

2012

## An intelligent computational approach to the optimization of inventory policies for single company

Qinglin Duan

*Louisiana State University and Agricultural and Mechanical College*

Follow this and additional works at: [https://digitalcommons.lsu.edu/gradschool\\_theses](https://digitalcommons.lsu.edu/gradschool_theses)



Part of the [Construction Engineering and Management Commons](#)

---

### Recommended Citation

Duan, Qinglin, "An intelligent computational approach to the optimization of inventory policies for single company" (2012). *LSU Master's Theses*. 1819.

[https://digitalcommons.lsu.edu/gradschool\\_theses/1819](https://digitalcommons.lsu.edu/gradschool_theses/1819)

This Thesis is brought to you for free and open access by the Graduate School at LSU Digital Commons. It has been accepted for inclusion in LSU Master's Theses by an authorized graduate school editor of LSU Digital Commons. For more information, please contact [gradetd@lsu.edu](mailto:gradetd@lsu.edu).

# AN INTELLIGENT COMPUTATIONAL APPROACH TO THE OPTIMIZATION OF INVENTORY POLICIES FOR SINGLE COMPANY

A Thesis

Submitted to the Graduate Faculty of the  
Louisiana State University and  
Agricultural and Mechanical College  
in partial fulfillment of the  
requirements for the degree of  
Master of Science in Industrial Engineering  
in

The Department of Mechanical and Industrial Engineering

by  
Qinglin Duan  
B.S., Shanghai University, 2007  
M.S., The University of New South Wales, 2009  
December 2012

## **ACKNOWLEDGEMENTS**

My special thanks to my advisor, Professor T. Warren Liao, who served as a patient guide and a source of ideas and impetus during this endeavor. I really appreciate the time and effort that he has given in mentoring me through this research process. I would also like to thank the other professors who were so kind as to serve on my thesis committee: Dr. Peter Kelle and Dr. Frank Tsai. Huge thanks for their input and support.

I benefitted from the teaching experience at the Department of Mechanical and Industrial Engineering, Louisiana State University and Agricultural and Mechanical College. The financial assistance from the teaching assistantship towards this research is hereby acknowledged. My sincere thanks are due to Professors Fereydoun Aghazadeh, Craig M. Harvey, Isabelina Nahmens, Emy M. Roider, Bhaba R. Sarker and Magd Zohdi for their support and encouragement.

Last but not least, I wish to extend my deepest thanks to my parents, my husband and my entire family who always believed in me and made me who I am today.

## TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	ii
ABSTRACT.....	v
CHAPTER 1 INTRODUCTION .....	1
CHAPTER 2 LITERATURE REVIEW .....	4
CHAPTER 3 PROPOSED APPROACH.....	10
CHAPTER 4 OPTIMIZATION OF FAST-MOVING NONPERISHABLE ITEMS .....	14
4.1 The problem .....	14
4.2 ACO <sub>R</sub> for determining optimal inventory policy .....	17
4.3 Experimental details and results .....	23
4.4 Discussion .....	37
4.5 Conclusion .....	52
CHAPTER 5 OPTIMIZATION OF SLOW-MOVING PERISHABLE ITEMS .....	54
5.1 The problem .....	54
5.2 Mathematical model for the considered problem .....	55
5.3 ACO <sub>R</sub> for optimizing ( <i>S-I</i> , <i>S</i> ) policies .....	62
5.4 Experimental details and results .....	64
5.5 Discussion .....	78
5.6 Conclusion .....	86
CHAPTER 6 OPTIMIZATION OF PERISHABLE ITEMS FOLLOWING COMPOUND POISSON DEMAND PROCESSES .....	88
6.1 The problem .....	88
6.2 Mathematical model for the considered problem .....	89
6.3 Barebones differential evolution for determining optimal inventory policy .....	95
6.4 Computational results .....	98
6.5 Conclusion .....	101
CHAPTER 7 SUMMARY AND CONCLUSION.....	102

REFERENCES .....	106
VITA .....	111

## **ABSTRACT**

This study develops and tests a computational approach for determining optimal inventory policies for single company. The computational approach generally comprises of two major components: a metaheuristic optimizer and an event-driven inventory evaluation module. Metaheuristic is a powerful search technique, under the intelligent computational paradigm. The approach is capable of determining optimal inventory policy under various demand patterns regardless their distribution for a variety of inventory items. Two prototypes of perishability are considered: (1) sudden deaths due to disasters and (2) outdateding due to expirations. Since every theoretical model is specially designed for a certain type of inventory problem while the real world inventory problems are numerous, it is desirable for the newly proposed computational approach to cover as many inventory problems/models as possible. In a way, the proposed meta-heuristic based approach unifies many theoretical models into one and beyond. Experimental results showed that the proposed approach provides comparable results to the theoretical model when demand follows their assumption. For demands not well conformed to the assumption, the proposed approaches are able to handle it but the theoretical approaches do not. This makes the proposed computational approach advantageous in that it can handle various types of real world demand data without the need to derive new models. The main motivation for this work is to bridge the gap between theory and practice so as to deliver a user-friendly and flexible computational approach for rationalizing inventory management for single company.

## **CHAPTER 1 INTRODUCTION**

Inventory management plays a key role in the functioning of industrial and commercial enterprises. A desirable inventory control policy is one that will guarantee a satisfactory service level without keeping too much unnecessary stocks that are costly and difficult to handle.

