FINITE PERTURBATION ANALYSIS METHODS FOR OPTIMIZATION OF PERIODIC (s,S) INVENTORY CONTROL SYSTEMS

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL

ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

by

Erdinç Mert

July, 2008

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Kağan Gökbayrak (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Osman Alp

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Ayşe Kocabıyıkoğlu

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

FINITE PERTURBATION ANALYSIS METHODS FOR OPTIMIZATION OF PERIODIC (s,S) INVENTORY CONTROL SYSTEMS

Erdinç Mert
M.S. in Industrial Engineering
Supervisor: Asst. Prof. Dr. Kağan Gökbayrak
July, 2008

We are dealing with single item inventory systems where the period time is constant and the unsatisfied demands are backordered. The demands are independent and identically distributed random variables, but the distribution of those variables are not known. The total cost of a period consists of; ordering cost "*K*" which is independent of the ordering quantity, holding cost "*h*" for each item that remains in stock, and penalty cost "*p*" for the each backordered item. In the considered system, it is known that when the parameters of an *(s,S)* inventory policy are chosen appropriate, then the expected period cost can be minimized. There are some exact methods or heuristics for finding the optimal *s* and *S* parameters in the literature for the case where the demand distribution is known. In our study, we introduce a perturbation analysis based method for finding the optimal *s* and *S* parameters where the demand distribution is not known. Our method anticipates the sensitivity of *(s,S)* parameters to the period cost for the observed demand quantities. This method's performance is compared with a method that uses Integer Programming with the past data and with a method that calculates the mean and standard variation values with the past data and feeds them to the Ehrhardt's Heuristic.

Keywords: Inventory Policies, Perturbation Analysis, Simulation

ÖZET

DAĞILIMI BİLİNMEYEN RASSAL AYRIK TALEPLER İÇİN PERİYODİK (s,S) ENVANTER DENETİM METODU

Erdinç Mert
Endüstri Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. Kağan Gökbayrak
Temmuz, 2008

Karşılanamayan taleplerin bekletilebildiği ve tedarik zamanlarının sabit olduğu tek öğeli envanter sistemlerini ele almaktayız. Gözlem periyotları içinde gelen talep miktarlarının bağımsız özdeşçe dağılmış ayrık rastgele değişkenler oldukları ancak bu değişkenlerin dağılımının bilinmediği kabul edilmektedir. Her periyot için maliyet, sipariş miktarından bağımsız *K* sipariş maliyeti, elde kalan her birim için *h* tutma maliyeti ve karşılanmak için bekletilen her birim talep için *p* yoksatma maliyeti toplamından oluşmaktadır. Ele alınan sistemde parametreleri doğru seçilmiş *(s,S)* envanter denetim politikasının beklenen periyot maliyetlerini en aza indirgeyebileceği bilinmektedir. Literatürde talep dağılımın bilinmesi durumunda en iyi *s* ve *S* parametrelerinin bulunabilmesi için önerilmiş tam veya sezgisel yöntemler bulunmaktadır. Bu çalışmamızda ise dağılımın bilinmediği durumda, yöntemimizin yakınsamasına yeterli süreler için dağılımın durağan kalacağı kabulu ile, en iyi *s* ve *S* değerlerinin bulunması için sarsım analizi tabanlı dağılım değişiklerine uyarlanabilir bir yöntem önermekteyiz. Gerçekleşen talep miktarları için periyot maliyetlerinin *s* ve *S* parametrelerine duyarlılıklarını tahmin eden ve bu parametreleri yenileyen yöntemimizin performansı geriye dönük tamsayı programlaması sonuçlarını uygulayan ve gerçekleşen miktarlar için hesaplanan ortalama ve standard sapma değerlerini Ehrhardt sezgiselline besleyen iki yöntemin performansları ile karşılaştırılmaktadır.

Anahtar Sözcükler: Envanter Politikaları, Sarsım Analizi, Simulasyon

To my family...

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Asst. Prof. Dr. Kağan Gökbayrak for all his attention and supports during my graduate study and for his valuable guidance and most importantly for his patience and trust.

I am indebted to members of my dissertation committee: Asst. Prof. Dr. Osman Alp and Asst. Prof. Dr. Ayşe Kocabıyıkoğlu for showing of kindness to accept to read and review this thesis. I am grateful to them for their effort, sparing their valuable time for me and for their support.

I am most thankful to members of Land and Missile Programs, especially to Mr. Aybars Küçük and Mr. Oğuz Yemişçiler, for their support and their patience. I would like also thank to my friends in Bilkent IE for their friendship.

Finally, I would like to express my deepest gratitude on my family for believing in my work and sacrifices that they have made for me. I feel very lucky to have such a wonderful sister, father and mother. Without their love and support, I would never have finished this thesis.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Inventories are assets held for sale in the ordinary course of business or in the process of production for such sale or supplies to be consumed in the production process or in the rendering of services. Some inventories have to be held due to the lead time of the goods and the ordering cost. Holding inventories in stock causes a cost, holding cost, because of the cost of storage, the opportunity cost, etc. On the other hand, if no inventories are held, the customers cannot find what they want and since the customer is not satisfied, there is a potential for the loss of future orders. Thus, having no inventories will also cause a cost; namely penalty cost. The third type of cost is named as ordering cost and it is the money given for the transportation of the goods, all possible administration tools to place an order, etc.

Inventory policies are methods for determining when and how many units to order in order to minimize an inventory cost function. There are two types of inventory policies; continuous review policies and periodic review policies. In the first type, the inventory position is monitored continuously and when the inventory position falls below a pre-specified level, an order is given. In the periodic review policies, the inventory position is monitored in periods and the same action is taken. One of the periodic review policy is the *(R,s,S)* policy where the inventory position is checked in every period of R time units and if the inventory position is less than or equal to s, an order is placed to raise the position up to S. The inventory policy discussed in this thesis will be the *(R,s,S)* policy for given R time units, and the aim is to find the optimal *(s,S)* values to minimize the expected summation of holding cost, penalty cost and the ordering cost.

The aim is to design inventory control policies for random demand distributions that can adapt themselves to sudden demand changes. For that purpose, Perturbation Analysis Algorithm will be defined and will be compared with two other algorithms; Ehrhardt's Heuristic and Retrospective Integer Programming Method. Perturbation Analysis Algorithm is a perturbation analysis based method for finding the optimal s and S parameters. This method will anticipate the sensitivity of s, S parameters to the period cost for the observed demand quantities. This algorithm can find the real optimal s, S parameters without knowing the distribution, mean and the variance of the demand distribution.

Ehrhardt's Heuristic is a simple heuristic that calculates near-optimal (s, S) values by calculating the mean and the standard deviation by the observed data. Retrospective Integer Programming Method also uses the past data and finds the minimum cost by integer programming.

In the second chapter, some background information about the Inventory Systems and Inventory Policies are given. The problem formulation is defined and a literature review about how to determine optimal (s, S) parameters is mentioned.

In Chapter 3, the main method, Perturbation Analysis, is introduced. At the end of this chapter, methods to find the optimal (s, S) pair are given.

In Chapter 4, the logical model is defined, and the simulation models are introduced. Chapter 5 consists of numerical examples. In this chapter, several distributions are taken as the demand distributions and the algorithms defined in Chapter 3 are tested. Also comparison of the algorithms (Perturbation Analysis Algorithm, Ehrhardt's Heuristic and Ret-

rospective Integer Programming Method) can be found in this chapter. This chapter ends with the adaptability issue.

The last chapter includes the conclusion and general discussion about the outputs of numerical examples. At the end of this chapter, future directions are given and the thesis ends with the appendices.

CHAPTER 2

BACKGROUND

2.1 Inventory Systems

Inventories are assets held for sale in the ordinary course of business or in the process of production for such sale or supplies to be consumed in the production process or in the rendering of services. Fogarty & Hoffman (1983) defines inventory as:

> Inventory ties up capital, uses storage space, requires handling, deteriorates, sometimes becomes obsolete, incurs taxes, requires insurance, can be stolen, and sometimes is lost ... the absence of the appropriate inventory will halt a production process; ... an expensive piece of earth moving equipment may be idled by lack of a small replacement part; a patient may die due to the unavailability of plasma; ... and, in many cases, a good customer may become irate and take his business elsewhere if the desired product is not immediately available.

Thus, inventories serve the function of providing a buffer between customer demands and production capabilities (Aft 1987). There are three basic classes of inventories for a company;

- *Raw materials* are the items that are purchased from another vendor to be processed further.
- *Work in process inventories* are still not finished goods but processed raw materials.
- *Finished goods* are the products that are available for sale.

In this thesis, inventory will only represent the "finished goods" which are considered as discrete units.

An inventory system is a system in which the following three kinds of costs are significant; the cost of carrying inventories, the cost of incurring shortages, and the cost of replenishing inventories (Eliezer 1982). After this part, these costs will be recalled as holding cost, penalty cost, and ordering cost, respectively. Holding cost applies if there is a positive inventory on hand, penalty cost applies if there is unsatisfied order, and ordering cost applies if order was made at that time.

2.2 Inventory Policies

Inventory policies are methods for determining when and how many units to order in order to minimize an inventory cost function. Based on the inventory level, anticipated demand and cost function, these policies answer the following questions:

1. When should the inventory be replenished?

2. How many units should be added to the inventory? (Eliezer 1982)

For the first question, following answers can be given;

- Inventory should be replenished when the amount in inventory is equal to or below s quantity units

- Inventory should be replenished every R time units

For the second question, the answers can be;

- The quantity to be ordered is q quantity units

- A quantity should be ordered so that the amount in inventory is brought to a level of S quantity units

If the demand is deterministic, then some methods can be used to find the answers of these questions. However, if the demand is stochastic, then some inventory policies should be used. The quantities s, R, q, S used above are the policy parameters and defined as the reorder point, scheduling period, lot size, and order-up-to level, respectively.

Based on the reviewing process, inventory policies are divided into two main groups; continuous review policies and periodic review policies.

2.2.1 Continuous Review Policies

In continuous review policies, the inventory control system is designed so that the inventory position (inventory on hand plus orders given but not received yet) is monitored continuously. When the inventory level is sufficiently low, an order is given at that time and the given order is received after a certain lead time. Well known continuous review policies are (s, S) Policy and (s, Q) Policy.

(s,S) Policy

In (s, S) Ordering Policy, whenever the inventory position falls to the *reorder point* s, an order is placed enough to raise the position to the *order-up-to level* S.

(s,Q) Policy

An (s, Q) Ordering Policy means that when the level of inventory position is less than or equal to s, an order of fixed amount of Q is placed.

2.2.2 Periodic Review Policies

Another alternative of Continuous Review is to consider the inventory position only at certain given points in time. The intervals between these reviews are constant and that is called the Periodic Review. Well known periodic review policies are (R, s, S) Policy, (R, s, Q) Policy and (R, S) Policy.

(R,s,S) Policy

In intervals of R time units, the inventory position is checked and if the inventory position is less than or equal to s, an order is placed to raise the position up to S.

(R,s,Q) Policy

In (R, s, Q) policy, the inventory position is checked in every period of R time units and if the inventory position is less than or equal to s, an order of amount Q is placed.

As can be seen easily, in the first policy, the inventory position is always increased to same value after orders, where in the second policy, always the same quantity of orders are given.

(R,S) Policy (Base Stock Policy)

In this policy, an order up to S at the end of each period of R time units is placed, unless the period demand is zero.

In this thesis, **Periodic** (R, s, S) **Policy** will be analyzed for a fixed and given time period, R. After this part, "(s, S) Policies" will be used for "(R, s, S) Policies" and the aim is to find optimal s and S values that minimize a total cost. For that purpose, we consider a single-item inventory system where unfilled demand is backlogged with a fixed lead time L between placement and delivery of an order where the holding cost and penalty cost are linear, and the demand is random.

2.3 Problem Formulation

We have a stochastic inventory control problem with an unknown demand distribution. The demand distribution shall not have any known distribution; the algorithms in this thesis work for all kinds of distributions and also for empirical distributions. There is a non-negative constant lead time and three types of costs (holding cost, penalty cost, and ordering cost). Detailed information of the costs are given below.

Holding Cost: Holding cost applies while holding inventory on stock. Holding inventory causes a cost because there is an opportunity cost (money invested in inventory could be placed on bank for interest), storage cost (need for place to store the inventory), deterioration cost (materials deteriorate over time), obsolescence problem (products may become outdated), insurance cost (insurance protects against potential loss), etc.

Thus, holding cost is applied if the inventory on hand is positive at the end of the day. The holding cost per unit "h" is multiplied with the "Inventory on Hand", and the total holding cost is calculated.

Penalty Cost: It is also known as stockout cost or shortage cost. There are two types of penalty cost; in the first type, the customer is willing to backorder and to wait for the delivery, where in the second, customer does not wait and the order is lost. In this thesis, we assume full backlogging. However, it also causes a cost since you cannot satisfy your customer and there is a potential for the loss of future orders (Aft 1987).

Thus, if the inventory on hand at the end of the day is negative, penalty cost is applied. The penalty cost per unit "p" is multiplied by the total "Unsatisfied Demand" at the end of that period and the total penalty cost is calculated. Since backlogging policy is applied, the unsatisfied demand is forwarded to the following day until all of the past demands are satisfied.

Ordering Cost: There are usually fixed costs associated with a replenishment (independent of the batch size). It may be described as a truck with infinite capacity delivering the orders and even if one order is placed, the same money will be given to the truck. Thus, if order is given in a day, Ordering Cost (Set-Up Cost) "K" is also incurred. In other words, this cost is independent of the ordered unit size, i.e. ordering 1 unit and 100 units will have the same ordering cost.

Then, the total cost is defined as:

$$J = I\{Y \leq s\}K + h\max(X, 0) + p\max(-X, 0)$$

where Y is the inventory position at the end of the period, X is the inventory level at the end of the period, and I is the indicator function.

2.3.1 Mathematical Model

The model of the problem can be formulated as:

$$\min_{s,S} E[(J = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} J_i)]$$

$$J_i = I\{Y_i \leq s\}K + h\max(X_i, 0) + p\max(-X_i, 0)$$

$$X_i = X_{i-1} + R_i - D_i$$

$$Y_i = X_{i-1} + O_{i-1} - D_i$$

$$O_i = O_{i-1} + Q_i - R_i$$

$$Q_i = I\{Y_i \leq s\}(S - Y_i)$$

$$R_i = Q_{i-L-1}$$

$$X_0 = S$$

$$O_0 = 0$$

$$Q_0 = 0$$

where the problem parameters are as below:

D_i : Demand Occurred at Period i

L : Constant Lead Time

h : Holding Cost per unit

p : Penalty Cost per unsatisfied units

K : Ordering cost for placing an order, independent of number of units

s : Inventory control policy parameter, Reorder Point

S : Inventory control policy parameter, Order Up-To Level

and the state variables/decision variables of the problem are as below:

J_i : Total Cost at the end of Period i

Y_i : Inventory Position at the beginning of Period i

X_i : Inventory Level at the beginning of Period i

R_i : Received Orders at the beginning of Period i

Q_i : Order Quantity at the beginning of Period i

O_i : Outstanding Orders at the beginning of Period i

By Inventory Position, we mean inventory on hand plus on order minus backorders. By Inventory Level, we mean inventory on hand minus backorders. Lead Time is the time between the placement of an order and the arrival of that order.

For linearizing the problem, the model was rewritten as below:

$$
\begin{aligned}
\min Z &= \sum_i J(i) \\
J(i) &\geq KI_1(i) + hX(i) \\
J(i) &\geq KI1(i) - pX(i) \\
X(1) &= S - D(1) \\
X(i) &= X(i-1) + R(i) - D(i), \quad \text{for } i > 1 \\
Y(1) &= X(1) \\
Y(i) &= X(i-1) + O(i-1) - D(i), \quad \text{for } i > 1 \\
O(1) &= Q(1) \\
O(i) &= O(i-1) + Q(i) - R(i), \quad \text{for } i > 1 \\
Q(i) &\geq S - Y(i) - M(1 - I_1(i)) \\
Q(i) &\leq S - Y(i) + M(1 - I_1(i)) \\
Q(i) &\leq MI_1(i) \\
R(1) &= 0 \\
R(i) &= Q(i - L - 1), \quad \text{for } i > 1 \\
s &\leq S \\
MI_1(i) &\geq s - Y(i) + 1 \\
MI_2(i) &\geq Y(i) - s \\
I_1(i) &\leq 1 - I_2(i)
\end{aligned}
$$

where

M is a very big number,

I_1 and I_2 are binary variables,

s and S are integers,

J, R, O and Q are positive variables.

2.4 Determining Optimal (s,S) Policy Parameters

In this section, the literature about determining the optimal (s, S) inventory policy parameters will be reviewed. The analysis of (s, S) inventory policies started out as a research topic after the introduction of the multistage periodic review inventory model in the paper by Arrow, Harris & Marschak (1951). In that paper, they studied uncertainty models–a static and a dynamic one–in which the demand flow is a random variable with a known probability distribution. The best maximum stock and the best reordering point are determined as functions of the demand distribution, the cost of making an order, and the penalty of stock depletion.

Scarf (1960) and Iglehart (1963) showed that an optimal policy can be found within the class of (s, S) policies where there are linear costs, stochastic demand, and a fixed lead time. Thus, since our assumptions satisfy these requirements, there will always be an optimal (s, S) policy that minimizes the average cost. However, computing the optimal policy that minimizes the long term average cost has become a big problem for years since there are a potentially infinite number of (s, S) combinations. There are mainly two categories for finding the optimal (s, S) pair; exact method and heuristics. Iglehart (1963),

Veinott&Wagner (1965), Bell (1970), Archibald&Silver (1978), Federgruen&Zipkin (1984), Zheng&Federgruen (1991), Feng&Xiao (2000) and Daniel&Rajendran (2005) are all examples about the first category, exact method. On the other side, Naddor (1975), Ehrhardt (1979, 1984) and Roundy&Muckstadt (2000) are the most known papers that introduce heuristics for (s, S) optimization.

Exact Methods (For known demand distribution):

Iglehart (1963) gives bounds for the sequences $\{s_n\}$ and $\{S_n\}$ and discusses their limiting behavior. The limiting (s, S) policy characterizes the optimal ordering policy for the infinite horizon problem. In Veinott&Wagner (1965), a complete computational approach for finding optimal (s, S) inventory policies is developed. The model is named as a dynamic inventory model in which demands for a single product are independent, identically distributed (i.i.d) discrete random variables. There is a constant lead time, a discount factor $0 \leq \alpha \leq 1$, a fixed non-negative set up cost, a linear purchase cost, a convex expected holding and penalty cost function, and total backlogging of unfilled demand. The objective of the paper is to choose the control parameters (s, S) which minimizes the long run average cost per period. The methods of Veinott&Wagner (1965), Bell (1970), and Archibald&Silver (1978) apply complete enumeration method where it is bounded by lower bounds $\underline{s}$, $\underline{S}$ and upper bounds $\overline{s}$, $\overline{S}$ for finding the optimal values of s^* and S^*.

Sahin (1982) analyzes the behavior of (s, S) inventory models. Both for periodic and continuous review systems, Sahin states that for constant lead times and full backlogging, expected cost $E(s, \Delta)$ is convex in s where $\Delta = S - s$. From that point, he proves that both

for periodic and continuous review systems with full backlogging, constant lead times, and linear holding and shortage costs, $E(s_1(\Delta), \Delta)$ is pseudoconvex on $\Delta \geq 0$. After that, in Federgruen&Zipkin (1984), an iterative algorithm to compute an optimal (s, S) policy under standard assumptions (stationary data, well-behaved one-period costs, discrete demand, full backlogging) is presented. The algorithm starts with a given (s, S) policy and evaluates a sequence of policies, and converges to an optimal one in a finite number of iterations.

In Zheng&Federgruen (1991), a new algorithm for computing optimal (s, S) policies is derived based upon a number of new properties of the infinite horizon cost function $c(s, S)$ as well as a new upper bound for optimal order-up-to levels S^* and a new lower bound for optimal reorder levels s^*. In other words, Zheng&Federgruen (1991) propose a simple and efficient algorithm that searches directly on (s, S) plane. While improving the average cost $c(s, S)$, the search moves vertically up and horizontally right to upgrade the s and S values at each step until reaching the optimal s^*, S^* pair. The algorithm's computational complexity is 2.4 times that required to evaluate a single (s, S) policy and this algorithm can be applied both to periodic review and continuous review inventory systems. Zheng&Federgruen defines:

$D =$ the one-period demand (random variable);

$p_j = \Pr\{D = j\}, j = 0, 1, 2, ...;$

$K =$ the fixed cost to place an order;

$G(y) =$ the one-period expected costs where,

$$G(y) = \begin{cases} C_h * \sum_{d=0}^{y}(y-d) * \Pr(D=d) + C_p * \sum_{d=y+1}^{\infty}(d-y) * \Pr(D=d) & \text{for } y \geq 1 \\ C_p * \sum_{d=0}^{\infty}(d-y) * \Pr(D=d) & \text{for } y < 1 \end{cases}$$

After defining this G function, the long-run average cost $c(s, S)$ is calculated as:

$$c(s,S) = M(S-s)K + \sum_{j=0}^{S-s-1} m(j)G(S-j)$$

where

$$\begin{aligned} m(0) &= (1-p_0)^{-1} \\ M(0) &= 0 \\ m(j) &= \sum_{l=0}^{j} p_l m(j-l), \quad j = 1, 2, ... \\ M(j) &= M(j-1) + m(j-1), \quad j = 1, 2, ... \end{aligned}$$

Feng&Xiao (2000) uses the method that Zheng&Federgruen (1991) introduced, and proposes a new approach to solve the optimal (s, S) inventory policy. A dummy cost factor and an auxiliary function were introduced in this paper. The algorithm searches for the optimal dummy cost through continuously evaluating the auxiliary function.

Daniel&Rajendran (2005) studies the performance of a single-product serial supply chain operating with a base-stock policy and optimizes the inventory levels in the supply to minimize the total supply chain cost with holding and shortage costs. A genetic algorithm is proposed to optimize the base-stock levels with the objective of minimizing the sum of holding and shortage costs in the entire supply chain. Simulation is used to evaluate the base-stock levels generated by the genetic algorithm. The effectiveness of this algorithm is compared with a random search procedure. Optimal base-stock levels are obtained through

complete enumeration of the solution space and compared with those yielded by this algorithm. It is found that the solutions generated by the proposed algorithm do not significantly differ from the optimal solution obtained through complete enumeration for different supply chain settings, thereby showing the effectiveness of the proposed algorithm.

Heuristics (For known demand distribution):

As mentioned before, there are also some papers that introduce heuristics to find the optimal pair of (s, S). Naddor (1975) derives a heuristic decision rule that computes optimal s^* and S^* values. The rule requires the knowledge of the mean and standard deviation of demand, probability of no demand, carrying and replenishing costs, desired availability, and leadtime. Ehrhardt (1979) presents an analytic approximation for computing (s, S) policies for single items under periodic review with a set up cost, linear holding and shortage costs, fixed lead time, and backlogging of unfilled demand. He calls the approximation as the "Power Approximation". By that approximation, it is very easy to compute the optimal s and S values by only knowing the mean and the variance of the demand.

The Power Approximation is defined as follows:

Let $\mu_L = (L+1)\mu$ and $\sigma_L = \sigma\sqrt{L+1}$.

Then,

$$\begin{aligned} q_p &= 1.463\mu^{0.364}(k/h)^{0.498}\sigma_L^{0.138}, \\ z &= \{q_p/[(1+p/h)\sigma_L]\}^{0.5}, \\ s_p &= \mu_L + \sigma_L^{0.832}(\sigma^2/\mu)^{0.187}(0.220/z + 1.142 - 2.866z) \end{aligned}$$

And if q_p/μ is greater than 1.5, then $s^* = s_p$ and $S^* = s_p + q_p$. If demands are to be integer values, s_p and q_p values are rounded to the nearest integer.

Ehrhardt (1984) introduces the Revised Power Approximation. In Ehrhardt (1979), it is likely that the accuracy of the Power Approximation will suffer when the variance of demand is very small. Thus, a revised version of the Power Approximation is published. The Revised Power Approximation is defined as follows:

Let $\mu_L = (L+1)\mu$ and $\sigma_L = \sigma\sqrt{L+1}$.

Then,

$$\begin{aligned} q_p &= 1.3\mu^{0.494}(k/h)^{0.506}(1+\sigma_L^2/\mu^2)^{0.116}, \\ z &= \left[q_p/(\sigma_L * p/h)\right]^{1/2}, \\ s_p &= 0.973\mu_L + \sigma_L(0.183/z + 1.063 - 2.192z) \end{aligned}$$

Roundy&Muckstadt (2000) study the problem of determining production quantities in each period of an infinite horizon for a single item produced in a capacity-limited facility. The demand for the product is random, and it is independent and identically distributed from period to period. Unfilled demand is backordered. A base stock or order-up-to policy is used. They develop a new approximation for this distribution, and perform extensive computational tests of existing approximations. Their new approximation works extremely well as long as the coefficient of variation of the demand is less than two.

Retrospective Approaches (If demand distribution is not known):

All of the methods until this part, we had the assumption of knowing the demand distribution or at least the mean and the variance. After this part, some retrospective approaches will be introduced. In Fu&Healy (1992), a periodic review (s, S) inventory system is considered. The problem in this paper considers the inventory control of an item which is measured in continuous units. The infinite horizon problem is considered and simulation is used to find the optimal s and S values that minimize the average cost per period. The general assumptions of this paper are; general independent and identically distributed (i.i.d.) continuous demands, zero lead time, full backlogging of orders, and linear ordering, holding and shortage costs.

Two approaches are introduced in this paper: a deterministic "retrospective" algorithm and a gradient-based, steepest-descent algorithm. The more important approach is the second one which involves estimating the gradient of the performance measure of interest and adjusting the parameters according to the gradient during the evolution of the simulation. The simulation terminates when the gradient is "close enough" to zero. As L'Ecuyer (1991) states, there are two main techniques for gradient estimation; perturbation analysis and likelihood ratio. Perturbation analysis is a technique for gradient estimation from a single simulation of a discrete-event system.

The perturbation analysis estimators for continuous demand is derived in Fu (1994) and PA algorithms for $\partial J/\partial s$ and $\partial J/\partial q$ (for continuous demand and zero lead time) are given where J denotes the average cost per period.

In Fu (1994), sample path derivatives of performance measures were derived for (s, S) inventory systems. The objective is again to find optimal s and S values that minimize the average total cost. In this paper, nondiscounted periodic review system with i.i.d continuous demands, full backlogging, constant lead time and general holding, shortage costs are considered. The approach is sample path analysis with the focus on sample path derivatives. Since it deals with derivative estimation, the paper assumes that demands take continuous values. Also, in this paper Fu focuses on infinitesmall perturbation analysis (IPA) and smoothed perturbation analysis (SPA) for finding the estimators of s and S.

Fu&Healy (1997) utilizes the perturbation analysis estimators derived in Fu (1994) for zero lead time. In this paper, two approaches (gradient-based methods and retrospective approach) for the optimization of a periodic review (s, S) inventory systems are compared, and a Perturbation Analysis algorithm that combines these two approaches is proposed.

Glasserman&Tayur (1995) develop simulation-based methods for estimating sensitivities of inventory costs with respect to policy parameters. They consider capacitated, multiechelon systems operating under base-stock policies and develop estimators of derivatives with respect to base-stock levels. They formulate and validate infinitesimal perturbation analysis derivative estimates for multiechelon capacitated production-inventory systems. It means that they introduce appropriate algorithms and show that they converge to the correct values. The convergence is shown over the number of independent simulation runs for derivatives of finite-horizon costs and over an infinite horizon for derivatives of steady-state costs. The model can be defined as a periodic-review, continuous-demand, fixed lead time multiechelon system with limited production capacity at each stage. There

are linear holding and shortage costs, all unsatisfied orders are backlogged, where there is no fixed ordering cost. When compared with Fu (1994), Fu's estimator is more complex than the Glasserman&Tayur's estimator.

Zhao&Melamed (2004) considers single-stage, single-product Make-to-Stock systems with random demand and random service (production) rate, where demand shortages at the inventory facility are backordered. IPA (Infinitesimal Perturbation Analysis) gradients of various performance metrics are derived with respect to parameters of interest and are showed to be easy to compute.

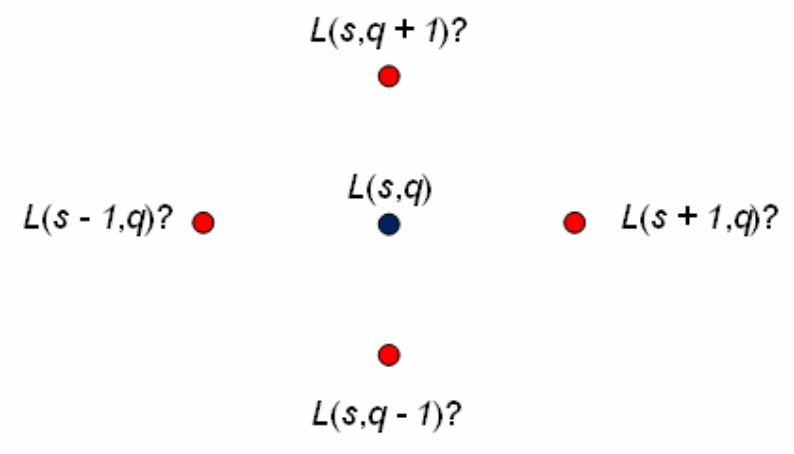
In all of the papers defined in this section, the main purpose is to find the optimal (s, S) pair. For that purpose, this thesis will present a method named Perturbation Analysis and this method will find the optimal inventory control parameters by calculating the delta costs of neighbor points by only one simulation.

