simple as evaluating a single policy. Operations research, 39(4), 654–665.