Determining optimal inventory policy is a typical inventory research problem. Various inventory models tackling this problem have been developed. However, even though almost all companies and enterprises are increasingly trying to apply scientific methods for better handling their inventories, the use of those methods is often limited to some basic tools like the computation of economic order quantities and rough approximations of safety stock. The wide use of more elaborate and appropriate methods for inventory management in practice is hindered by the following most notable reasons:

When dealing with inventory models, the probability distribution of demand is often used to capture the demand uncertainty and is an important characteristic in inventory management. The traditional approaches on inventory control almost are all based on the assumption that lead time demand follows a certain type of distribution. This is often not the case in real practice. It is not unexpected that real world data simply does not fit perfectly the demand distribution assumed by those models. Using those traditional approaches as approximations in cases where real world data significantly deviates from the assumed distribution can lead to very unsatisfactory results.

The problem gets more complicated when dealing with stochastic demand. It is found through previous literature that stochastic theoretical analysis leads to tractable expressions only under specific assumption. Furthermore, most theoretical models assume that the demand data are independently and identically distributed (i.i.d.) while some real-world data may be auto-correlated. Failing to account for the auto-correlation between demand data can also cause serious inaccuracy in calculating inventory level.

Since the feasibility of a theoretical model depends on whether or not it is mathematically tractable for the subject cost function under the distribution of lead-time demand, it is almost impossible to develop a theoretical model that covers more than one type of distribution. Besides, when considering real world application, “in a context where the optimization must be carried out relatively frequently for many thousands of items the computational effort can be regarded as too heavy”, quoted from Axsäter (2006).

Based on those observations, an intelligent computational approach is proposed here to optimize inventory, totally different from the traditional theoretical/analytical approach. The computational approach comprises of two major components: a metaheuristic optimizer and an event-driven evaluation module (without the need of any explicit mathematical function/model). The proposed approach of optimizing inventory is fully developed for single company. The proposed computational approach offers unique features that existing theoretical/analytical models are lack of. It is capable of finding optimal inventory policy regardless the distribution type and the dependence among data points. The cost function is not explicit and does not need



to be convex and mathematically tractable. This makes the proposed computational approach advantageous in that it can handle various types of real world demand data without the need to derive new models.

## CHAPTER 2 LITERATURE REVIEW

The scientific literature on determining optimal inventory policy is huge and it is impossible to provide a comprehensive overview of all literature. Therefore, I will focus on those publications motivated by practical applications and directly related to this study.

The problem of computing optimal inventory policies under stochastic demand subject to either a service level constraint or a backorder/lost sale cost is a typical inventory research problem. Since the cost function is, in general, ill behaved, it implies that almost all possible combinations of policy parameters need to be evaluated in a direct search procedure. When considering real world situation, the function, indeed, usually fails to be quasiconvex and may have several local optima. Therefore, various inventory models have been developed for tackling this problem. Typically these mathematical models more or less rely on certain approximations and assumptions so as to reduce the complexity of the model to a relatively simple form for the ease of model solution. These models differ in the assumptions made, inventory policies used and/or in the service level criterion used. The most widely-used inventory policies are  $(R, Q)$ ,  $(s, S)$  and base-stock: usually  $(R, Q)$  and  $(s, S)$  for fast-moving items while base-stock for slow-moving items.

Hadley and Whitin (1963) offers a clear overview of both lost sales and backordering inventory modeling. In many cases, closed form solutions cannot be obtained and iterative algorithms must be used to converge on optimal solutions. The existing computation schemes

can be classified into two categories: exact method and heuristic/approximate method. Exact methods are rarely used in real world practice because they are considered prohibitively expensive. Therefore, the remaining review only focuses on those heuristic/approximate methods considering real world applications.

For  $(R, Q)$  inventory policy, the classical approximate approach is that of Hadley and Whitin (1963), followed by Brown (1967) and Wagner (1969). The approximate approach by Hadley and Whitin (1963) was initially developed for lost sale case and a version adapts the corresponding backorder solution procedure was also developed. Nahmias (1976 a) developed the stream-lined versions of the above three algorithms by demonstrating their equivalence. Yano (1985) proposed a heuristic method to minimize the average annual inventory investment subject to a service-level constraint. Federgruen and Zheng (1992) derived an efficient algorithm for computing optimal  $(R, Q)$  inventory control policy assuming a unimodal cost structure. Specifically, their algorithm is restricted to demands arising on a unit-by-unit basis.

Johansen and hill (2000) devised an improved solution procedure for a periodic review  $(R, Q)$  policy with lost sales using asymptotic renewal theory, assuming a continuously distributed demand and only a single outstanding replenishment order at any time. Matheus and Gelders (2000) considered an inventory subject to a compound Poisson demand pattern and proposed an exact and an approximate reorder point calculation method for the  $(R, Q)$  inventory policy given the pre-determined reorder quantities  $Q$ . Rosling (2002) demonstrated how the optimization of  $R$  and  $Q$  can be carried out by an iterative procedure under a normally distributed lead-time

demand, aiming to minimize the average sum of holding, ordering and backorder penalty cost per unit time. Axsäter (2006) considered the same problem but replace the backorder penalty cost with a fill-rate constraint instead to minimize the average sum of holding and ordering costs per unit time. Based on the observation that the considered problem for a given fill rate could be reformulated to involving only one single parameter, Axsäter (2006) came up with a simple approximate interpolation procedure. Al-Rifai (2007) developed an iterative heuristic optimization algorithm for a two-echelon non-repairable spare parts  $(R, Q)$  inventory system in order to minimize total annual inventory investment subject to average annual ordering frequency and expected number of backorder constraints. However, their heuristic cannot be applied to single-echelon models separately to determine the policy parameters for the inventory system under consideration.