CHAPTER 3

PERTURBATION ANALYSIS METHODS FOR OPTIMIZATION OF POLICY PARAMETERS

3.1 Perturbation Analysis Algorithms

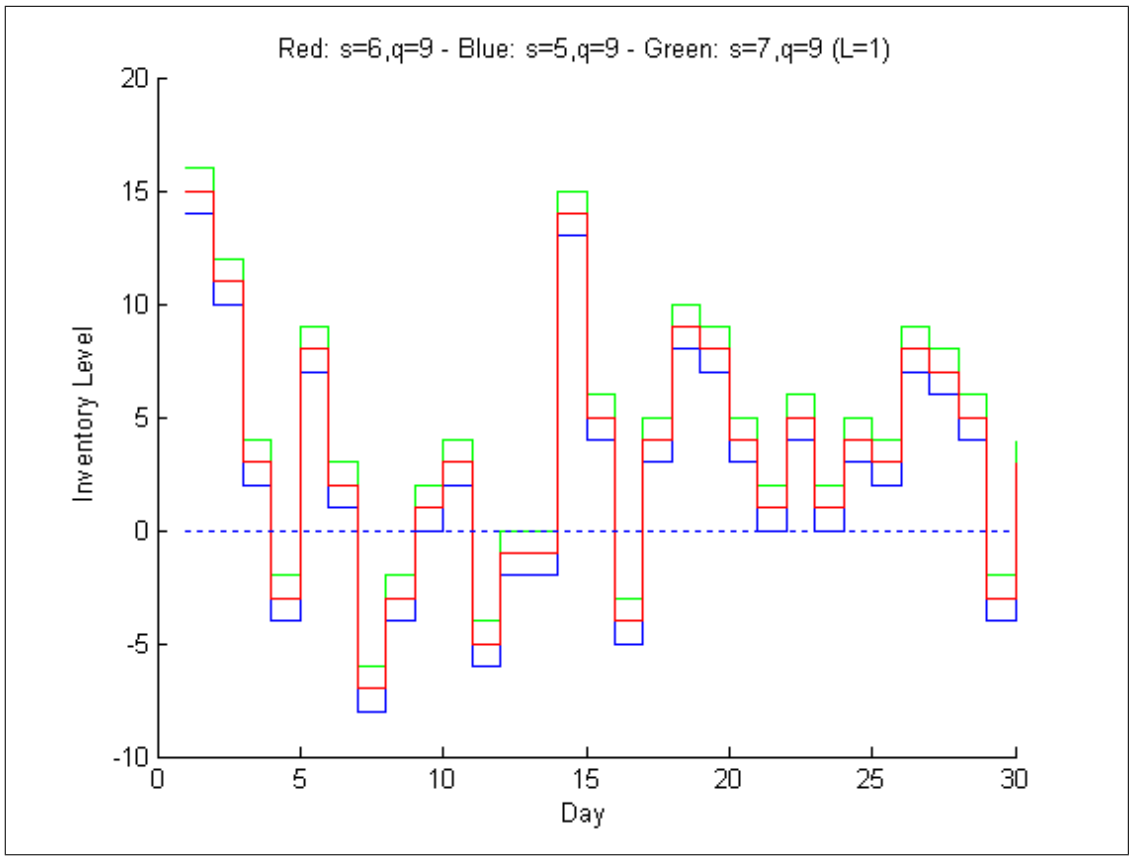
Perturbation Analysis is a technique for gradient estimation from a single simulation. For Perturbation Analysis algorithms, $q=S-s$ will be defined for simplifications.



As can be seen from the graph above, the aim is to find the costs of four neighbors of point *(s,q)* by only making the simulation for *(s,q)*. Firstly the Perturbation Analysis Algorithm is applied to the parameter *s* to calculate the delta costs of *(s-1,q)* and *(s+1,q)*. After that, the delta costs of *(s,q+1)* and *(s,q-1)* are calculated by Perturbation Analysis on q parameter. Detailed information about these algorithms can be found below.

3.1.1 Perturbation Analysis on "s" (Algorithm 1)

In this algorithm, the perturbation is done on "s". It means that we leave the $S - s$ value (q value) constant and perturbate the system on s to determine the descent direction for the given constant q.



Perturbation on s one by one (Constant q)

Figure that is above shows the effect on sample path where q is fixed and s is perturbed. Red lines denote the On-Hand Inventory when $s = 6$, $q = 9$. Green lines denote the On-Hand Inventory when $s = 7$, $q = 9$, and blue lines denote the On-Hand Inventory when $s = 5$, $q = 9$. It can be easily seen that a perturbation in s shifts the entire sample path by the same perturbation. When s is perturbed by 1 unit, all of the sample path

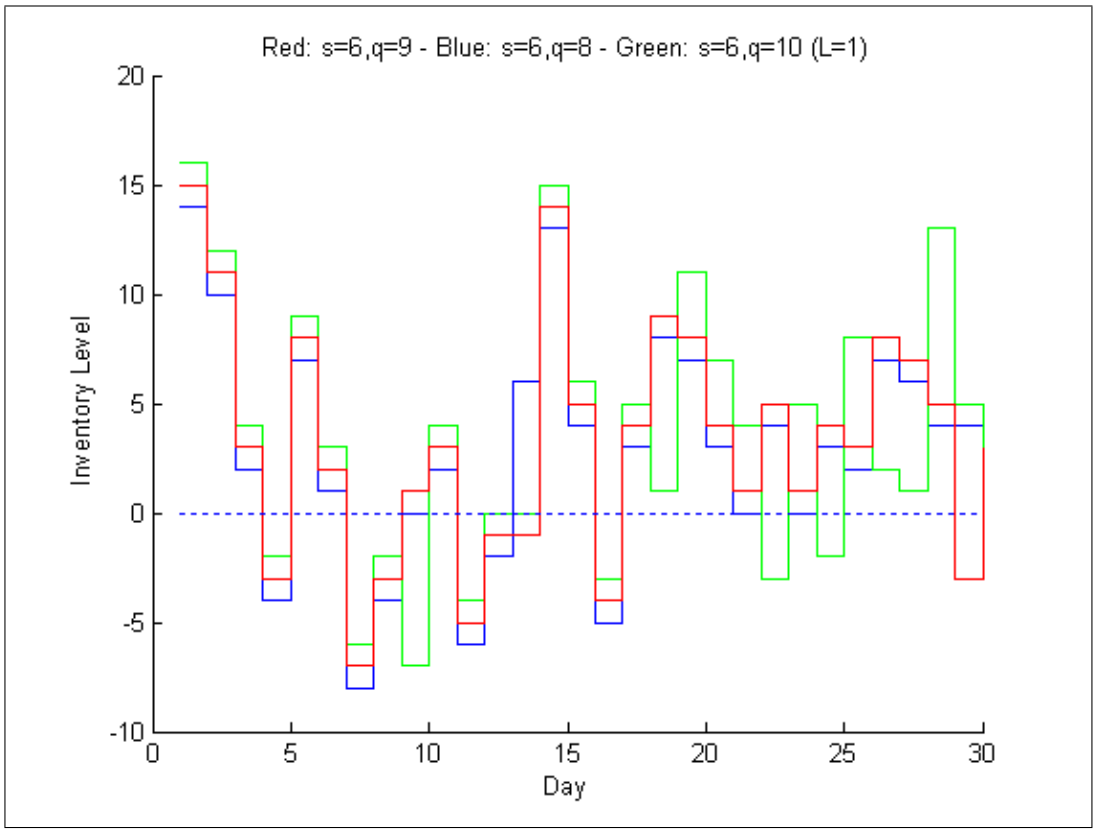
is perturbed 1 unit. It means that, when we change s value and keep q value constant, the ordering policy does not change, i.e., we still order the same amount at the same periods. The effect of increasing s is to increase the on hand inventory or to decrease the shortage amount resulting with the change in cost determined by the following algorithm.

Algorithm 1:

The flowchart of Algorithm 1 can be found in Appendix A.

3.1.2 Perturbation Analysis on "q" (Algorithm 2)

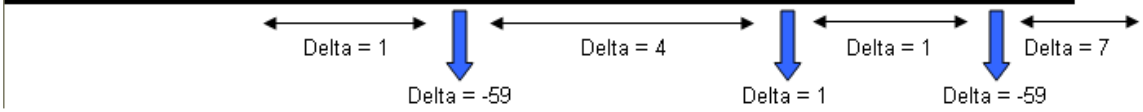
In this section s value is kept constant and q value is perturbed.



As seen from the figure above, the perturbation in q may result with a major change on the sample path. Thus, the ordering policies change, i.e. in perturbed path an order may be placed where there was no order in the nominal path.

In the table below, the policy $(s = 1, S = 61)$ is perturbed to become $(s = 1, S = 62)$ for the case of zero Lead Time.

DAY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Demand	-	8	6	3	5	20	21	9	7	3	16	33	8	6	4
Inventory Level for (1,60)	15	7	1	58	53	33	12	3	-4	58	42	9	1	55	51
Inventory Position	15	7	61	58	53	33	12	3	61	58	42	9	61	55	51
Inventory Level for (1,61)	16	8	2	-1	57	37	16	7	0	59	43	10	2	-4	58



In the table above, the first row is the demand occured that period. The second row is the Ending Inventory if (s, S) policy is used (Nominal Path). The third row shows the Inventory Position for (s, S) policy. And the last row is the Ending Inventory if $(s, S + 1)$ policy is used (Perturbed Path). To be short, POL1 will be used for (s, S) policy and POL2 will be used for $(s, S + 1)$ policy afterwards. From the table above, we can also see the following points:

Firstly, it is seen that we have to deal with "Delta" values to construct the perturbed sample path. "Delta" value can be defined as the value in the fourth row minus the value in the second row (how many more units to hold in the perturbed path compared to the nominal path). At the beginning, the perturbed path can be easily calculated as the previous section (only one unit difference between the two paths). However, after some point, the

perturbed path starts to become different from the nominal path. When the path is analyzed carefully, it can be seen that the system starts to get out of order ("delta" values change) after the numbers with red circles in the third row. 61 is the S value of the nominal path. Thus, we can see numerically that the perturbed path gets out of order when Inventory Position reaches the S value. It means that when we reach the S value in POL1, we place an order. However, the S value of POL2 is 62, i.e. we do not place an order in POL2 in the cases where Ending Inventory falls exactly to s in POL1. So, there is an *ordering change*.

From now on, we focus on the "Delta" values. If POL1 orders and POL2 does not order, we have a "Delta"=-59 for one period and after that period, we have a completely different "Delta" value until we reach another red circle. When we reach the red circle, if both policies order, everything becomes fine, "Delta" again decreases to 1 and continues as 1 until reaching another red circle.

After calculating the perturbed path, things will become similar to Section 3.1.1. We consider the cost differences if we increase or decrease q by 1, and try to solve this optimization problem until we reach the optimal q value for the specific constant s value.

Algorithm 2:

The flowchart of Algorithm 2 can be found in Appendix B.

3.2 Finding Optimal (s*,q*) Pair

For finding the optimal parameters by the perturbation analysis method, the cost graph should be convex. Thus, the cost graph must be convex in s when q is fixed, and vice

versa. Sahin (1982) showed that the cost graph is convex in s for fixed q values. Thus, the perturbation analysis method can find the optimal s value by Algorithm 1. However, there are no proofs for Algorithm 2's optimality. When s is fixed, there may be some local optimal values such that the perturbation algorithm method can stuck on these values. Since we have different demand sample paths in each period, the local optimal values, if any, change from period to period. Thus, if there are some local optimal values, we can get rid of them while the simulation runs long enough. The convergence to the global optimal value can be satisfied since the global optimal value remains the same in each period.

This chapter will be a combination of Section 3.1.1 and Section 3.1.2. In those sections, how to update one of the control parameters are discussed. In addition to this, by using the methods in the mentioned sections, how to find the optimal pair will be discussed in this section. s and q values will be updated one by one until the optimal pair is reached. Firstly, we start with an arbitrary (s, q) pair. After that, using Algorithm 1, s value will be updated to $s+1$, s or $s-1$. Then, we have the updated $s+1$ value and initial q value. Now, we will perturbate on q value, and update it to $q+1$, q or $q-1$. The process continues on until q and s values do not oscillate.

Algorithm 3:

- **STEP1:** Start with initial s and q values
- **STEP2:** Use Algorithm 1 and update s value ($s+1$, s or $s-1$)
- **STEP3:** Use Algorithm 2 and update q value ($q+1$, q or $q-1$)

- **STEP4:** if s and q values were both same at the previous iteration

  $s^* = s$ and $q^* = q$

  else

  go back to STEP2.

Example 1 *Finding optimal* (s, S) *pair for the problem below by using Algorithm 3.*

Demand ~$N(5, 2)$*: round the numbers to have integer demands, negative demands are assumed to be zero*

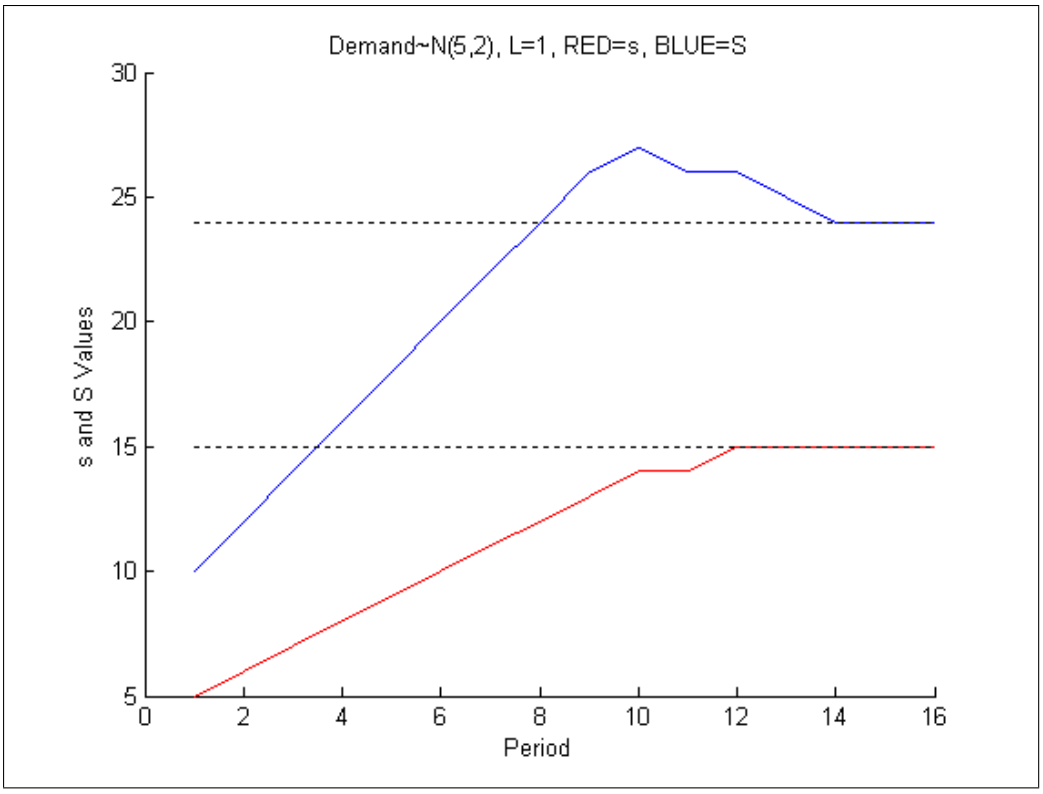
Lead Time: 1 day

Holding cost/unit: $0.1

Ordering cost: $1

Penalty cost/unit: $30

After starting with initial $s = 5$ and $q = 5$, $s^* = 15$ and $q^* = 9$ ($S^* = 24$) is found by running code in Appendix C. The optimal solution could be found in 15 iterations and the figure below shows the iterations from (5,10) to (15,24).



Iterations on s and S

3.3 Initializing Perturbation Analysis Parameters

The Perturbation Analysis will be initialized with values that are the outputs of the Ehrhardt's Heuristic. Thus, the Perturbation Analysis Algorithm will have two main stages:

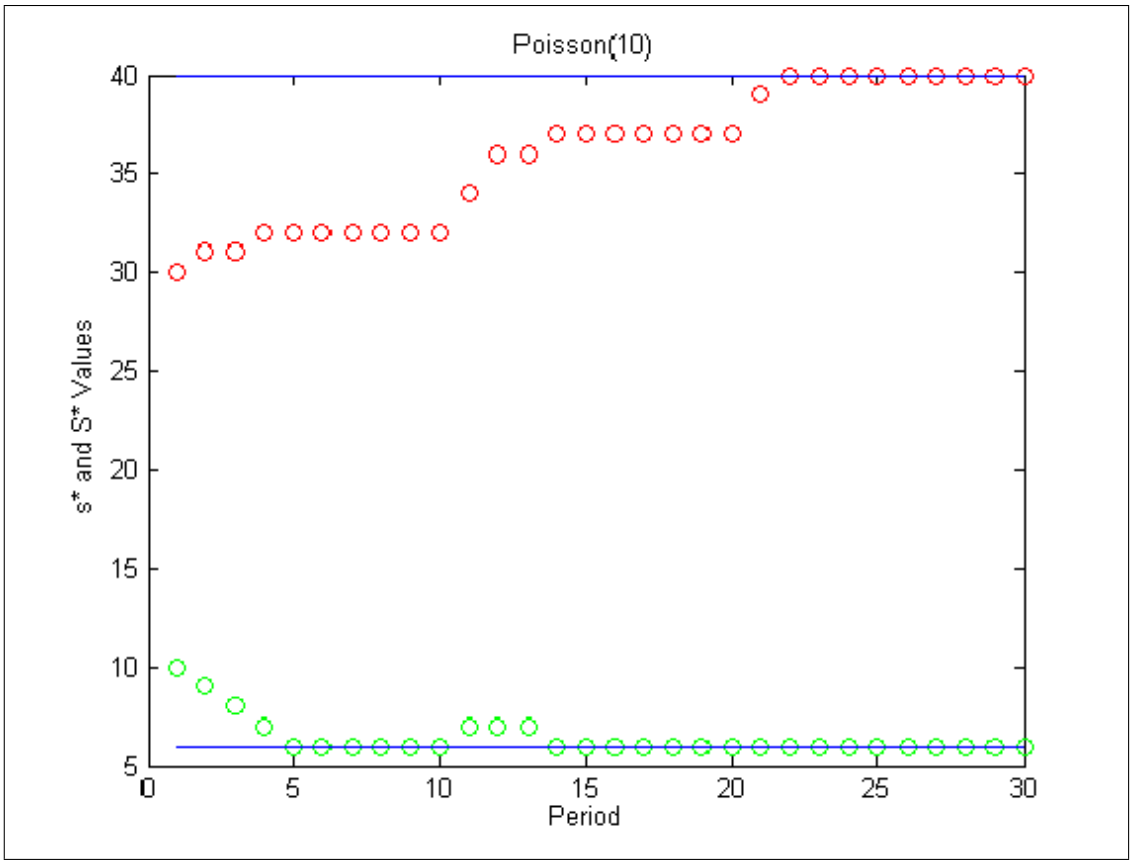
1. Finding Rough Estimate by using Ehrhardt's Revised Power Approximation
2. Fine Tuning by Perturbation Analysis

As mentioned in Section 2.4, Ehrhardt introduces two heuristics that compute approximation of optimal s_E and S_E values with simple equations; Power Approximation and Revised Power Approximation. These values are really good approximations of the

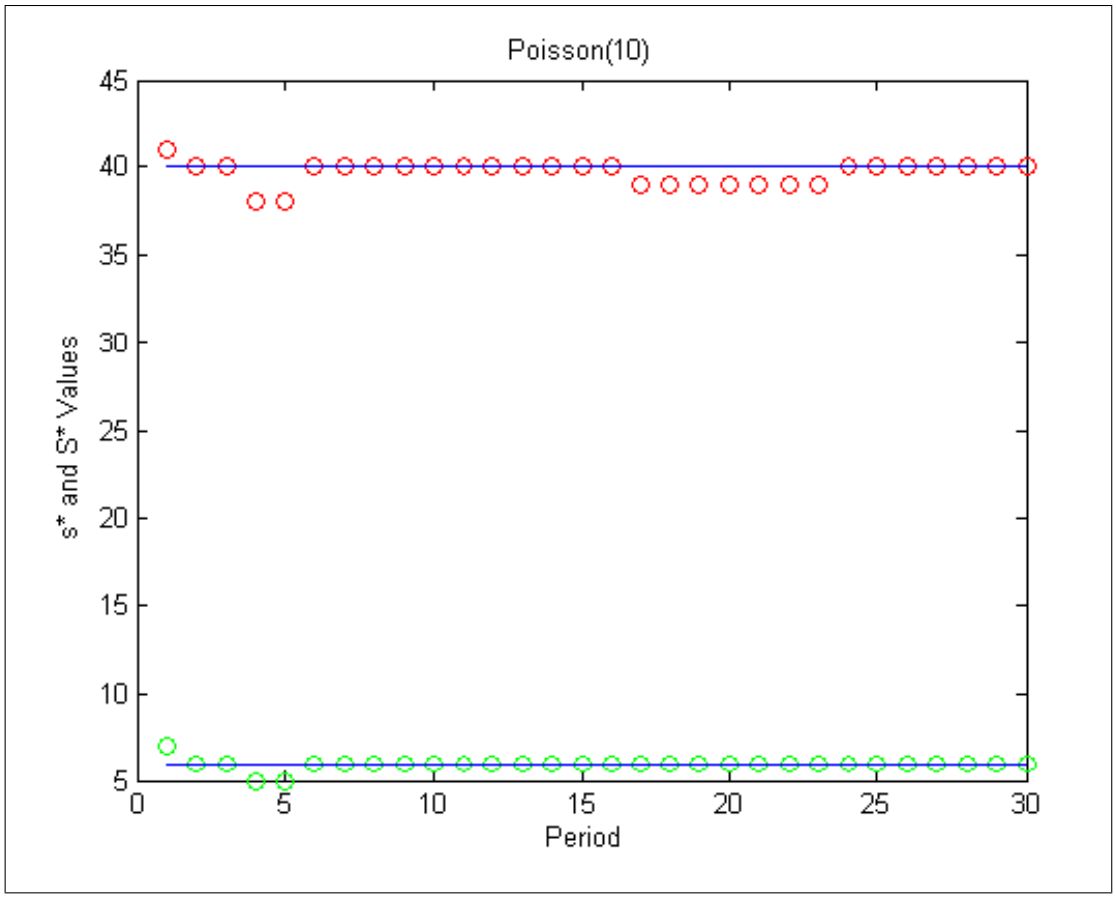
optimal values. Both of these heuristics need the knowledge of mean and variance of the demand distribution.

In this thesis, Ehrhardt's Revised Power Approximation will be used without the knowledge of mean and variance. This new heuristic will work as follows; the demand will be observed for some time period. The mean and the variance of this observed demand will be calculated and these calculated values will be used to find the optimal s and S values. After that period, the policy parameters will be updated and the new mean and variance will be calculated with the new demand data. The MATLAB Code of the Ehrhardt's Heuristic can be found in Appendix D.

The main issue is why to use Ehrhardt's Revised Power Approximation to initialize the problem. For running the Perturbation Analysis Algorithm, it is a must to start with some initial values. Ehrhardt's Revised Power Approximation is a very good approximation for the optimal parameters. It cannot always find the optimal but is a really good approximation that only requires some nanoseconds. If the Ehrhardt's Revised Power Approximations is not used, you have to start from an initial point; let's say $(10, 30)$. Then, the Perturbation Analysis will work as the graph below:



However, if the Ehrhardt's Revised Power Approximation were used to initialize the values, the following graph would be obtained:



As can be seen, the PA Algorithm can find the optimal solution in both cases. However, since the PA Algorithm can update itself one by one, it takes some time to reach the optimal solution in the first case. Thus, initializing the (s, S) values by Ehrhardt's Revised Power Approximation Method is very advantageous. The graph converges to the optimal value in the 6^{th} period. However, after a while, the S value again starts to oscillate because of different demand sample paths. If the sample path was always the same, after the convergence there would be no more oscillations. The converged value would be the optimal value where the Perturbation Analysis method cannot move in either sides for the same demand data.

To conclude, the Perturbation Analysis Algorithm works in the following way:

- **STEP1:** Determine a short time period, R (i.e. 7 days)
- **STEP2:** Observe the problem environment for R days
- **STEP3:** Calculate the mean and variance of the demand for the first R days
- **STEP4:** Use Ehrhardt's Heuristic to calculate the approximate values of s_E and S_E, and start with $s = s_E$ and $q = S_E - s_E$
- **STEP5:** Use Algorithm 1 and update s value ($s + 1$, s or $s - 1$)
- **STEP6:** Use Algorithm 2 and update q value ($q + 1$, q or $q - 1$)
- **STEP7:** if s and q values were both same at the previous iteration

    $s^* = s$ and $q^* = q$

  else

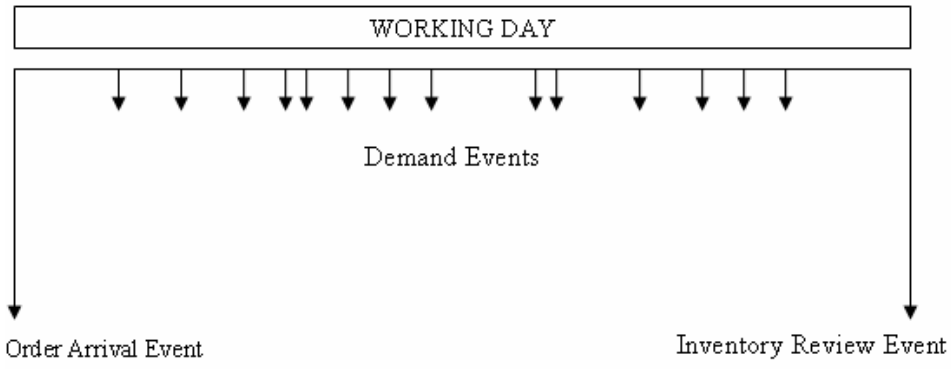
    go back to STEP5

CHAPTER 4

SIMULATION

4.1 Logical Model

The logical model of the problem can be summarized as the figure below:



As seen from the figure, there are mainly three types of events in our problem; order arrival event at the beginning of the day, demand event during the day, and inventory review event at the end of the day.

Orders arrive at the beginning of the working days. If there is a lead time of zero, the orders arrive in the following morning. On the other hand, the orders will not arrive before the lead time passes. At the beginning of the first day, no orders arrive since no orders were given before. Demands occur during the whole day as integers. Random demands occur in the problem and the distribution of the demands can be any kind of distribution or may have empirical distribution. Thus, during the day, demand *D* occurs and the ending inventory of the first day becomes "*S-D*". Inventory position also becomes "*S-D*".

The inventory is reviewed at the end of every day. At the end of the day, the inventory position is calculated and order is given if the inventory position is below or equal to s. "$S-$ Inventory position" units are ordered and waiting orders is updated. Since order is given, the new inventory position will be S. The inventory on hand is also calculated when the shop is closed. Then, we calculate the cost of the first day from the inventory level. If we have inventory on hand, we will have holding cost at the end of the first day. On the other hand, if there are unsatisfied demands, there will be penalty cost at the end of the day. Also, an ordering cost will be applied if we have given an order.

The objective of the problem is to choose s and S values such that the summation of the daily costs is minimized.

The general flowchart of the problem can be seen in Appendix E.

Numerical Example

To visualize the problem, a simple numerical example is given below. The example is given for a period of 10 days and the (s, S) values are defined as (10,15). We have a constant Lead Time of 2 days and demand is uniformly distributed between 1 and 10 units. Demands arrive as integers and the costs are defined as; holding cost/unit=$1, penalty cost/unit=$100, ordering cost=$10.

As shown from Figure 4.1 and 4.2, at the beginning of day one, the beginning inventory is 15. During day one, 8 units are demanded. Since we had inventory on hand, we have sold all of the 8 items to the customers and at the end of the day we have remaining 7 items on inventory. We order $S - Y_1$ (15-7=8) items since the "inventory position" is be-

DAY	1	2	3	4	5	6	7	8	9	10
Orders Received	-	-	-	8	6	5	5	7	-	9
Beginning Inventory	15	7	1	4	5	3	6	6	3	8
Demand	8	6	5	5	7	2	7	3	4	5
Ending Inventory	7	1	-4	-1	-2	1	-1	3	-1	3
Inv. Pos. Before Order	7	9	10	10	8	13	6	12	8	10
Orders Given	8	6	5	5	7	-	9	-	7	5
Inv. Pos. After Order	15	15	15	15	15	13	15	12	15	15
Holding Units	7	1	-	-	-	1	-	3	-	3
Backlogged Units	-	-	4	1	2	-	1	-	1	-
Total Cost	17	28	438	548	758	759	869	872	982	995

Fig. 4.1. Numerical Example of Inventory Simulation for s=10, S=15

low the s value. For the first day, the cost calculation is done as follows; we have 7 units in inventory, which means a cost of 7x(unit holding cost)=$7. We have no shortages; thus no costs occur from shortages. On the other hand, since we have given an order, an ordering cost (=$10) is also occurred. As can be seen, at the end of first day, we have a total cost of $17.