Studies of heuristic/approximate methods on  $(s, S)$  policy can be dated back into Roberts (1962), Veinott and Wagner (1965), Wagner (1969), Sivazlian (1971), Naddor (1975), Schneider (1978), Ehrhardt (1979, 1984), Porteus (1979), Freeland and Porteus (1980), Tijms and Groenevelt (1984), and Sahin and Sinha (1987). It was until Zheng and Federgruen (1991) that they provided an efficient algorithm that search in the  $(s, S)$  plane directly, based on a number of properties of the cost function as well as new tight lower and upper bounds for  $s^*$  and  $S^*$ . The computational complexity of the proposed algorithm is proven to be only 2.4 times that required to evaluate a specific single  $(s, S)$  policy. Later, Feng and Xiao (2000) developed a new algorithm for finding optimal  $(s, S)$  policy by introducing a dummy cost factor and an auxiliary

function. Their numerical tests showed that on average, their proposed algorithm further reduced the search effort by more than 30% comparing to Zheng and Federgruen's method.

Base stock policy is actually a special case of  $(s, S)$  policy with  $s=S-I$ . Realizing many real-world inventory items are perishable, I decide to focus only on those studies that take perishable into account when considering base stock policy so as to increase the complexity of modeling and better meet the comprehensive needs of real-world application. Perishable inventory models with stochastic demands are typically difficult to analyze (Nahmias 1982). The easiest case occurs when the lifetime of the stock is exactly one period. When units will perish after one single period and inventories are reviewed periodically, the ordering decisions are independent and the problem is simplified to a sequence of newsboy problems (Arrow et al. 1958). Van Zyl (1964) analyzed the perishable inventory problem where the lifetime is exactly two periods using dynamic programming and showed the existence of an optimal order-up-to policy. Nahmias and Pierskalla (1973) also considered the two-time-period perishing problem with a different cost structure for both lost sales and complete backorder cases. Some analytical results on the system performance were obtained.

Extending those early models to the general  $m$ -period periodic review models is far more complex due to the required multi-dimensional state space. Early pioneers include Fries (1975) and Nahmias (1975). The main difficulty of mathematical modeling lies in that, when demand is uncertain and the product lifetime exceeds one period, it is no longer possible to obtain a replenishment ordering policy so that there is no perishing. The problem state vector must

include the stock level of each possible age category. Due to the complicated nature of the problem, it is unlikely to find optimal ordering policies for general perishable inventory models with positive lead times. Instead, later efforts have been largely focused on finding approximations of the true optimal policy (Chazan and Gal 1977; Cohen 1976; Nahmias 1976 b, 1978; Nansakumar and Morton 1993). Comparing to general m-period periodic review model, fewer perishable inventory studies focused on continuous review model. Weiss (1980) investigated a continuous review perishable inventory model with a Poisson demand process and zero lead time. Weiss (1980) showed that in the backorder case, when the shortage cost is increasing convex in response time, the  $(s, S)$  policy is optimal.

Schmidt and Nahmias (1985) considered an  $(S-I, S)$  continuous review perishable inventory model with Poisson demand, fixed lifetime and fixed leadtime. They assumed the lost sales case, where the problem state vector was kept to be finite-dimensional. However, the backorder case will definitely complicate the problem and make the state space infinite-dimensional. Schmidt and Nahmias (1985) concluded that with fixed lifetime and fixed leadtime, it is extremely difficult to build an exact model to obtain an optimal policy. Ravichandran (1995) analyzed the perishable model with Poisson demand and positive random leadtime. Ravichandran (1995) analyzed the inventory system performance, however, based on the unrealistic assumption that the aging of new stock only begins after the complete depletion of the existing stocks. Gürlü and Özkaya (2008) analyzed the continuous review  $(s, S)$  policy for perishables with zero lead time and backlogging. They observed that the shape of the shelf life distribution has significant impact

on the cost function and the loss incurred by ignoring the randomness of the shelf life can be drastic. Based on their observation, they express the expected total cost function using integrations and sums of relevant shelf life distribution functions. The expected cost function developed can be evaluated only by a computer numerical method and therefore are difficult to use in optimization. Latest work of Olsson and Tydesjö (2010) derived their  $(S-I, S)$  optimal solution for the backorder case from the solution procedure for the lost sale case developed by Schmidt and Nahmias (1985) and their results are compared to Chiu (1995), which considers  $(R, Q)$  policies. Baron et al. (2010), which is an extension to Gürler and Özkaya (2008) but does not allow back orders, derived closed form expressions for the relevant cost in their model and theoretically, their model can be extended to demand-sizes of various phase-type distributions. However, complicated phase-type distributions lead to cumbersome expressions of the relevant cost functions which need for optimization.

Without exception, those mathematical models in literature are very complicated and their validity holds only when the many restrictive assumptions including the demand distribution are satisfied. Since the feasibility of a theoretical model depends on whether or not it is mathematically tractable for the subject cost function under the distribution of lead-time demand, it is almost impossible to develop a globally comprehensive theoretical model that covers more than one type of distribution. This major observation motivates me to design a globally comprehensive computational approach which can cover as wide scope of inventory problems as possible so that it is user-friendly for the management of inventory in a business environment.

### **CHAPTER 3 PROPOSED APPROACH**

The proposed intelligent computational approach comprises of two major components: a meta-heuristic and an event-driven evaluation module (without the need of any explicit mathematical function/model). The meta-heuristic can be any meta-heuristic algorithm. In this study, Ant Colony Optimization for continuous domains (ACO<sub>R</sub>) and barebone differential evolution (BBDE) are chosen (and have been implemented) because they are relatively new and have been shown to produce good results in various applications. The author does not claim that these two are the best choice. It is very likely that some other meta-heuristic algorithms might work better than these two for the subject application. A comprehensive comparative study is needed to address this issue, which will be out of the scope of this study.

Note that the evaluation module is a key component of the proposed approach in order for the meta-heuristic algorithm to find the optimal solution. The evaluation module essentially implements the inventory policy of concern, considering different problem settings. The implementation is based on an event-driven updating mechanism identical to discrete-event simulation, starting from the first event to the last event. Each customer arrival is considered as an event. At each event occurrence, the demand size required by the subject customer is read and the corresponding sequence of operations relevant to the inventory policy (holding cost calculation, backordering/lost sale cost calculation, service level calculation, and etc.) is implemented. A key input to the evaluation module is the demand data actually collected from



the real world and does not have to fit into any distribution at all. Nevertheless, it should be pointed out that one single optimal inventory policy might not be the best if the demand is non-stationary. An adaptive inventory policy will be needed in such cases. In this study, one single optimal policy is assumed.

It is not difficult to see that the evaluation module does not optimally solve any explicit mathematical model by itself. It reads in raw demand data for each event occurrence and evaluates the associated cost, starting from the first to the last, for a given combination of inventory policy parameters. It does not care what distribution the demand data is. It also circumvents the problem that the cost function may become non-convex and mathematically intractable. Furthermore, the evaluation module can be easily adapted to fit various inventory policies and problem settings. For example, a new evaluation module can be easily developed based on the backordering one to adapt to the lost sale case, without any re-derivation of mathematical equations. To adapt to any other inventory system, one only needs to understand how the inventory system works and how to calculate the corresponding cost. In summary, the differences between the proposed approach and theoretical/analytical approaches are summarized in Table 3.1.

In the following sections, the successful application of the proposed approach to different inventory problems will be demonstrated in detail. Table 3.2 gives a summary of the inventory problems studied in this thesis research.

Table 3.1 Comparison between the proposed approach and traditional analytical approaches.

	Traditional approach	Proposed approach
Assumptions on demand distribution	Required	None
Parameter fitting for raw demand data	Yes	No
Explicit mathematical models (cost equations)	Yes	No
Solution methods	Exact, approximation, heuristic, meta-heuristic	Any meta-heuristic with a specially designed evaluation module
Demand data that can be handled	Only those satisfy the assumption	Any
Adaptability to other situations	Low	High

Table 3.2 Summary of the inventory problems studied

Inventory problem	Counterpart theoretical/analytical models	Inventory policy
Fast-moving nonperishable	Rosling (2002) Axsäter (2006)	$(R, Q)$ policy
Slow-moving perishable	Schmidt and Nahmias (1985) Olsson and Tydesjö (2010)	Order-up-to policy
fast-moving/slow-moving, nonperishable/perishable	Baron et al. (2010)	$(s, S)$ policy

Basically, three categories of inventory problems that utilize different inventory policies have been studied in detail. For each category, the results obtained by the proposed approach are compared with its counterpart theoretical model and the advantages offered by the newly computational approach will be identified.

## CHAPTER 4 OPTIMIZATION OF FAST-MOVING NONPERISHABLE ITEMS

### 4.1 The problem

This chapter considers the problem of determining the optimal parameters for one of the most commonly used inventory control policies, i.e.,  $(R, Q)$  policy with complete backorder, in a single-echelon inventory system. When the inventory position (stock on hand, plus outstanding orders, and minus backorders) drops to or below the reorder point  $R$ , a batch quantity of size  $Q$  is ordered. Any unmet demand is backordered in full amount. The backorder is evaluated in two ways: by computing penalty cost or fill rate (a.k.a. service level). The optimization problem is to determine the reorder point and the batch quantity jointly so that the total cost per unit time is minimized. For the former case, the total cost consists of ordering, holding and backorder penalty costs. The latter case differs from the former case in that there is no backorder penalty cost, but the optimization problem is subject to the fill rate constraint.

Using the following notations, the mathematical model of the inventory problem is presented.

<b>Subscripts</b>	
$t$	Period $t$ in the planning horizon, $t=1, 2, \dots, T$
<b>Notations</b>	
$D_t$	demand for period $t$
$L$	leadtime
$A$	Fixed ordering cost
$h$	holding cost per product unit and unit time
$b$	backorder penalty cost per product unit and unit time
$S$	target fillrate
$R$	reorder point
$Q$	ordering quantity

**Notations (continued)***TC/period*                      average total cost per unit time

Given the initial inventory level  $x_0$  at the period ( $t=0$ ) and the backorder quantity at period ( $t=0$ ) is zero,  $bx_0 = 0$ , for each time period,  $t=1, 2, \dots, T$ , do the following:

All orders placed at the end of period  $t$  will be due after the lead time (at the beginning of period  $t+L$ ). As a result, the available inventory used to satisfy the customer (including the backorder quantity and the current demand) at the beginning of period  $t$ ,  $ax_t$ , is given by

$$ax_t = \begin{cases} x_{t-1}, & t \leq L \\ x_{t-1} + (y_{t-L} - x_{t-L}), & t > L \end{cases} \quad (4.1)$$

In which  $x_{t-1}$  is the inventory level at the end of previous time period;  $y_{t-L} - x_{t-L}$  is the order quantity due at current time period. How to calculate  $x_t$  and  $y_t$  is detailed in Eq. (4.5)-(4.6).