After the first day, the second day starts. We had an inventory of 7 units from the first day. Also we had ordered 8 items the day before. However, since the leadtime is 2 days, we do not receive any orders this day. At the end of second day, we have a 1 unit inventory on hand, and order 6 items since the "inventory position" is 9 which is less than the "s value".

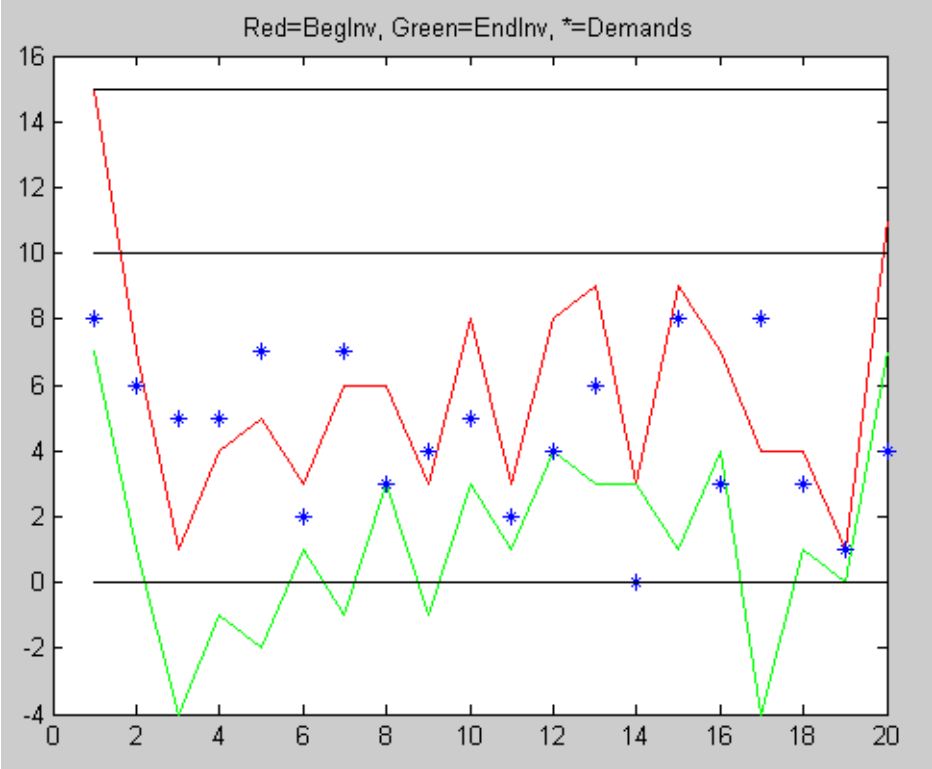


Fig. 4.2. Inventory Simulation Example

Similarly, at the end of the third day, we have 4 backlogged units. From these shortages, we have a cost of 4x(unit shortage cost)=$400. At the beginning of fourth day, the orders given in first day are received. And by these manner, the simulation continues. For these 10-days simulation, we have an average cost of $99.5. The algorithms defined in Chapter 3.1, calculates the optimal s and S values that minimize this average cost by using the method of Perturbation Analysis.

4.2 Simulation Models

Several simulation models were constructed to simulate the problem environment.

- ARENA Model: An ARENA Model was constructed to simulate the problem environment. The distribution of the demand, s, S values, the cost rates, and the lead time are given into the system as the inputs. The outputs are number of holding inventories, number of shortages, number of orders, and the totalcost. The ARENA model can be seen in Appendix F.

- EXCEL Model: In the EXCEL Model, the inputs are the same as the ARENA model. However in this model, the outputs can be seen period by period. For example, it can be seen how many units were backordered in 17^{th} period, what is the total cost of 26^{th} period, etc. Also the probability of shortage can also be seen in this model. The EXCEL model can be seen in Appendix G.

- MATLAB Model: The MATLAB model works the same way with the EXCEL model. However, it works much faster when compared to the EXCEL Model. Thus, the MATLAB model will be used in the following applications. The MATLAB code that models the simulation can be found in Appendix H.

CHAPTER 5

NUMERICAL EXAMPLES

In this chapter, the Perturbation Analysis Algorithm defined in Section 3.3 will be used for different types of demand distributions. The aim of this chapter is to show that the algorithms introduced in this thesis can find the optimal solutions for all different types of demand distributions. In addition to this, the adaptability of the methods will be compared. Adaptability means to find out the new optimal policy parameters after a sudden change in demand.

Three new parameters will be introduced in this chapter;

- Delta Day (δ): Period Length (in days)
- Period (Φ): Simulation Length (in periods)
- Demand Period (γ): Moving window's size (in periods)

δ is the time period that a single policy is run. After δ day passes, the (s,S) parameters will be updated and the second Φ starts. As can be understood, Φ is the time period of δ. To give a simple example, lets take $\delta = 7$ days, and $\Phi = 30$. Then, the parameters will be updated in every 7 days, and the total runtime will be 210 days.

γ is the time that is used to compute the new optimal parameters. For example, if we have $\delta = 7$ days, $\gamma = 3$, the new (s,S) parameters will be calculated by observing the last 21 days' demand history. When γ is larger, iteration number to find the optimal pair will be less. On the other side, when you have very large γ, then the adaptability of the code will be worse. The effects of these parameters will be seen in Section 5.1 and Section 5.3.

In this chapter, firstly, the demand distribution will be taken as Poisson Distribution. In this section, I will compare my results with the results of Zheng&Federgruen (1991) to discuss about the optimal solution. Also, the effects of δ and γ values will be discussed in this section.

After that section, the algorithm will be tested with Discrete Uniform and Discretized Normal distributions, respectively. Also, a demand distribution that is created by composition method (Poisson(10) with 40% probability and Poisson(25) with 60% probability) will be taken as the input; and it will be shown that the algorithm can also work with empirical distributions. In all of these examples, the Perturbation Analysis Algorithm will be compared to the Retrospective Integer Programming Method and Ehrhardt's Heuristic. Retrospective Integer Programming Method uses the past data and finds the optimal (s, S) pair by using integer programming. This problem is solved in GAMS environment and the GAMS code can be found in Appendix I.

Finally, at the last section of this chapter, the mean of the demand distribution will change suddenly after some time, and the adaptability of the Perturbation Analysis Algorithm, Retrospective Integer Programming Method and Ehrhardt's Heuristic will be compared for different δ and γ values.

5.1 Poisson Distribution

In this section, Poisson Distribution with mean 10, and with mean 25 will be taken as the demand distributions. Zheng&Federgruen (1991) studied about the Poisson Distribution and founded the following results:

μ	s^*	S^*	c^*
10	6	40	35.022
25	19	56	54.262

The table shows that, for Poisson(10), the optimal solution is found as $(6, 40)$ with an average cost of 35.022. When the mean is 25, the optimal solution becomes $(19, 56)$ with an average cost of 54.262. The costs are taken as; holding cost rate $h = 1$, penalty cost rate $p = 9$, and fixed setup cost $K = 64$ with no leadtime $(L = 0)$. In this section, to compare the Perturbation Analysis Algorithm with the algorithm defined in Zheng&Federgruen (1991), in all of the examples the input parameters will be taken as given above ($h = 1$, $p = 9$, $K = 64$, $L = 0$).

5.1.1 Poisson Distribution (10) with No Lead Time

The optimal (s, S) parameters are found as $(6, 40)$ in Zheng&Federgruen (1991) for the input parameters defined above. As defined in Section 3.3, the Perturbation Analysis Algorithm initializes the s and S values with the Ehrhardt's Algorithm. While the Ehrhardt's Heuristic is run for the first Period, $s_E = 7$, $S_E = 41$ is found. Thus, the algorithm starts with $s(1) = 7, S(1) = 41$. Then, different sample paths are generated for each runs. The output of Perturbation Analysis Algorithm for 20 runs can be seen in Fig.J.1-Fig.J.2 in Appendix J.

For $\gamma = 3$, when we look at the graphs Fig.J.2, Fig.J.4, Fig.J.6, it is seen that the optimal values oscillate a lot, and in the periods, the algorithms find different optimal values in different runs. When we look at the ensemble average graph (Fig.J.8), it is seen that the average values in Perturbation Analysis Algorithm converges to the optimal values. There

is a dispersity because of the small γ value. It means that the algorithms only take the previous 3 Periods' history and it cannot be enough to find the exact optimal values. When the algorithms' cost are compared (Fig.J.10), it can be seen that the Perturbation Analysis Algorithm has the smallest cost where the Ehrhardt's Heuristic has the largest cost.

When the γ value is increased, the optimal (s, S) values start to oscillate near the optimal values in Perturbation Analysis Algorithm (Fig.J.1). All of the runs give the same optimal (s, S) values after some time. The Retrospective Integer Programming Method and the Ehrhardt's Heuristic also converge to some value, but they are not the optimal values (Fig.J.3, Fig.J.5). Ehrhardt's Heuristic overestimates the (s, S) values where the Retrospective Integer Programming Method usually underestimates. When looked at the cost functions, Fig.J.9, as in the first part Perturbation Analysis Algorithm gives the least cost where Ehrhardt's Heuristic gives the most.

5.1.2 Poisson Distribution (25) with No Lead Time

This part is another example for the Poisson distribution. In Zheng&Federgruen (1991), the optimal solution is calculated as $(19, 56)$ and by the Perturbation Analysis Algorithm, the graphs (Fig.J.11, Fig.J.12) in Appendix J are obtained after 20 runs.

The analysis made in Subsection 5.1.1 are also correct for this subsection. When the γ value is small, the optimal solutions of the algorithms oscillate a lot. However, the ensemble average of the Perturbation Analysis Algorithm converges to the optimal values (Fig.J.18). When this value is increased, the algorithms start to get better results. In addition to this, Ehrhardt's Heuristic again overestimates the policy parameters and can only

find values that are near optimal, not optimal. (Fig.J.13 - Fig.J.14) The Retrospective Integer Programming Method usually underestimates the policy parameters (Fig.J.15, Fig.J.16) and the Perturbation Analysis Algorithm, again, has the minimum cost (Fig.J.19, Fig.J.20).

Zheng&Federgruen analyze 24 examples in their paper. I had also run all of these examples with the Perturbation Analysis Algorithm, Ehrhardt's Heuristic, and Retrospective Integer Programming Method, and found the following results (these results are the averages of 20 runs for each example):

μ	s* (ZF)	S* (ZF)	μ	s* (PA)	S* (PA)	s* (EHR)	S* (EHR)	s* (RIP)	S* (RIP)
10	6	40	**10**	6	40	7	42	5	39
15	10	49	**15**	10	49	11	51	9	48
20	14	62	**20**	14	62	15	64	13	60
25	19	56	**25**	19	56	21	59	18	55
30	23	66	**30**	23	66	24	68	22	65
35	28	77	**35**	28	77	29	79	27	76
40	33	87	**40**	33	87	35	90	32	86
45	37	97	**45**	37	97	38	99	36	96
50	42	108	**50**	42	108	44	111	42	107
55	47	118	**55**	47	118	48	121	46	117
60	52	129	**60**	52	129	53	131	51	127
65	56	75	**65**	56	75	57	77	55	74
70	62	81	**70**	62	81	63	83	61	80
75	67	86	**75**	67	86	69	89	66	85
21	15	65	**21**	15	65	17	68	14	64
22	16	68	**22**	16	68	18	71	15	67
23	17	52	**23**	17	52	19	55	16	51
24	18	54	**24**	18	54	20	57	17	53
51	43	110	**51**	43	110	44	112	43	109
52	44	112	**52**	44	112	45	114	44	111
59	51	126	**59**	51	126	52	128	50	125
61	52	131	**61**	52	131	53	133	51	129
63	54	73	**63**	54	73	55	75	53	72
64	55	74	**64**	55	74	56	76	54	73

In the table above, the green color represents optimal values, yellow values represent the values that are above optimal and the red color represents the values that are below the optimal. It is easy to see that the Perturbation Analysis Algorithm can find the same op-

timal solutions as Zheng&Federgruen's Algorithm for all of the 24 examples. Ehrhardt's Heuristic always overestimates the policy parameters and Retrospective Integer Programming Method usually underestimates.

5.2 Comparison with Other Algorithms

In this section, several demand distributions will be compared by other algorithms; i.e. Ehrhardt's Heuristic and Retrospective Integer Programming Method. The aim is to check whether the algorithms can find the optimal solutions for these distributions.

5.2.1 Uniform Distribution (0,10) with Lead Time=1 Day

In this section, Discrete Uniform Distribution with minimum value of 0, and maximum value of 10 is taken as the demand distribution.

By running the algorithms presented in this thesis, the following results are obtained:

After running the code which is in Appendix H, the optimal s^* and S^* values are found as 15 and 24 in a time of 40.12 seconds. It means that when all of the (s, S) pairs' cost is calculated, the minimum cost will be satisfied with the $(15, 24)$ pair.

For this distribution, as seen from Fig.J.21-Fig.J.22 Perturbation Analysis Algorithm again converges to the optimal parameters and has the minimum cost when compared to the other two methods.

5.2.2 Normal Distribution (5,1) with Lead Time=1 Day

In this section, Normal Distribution is taken as the demand distribution with the parameters of Mean=5, Std.Dev=1. Since normal distribution is a continuous distribution, the values had been rounded to the nearest integer to have integer valued demands, and the negative demands will be taken as no-demand.

By running the algorithms presented in this thesis, the following results are obtained:

After running the code which is in Appendix H, the optimal s^* and S^* values are found as 11 and 18. It means that when all of the (s, S) pairs' cost is calculated, the minimum cost will be satisfied with the $(11, 18)$ pair.

Again for the Normal Distribution, the Perturbation Analysis Algorithm converges to the optimal parameters and has the minimum cost when compared to the other two methods. The figures (Fig.J.23-Fig.J.24) can be seen in Appendix J.

5.2.3 Empirical Distribution

In this section, the demand distribution does not have a specific distribution. This distribution is a combination of Poisson(10) and Poisson(25) distributions with 40% and 60% probabilities, respectively where the demand distribution is created by the Composition Method. In this case, all of the three algorithms can work well since all of them use the previous data, without the knowledge of variance and mean. The optimal values of the (s, S) parameters are found by the code in Appendix H and the optimal (s^*, S^*) values are found as $(14, 49)$. Perturbation Analysis Algorithm has the minimum cost when compared

by the other two methods. The graphs about this section (Fig.J.25-Fig.J.26) can be seen in Appendix J.

5.3 Adaptability

For the adaptability issue, in the first 20 periods the demand distribution is taken as POI(10). After the 21^{st} period, the demand distribution changes to POI(25). In this section, the aim is to check if the methods can adapt themselves to the sudden demand changes.

5.3.1 $\delta = 15$ Days, $\gamma = 3$ Periods

When γ value is very small, from Fig.J.27-30, it is seen that all of the methods (Perturbation Analysis Algorithm, Retrospective Integer Programming Method and Ehrhardt's Heuristic) are all adaptive for sudden changes. When we analyze the rate of adaptability, it is seen that the Perturbation Analysis Algorithm cannot adapt itself as quick as the other methods. It is an expected result since the Perturbation Analysis Algorithm can update itself one by one. The Retrospective Integer Programming Method and Ehrhardt's Heuristic update themselves very quickly since they only consider the last three periods' demand data. On the other hand, all of the algorithms cannot converge to a single value quickly. It means that when the γ value is very small, the methods cannot converge to the optimal value quickly but the adaptability for sudden demand changes is quite good.

When the ensemble averages of the 20 runs are considered (Fig.J.30), it is seen that the average optimal values of Perturbation Analysis Algorithm converge to the real optimal

values. On the other hand, Ehrhardt's Heuristic overestimates and Retrospective Integer Programming Method underestimates the optimal values as also discussed in Section 5.1.

5.3.2 $\delta = 15$ Days, γ =All History

When we keep the δ same but only increase the γ value, it is seen that convergence rate decreases (Fig.J.31-34).

For the Perturbation Analysis Algorithm, the S value starts to converge around a value in 35^{th} period where it was 33^{rd} period in Section 5.3.1. The s value also starts to converge around a value in 39^{th} period where it was 35^{th} period in the previous section.

The change of the convergence rate can be seen better in Ehrhardt's Heuristic. That method could converge in five periods when γ was very small. However, when it is increased, it needs about ten periods to converge to the new values.

On the other hand, the values oscillate in specific values instead of a big range of values when the γ value is increased. When looked at Fig.J.28, the S value in Perturbation Analysis Algorithm converges to values between 54-58 when γ value was small. However in Fig.J.32, the algorithm converges to only three values; 55, 56, 57.

The ensemble average graph is very similar to the previous section. Perturbation Analysis Algorithm can converge to the real optimal value where Ehrhardt's Heuristic overestimates and Retrospective Integer Programming Method underestimate.

5.3.3 $\delta = 90$ Days, $\gamma =$All History

In this section, the effect of δ will be analyzed. From Fig.J.35-38, it is seen that when the δ value increases, convergence to a single value can be satisfied earlier since much more days' demands are observed. After observing 90 days' demand, the (s, S) values are updated and it helps to converge to a specific value. The disadvantage of increasing δ cannot be seen from the graph. However, when the δ is taken as a big value, then the s, S values cannot be updated for a large number of days and it decreases the flexibility of the inventory policy since the period time increases.

CHAPTER 6

CONCLUSIONS AND FUTURE DIRECTIONS

Inventory policies are used to determine when and how many units to order. For the periodic review policies, the most common inventory policy is the *(R,s,S)* policy. In this policy, it is known that if the parameters of the system are chosen appropriate, there is always an optimal solution, i.e. the expected period cost can be minimized.

For finding the optimal policy parameters, if the demand distribution is known, exact methods and heuristics are used in the literacy. In this thesis, it is assumed that the demand distribution is not known. The mean and the variance of the demand distribution are also not known. For this case, a perturbation analysis based method is introduced. This method (Perturbation Analysis Algorithm) initializes itself from a heuristic named Ehrhardt's Revised Power Approximation. After the initialization, the method anticipates the sensitivity of *(s,S)* parameters to the period cost and updates itself to find the optimal *(s*,S*)* pair.

The outputs of Perturbation Analysis Algorithm were firstly compared with the results of Zheng&Federgruen (1991) for the Poisson Distribution. In all of the twenty-four examples, the Perturbation Analysis Algorithm could find the same optimal values. After that, the Perturbation Analysis Algorithm was compared by two other methods, Ehrhardt's Heuristic and Retrospective Integer Programming Method for several demand distributions, including the empirical distribution for twenty runs. In all of the examples, the Perturbation Analysis Algorithm could oscillate on the optimal solution.

When the ensemble average of these twenty runs are taken, the Perturbation Analysis Algorithm can find the real optimal (s^*, S^*) values. On the other hand, Ehrhardt's Heuristic always overestimates the optimal values where the Retrospective Integer Programming Method usually underestimates. When the cost graphs were compared, in all of the cases, Perturbation Analysis Algorithm gives the minimum cost.

Adaptability to sudden demand changes was also discussed. When a sudden demand change occurs, all of the three methods can adapt themselves to the new demand distribution. However, the Perturbation Analysis Algorithm adapts itself very slowly when compared with the other two methods. This is an expected result since the Perturbation Analysis Algorithm, by definition, can update itself one by one while the others can update themselves suddenly. On the other hand, the Perturbation Analysis Algorithm again finds the optimal value after a time and oscillates there where the other two methods oscillate in near-optimal values.

To conclude, the algorithm defined in this thesis works really well for all kinds of distributions and does not require any knowledge of the distribution, mean or variance. It can adapt itself to demand changes and always finds the optimal solution. Another advantage of this algorithm is, it finds the optimal solution in a very short time. The Complete Enumeration Method finds the optimal solution in 3-4 minutes (with MATLAB code) and the Retrospective Integer Programming Method solves in several hours (with GAMS code); however, the Perturbation Analysis Algorithm finds the solution in average of 7 seconds.

There are also some future works that can be done for improving this algorithm. As mentioned before, the weakest point of the algorithm is that it can adapt itself very slowly

since it updates the values one by one. For that point, a "Step Size" can be defined and while the cost difference between (s, q) and $(s + 1, q)$ is very high, the s value can be increased more than 1 (regarding to Step Size). By that way, the time to adapt to changes would be decreased, but it may start not finding the exact optimal point if Step Size is very large.

Another solution would be to update the Perturbation Analysis Algorithm from the output of Ehrhardt's Heuristic. This heuristic can not find the exact optimal point but gives a point which is not so far from the optimal in a very short time. Thus, if a sudden demand change has occurred, the Ehrhardt's Heuristic could be used for one period to re-initialize the Perturbation Analysis Algorithm and then continue with Perturbation Analysis Algorithm.

Another trade-off is about the δ and γ values. When these values are taken very small, the algorithms cannot converge to a specific value. When these values are increased, then the algorithms cannot adapt themselves quickly in sudden demand changes. Thus, there may be some varying δ, γ values (not fixed). These values can increase if the demand variance decreases for finding the exact optimal solutions, and vice versa for quick adaptability.

References

AFT, L.S. 1987. "Production and Inventory Control", *San Diego: Harcourt Brace Jovanovich.*

ARCHIBALD, B., SILVER, E. 1978. "(s,S) Policies Under Continuous Review and Discrete Compound Poisson Demands", *Management Science,* Vol. 24, 899-908.

ARROW, K.J., HARRIS, T., MARSCHAK, J. 1951. "Optimal Inventory Policy", *Econometrica*, Vol. 19, No. 13, 250-272.

AXSATER, S. 2004. "Inventory Control", *Boston: Kluwer Academic.*

BASHYAM, S., FU, M.C. 1998. "Optimization of (s,S) Inventory Systems with Random Lead Times and a Service Level Constraint", *Management Science*, Vol. 44, No. 12, 243-255.

BELL, C., 1970. "Improved Algorithms for Inventory and Replacement Stock Problems", *SIAM Journal of Applied Mathematics,* Vol. 18, 558-566.

CASSANDRAS, C.G., DAI, L., PANAYIOTOU, C.G. 1997. "Ordinal Optimization for Deterministic and Stochastic Discrete Resource Allocation", *Decision and Control Proceedings of the 36th IEEE Conference,* Vol. 1, 662-667.

DANIEL, J.R., RAJENDRAN, C. 2005. "A Simulation-Based Genetic Algorithm for Inventory Optimization in a Serial Supply Chain", *International Transactions in Operational Research*, Vol. 12 (1), 101–127.

EHRHARDT, R. 1979. "The Power Approximation for Computing (s,S) Inventory Policies", *Management Science*, Vol. 25, No. 8, 777-786.

EHRHARDT, R. 1984. "(s,S) Policies for a Dynamic Inventory Model with Stochastic Lead Times", *Operations Research*, Vol. 32, No. 1, 121-132.

FEDERGRUEN, A., ZIPKIN, P. 1984. "An Efficient Algorithm for Computing Optimal (s,S) Policies", *Operations Research*, Vol. 32, No. 6, 1268-1285.

FEDERGRUEN, A., ZHENG, Y.S. 1992. "An Efficient Algorithm for Computing an Optimal (r,Q) Policy in Continuous Review Stochastic Inventory Systems", *Operations Research*, Vol. 40, No. 4, 808-813.

FENG, Y., XIAO, B. 2000. "A New Algorithm for Computing Optimal (s,S) Policies in a Stochastic Single Item/Location Inventory System", *IIE Transactions,* Vol. 32, 1081-1090.

FOGARTY, D.W., HOFFMAN, T.R. 1983. "Production and Inventory Control", *Cincinnati: South-Western Publishing*

FU, M.C., HEALY, K.J. 1992. "Simulation Optimization of (s,S) Inventory Systems", *Proceedings of the 1992 Winter Simulation Conference*, 506-514.

FU, M.C., HEALY, K.J. 1997. "Techniques for Optimization via Simulation: An Experimental Study on an (s,S) Inventory System", *IIE Transactions*, Vol. 29, 191-199.

FU, M.C. 1994. "Sample Path Derivatives for (s,S) Inventory Systems", *Operations Research*, Vol. 42, No. 2, 351-364.

GLASSERMAN, P., TAYUR, S. 1995. "Sensitivity Analysis for Base-Stock Levels in Multiechelon Production-Inventory Systems", *Management Science*, Vol. 41, No. 2, 263-281.

GREENE, J.H. 1987. "Production and Inventory Control Handbook", *New York: McGraw-Hill.*

IGLEHART, D.L. 1963. "Optimality of (s,S) Policies in the Infinite Horizon Dynamic Inventory Problem", *Management Science*, Vol. 9, 259-267.

KAPLAN, R.S. 1970. "A Dynamic Inventory Model with Stochastic Lead Times", *Management Science*, Vol. 16, No. 7, 491-507.

L'ECUYER, P. 1991. "An Overview of Derivative Estimation", *Proceedings of the 1991 Winter Simulation Conference*, 207-217.

NADDOR, E. 1975. "Optimal Heuristic Decisions for the s,S Inventory Policy", *Management Science*, Vol. 21, No. 9, 1071-1072.

NADDOR, E. 1982. "Inventory Systems", *Florida: Krieger.*

ROUNDY, R.O., MUCKSTADT, J.A. 2000. "Heuristic Computation of Periodic-Review Base Stock Inventory Policies", *Management Science,* Vol. 46, 104-109.

SAHIN, I. 1982. "On the Objective Function Behavior in (s,S) Inventory Models", *Operations Research*, Vol. 30, No. 4, 709-724.

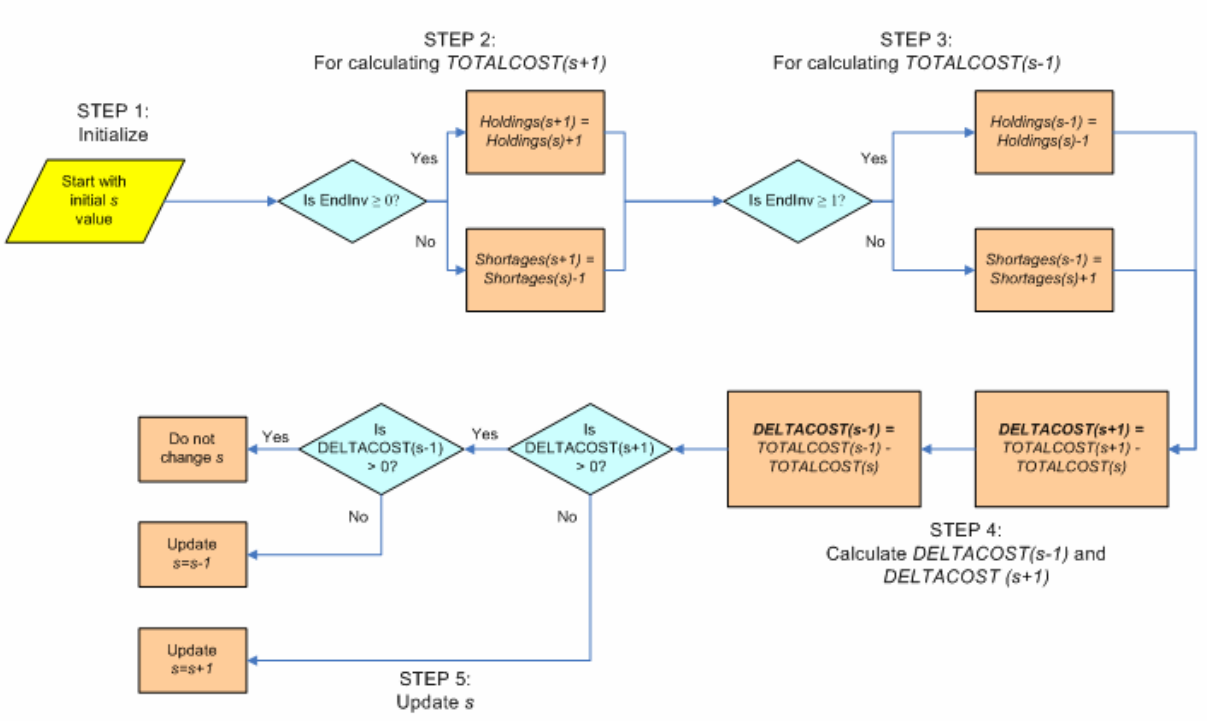
SCARF, H. 1960. "The Optimality of (S,s) Policies in the Dynamic Inventory Problem", *Mathematical Methods in the Social Sciences*.

VEINOTT, A.F., WAGNER, H.M. 1965. "Computing Optimal (s,S) Inventory Policies", *Management Science,* 525-552.

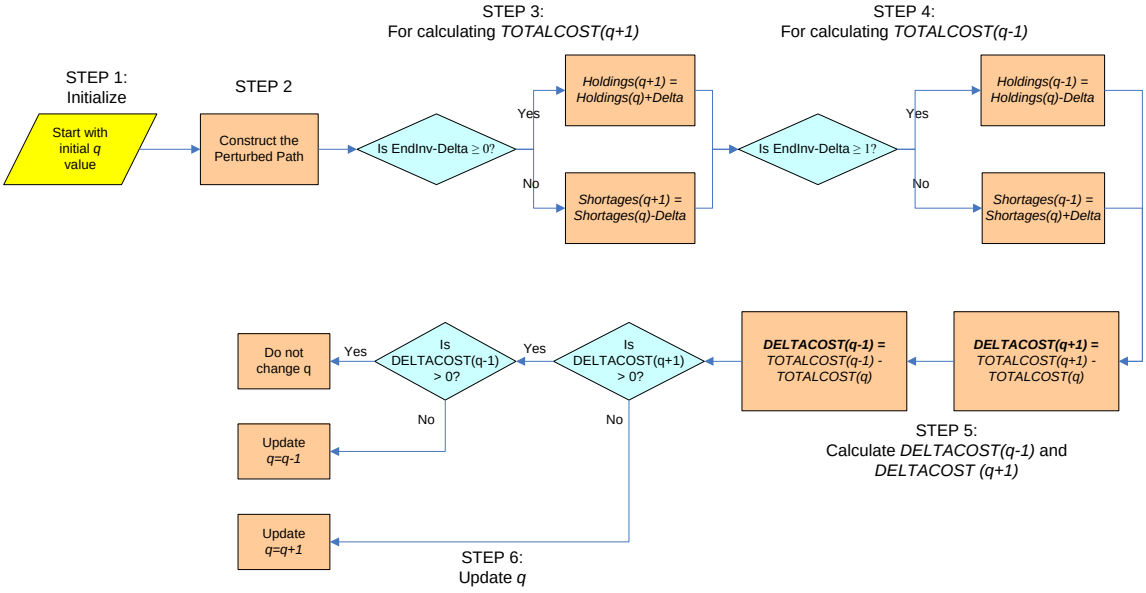
ZHAO, Y., MELAMED, B. 2004. "Make-to-Stock Systems with Backorders: IPA Gradients", *Winter Simulation Conference*, 559-567.

ZHENG, Y.S., FEDERGRUEN, A. 1991. "Finding Optimal (s,S) Policies is about as Simple as Evaluating a Single Policy", *Operations Research*, Vol. 39, No. 4, 654-665

Appendix A
PA on "s" – Algorithm 1



Appendix B
PA on "q" – Algorithm 2



Appendix C
Finding Optimal (s^*, q^*) Pair: Algorithm3.m

C.1 Main Code

```
clc
    clear
    global opts
    global day
    global D
    global BegInv
    global EndInv
    global IP
    global shortage
    global order_amount
    global WaitingOrders
    global hold_array
    global short_array
    global holdingcost
    global shortagecost
    global orderingcost
    global holdings
```

```
global shortages

global totalcost

global totalcost_array

global order

global maxs

global loop

global L

global S

global s

global q

global Cs

global Co

global Ch

global TOTALCOST

load demNOR_5_1.mat

t=cputime;

day=20000;

q_array(1)=5;

s_array(1)=5;

q=q_array(1);

s=s_array(1);

L=2;
```

```
Ch=1;

Cs=100;

Co=10;

maxs=50;

totalcost=0;

finish=0;

loop=1;

while finish==0

        init

        q=q_array(loop);

        create_path

        calc_cost1

        der_s

        s_array(loop+1)=s;

        init

        der_q_prep

        calc_cost


%%%%%%%%% PA FOR Q %%%%%%%%%%

        order1=zeros(1,day); % EndInv

        order2=zeros(1,day); % PLUS

        order3=zeros(1,day); % MINUS
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%      for     q+1     %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        for i=1:(L+1)
                PL(i)=EndInv(i)+1;
        end


        IPplus(1)=PL(1);
        if IPplus(1)<=s
                IPplus(1)=s+q+1;
        end
        if (IPplus(1)==s+q+1 & D(1)>0)
                order2(1)=1;
        end


        for i=1:L
                IPplus(i+1)=IPplus(i)-D(i+1);


                if IPplus(i+1)<=s
                        IPplus(i+1)=s+q+1;
                end
```

```
        if (IP(i+1)==s+q & D(i+1)>0)
                order1(i+1)=1;
        end

        if (IPplus(i+1)==s+q+1 & D(i+1)>0)
                order2(i+1)=1;
        end
end

delta=1;
for i=(L+2):day
        if (order1(i-L-1)==0 & order2(i-L-1)==0)
                PL(i)=EndInv(i)+delta;
        elseif (order1(i-L-1)==1 & order2(i-L-1)==1)
                delta=1;
                PL(i)=EndInv(i)+delta;
        elseif (order1(i-L-1)==1)
                PL(i)=EndInv(i)+IPplus(i-L-1)-q-s;
                delta=IPplus(i-L-1)-q-s;
        elseif (order2(i-L-1)==1)
                PL(i)=EndInv(i)+q+s+1-IP(i-L-1);
```

```
            delta=q+s+1-IP(i-L-1);
        else
            %
        end

        IPplus(i)=IPplus(i-1)-D(i);

        if IPplus(i)<=s
            IPplus(i)=s+q+1;
        end

        if (IP(i)==s+q & D(i)>0)
            order1(i)=1;
        end

        if (IPplus(i)==s+q+1 & D(i)>0)
            order2(i)=1;
        end

    end

%%%%%%%%%%% COST FOR q+1 %%%%%%%%%%%%%%%%
```

```
holdingcost_plus=0;
shortagecost_plus=0;
orderingcost_plus=0;

for i=1:day
    if PL(i)>0
        holdingcost_plus=holdingcost_plus+PL(i)*Ch;
    else
        shortagecost_plus=shortagecost_plus-PL(i)*Cs;
    end

    if order2(i)==1
        orderingcost_plus=orderingcost_plus+Co;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%        for q-1        %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
  for i=1:(L+1)
      MIN(i)=EndInv(i)-1;
  end
```

```
IPminus(1)=MIN(1);

if IPminus(1)<=s

        IPminus(1)=s+q-1;

end


if (IPminus(1)==s+q-1 & D(1)>0)

        order3(1)=1;

end


for i=1:L

        IPminus(i+1)=IPminus(i)-D(i+1);


        if IPminus(i+1)<=s

                IPminus(i+1)=s+q-1;

        end


        if (IPminus(i+1)==s+q-1 & D(i+1)>0)

                order3(i+1)=1;

        end

end


delta=1;
```

```
for i=(L+2):day

    if (order1(i-L-1)==0 & order3(i-L-1)==0)

        MIN(i)=EndInv(i)-delta;

    elseif (order1(i-L-1)==1 & order3(i-L-1)==1)

        delta=1;

        MIN(i)=EndInv(i)-delta;

    elseif (order1(i-L-1)==1)

        MIN(i)=EndInv(i)-(s+q-IPminus(i-L-1));

        delta=s+q-IPminus(i-L-1);

    elseif (order3(i-L-1)==1)

        MIN(i)=EndInv(i)+q+s-1-IP(i-L-1);

        delta=-(q+s-1-IP(i-L-1));

    else

        %

    end


  IPminus(i)=IPminus(i-1)-D(i);


    if IPminus(i)<=s

        IPminus(i)=s+q-1;

    end
```

```
        if (IPminus(i)==s+q-1 & D(i)>0)
            order3(i)=1;
        end

    end
%%%%%%%%%%% COST FOR q-1 %%%%%%%%%%%%%%%%
    holdingcost_minus=0;
    shortagecost_minus=0;
    orderingcost_minus=0;

    for i=1:day
        if MIN(i)>0
            holdingcost_minus=holdingcost_minus+MIN(i)*Ch;
        else
            shortagecost_minus=shortagecost_minus-MIN(i)*Cs;
        end

        if order3(i)==1
            orderingcost_minus=orderingcost_minus+Co;
        end
    end
```

```
TOTALCOST_PLUS=holdingcost_plus+shortagecost_plus+orderingcost_plus;

TOTALCOST_MINUS=holdingcost_minus+shortagecost_minus+orderingcost_minus;


COST_DIF_PLUS=TOTALCOST_PLUS-TOTALCOST;

COST_DIF_MINUS=TOTALCOST_MINUS-TOTALCOST;


if (COST_DIF_PLUS>=0 & COST_DIF_MINUS>=0)

        q_array(loop+1)=q_array(loop);

elseif COST_DIF_PLUS>=0

        q_array(loop+1)=q_array(loop)-1;

elseif COST_DIF_MINUS>=0

        q_array(loop+1)=q_array(loop)+1;

else

        if COST_DIF_PLUS<COST_DIF_MINUS

                q_array(loop+1)=q_array(loop)+1;

        else

                q_array(loop+1)=q_array(loop)-1;

        end

 end



if loop>=2
```

```
            if (s_array(loop)==s_array(loop-1) & q_array(loop)==q_array(loop-1))
                finish=1;
                optimals=s_array(loop);
                optimalq=q_array(loop);
            end
        end

        loop=loop+1;

end

time=cputime-t
optimals
optimalS=optimals+optimalq
%loop
plot(s_array+q_array,'g')
hold
plot(s_array,'b')
plot(q_array,'r')
title('BLUE=s, RED=q, GREEN=S')
```

C.2 Initialization (init.m)

```
function init

        global BegInv

        global EndInv

        global IP

        global shortage

        global order_amount

        global WaitingOrders

        global hold_array

        global short_array

        global totalcost_array

        global order

        global maxs

        global S

        global s

        global q

        global day

        global holdingcost

        global shortagecost

        S=s+q;


        BegInv=zeros(1,day);

        EndInv=zeros(1,day);
```

```
IP=zeros(1,day);

shortage=zeros(1,day);

order_amount=zeros(1,day);

BegInv(1)=S;

WaitingOrders=0;

order=0;

hold_array=zeros(1,maxs);

short_array=zeros(1,maxs);

totalcost_array=zeros(1,maxs);


holdingcost=0;

shortagecost=0;
```

C.3 Creating the Path (create_path.m)

```
function create_path

    global day

    global loop

    global L

    global S

    global s

    global q

    global day
```

```
global EndInv

global BegInv

global D

global WaitingOrders

global IP

global order

global order_amount

global shortage

S=s+q;


for i=1:day

        EndInv(i)=BegInv(i)-D(i);

        IP(i)=EndInv(i)+WaitingOrders;

        if IP(i)<=s

                order=order+1;

                order_amount(i)=S-IP(i);

                WaitingOrders=WaitingOrders+order_amount(i);

                IP(i)=EndInv(i)+WaitingOrders;

        end


        if D(i)>BegInv(i)

                shortage(i)=D(i)-BegInv(i);
```

```
        end


        if i<=L
            BegInv(i+1)=EndInv(i);
        else
            BegInv(i+1)=EndInv(i)+order_amount(i-L);
            WaitingOrders=WaitingOrders-order_amount(i-L);
        end
    end
```

C.4 Calculating the Cost1 (calc_cost1.m)

```
function calc_cost1
    global day
    global EndInv
    global Ch
    global Cs
    global Co
    global order
    global TOTALCOST
    global holdingcost
    global shortagecost
    global orderingcost
```

```
global holdings
global shortages
global shortage
holdings=0;
shortages=0;
for i=1:day
    % holding cost
    if EndInv(i)>0
        holdings=holdings+EndInv(i);
    else
        % shortage cost
        shortages=shortages+shortage(i);
    end
end
holdingcost=holdings*Ch;
shortagecost=shortages*Cs;
orderingcost=order*Co;
TOTALCOST=holdingcost+shortagecost+orderingcost;
```

C.5 PA on "s" (der_s.m)

```
function der_s
    global EndInv
```

```
global day

global Ch

global Cs

global s

global orderingcost

global TOTALCOST

global opts

global holdings

global shortages

state1=0;

state2=0;

state3=0;

state4=0;

for i=1:day

        if EndInv(i)>=0

                state1=state1+1;

        else

                state2=state2+1;

        end

        if EndInv(i)>=1

                state3=state3+1;

        else
```

```
            state4=state4+1;
        end
end
holding_plus=holdings+state1;
hc_plus=holding_plus*Ch;
shortages_plus=shortages-state2;
sc_plus=shortages_plus*Cs;
totalcost_plus=hc_plus+sc_plus+orderingcost-TOTALCOST;

%%% for s-1
holding_minus=holdings-state3;
hc_minus=holding_minus*Ch;
shortages_minus=shortages+state4;
sc_minus=shortages_minus*Cs;
totalcost_minus=hc_minus+sc_minus+orderingcost-TOTALCOST;
if (totalcost_plus>0 & totalcost_minus>0)
        opts=s;
elseif (totalcost_plus<=0)
        opts=s+1;
elseif (totalcost_minus<=0)
        opts=s-1;
else
```

```
        if (totalcost_plus<totalcost_minus)
            opts=s+1;
        else
            opts=s-1;
        end
    end
    s=opts;
```

C.6 Preparation for PA on "q" (der_q_prep.m)

```
function der_q_prep
    global day
    global EndInv
    global BegInv
    global D
    global IP
    global s
    global order
    global order_amount
    global S
    global WaitingOrders
    global shortage
    global L
```

```
for i=1:day
        EndInv(i)=BegInv(i)-D(i);
        IP(i)=EndInv(i)+WaitingOrders;
        if IP(i)<=s
                order=order+1;
                order_amount(i)=S-IP(i);
                WaitingOrders=WaitingOrders+order_amount(i);
                IP(i)=EndInv(i)+WaitingOrders;
        end

        if D(i)>BegInv(i)
                shortage(i)=D(i)-BegInv(i);
        end

        if i<=L
                BegInv(i+1)=EndInv(i);
        else
                BegInv(i+1)=EndInv(i)+order_amount(i-L);
                WaitingOrders=WaitingOrders-order_amount(i-L);
        end
end
```

C.7 Calculating the Cost2 (calc_cost.m)

```
function calc_cost

        global day

        global EndInv

        global Ch

        global Cs

        global Co

        global order

        global TOTALCOST

        global holdingcost

        global shortagecost

        global orderingcost

        for i=1:day

                % holding cost

                if EndInv(i)>0

                        holdingcost=holdingcost+EndInv(i)*Ch;

                else

                        % shortage cost

                        shortagecost=shortagecost-EndInv(i)*Cs;

                end

        end

        orderingcost=order*Co;
```

```
TOTALCOST=holdingcost+shortagecost+orderingcost;
```

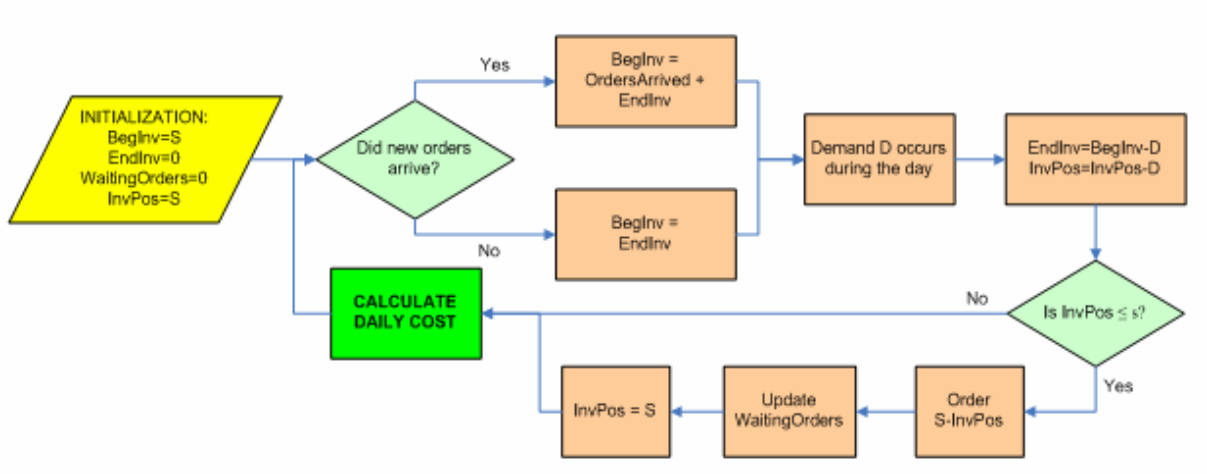
Appendix D
Ehrhardt's Heuristic Code

```
clc

clear

t=cputime;

load demNOR_5_2.mat

delta_day=30;

period=30;

dem_per=15;

h=1;

p=9;

K=64;

L=0;

for i=1:period



for j=1:(delta_day*i)

      demand(j)=D(j);

end



 day=min(dem_per,i)*delta_day;

 difference=(delta_day*i)-day;
```

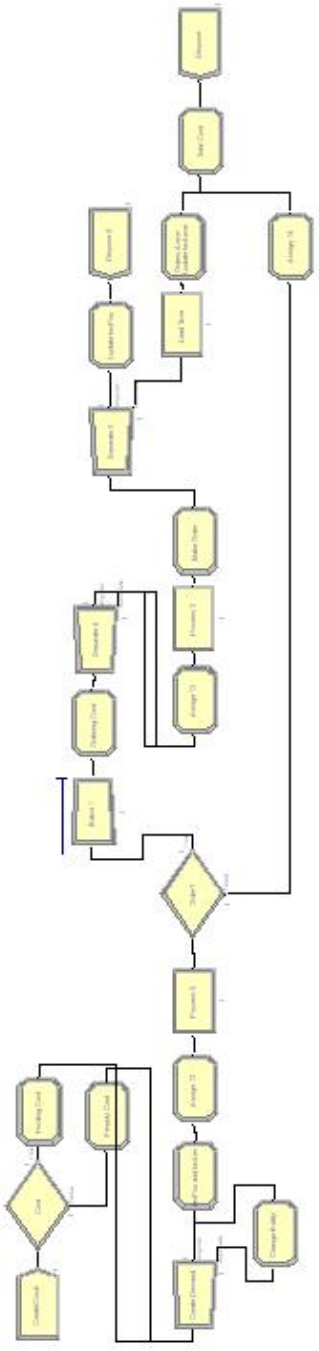
```
for j=1:day
    demand(j)=D(j+difference);
end

meannn=mean(demand);
varia=var(demand);
sigmaL=sqrt((L+1)*varia);
meanL=(L+1)*meannn;
Dopt=1.3*(meannn^0.494)*((K/h)^0.506)*(1+sigmaL^2/meannn^2)^0.116;
z=(Dopt/(sigmaL*p/h))^0.5;
sopt=0.973*meanL+sigmaL*(0.183/z+1.063-2.192*z);
Dopt/meannn;
sopt;
Sopt=sopt+Dopt;
opt_s(i)=round(sopt);
opt_S(i)=round(Sopt);
end

figure
plot(opt_s,'gx')
hold
```

```
plot(opt_S,'rx')
```

Appendix E
Flowchart

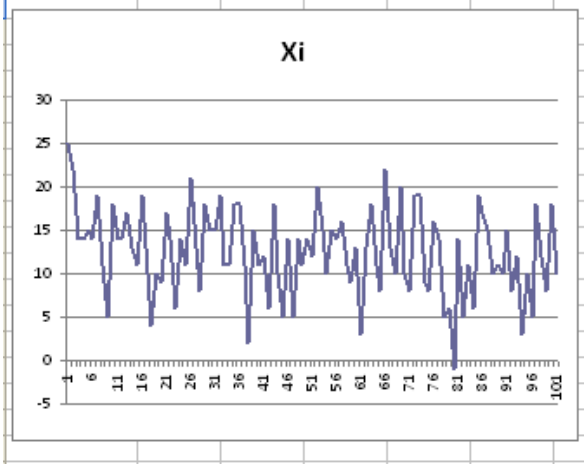


Appendix F
ARENA Simulation Model



Appendix G
EXCEL Simulation Model

L	1					Period i	D_i	X_i	Y_i	Q_i	O_i	R_i	J_i
K	10					0		25		25	0		
c	0					1	3	22	22	0	0	0	22
h	1					2	8	14	14	11	11	0	24
p	100					3	0	14	25	0	11	0	14
						4	10	15	15	10	10	11	25
s	15					5	1	14	24	0	10	0	14
S	25					6	5	19	19	0	0	10	19
						7	8	11	11	14	14	0	21
Total Cost	17697					8	6	5	19	0	14	0	5
Average Cost	17.697					9	1	18	18	0	0	14	18
						10	4	14	14	11	11	0	24
#Shortage Days	14					11	0	14	25	0	11	0	14
Prob. Shortage	0.014					12	8	17	17	0	0	11	17
						13	4	13	13	12	12	0	23
						14	2	11	23	0	12	0	11
						15	4	19	19	0	0	12	19
						16	6	13	13	12	12	0	23
						17	9	4	16	0	12	0	4
						18	6	10	10	15	15	12	20
						19	1	9	24	0	15	0	9
						20	7	17	17	0	0	15	17
						21	3	14	14	11	11	0	24
						22	8	6	17	0	11	0	6
						23	3	14	14	11	11	11	24
						24	3	11	22	0	11	0	11
						25	1	21	21	0	0	11	21
						26	8	13	13	12	12	0	23
						27	5	8	20	0	12	0	8
						28	2	18	18	0	0	12	18
						29	3	15	15	10	10	0	25
						30	0	15	25	0	10	0	15
						31	6	19	19	0	0	10	19



EXCEL Simulation Model

Appendix H
MATLAB Simulation Model

```
clc

clear

day=20000;

%%% Creating Demand

for i=1:day

    D(i)=random('Normal',5,2);

    D(i)=round(D(i));

    if D(i)<0

        D(i)=0;

    end

end


L=2;

Ch=1;

Co=10;

Cs=100;

maxs=90;

maxS=100;


%%% Initializations

for i=1:maxs
```

```
    for j=1:maxS
        holdingcost(i,j)=0;
        shortagecost(i,j)=0;
        orderingcost(i,j)=0;
    end
end

for s=1:maxs
    for S=s+1:maxS
        BegInv=zeros(1,day);
        EndInv=zeros(1,day);
        IP=zeros(1,day);
        shortage=zeros(1,day);
        order_amount=zeros(1,day);
        WaitingOrders=0;
        BegInv(1)=S;
        order=0;
        for i=1:day
            EndInv(i)=BegInv(i)-D(i);
            IP(i)=EndInv(i)+WaitingOrders;
            if IP(i)<=s
                order=order+1;
                order_amount(i)=S-IP(i);
```

```
            WaitingOrders=WaitingOrders+order_amount(i);
            IP(i)=EndInv(i)+WaitingOrders;
        end
        if D(i)>BegInv(i)
            shortage(i)=D(i)-BegInv(i);
        end
        if i<=L
            BegInv(i+1)=EndInv(i);
        else
            BegInv(i+1)=EndInv(i)+order_amount(i-L);
            WaitingOrders=WaitingOrders-order_amount(i-L);
        end
    end
%%% cost calculations
    for i=1:day
    % holding cost
        if EndInv(i)>0
            holdingcost(s,S)=holdingcost(s,S)+EndInv(i)*Ch;
        end
    % shortage cost
        shortagecost(s,S)=shortagecost(s,S)+shortage(i)*Cs;
    end
```

```
            % ordering cost
                orderingcost(s,S)=order*Co;
                totalcost(s,S)=holdingcost(s,S)+shortagecost(s,S)+orderingcost(s,S);
                averagecost=totalcost/day;
        end
end
%%% Finding the minimum average cost
for i=1:maxs
        for j=1:maxS
            if averagecost(i,j)==0     % where S<s
                new_tot(i,j)=9999999999; % very large number
            else
                new_tot(i,j)=averagecost(i,j);
            end
        end
end
min_av_cost=min(min(new_tot));
%%% Finding the optimal (s,S) pair
for i=1:maxs
        for j=1:maxS
            if new_tot(i,j)==min_av_cost
                opts=i;
```

```
                optS=j;

            end

        end

end

opts

optS
```

Appendix I
GAMS Code

```
$include 'D:\0607tez\partSON\demand.txt';

Scalars

 h holding cost /1/

 p shortage cost /100/

 k order cost /10/

 eval /0.000000000001/

 Mval big M /100/;

Variables

 j(i) cost of period i

 x(i) inventory level at period i

 y(i) inventory position at period i

 o(i) outstanding orders at period i

 q(i) order quantity at period i

 r(i) received quantity at period i

 Inda(i) order indicator at period i

 Indb(i) secondary indicator at period i

 z total cost

 small reorder point

 Slarge order-up-to level
```

```
positive variables j,o,q,r;

binary variables Inda,Indb;

integer variables small,Slarge;

Equations

 cost define objective function

 percosta(i) first equation for period cost

 percostb(i) second equation for period cost

 initx first period inventory level

 xdyn(i) inventory level dynamics

 inity first period inventory position

 ydyn(i) inventory position dynamics

 inito first period outstanding order dynamics

 odyn(i) outstanding order dynamics

 ordquana(i) first inquality for order quantity dynamics

 ordquanb(i) second inquality for order quantity dynamics

 ordquanc(i) third inquality for order quantity dynamics

 con s and S relation

 con2

 initra

 initrb

 rdyn(i)

 indadyn(i) Inda variable dynamics
```

```
 indbdyn(i) Indb variable dynamics

 indabrel(i) Inda and Indb relation;

cost .. z=e=sum(i,j(i));

percosta(i).. j(i)=g=k*Inda(i)+h*x(i);

percostb(i).. j(i)=g=k*Inda(i)-p*x(i);

initx.. x('1')=e=Slarge-d('1');

xdyn(i)$(ord(i)>1).. x(i)=e=x(i-1)+r(i)-d(i);

inity.. y('1')=e=x('1');

ydyn(i)$(ord(i)>1).. y(i)=e=x(i-1)+o(i-1)-d(i);

inito.. o('1')=e=q('1');

odyn(i)$(ord(i)>1).. o(i)=e=o(i-1)+q(i)-r(i);

ordquana(i).. q(i)=g=(Slarge-y(i))-Mval*(1-Inda(i));

ordquanb(i).. q(i)=l=(Slarge-y(i))+Mval*(1-Inda(i));

ordquanc(i).. q(i)=l=Mval*Inda(i);

initra.. r('1')=e=0;

initrb.. r('2')=e=0;

rdyn(i)$(ord(i)>2).. r(i)=e=q(i-2);

con.. small =l= Slarge;

con2.. Slarge =l= 65;

indadyn(i).. Mval*Inda(i)=g=small-y(i)+1;

indbdyn(i).. Mval*Indb(i)=g=y(i)-small;

indabrel(i).. Inda(i)=l=1-Indb(i);
```

```
Model sSinventory /all/;

Solve sSinventory using mip minimizing z ;

option small:3:0:1;

option Slarge:3:0:1;

display Inda.l,x.l,y.l,q.l,o.l,r.l,j.l;

Display small.l, Slarge.l ;
```

Appendix J
Outputs of Numerical Examples

I.1. Poisson Distribution (10)