After replenishment, the company observes its customer demand  $D_t$  and try to satisfy  $D_t$  with its available on-hand inventory,  $ax_t$ . Any shortfall occurred at period  $t$  will be backordered to with a penalty cost,  $b$ . Any item held at stock at period  $t$  will be charged with inventory holding cost,  $h$ . Let  $G(ax_t)$  be the summation of the holding and backorder penalty costs for period  $t$  with on-hand inventory level  $ax_t$ , mathematically:

For using the fill rate as a constraint,

$$G(ax_t) = h(ax_t - D_t - bx_{t-1})^+ \quad (4.2.1)$$

For calculating backordering cost,

$$G(ax_t) = h(ax_t - D_t - bx_{t-1})^+ + b \cdot bx_t \quad (4.2.2)$$

where  $x^+ = \max \{x, 0\}$

The backorder quantity up to period  $t$ ,  $bx_t$ , is calculated as:

$$bx_t = \begin{cases} 0, & \text{if } ax_t \geq bx_{t-1} + D_t \\ bx_{t-1} + D_t - ax_t, & \text{else} \end{cases} \quad (4.3)$$

The fill rate achieved (the fraction of demand that is satisfied immediately from stock on hand),  $fr$ , is calculated as:

$$fr = \frac{\sum_{t=1}^T \min\{D_t, (ax_t - bx_{t-1})^+\}}{\sum_{t=1}^T D_t} \geq S \quad (4.4)$$

The inventory level at the end of period  $t$ , which is used to determine the next order quantity is calculated as follows:

$$x_t = (ax_t - D_t - bx_{t-1})^+ \quad (4.5)$$

The inventory order-up-to levels,  $y_t$ ,  $\forall t$ , are determined according to the subject inventory policy,  $(R, Q)$  policy, used in the system. The  $(R, Q)$  policy works as follows: when the inventory position (inventory on hand plus outstanding orders) declines to or below  $R$ , a fixed quantity of product units ( $Q$ ) is ordered. The decision variables thus are  $R$  and  $Q$  for the considered company. For the  $(R, Q)$  policy, the inventory order-up-to levels are determined as follows:

$$y_t = \begin{cases} x_t, & \text{if } x_t + \sum_{j=t-L+1}^{t-1} (y_j - x_j) > R \\ x_t + Q, & \text{else} \end{cases} \quad (4.6)$$

The objective is to minimize the total inventory-related cost over the entire planning horizon. The objective function is described next. Let  $\delta(x)$  be an indicator function as follows:

$$\delta(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \quad (4.7)$$

Denote by  $v_t(x)$  the total cost of the company with inventory level  $x$  at period  $t$ . The following equations are obtained:

$$v_t(x_t) = A\delta(y_t - x_t) + G(ax_t) \quad t = 1, \dots, T \quad (4.8)$$

Given the initial system state, the optimal system cost over the time horizon is

$$\text{Min } TC / \text{period} = \frac{\sum_{t=1}^T v_t(x_t)}{T} \quad (4.9)$$

For the fill-rate constrained optimization problem, it is to optimize the above equation under the constraint of Eq. (4.4).

## 4.2 ACO<sub>R</sub> for determining optimal inventory policy

Optimization algorithms inspired by the ants' foraging behavior (Dorigo, 1992) is one of those efficient metaheuristics for solving combinational optimization problems (COPs). It has been applied to many COPs such as travelling salesman (Dorigo, 1997a,b), assignment problem (Costa and Hertz, 1997; Stützle and Hoos, 2000; Wagner, 2000), vehicle routing (Dorigo, 1999; Reimann et al., 2004), feature selection (Liao, 2009), and project scheduling (Duan and Liao, 2010), just to name a few. Since determining an optimal  $(R, Q)$  policy deals with finding optimal combinations of  $R$  and  $Q$ , it can be represented as COP in a straightforward way. However, to be more compatible with those theoretical models in which both  $R$  and  $Q$  are assumed to be continuous variables, it is chosen to implement ACO<sub>R</sub> proposed by Socha and Dorigo (2008). ACO<sub>R</sub> is an extension of the original ACO to real or continuous domains without any major conceptual change to its structure. ACO<sub>R</sub> has been proven to be a very competitive approach. According to the experimental results reported by Socha and Dorigo (2008), ACO<sub>R</sub> has been shown to outperform several other continuous optimization methods.

The fundamental difference between ACO and ACO<sub>R</sub> is the shift from using a discrete probability distribution to a continuous one, i.e., a probability density function (PDF). The PDF employed by Socha and Dorigo (2008) is a Gaussian kernel PDF,  $G^i(x)$ . A Gaussian kernel PDF is defined as the weighted sum of several one-dimensional Gaussian functions  $g_l^i(x)$  as follows:

$$G^i(x) = \sum_{l=1}^k \omega_l g_l^i(x) = \sum_{l=1}^k \omega_l \frac{1}{\sigma_l^i \sqrt{2\pi}} e^{-\frac{(x-\mu_l^i)^2}{2\sigma_l^{i2}}} \quad (i = 1, \dots, N) \quad (4.10)$$

For each dimension  $i = 1, \dots, N$  of the problem, there exists a different Gaussian kernel PDF  $G^i$ . For each such  $G^i$ ,  $k$  number of solutions are maintained in the solution archive together with their objective function values. The Gaussian kernel  $G^i(x)$  as shown above is parameterized with three vectors of parameters:  $\omega$  is the vector of weights associated with the individual Gaussian functions;  $\mu^i$  is the vector of means; and  $\sigma^i$  is the vector of standard deviation. The solutions in the solution archive are used to calculate the values of these parameters, and hence shape the Gaussian kernel PDF used to guide the ants in their search processes. The solution in the archive are ranked according to their quality (ties are broken randomly). The weight  $\omega_l$  of the solution  $s_l$  is calculated by the following formula:

$$\omega_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}} \quad (4.11)$$

Where  $q$  is a parameter related to the standard deviation of the Gaussian function.

After computing the weight vector  $\omega$ , the sampling process is accomplished in two phases.

Phase one involves choosing one of the Gaussian functions that compose the Gaussian Kernel.



The probability  $p_l$  of choosing the  $l$ th Gaussian function is given by:

$$p_l = \frac{\omega_l}{\sum_{r=1}^k \omega_r} \quad (4.12)$$

Phase two is to sample the chosen Gaussian function. To this end, the value of the standard deviation  $\sigma_l^i$  at construction step  $i$  is calculated by multiplying the parameter  $\xi$  with the average distance from the chosen solution  $s_l$  to other solutions in the archive:

$$\sigma_l^i = \xi \sum_{e=1}^k \frac{|s_e^i - s_l^i|}{k-1} \quad (4.13)$$

The parameter  $\xi > 0$ , which is the same for all the dimensions, is similar to the pheromone evaporation rate in ACO. The higher is the value of  $\xi$ , the lower the convergence speed of the algorithm.

In order to present the ACO<sub>R</sub> algorithm tailored for determining optimal  $(R, Q)$  inventory policy, the key parameters used in ACO<sub>R</sub>-based algorithms are first defined. The following are the key algorithmic parameters employed in the ACO<sub>R</sub> algorithm.

<b>Key parameters used in ACO<sub>R</sub>-based algorithms</b>	
$k$	The size of the archive
$NumAnt$	Number of ants
$N$	Number of dimensions of the considered problem
$q$	Parameter ranging [0,1] that controls intensification vs. diversification (.7,.8,.9)
$\xi$	The higher the value of $\xi$ , the lower the convergence speed of the algorithm (.7,.8,.9)
$Maxeval$	Maximum number of evaluations for stopping each run
$NumRun$	Number of runs

The pseudo code of the implemented ACO<sub>R</sub> algorithm is given below.

---

**ACO<sub>R</sub> for determining optimal inventory policy**

- (1) Load the demand data. Both algorithm-related and problem-dependent parameters are first initialized.
- (2) Randomly generate  $k$  initial solutions within the bounds of the variables, evaluate and rank them in the archive according to the objective value. Each initial solution is evaluated by the evaluation module to compute its objective function value. In the fill rate constrained case, the actual fill rate is also computed and recorded for each solution generated. Set the number of evaluations to be  $k$  (i.e.,  $eval = k$ ). These  $k$  initial solutions are then computed for their relative weight in the archive according to Eq. (4.11). The probability for choosing each solution in the archive is computed according to Eq. (4.12).
- (3) While  $eval < Maxeval$  do
  - (3.1) For  $i = 1: NumAnt$ ;
    - (3.1.1) Select a solution from the archive by roulette selection based on the probability computed in Eq (4.12)
    - (3.1.2) The standard deviation associated with the selected solution is then computed according to Eq (4.13).
    - (3.1.3) A new trial solution is constructed by updating the selected solution by  $\pm randn \cdot \sigma_i$ , with  $randn$  being a normal random value.
    - (3.1.4) Check if any variable of the new solution is outside of the upper/lower bound. If so, the new trial solution is repaired by either randomly generating a new value within the bound or setting it to the bound value (with equal probability).
    - (3.1.5) The new trial solution is evaluated by the evaluation module to compute its objective function value. In the fill rate case, the fill rate value associated with the solution is also computed.
    - (3.1.6) Update the archive if the trial solution is better than the worst one in the archive

for each run. In the fill rate case, if the trial solution satisfies the pre-specified fill rate constraint and the objective function value is better than the worst one in the archive, the worst existing solution is replaced by the new solution

End for

(3.2) Update the best solution found so far

End while

Output the result of the optimal solution and its objective value. In the fill rate case, the corresponding fill rate is also reported.

---

Two versions of  $ACO_R$  are implemented with one optimizes  $(R, Q)$  subject to a fill rate constraint and the other optimizes  $(R, Q)$  with backorder penalty cost. For either version, the criterion used to determine the quality of a solution is the total cost per unit time. The size of the archive is consistently set equal to 10 times of the number of dimensions of the problem, i.e., 20.

The fill rate constraint in the first version of  $ACO_R$  is handled based on the parameter-less constraint handling method proposed by Deb (2000). This method was originally implemented in a genetic algorithm, which uses a tournament selection operator, where two solutions are compared at a time, and the following criteria are always enforced:

1. Any feasible solution is preferred to any infeasible solution.
2. Among two feasible solutions, the one having better objective function value (lower total cost per unit time) is preferred.
3. Among two infeasible solutions, the one having smaller constraint violation is preferred.

The superiority of this method lies in that penalty parameters are not needed because in any of the above three scenarios, solutions are never compared in terms of both the objective values and the amounts of constraint violation. This method avoids the difficulty in setting a good penalty coefficient and therefore is adopted here as a reliable and efficient constraint handling method. Note that this parameter-less constraint handling method has been successfully used in differential evolution based algorithms in a recent study (Liao, 2010).

For the second version of  $ACO_R$  the backorder penalty cost simply constitutes another term in the total cost expression. No other special technique is needed to deal with it. The evaluation module is modified to include the backorder penalty cost and is integrated into the  $ACO_R$  algorithm to search for optimal solution. This version of  $ACO_R$  thus implements an unconstrained optimization problem.

As mentioned earlier, the evaluation module is a key component of the proposed approach in order for the  $ACO_R$  algorithm to find the optimal solution. The evaluation module essentially implements the inventory policy of concern, continuous review  $(R, Q)$  replenishment policy with backordering in this study. The implementation is based on a period-by-period updating mechanism, starting from the first time point to the last time point. For each period, the sequence of events relevant to the inventory policy is implemented. A key input to the evaluation module is the demand data actually collected from the real world and does not have to fit into any distribution at all.

---

**The evaluation module for fast-moving nonperishable items**

Input: Demand data,  $R$  and  $Q$  values, and cost parameters

Output: TC/period

(1) For  $t = 1: T$

(1.1) Updating the on-hand inventory by receiving due order: Eq. (4.1)

(1.2) Fulfill the backorder demand first from the on-hand inventory. Fulfill the current demand if there are enough on-hand inventory otherwise backorder current demand.  
The corresponding holding cost and backorder cost is calculated in Eq. (4.2).