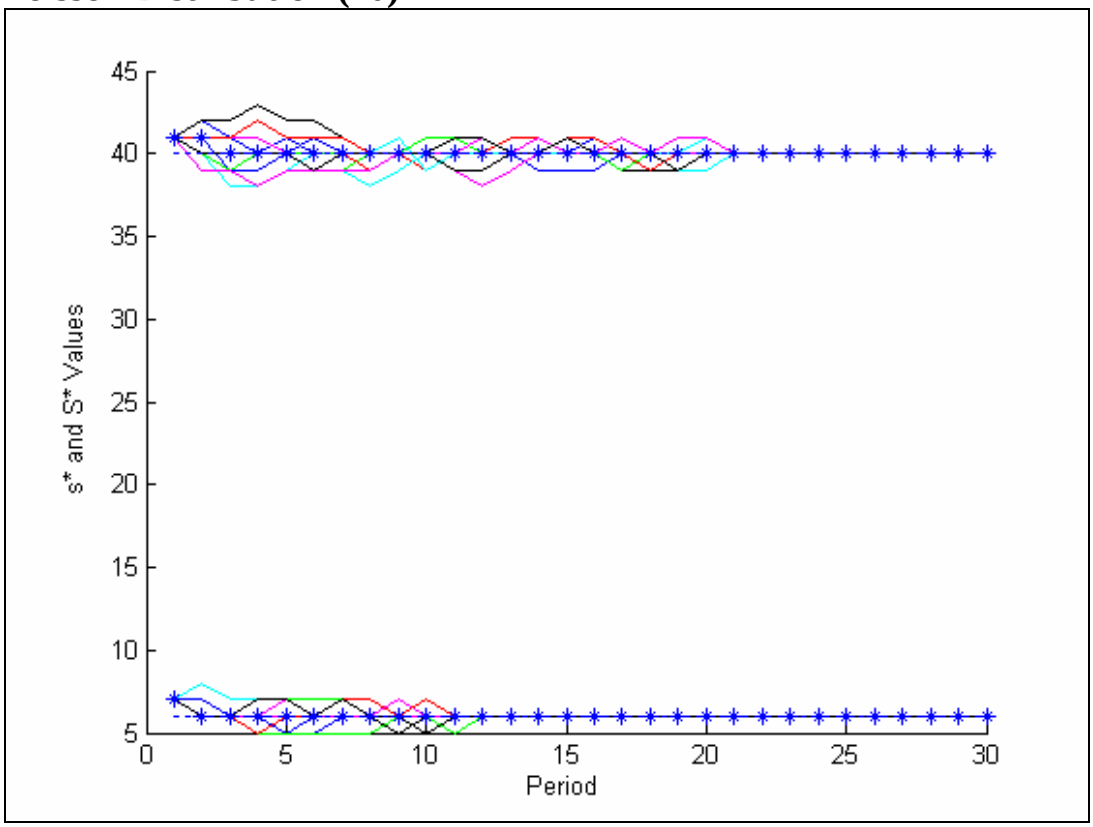


Fig. J.1. (s*,S*) Graph for **PA Alg.** POI(10), γ=ALL HISTORY

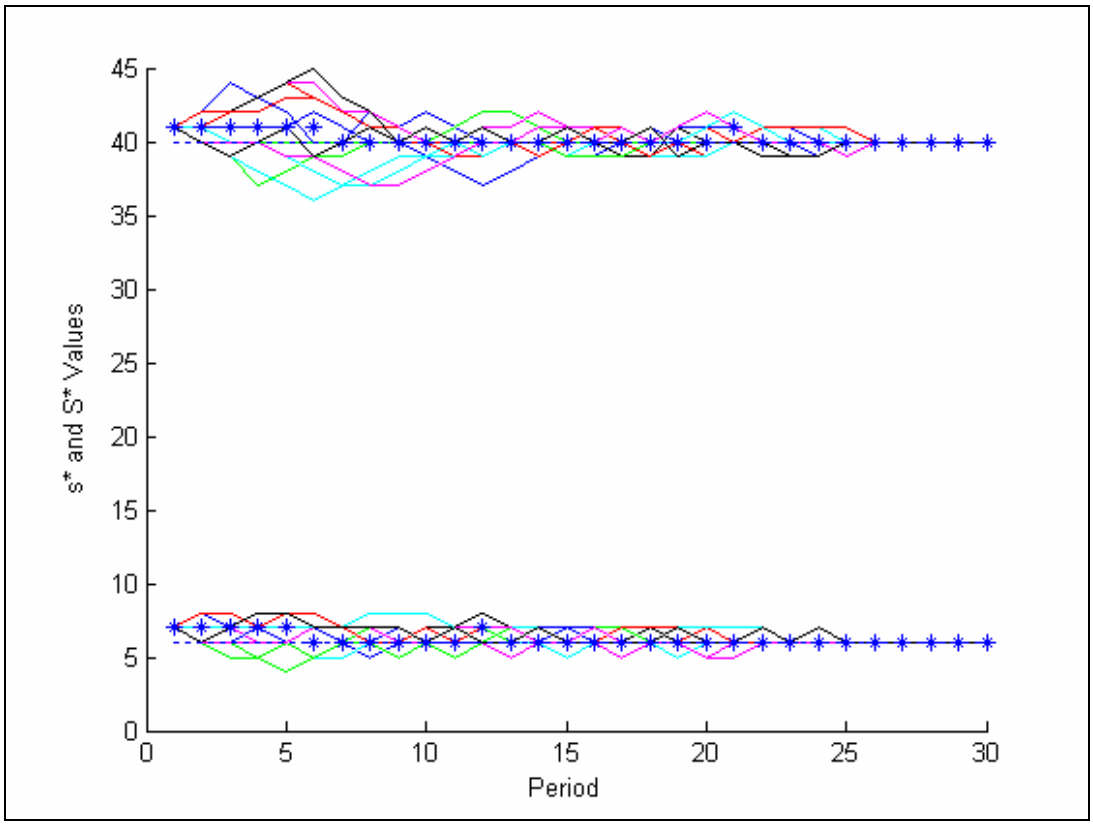


Fig. J.2. (s*,S*) Graph for **PA Alg.** POI(10), γ=3

Appendix J Outputs of Numerical Examples

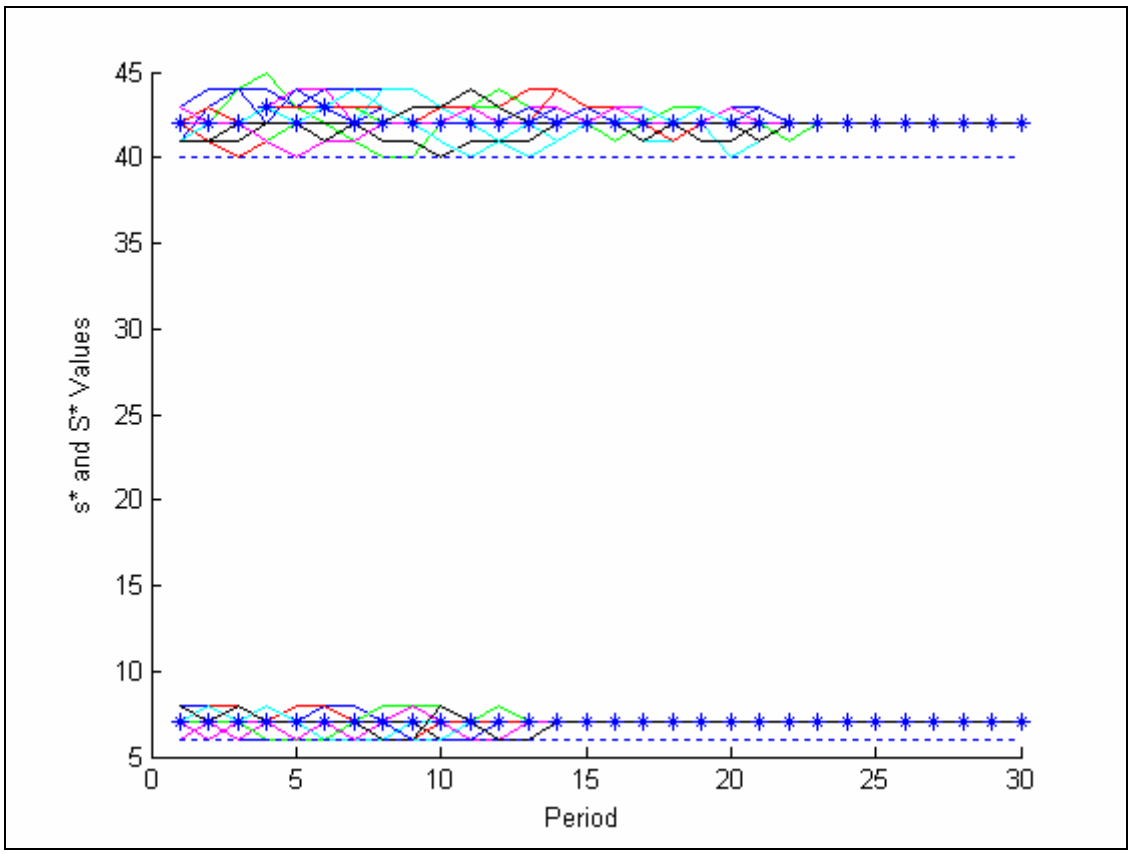


Fig. J.3. (s*,S*) Graph for **EHR Alg.** POI(10), γ=ALL HISTORY

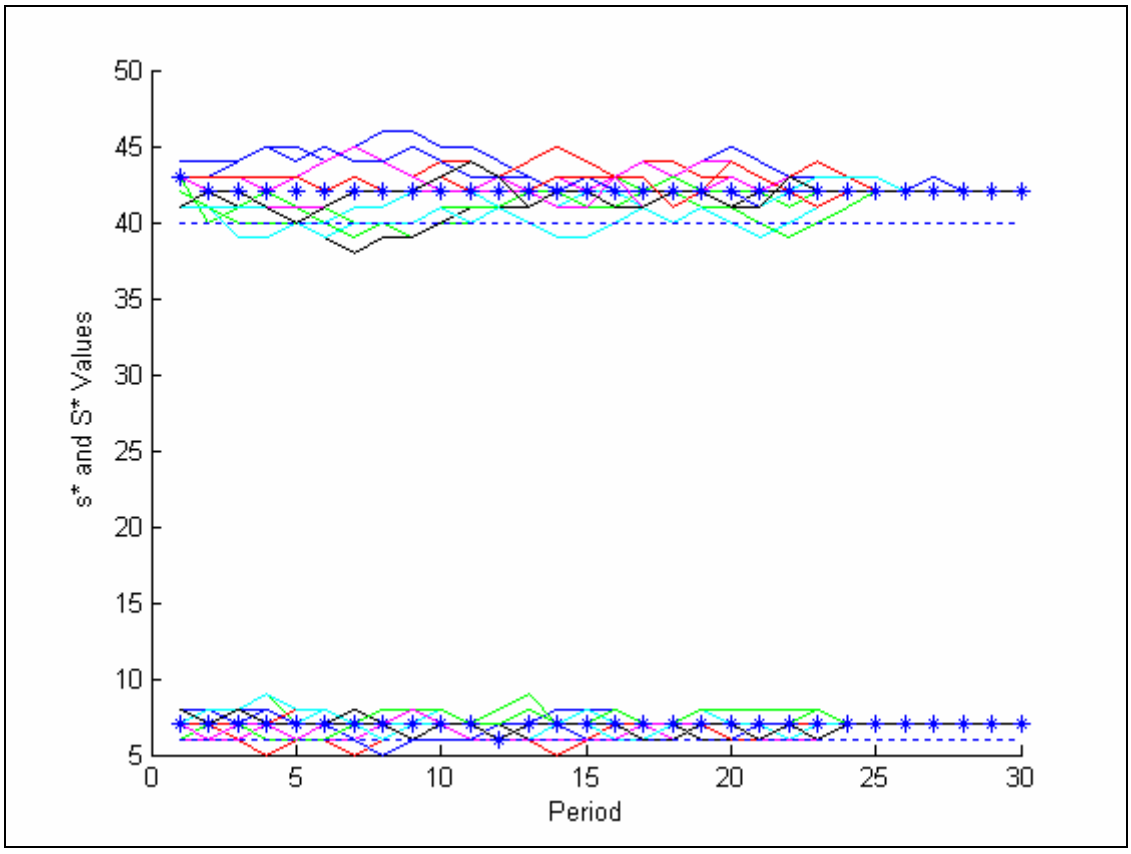


Fig. J.4. (s*,S*) Graph for **EHR Alg.** POI(10), γ=3

Appendix J Outputs of Numerical Examples

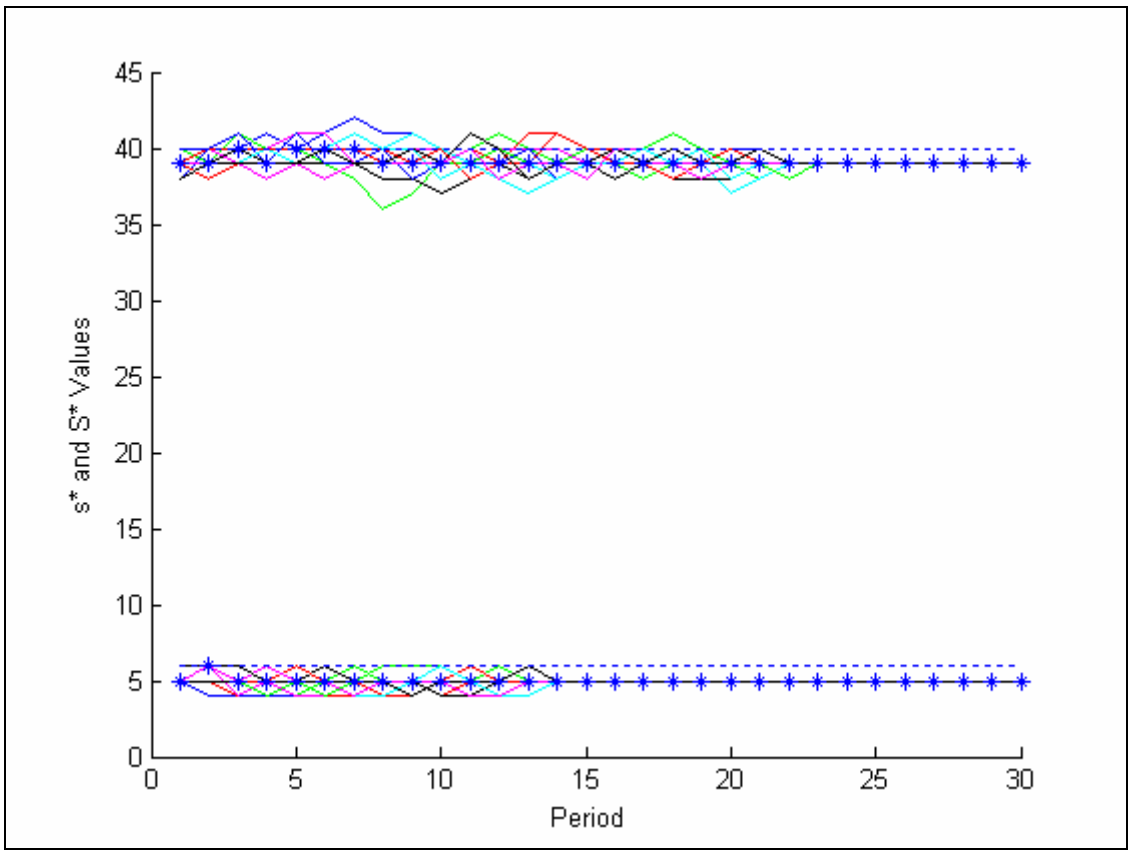


Fig. J.5. (s*,S*) Graph for **RIP Alg.** POI(10), γ=ALL HISTORY

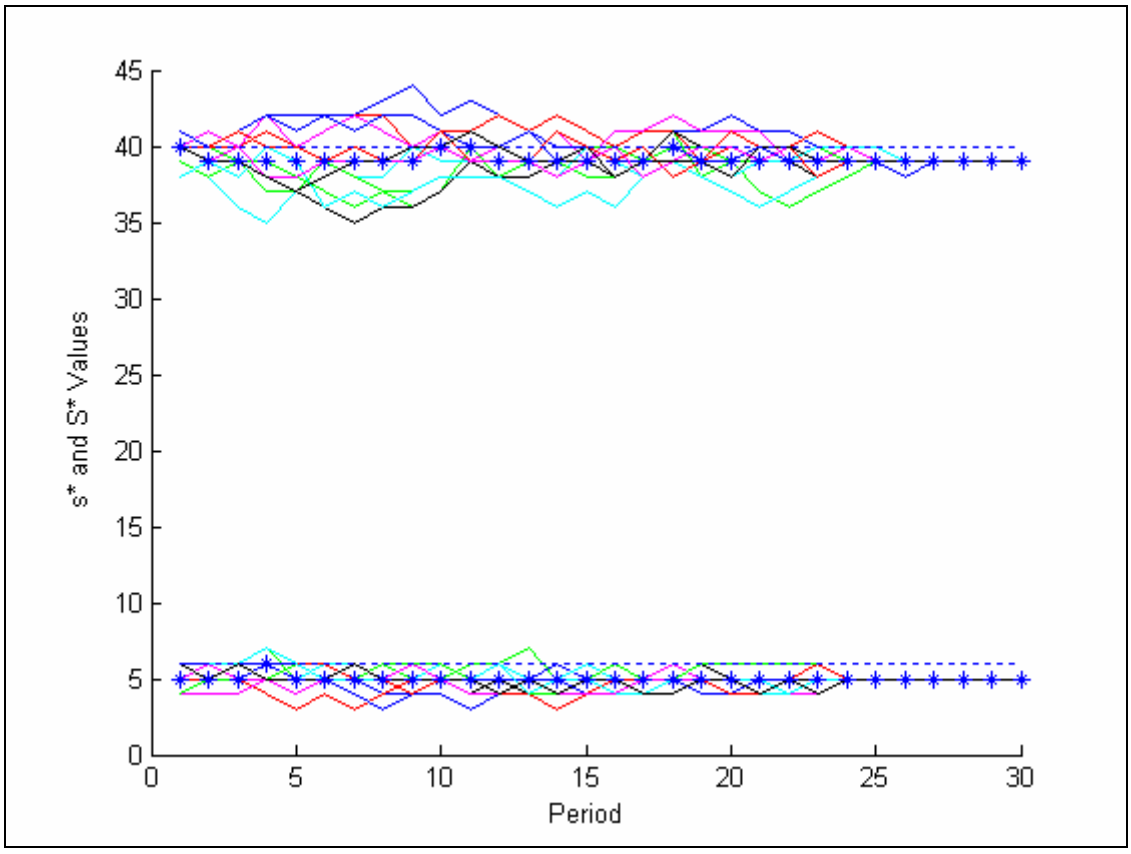


Fig. J.6. (s*,S*) Graph for **RIP Alg.** POI(10), γ=3

Appendix J Outputs of Numerical Examples

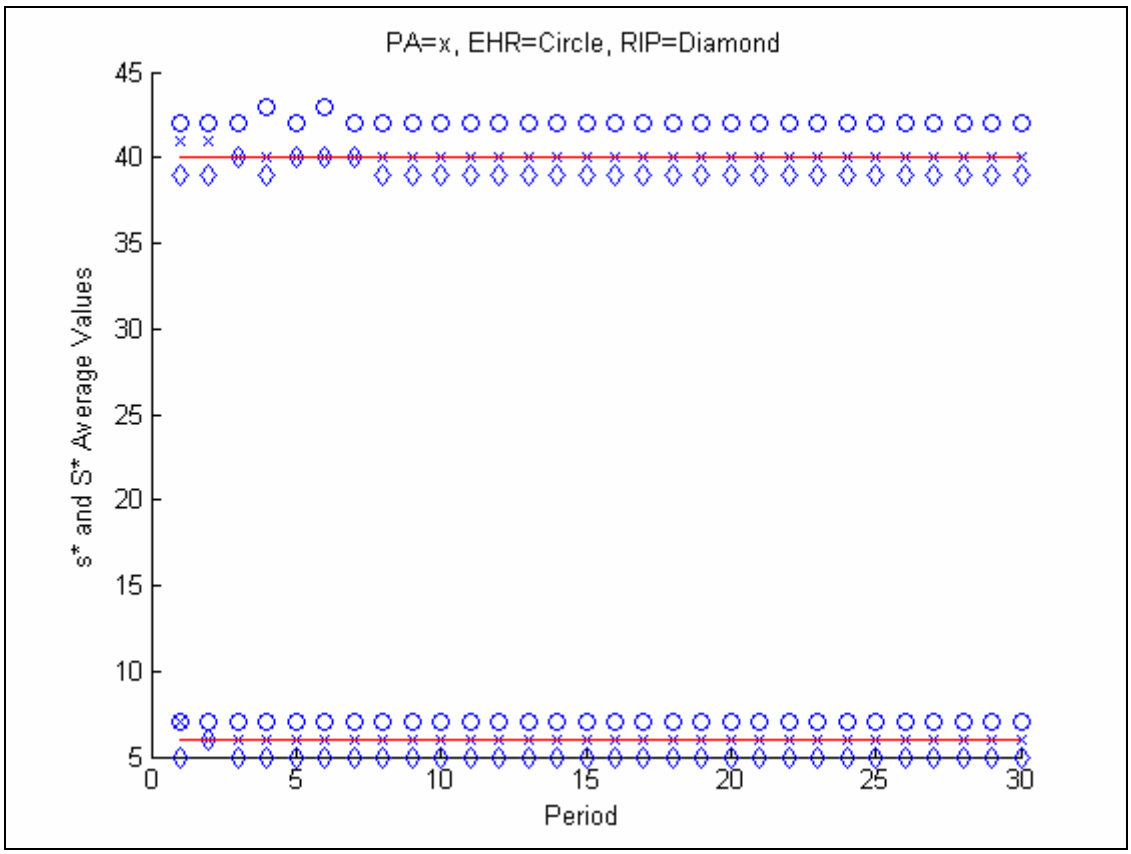


Fig. J.7. Comparison of (s*,S*) Graphs for POI(10), γ=ALL HISTORY

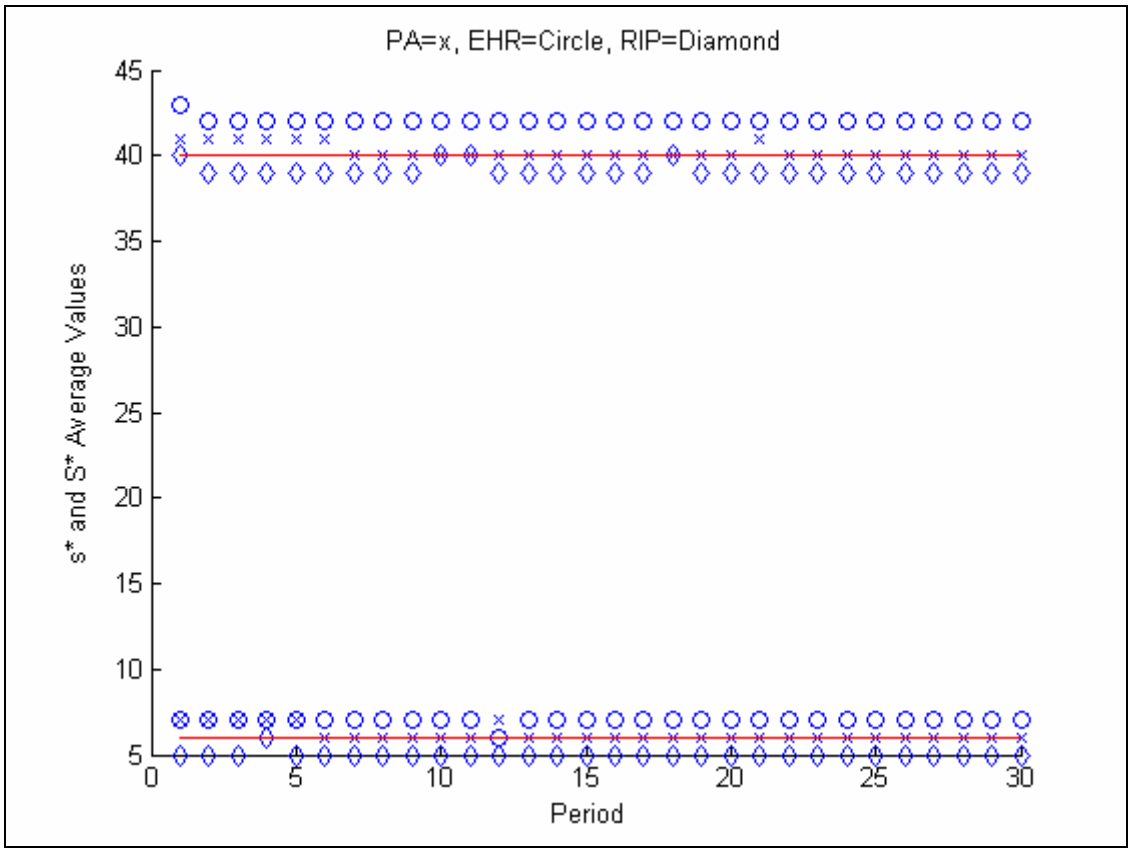


Fig. J.8. Comparison of (s*,S*) Graphs for POI(10), γ=3

Appendix J Outputs of Numerical Examples

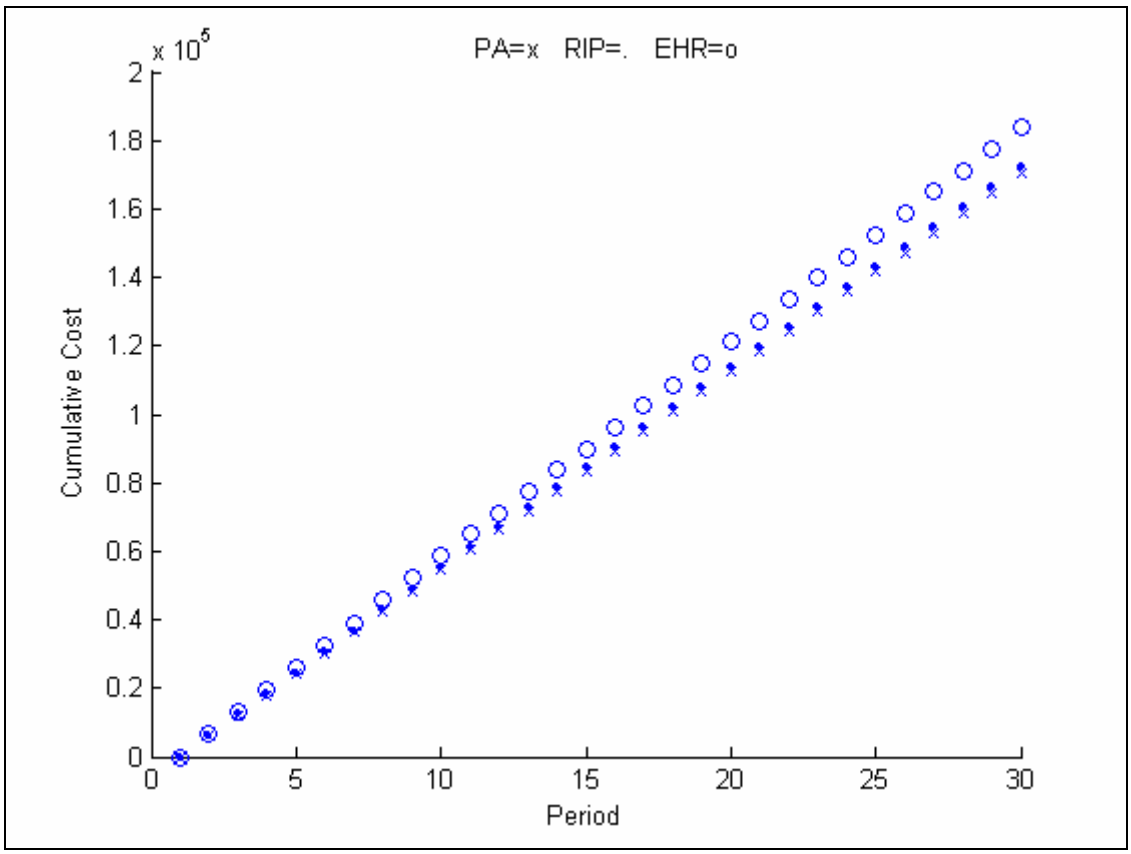


Fig. J.9. Comparison of Cost Graphs for POI(10), γ=ALL HISTORY

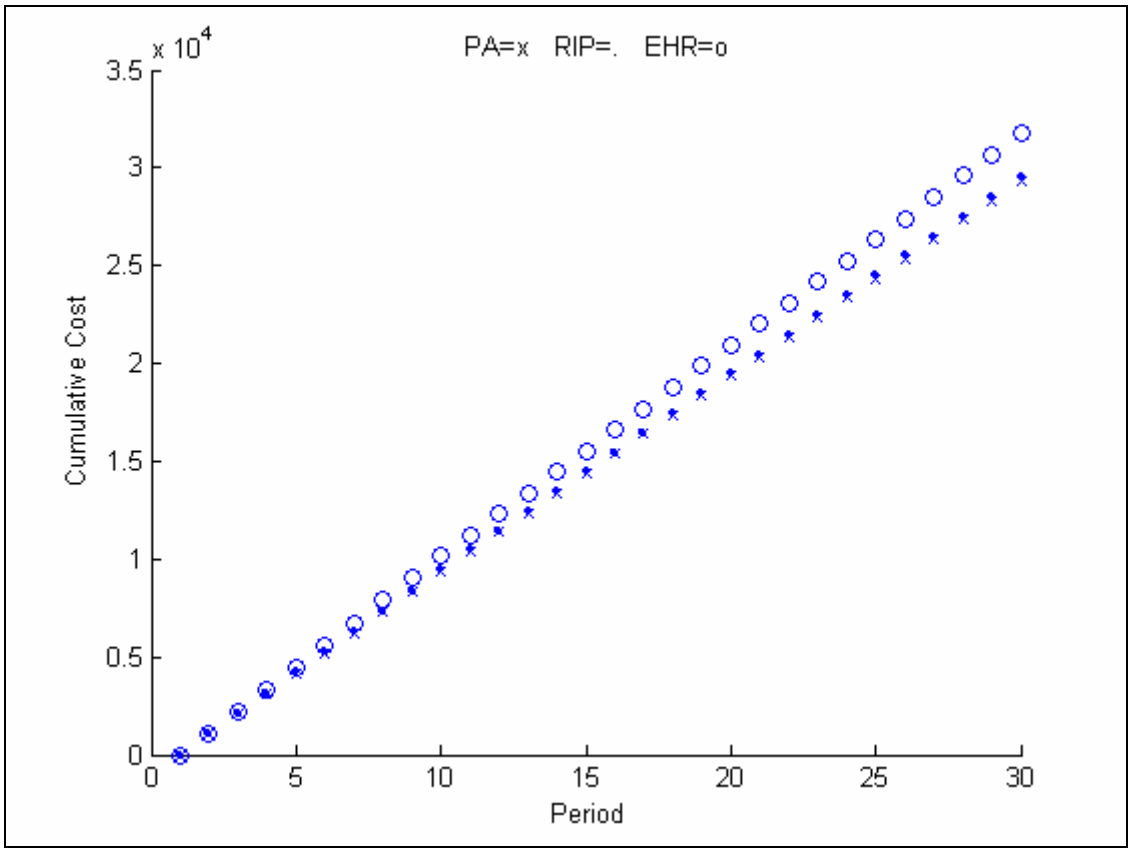


Fig. J.10. Comparison of Cost Graphs for POI(10), γ=3

I.2. Poisson Distribution (25)

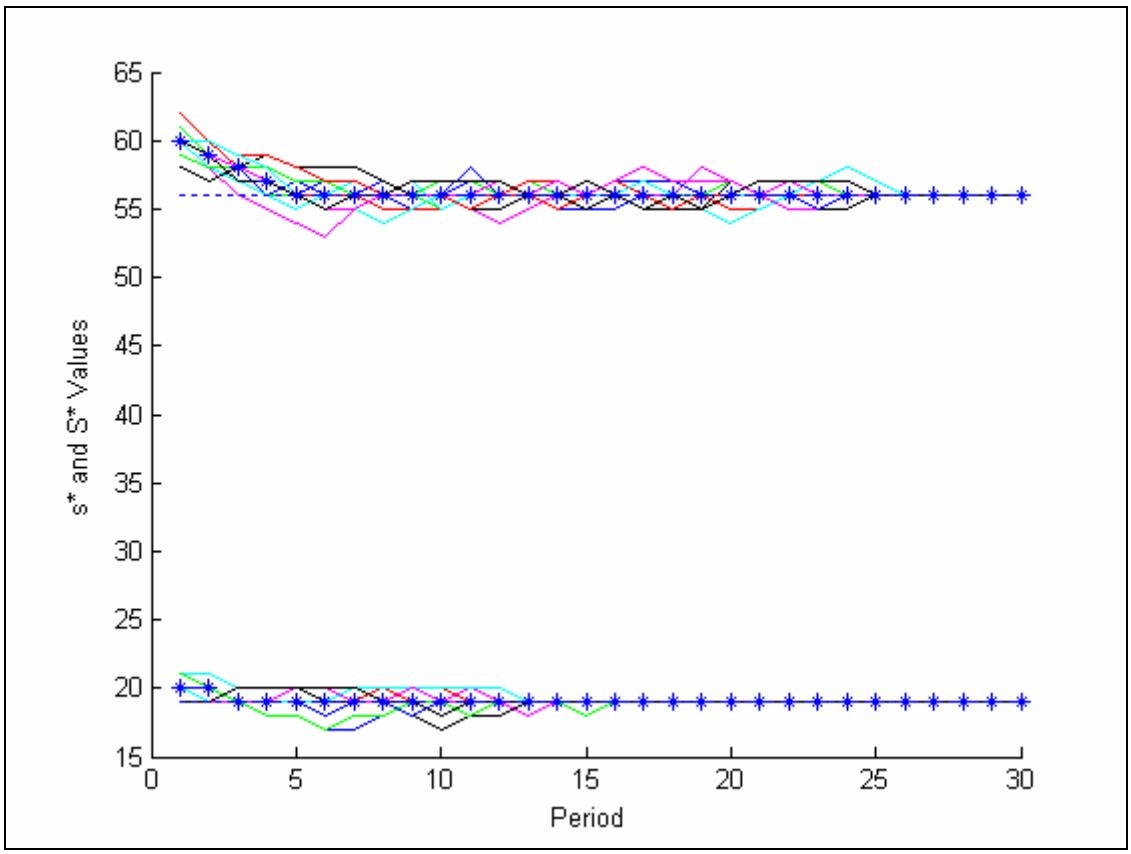


Fig. J.11. (s*,S*) Graph for **PA Alg.** POI(25), γ=ALL HISTORY

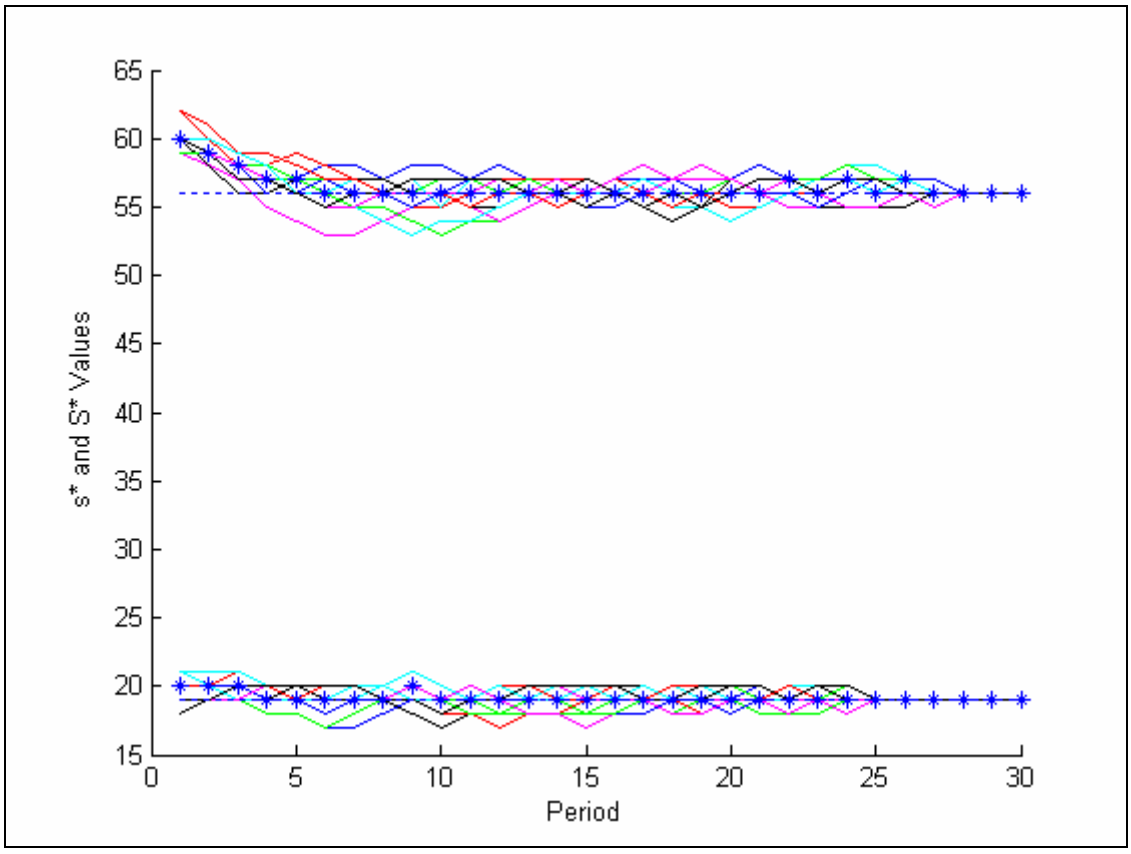


Fig. J.12. (s*,S*) Graph for **PA Alg.** POI(25), γ=3

Appendix J Outputs of Numerical Examples

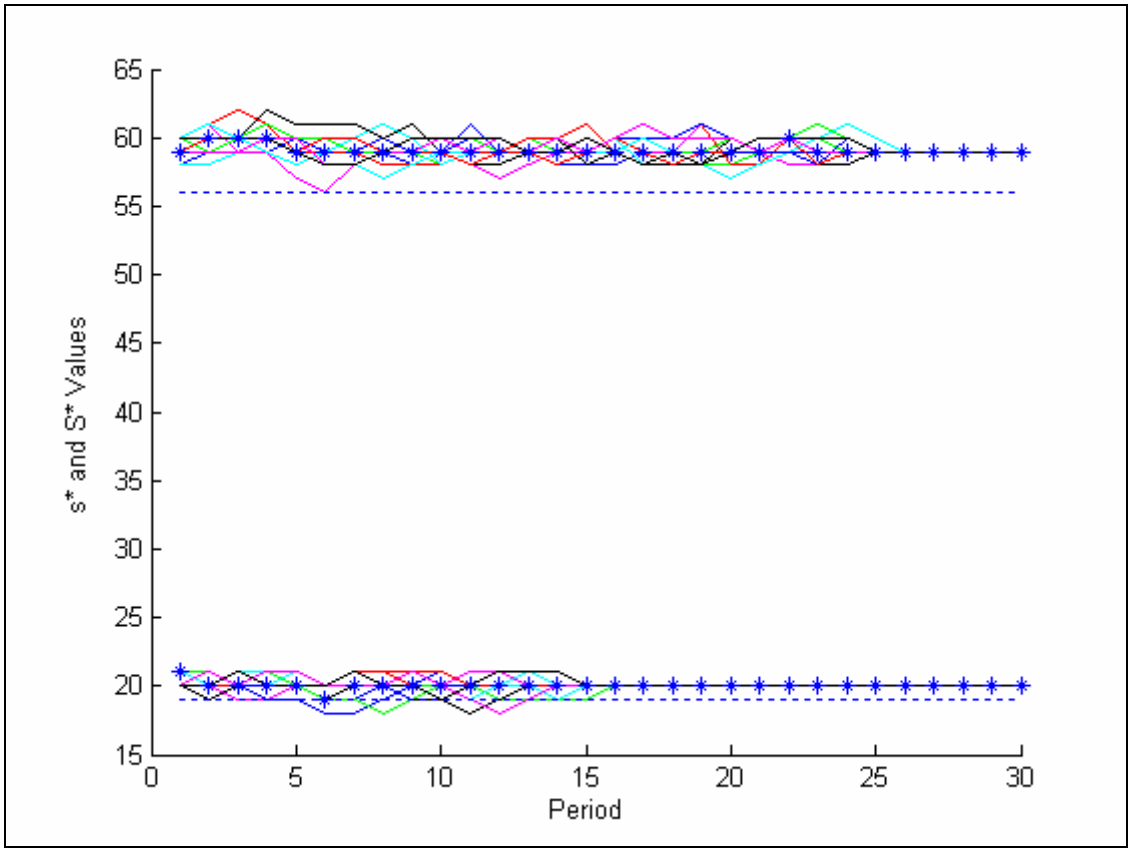


Fig. J.13. (s*,S*) Graph for **EHR Alg.** POI(25), γ=ALL HISTORY

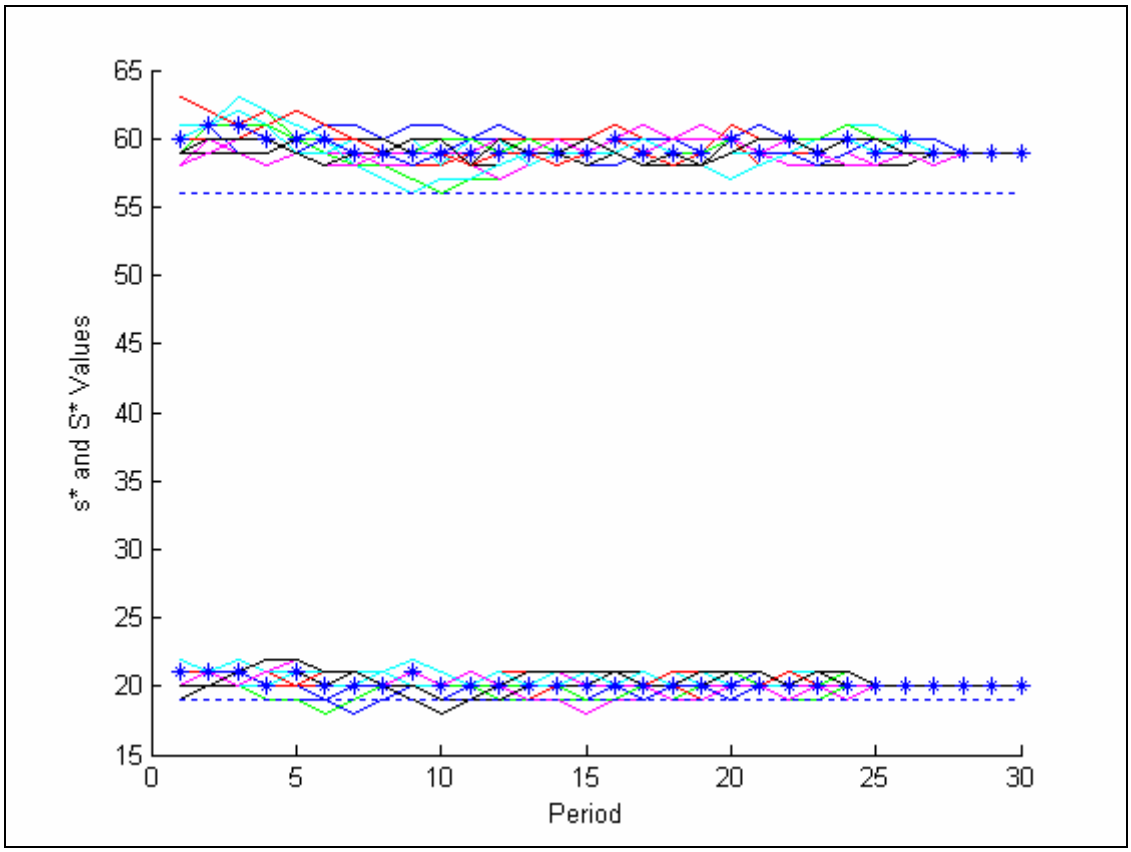


Fig. J.14. (s*,S*) Graph for **EHR Alg.** POI(25), γ=3

Appendix J Outputs of Numerical Examples

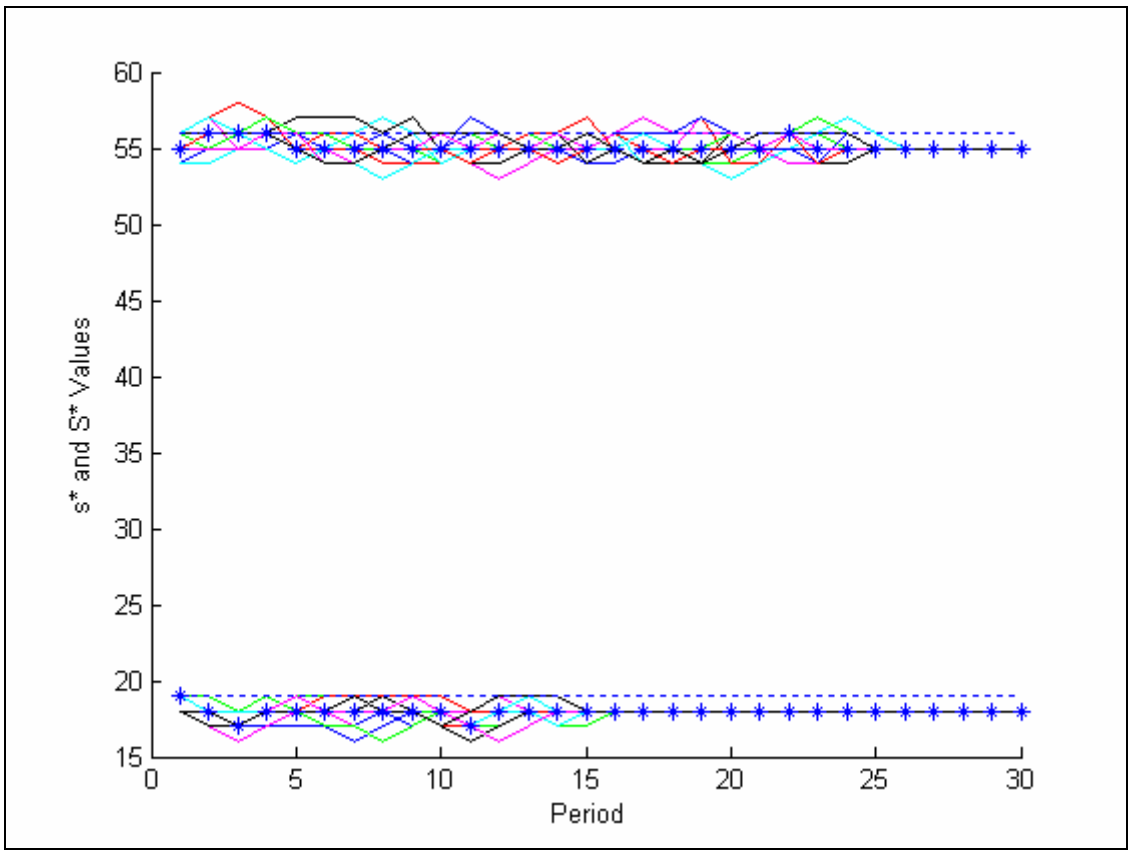


Fig. J.15. (s*,S*) Graph for **RIP Alg.** POI(25), γ=ALL HISTORY

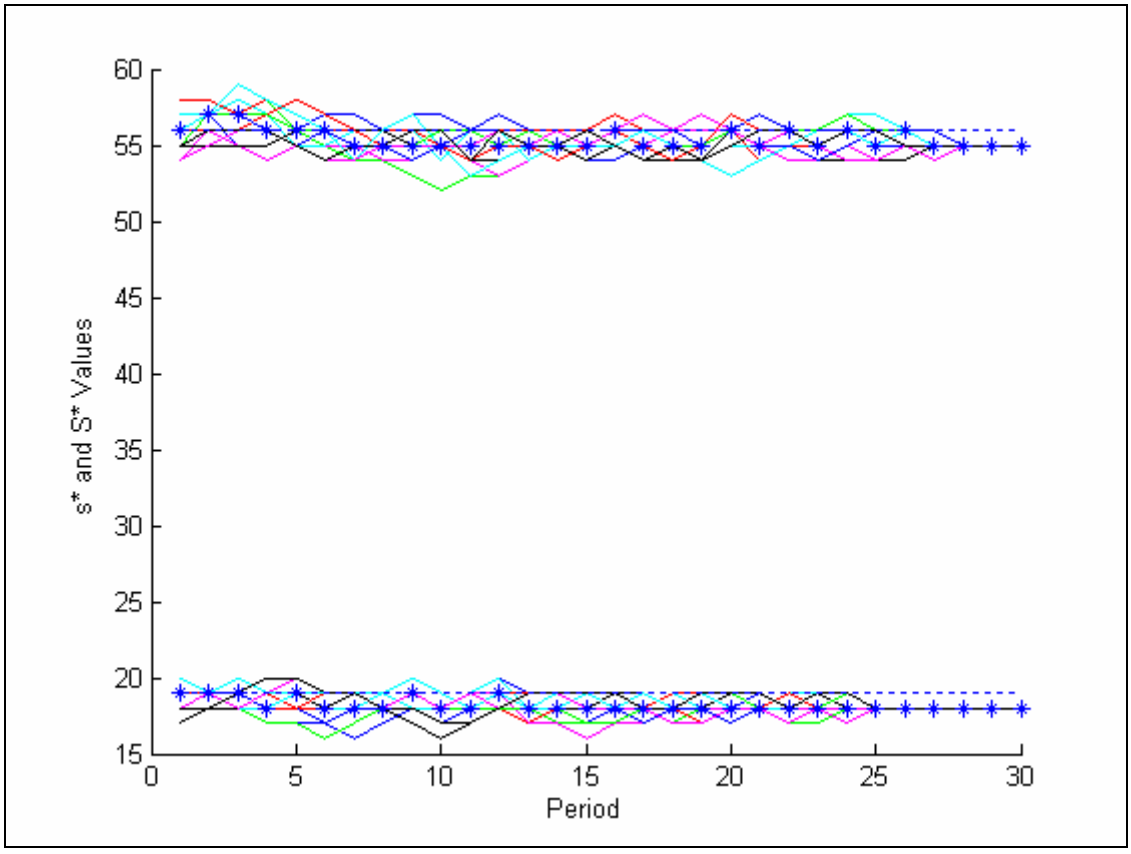


Fig. J.16. (s*,S*) Graph for **RIP Alg.** POI(25), γ=3

Appendix J Outputs of Numerical Examples

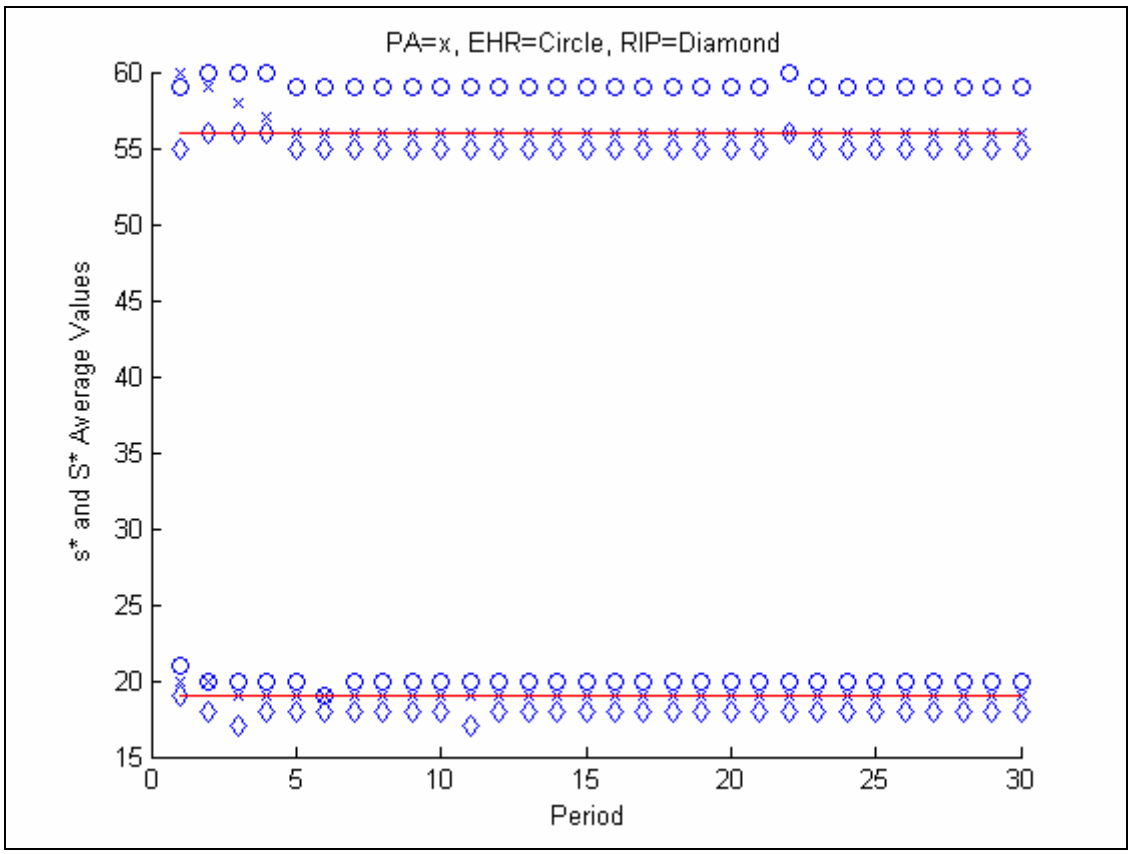


Fig. J.17. Comparison of (s*,S*) Graphs for POI(25), γ=ALL HISTORY

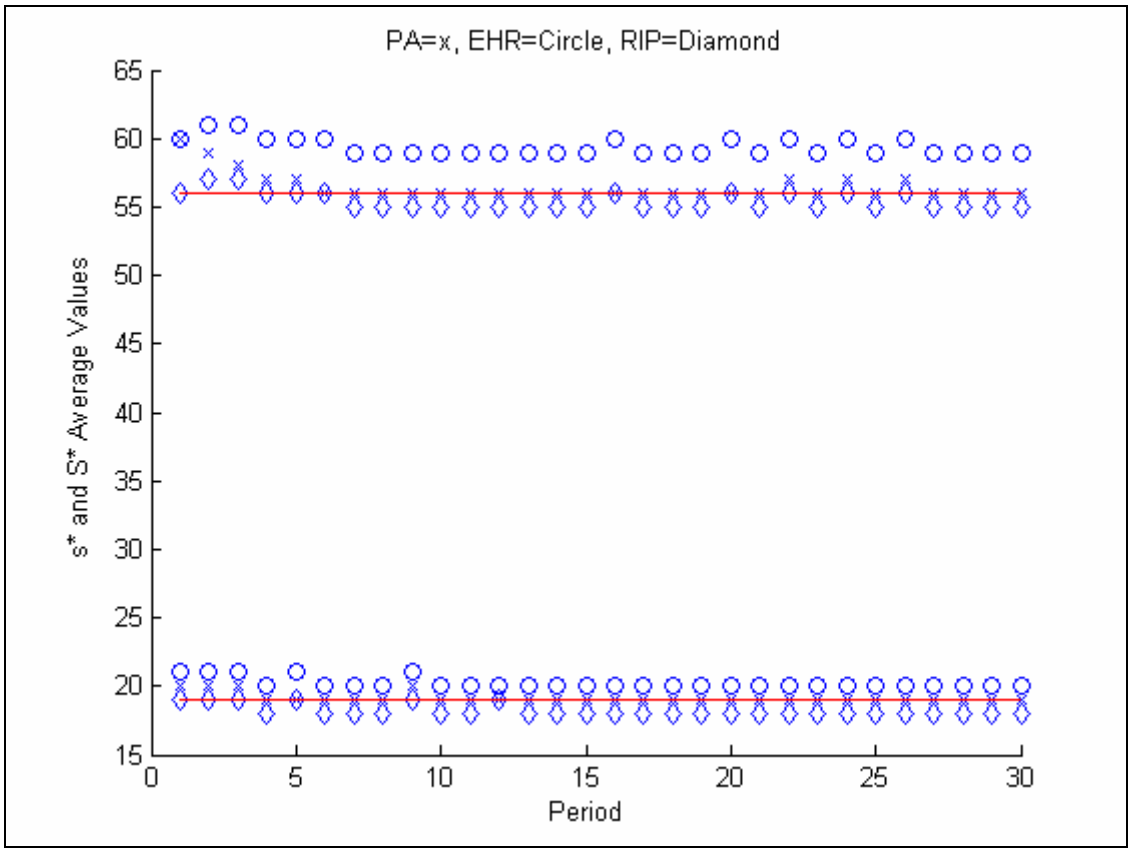


Fig. J.18. Comparison of (s*,S*) Graphs for POI(25), γ=3

Appendix J Outputs of Numerical Examples

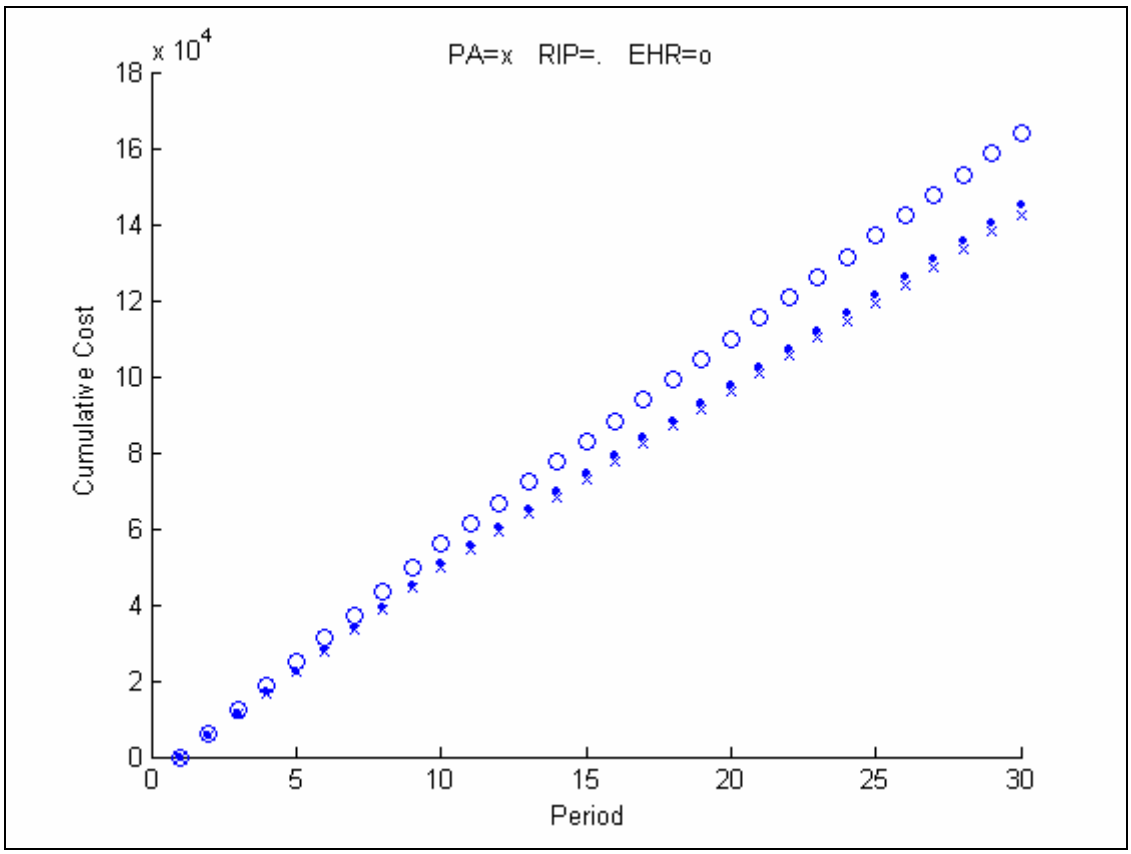


Fig. J.19. Comparison of Cost Graphs for POI(25), γ=ALL HISTORY

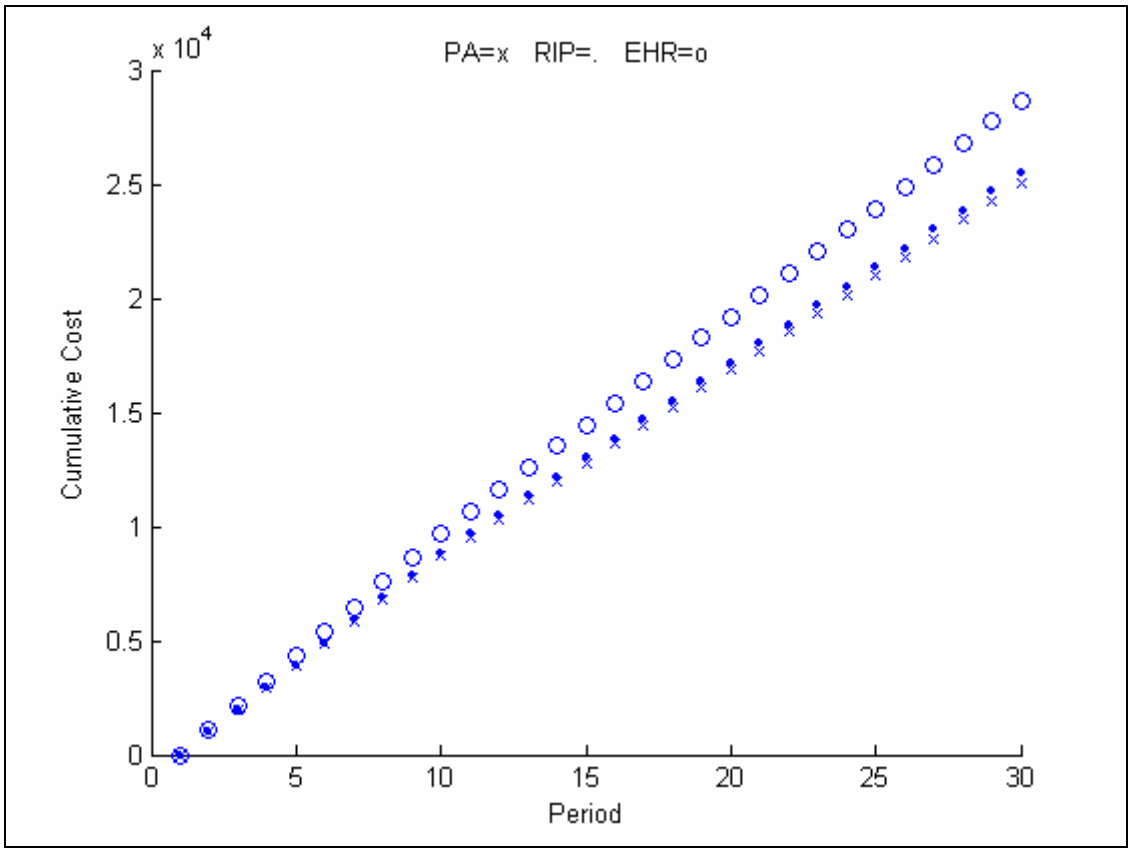


Fig. J.20. Comparison of Cost Graphs for POI(25), γ=3

I.3. Uniform Distribution (0,10)