(1.3) Calculate the backorder quantity according to Eq. (4.3)

(1.4) Update the inventory level according to Eq. (4.5)

(1.5) Determine whether to order a new batch of  $Q$  units or not based on Eq. (4.6).

(1.6) The ordering cost is applied if a new order is issued and calculate the total cost for period  $t$ , Eq. (4.8)

End for

(2) All costs incurred during the whole process are added to the total cost function Eq. (4.9) and the fill rate is computed according to Eq. (4.4) if needed.

---

### 4.3 Experimental details and results

In this section, the experimental details for evaluating the performance of the proposed approach are first presented, followed by the test results. The two versions of  $ACO_R$ -based algorithms are implemented in such a way so that they approximate to the two corresponding theoretical models as much as possible:  $ACO_R$  with backorder penalty cost versus Rosling(2002) and  $ACO_R$  with fill rate constraint versus Axsäter (2006).

In order to obtain a comprehensive overview of the performances of the two ACO<sub>R</sub>-based algorithms, various demand distributions and models which can be roughly divided into five groups below, are tested.

- Constant demand
- Normally distributed demand with constant mean and varying standard deviation
- Normally distributed demand with varying degree of imperfect fit
- Non-normal distributions
- Auto-correlated real world data

#### 4.3.1 Experimental details

Several experimental details should be clarified. First, steady-state inventory system is considered in this study, which is assumed by both theoretical models. In parameter initialization of ACO<sub>R</sub>, the initial inventory level ( $x_0$ ) is set to be the steady-state inventory level as calculated by the corresponding theoretical approach (note that some tuning of  $x_0$  might be necessary if  $(R, Q)$  values are restricted to integers; refer to Section 4.4.5 for more details). Nevertheless, the proposed ACO<sub>R</sub> is capable of determining an optimal  $(R, Q)$  policy with any initial inventory level. The steady-state inventory level is set because it is assumed by both theoretical models.

Secondly, the target fill rate ( $S$ ) in the theoretical model is set as the minimum fill rate in the proposed approach. However, the fill rate values reported in the following tables are the actual fill rates ( $Afr$ ) achieved. The actual fill rate of the optimal solution found may be slightly higher than the target fill rate ( $S$ ) due to the discrete nature of demand data points. By employing the

constraint handling method by Deb (2000), only those solutions that satisfy the target fill rate (solutions with  $Afr \geq S$ ) are feasible and will be qualified to compete for the final optimal solution. Thirdly, each set of demand data inputted to the ACO<sub>R</sub> is generated by a Minitab® random number generator designed for a given type of distribution without dependency among data points. It is chosen to generate 1000 data points for each demand set so as to simulate the long-run total cost per unit time. The 1000 demand points forms a demand pattern over 1000 time periods that follow a specified distribution. For example, if it is to test a demand pattern that follows normal distribution, a data set is generated by using the Minitab® normal random number generator by specifying its target mean ( $\mu$ ) and standard deviation ( $\sigma$ ). The value of each data point represents the demand size ordered at that particular time point.

Several differences between the random number generator and the theoretical assumption should be noted: (1) it is impossible to generate a data set that strictly follows the theoretical assumption, i.e., a data set generated by a normal random number generator will never be ideally normally distributed; (2) the actual computed  $\mu$  and  $\sigma$  of the data set will be more or less different from the pre-specified values of  $\mu$  and  $\sigma$  set to generate the data in the first place. Therefore, in order to minimizing the difference in modeling, a data set that approximately follows the target values of  $\mu$  and  $\sigma$  is first generated. After the data set is generated, they are checked by probability testing and the actual mean and standard deviation are calculated. The computed values are then used as input to the theoretical models to calculate the results. For example, considering the first data set in benchmark problem 4.3.2.3, the actual mean of the data

set is 50.08 instead of 50 and the actual standard deviation of the data set is 18.72 instead of 20. When it comes to compute the theoretical  $(R^*, Q^*)$  and its total cost per unit time, it is chosen to use  $\mu = 50.08$  and  $\sigma = 18.72$  even though this data set is initially generated by  $N(50, 20)$ . This explains any discrepancy between the resulting  $(R^*, Q^*)$  reported here and the  $(R^*, Q^*)$  reported in Axsäter (2006). This adjustment is needed to ensure that the proposed  $ACO_R$  and its corresponding theoretical approach are compared on equal footing. Nevertheless, it is important to note that this difference cannot be totally eliminated. In other words, it is always present and must be kept in mind in the subsequent analyses of experimental results.

All the following benchmark problems are tested using the following set of parameter values:  $N=2$  (2 variables,  $R$  and  $Q$ ),  $NumRun=30$ ,  $NumAnt=30$ ,  $q=0.7$ ,  $\xi=0.7$ ,  $Maxeval=1000$  (those algorithmic values are selected based on empirical testing results, for more detailed discussion on parameter value selection, please refer to section 4.4.1),  $S=0.9$ ,  $A=100$ ,  $h=2$ ,  $b=20$  and  $L=4$  (those inventory related parameters are the values that used by the corresponding theoretical model).

### **4.3.2 Experimental results**

#### **4.3.2.1 Constant demand**

In this subsection, the simplest type of demand distribution, constant demand, is considered. The average demand ( $\mu$ ) is constant for each testing data set and the values range from 10 per period to 100 per period. For each test data set, results are compared with those theoretical results and testify the validity of the proposed approach. Note that the two theoretical approaches were developed to handle stochastic demand and not for handling constant demand (with zero



standard deviation). To make them work for constant demand, the standard deviation,  $\sigma$ , is set at a very small value, i.e. 0.0000001, so that the demand can be approximated to constant. Table 4.1 summarizes the experimental results obtained. The left hand side of the table shows the comparison between the proposed and Axsäter's approach while the right hand side shows the comparison between the proposed and Rosling's approach.