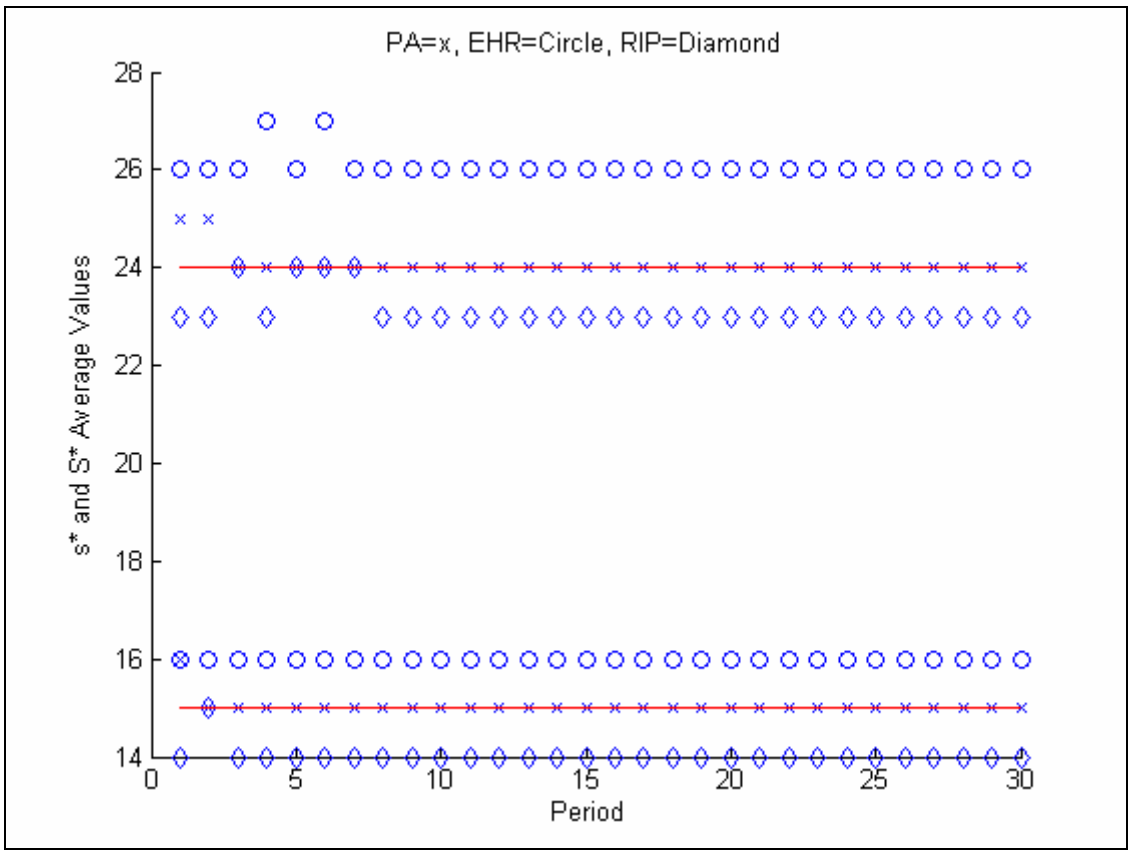


Fig. J.21. Comparison of (s*,S*) Graphs for UNI(0,10)

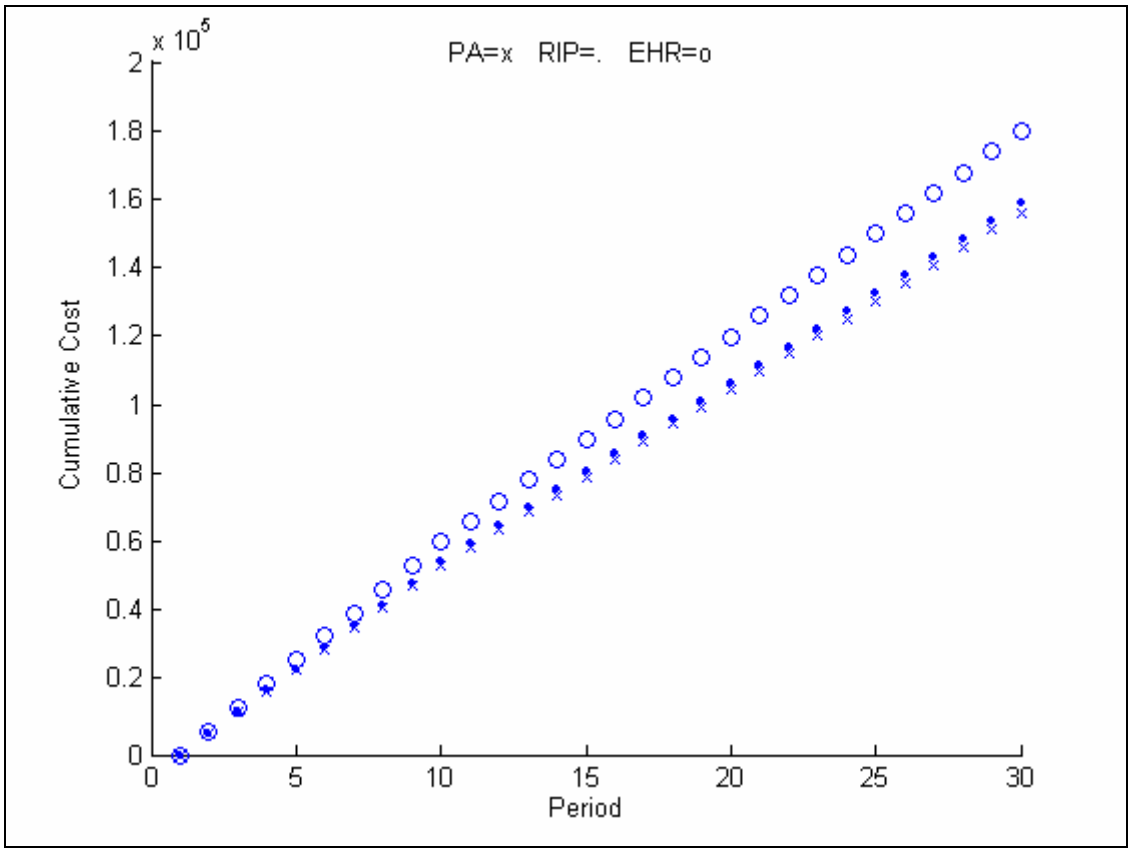


Fig. J.22. Comparison of Cost Graphs for UNI(0,10)

I.4. Normal Distribution (5,1)

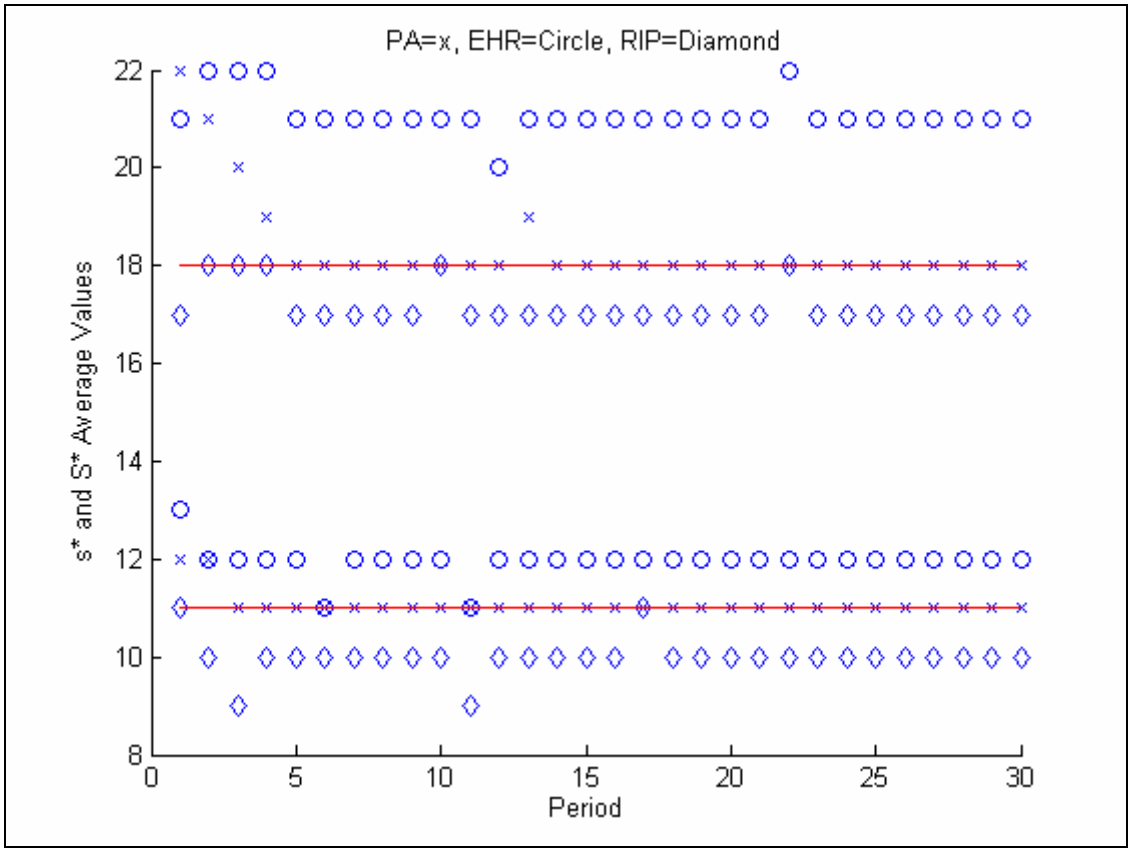


Fig. J.23. Comparison of (s*,S*) Graphs for NOR(5,1)

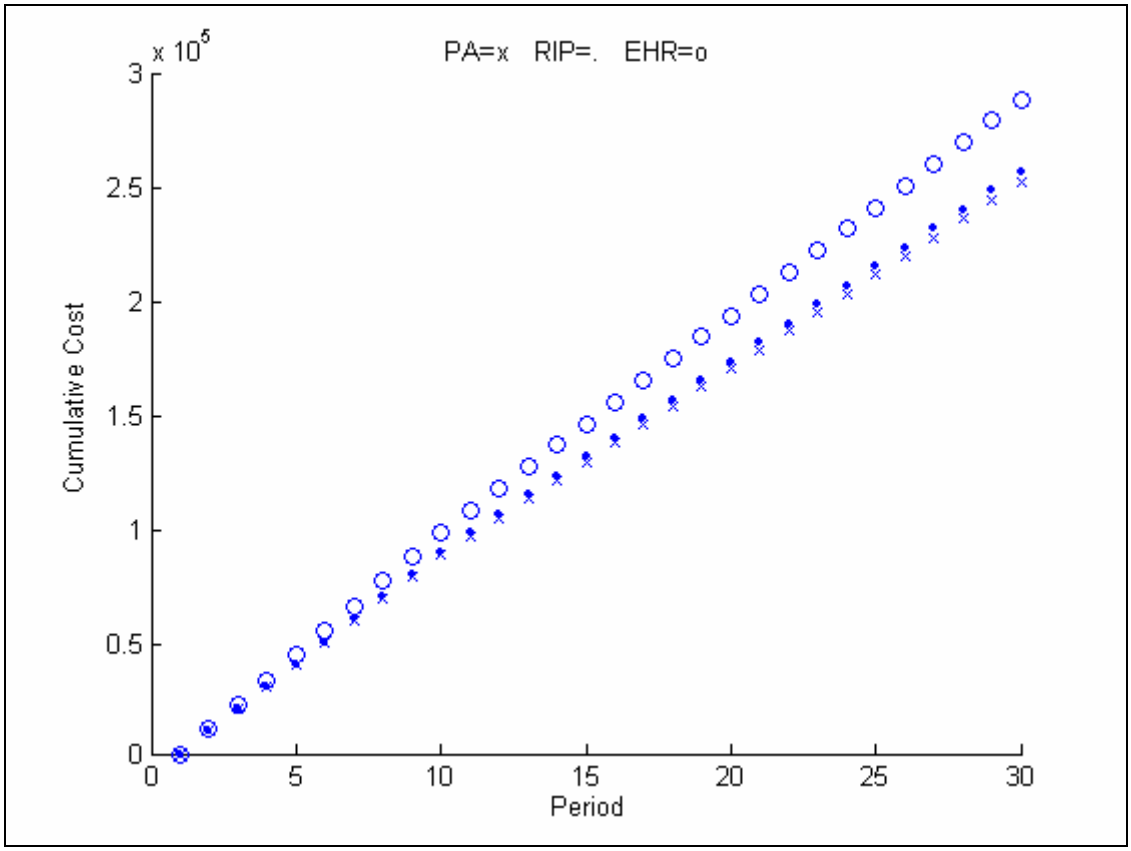


Fig. J.24. Comparison of Cost Graphs for NOR(5,1)

I.5. Empirical Distribution

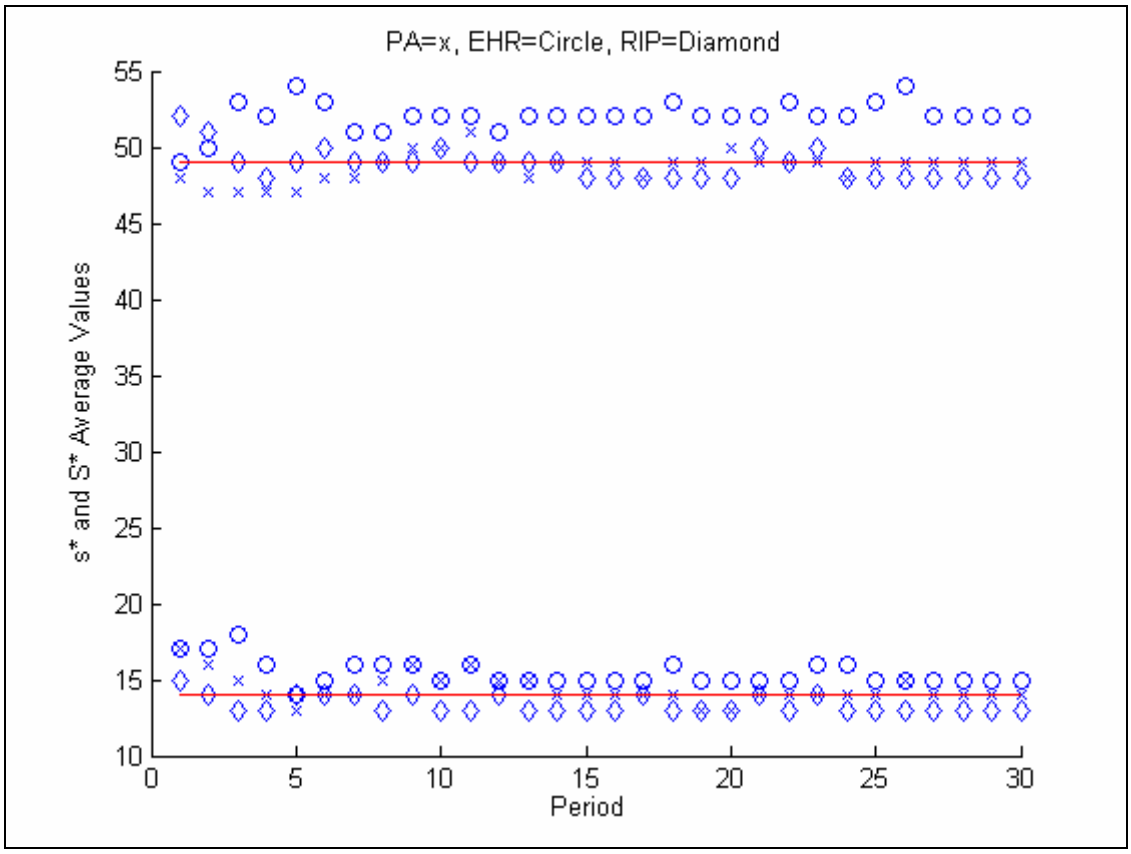


Fig. J.25. Comparison of (s*,S*) Graphs for Empirical Distribution

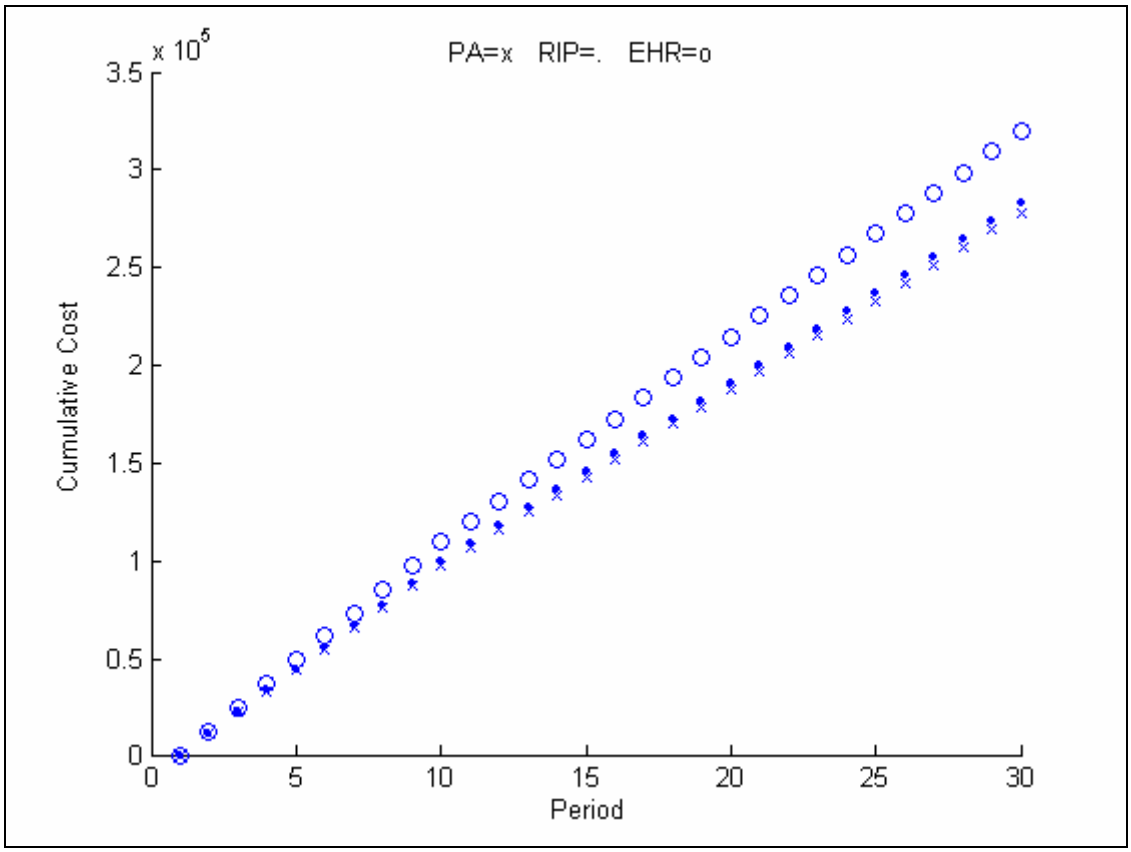


Fig. J.26. Comparison of Cost Graphs for Empirical Distribution

I.6. Adaptability

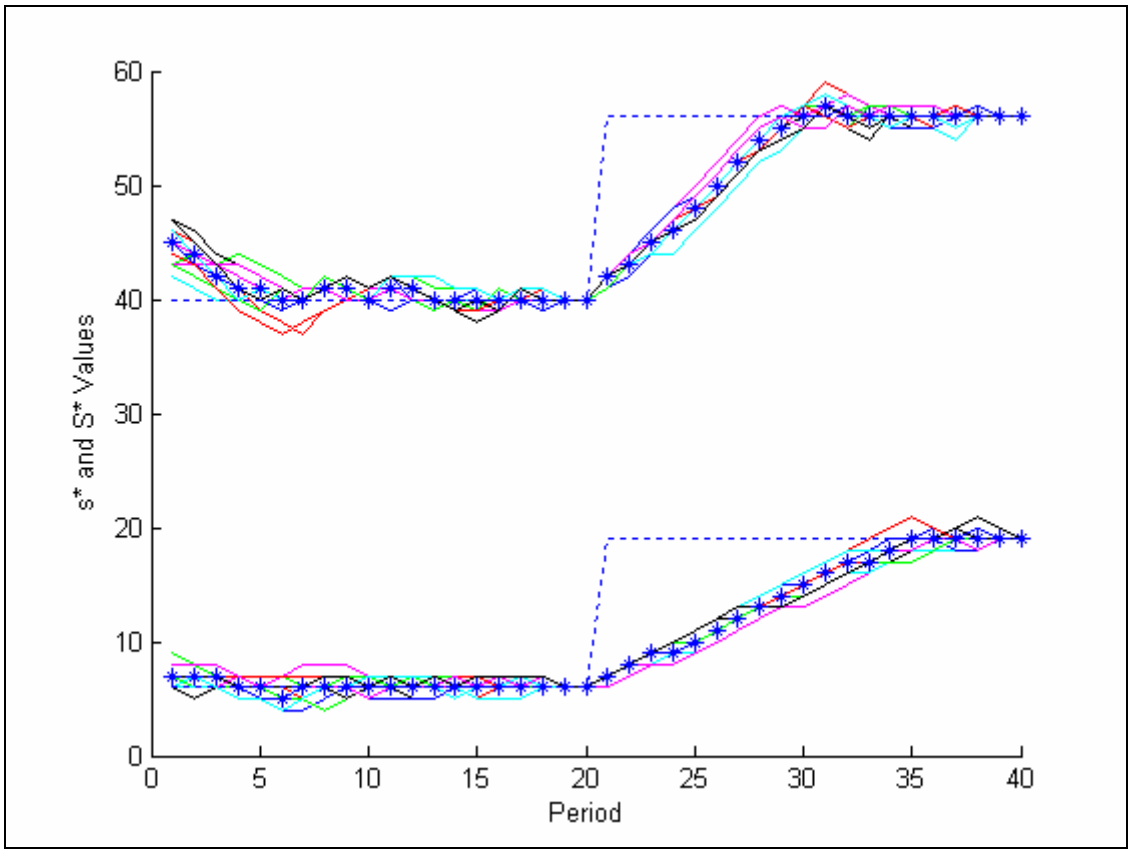


Fig. J.27. (s*,S*) Graph for **PA Alg.**, δ=15, γ=3

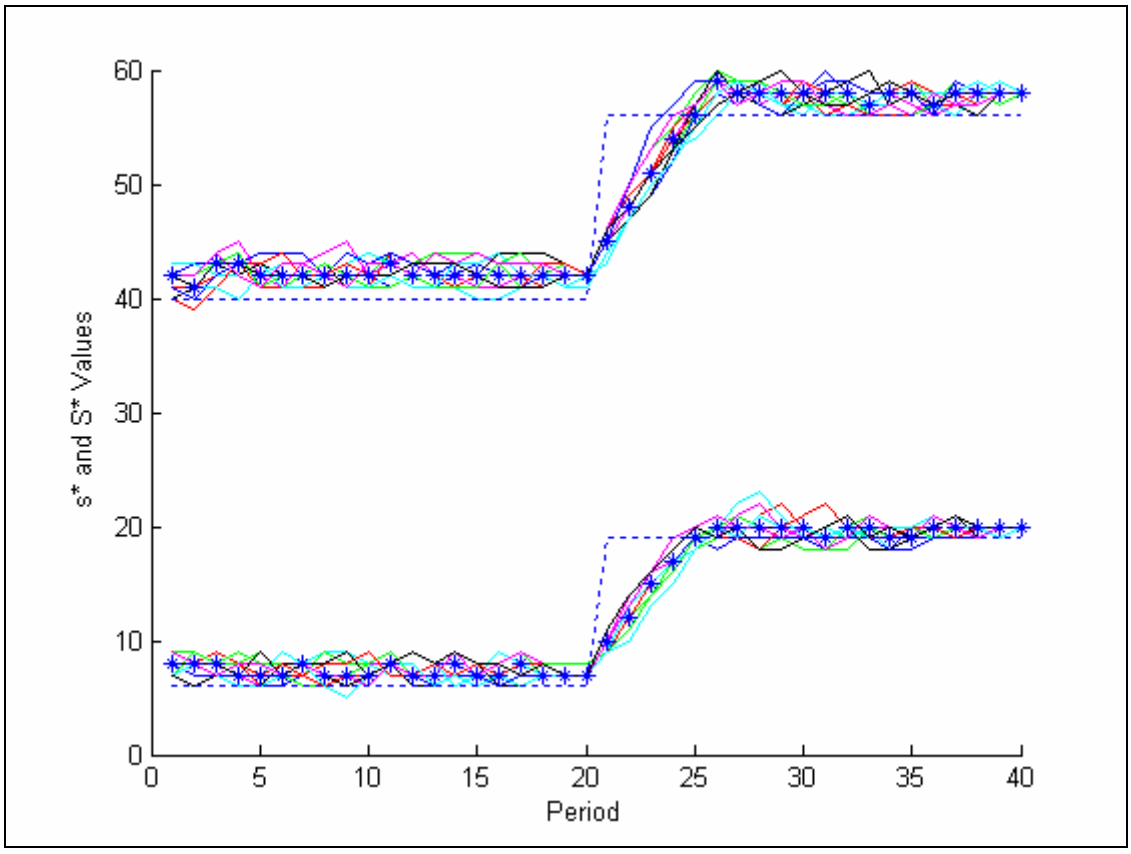


Fig. J.28. (s*,S*) Graph for **EHR Alg.**, δ=15, γ=3

Appendix J Outputs of Numerical Examples

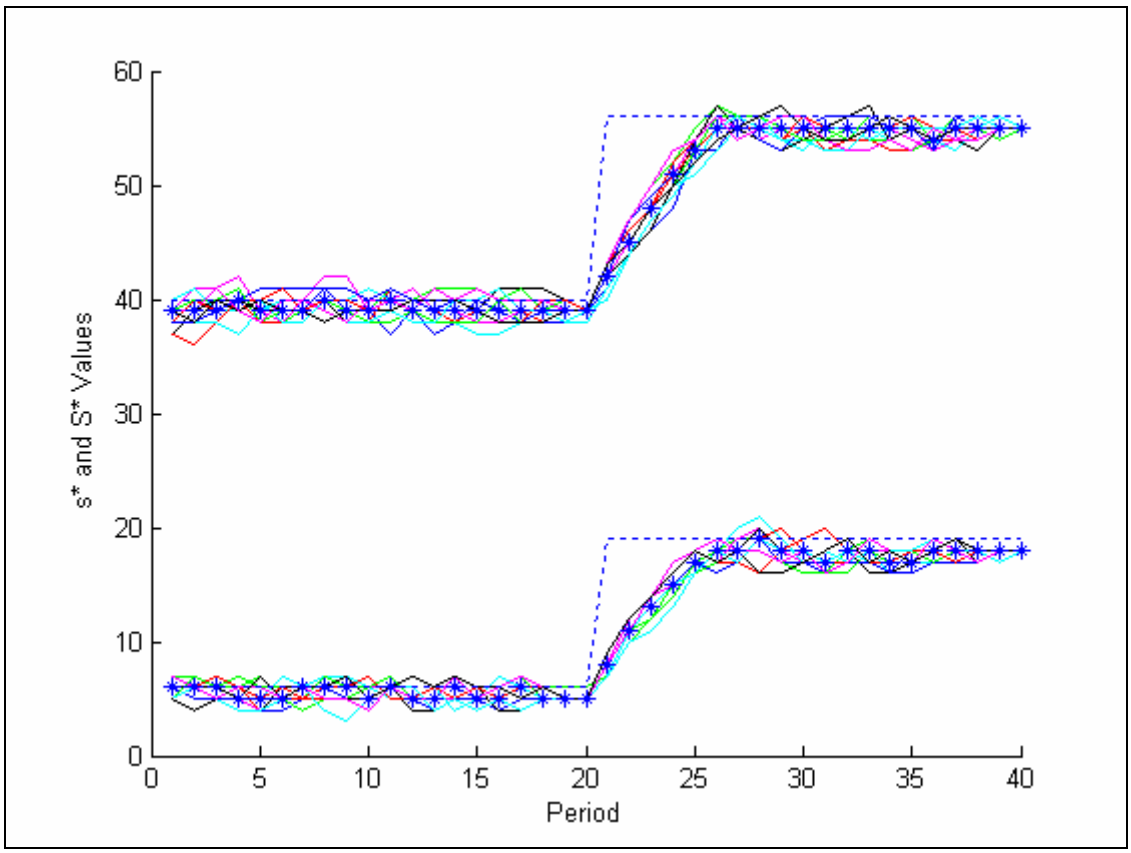


Fig. J.29. (s*,S*) Graph for **RIP Alg.**, δ=15, γ=3

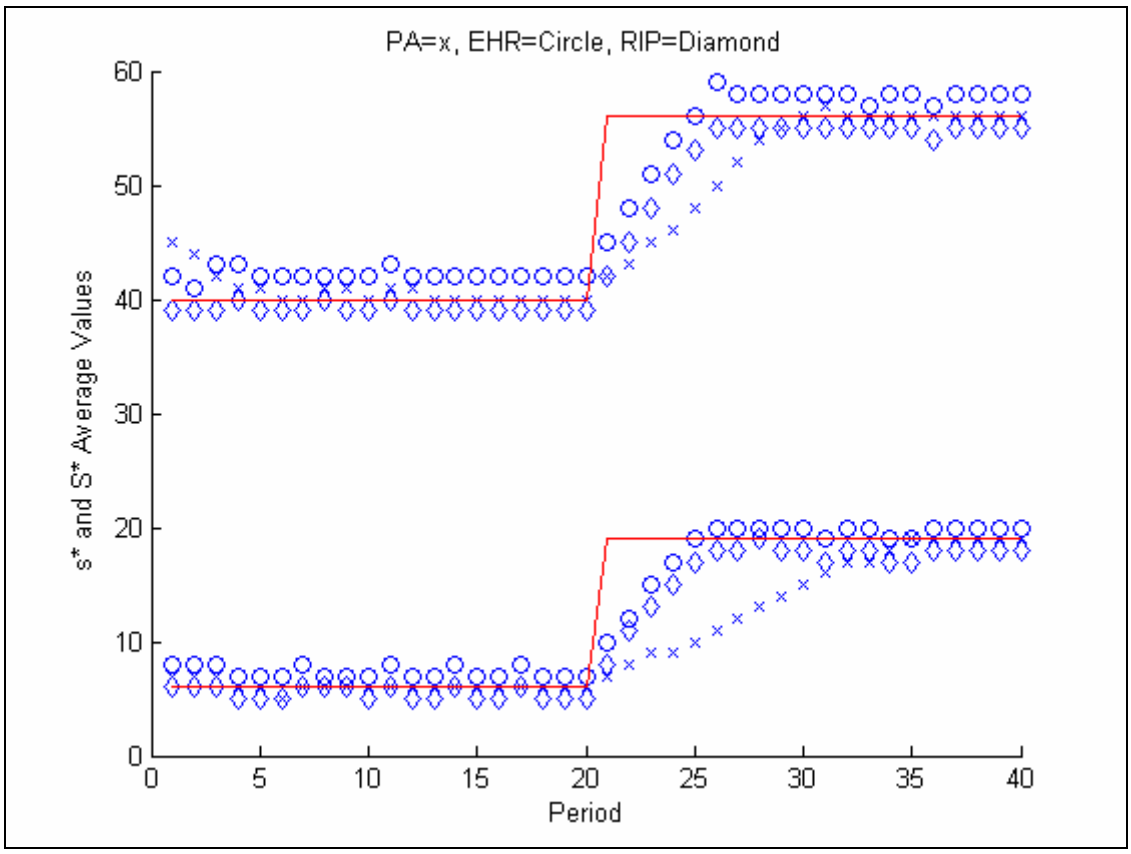


Fig. J.30. Comparison of (s*,S*) Graphs, δ=15, γ=3

Appendix J Outputs of Numerical Examples

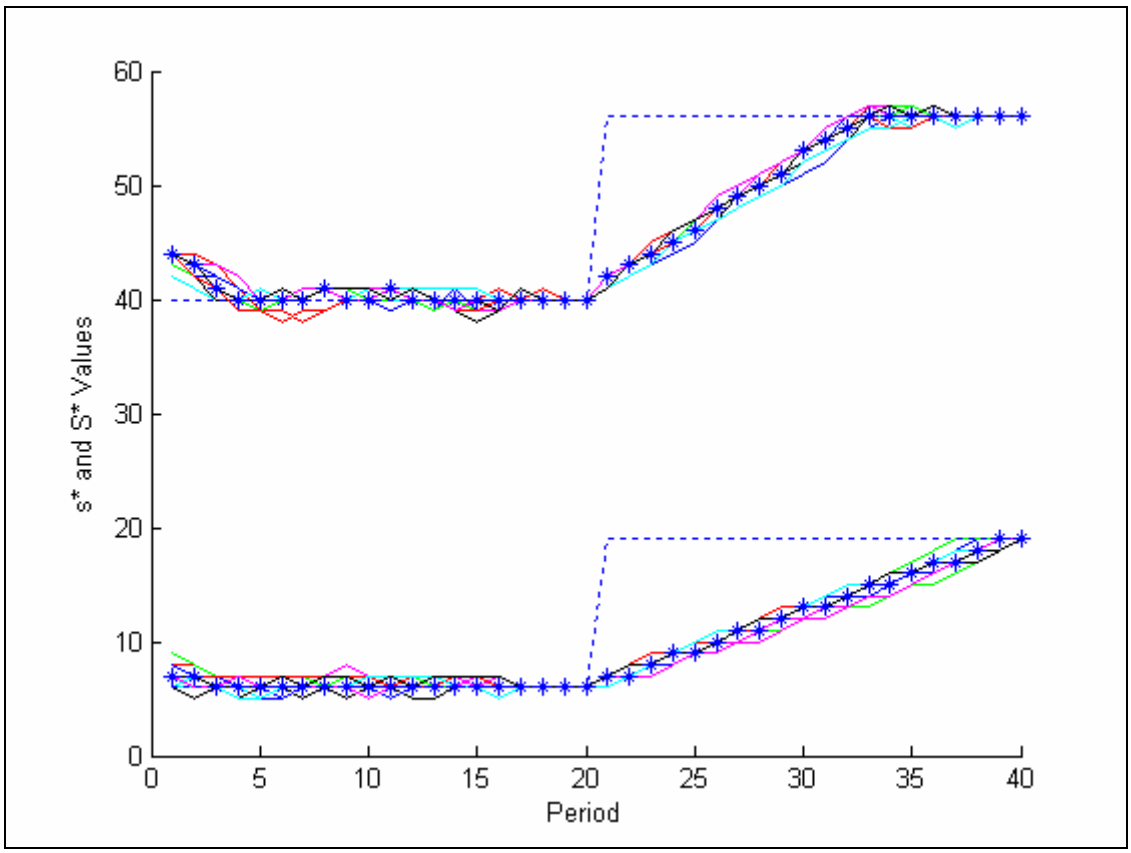


Fig. J.31. (s*,S*) Graph for **PA Alg.**, δ=15, γ=ALL HISTORY

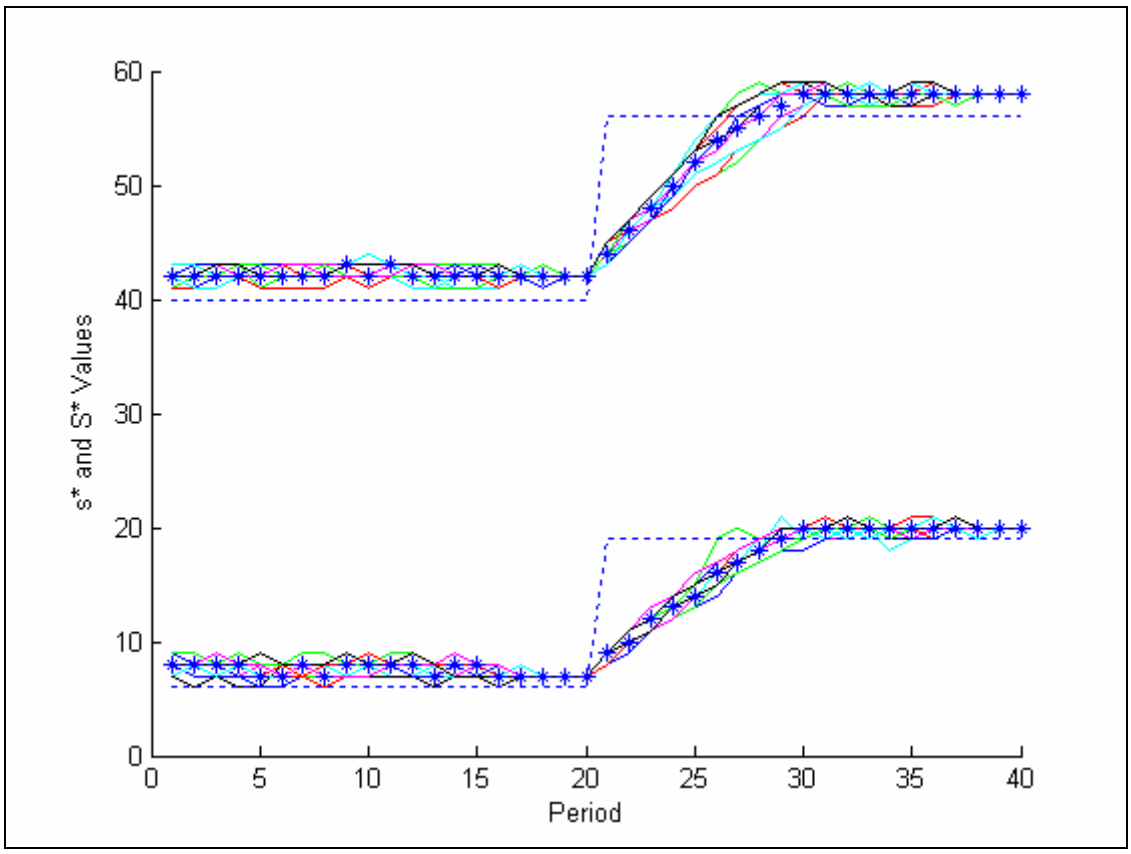


Fig. J.32. (s*,S*) Graph for **EHR Alg.**, δ=15, γ=ALL HISTORY

Appendix J Outputs of Numerical Examples

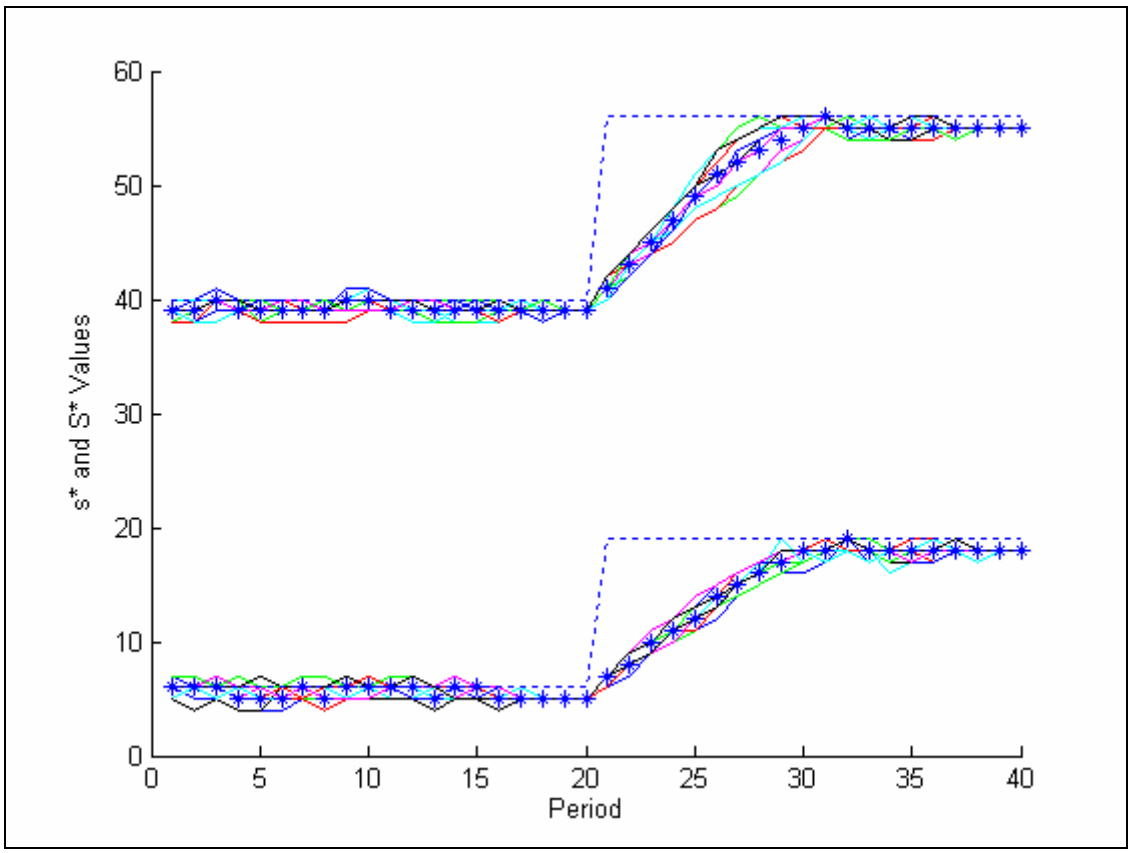


Fig. J.33. (s*,S*) Graph for **RIP Alg.**, δ=15, γ=ALL HISTORY

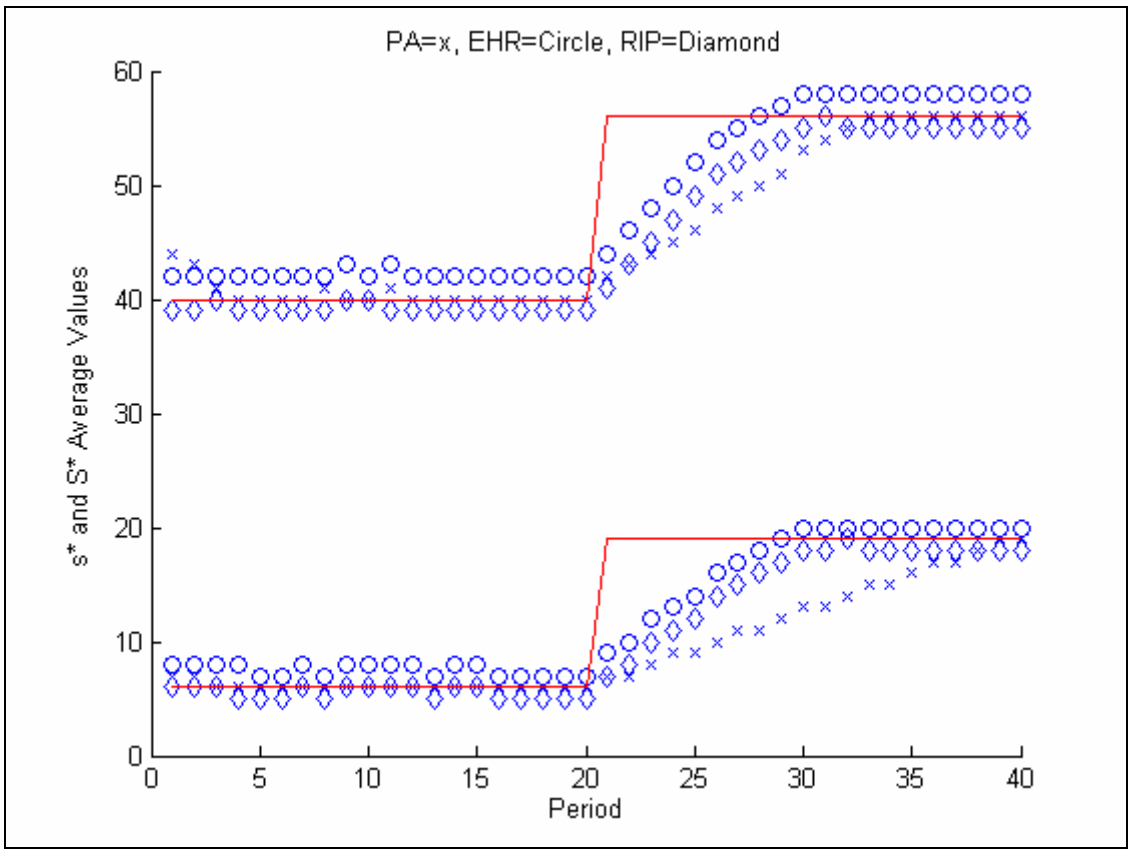


Fig. J.34. Comparison of (s*,S*) Graphs, δ=15, γ=ALL HISTORY

Appendix J Outputs of Numerical Examples

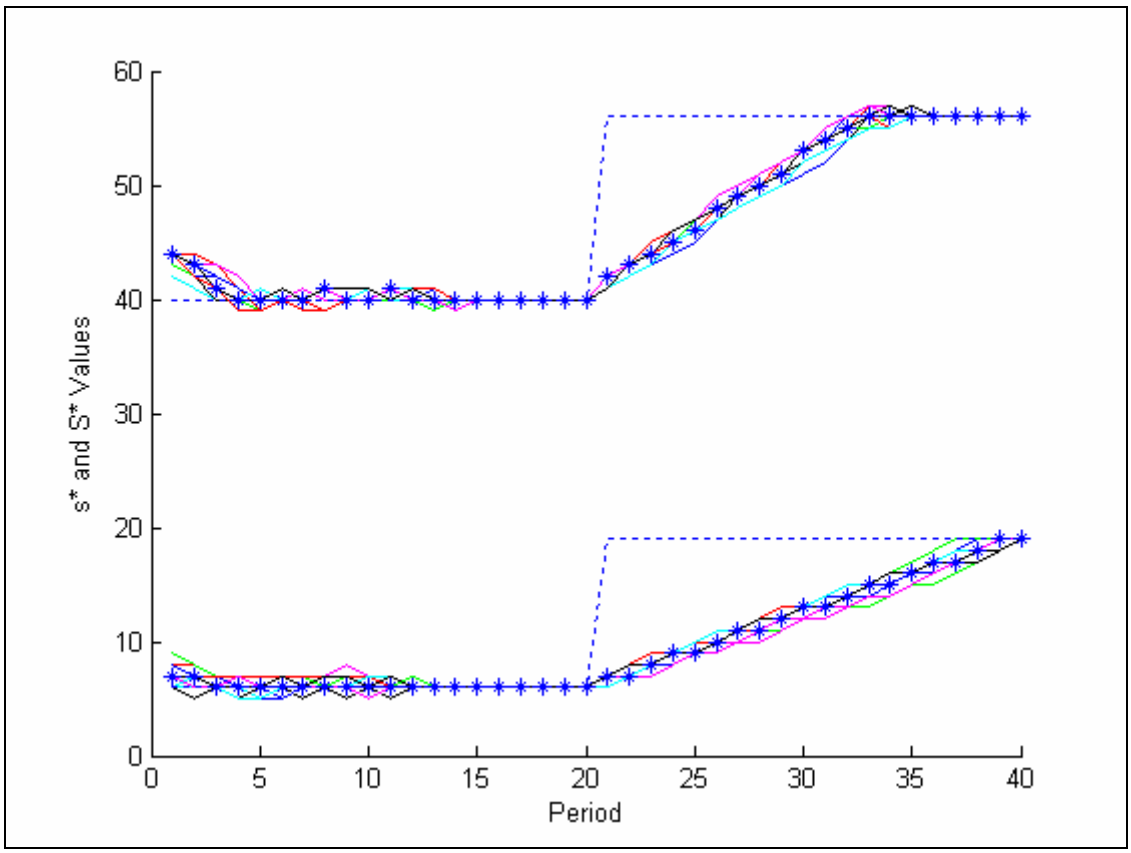


Fig. J.35. (s*,S*) Graph for **PA Alg.**, δ=90, γ=ALL HISTORY

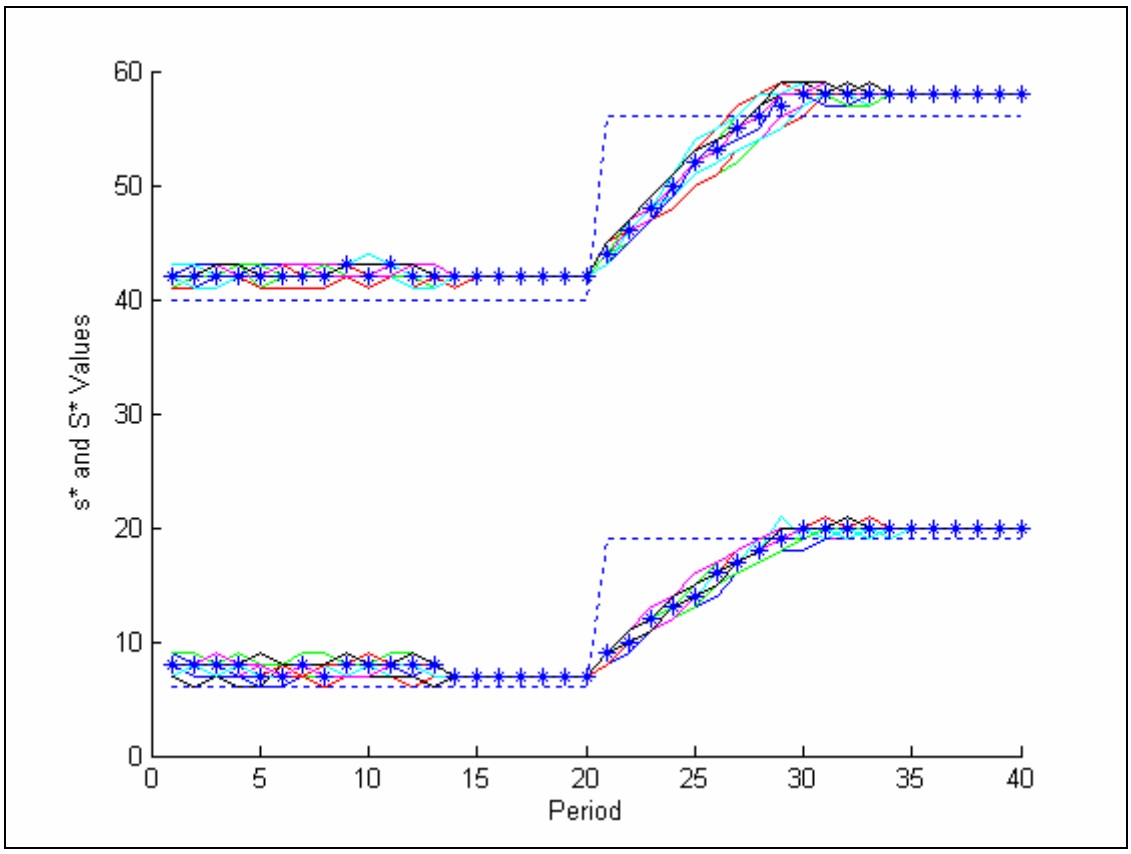


Fig. J.36. (s*,S*) Graph for **EHR Alg.**, δ=90, γ=ALL HISTORY

Appendix J Outputs of Numerical Examples

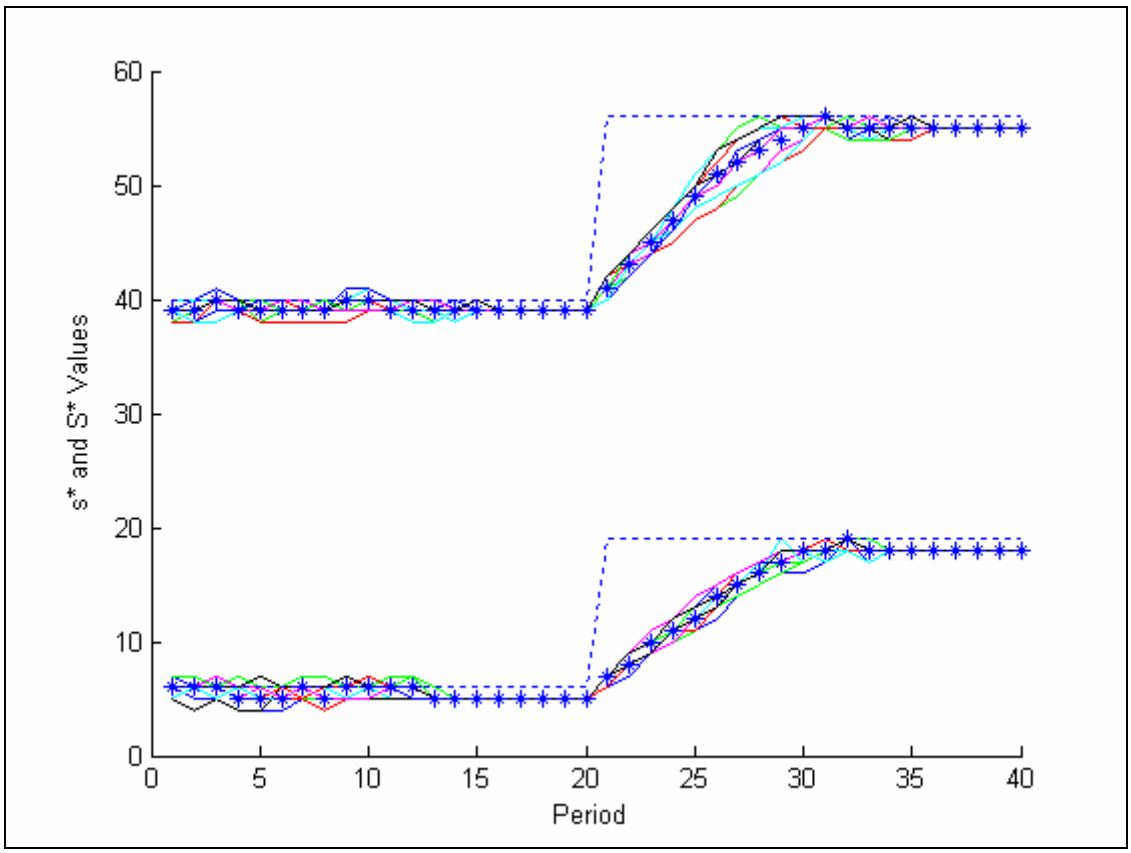


Fig. J.37. (s*,S*) Graph for **RIP Alg.**, δ=90, γ=ALL HISTORY

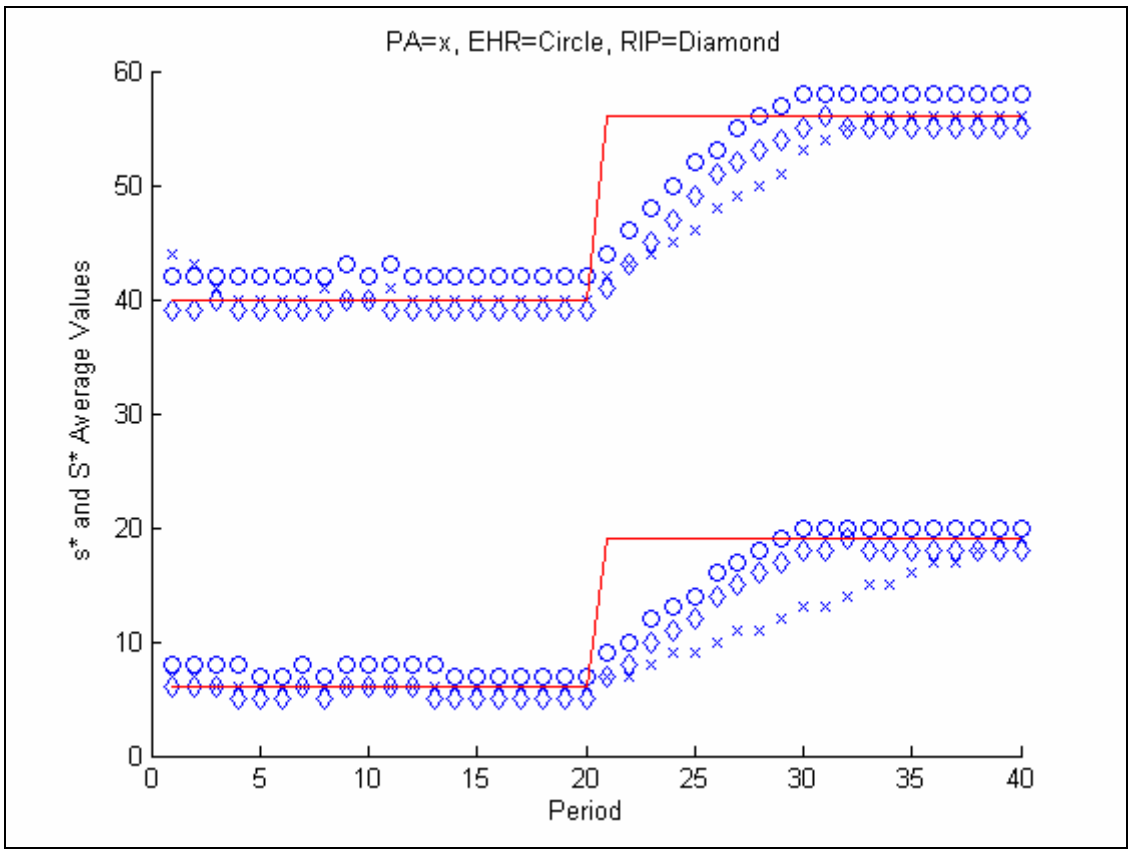


Fig. J.38. Comparison of (s*,S*) Graphs, δ=90, γ=ALL HISTORY