For each test problem, the optimal  $R^*$  and  $Q^*$  found and its corresponding TC/period is first reported. Each value under the “%” column is percent deviation of  $ACO_R$  from the theoretical model, computed as  $\frac{TC/period(ACO_R) - TC/period(theoretical)}{TC/period(theoretical)} \times 100$ . In Table 4.1, the TC/period is further partitioned into holding cost, ordering cost, and fill rate/backorder penalty cost for more detailed comparison. Such detailed cost breakdown is omitted in the remaining tables to save space and for each testing problem, only  $R^*$ ,  $Q^*$ , and TC/period are reported and the corresponding percent deviation in TC/period is recorded under the “%” column.

Table 4.1 shows that in either case, the proposed approach constantly offers good solutions: The optimal  $(R, Q)$  policy values found by  $ACO_R$  are always close to the theoretical optimal values; the total cost per unit time (TC/period) for each constant demand tested is within  $\pm 5\%$  difference of the theoretically calculated value, with the smallest difference to be slightly less than 0.1%. The proposed approach is capable of providing comparable results to its counterparts.

In the backorder cost case, the performance of the proposed approach keeps generating comparably good solutions as Rosling's approach. Note that since there is no variation in the average demand, both approaches incur no backorder penalty. In the fill rate constrained case,

the solutions obtained by  $ACO_R$  achieve higher fillrates (0.91 to 1) than the target value (i.e., 0.9) specified in Axsäter' approach, leading to slightly higher total cost. This can be explained by the discrete nature of the proposed approach; specifically the continuous review model is approximated by 1,000 time points. In Section 4.4.4, it is confirmed that  $ACO_R$  indeed does find the optimal solution and there is no better one around the neighborhood.

Table 4.1. Optimal  $(R, Q)$  policy found by each approach for constant demand (benchmark problem 4.3.2.1).

	ACOR		Axaster's model		%		ACOR		Rosling's model		%
	$(R^*, Q^*, TC/period)$	$Afr$	$(R^*, Q^*, TC/period)$	S			$(R^*, Q^*, TC/period)$	$(R^*, Q^*, TC/period)$			
1. $\mu=10, \sigma=0$	(36, 30; 59.33)	1	(36.49, 35.14; 56.92)	0.9	4.238	1. $\mu=10, \sigma=0$	(36, 30; 57.42)	(36.98; 33.17; 57.29)	0.232		
2. $\mu=25, \sigma=0$	(94, 55; 94)	0.909	(94.44, 55.56; 90)	0.9	4.444	2. $\mu=25, \sigma=0$	(95, 50; 95.2)	(95.23, 52.44; 90.58)	5.102		
3. $\mu=50, \sigma=0$	(192, 70; 133.43)	1	(192.14, 78.57; 127.28)	0.9	4.831	3. $\mu=50, \sigma=0$	(195, 75; 127.3)	(193.26, 74.16; 127.42)	-0.094		
4. $\mu=100, \sigma=0$	(389, 100; 184)	1	(388.89, 111.11; 180)	0.9	2.222	4. $\mu=100, \sigma=0$	(400, 100; 185.2)	(390.47, 104.88; 181.16)	2.231		
Cost partition											
1. Constant Average Demand=10											
	ACOR		Axaster's model		%		ACOR		Rosling's model	%	
TC/period	59.33		56.92		4.238	TC/period	57.42		57.29	0.232	
holding cost	26.03		28.46		-8.528	holding cost	24.12		27.14	-0.111	
ordering cost	33.3		28.46		17	ordering cost	33.3		30.15	0.104	
						backorder cost	0		0	0	
2. Constant Average Demand=25											
	ACOR		Axaster's model		%		ACOR		Rosling's model	%	
TC/period	94		90		4.444	TC/period	95.2		90.58	5.102	
holding cost	48.5		45		7.778	holding cost	45.2		42.91	5.347	
ordering cost	45.5		45		1.111	ordering cost	50		47.67	4.88	
						backorder cost	0		0	0	
3. Constant Average Demand=50											
	ACOR		Axaster's model		%		ACOR		Rosling's model	%	
TC/period	133.43		127.28		4.831	TC/period	127.3		127.42	-0.094	
holding cost	62.03		63.64		-2.531	holding cost	60.6		60	1	
ordering cost	71.4		63.64		12.194	ordering cost	66.7		67.42	-1.068	
						backorder cost	0		0	0	
4. Constant Average Demand=100											
	ACOR		Axaster's model		%		ACOR		Rosling's model	%	
TC/period	184		180		2.222	TC/period	185.2		181.16	2.231	
holding cost	84		90		-6.667	holding cost	85.2		85.81	-0.713	
ordering cost	100		90		11.111	ordering cost	100		95.35	-4.881	
						backorder cost	0		0	0	

A notable observation is that employing the optimal  $(R, Q)$  policy determined by  $ACO_R$ , the inventory levels over time all show a cyclic pattern after a transition period of irregularity lasting only for a few time points in the beginning. As an illustration, Figure 4.1 shows the cyclic pattern in the first 100 time units for the case constant demand equaling to 25. In the first few time units,

the inventory levels do not seem to show any pattern, but after that a clear replenishment cycle emerges. This cycle then repeats itself throughout the remaining time, which indicates that the inventory control system is in steady state. This again verifies the validity of the proposed  $ACO_R$  inventory optimization algorithms. Similar patterns were observed in all other demand streams tested; but the corresponding figures are omitted due to space restriction.

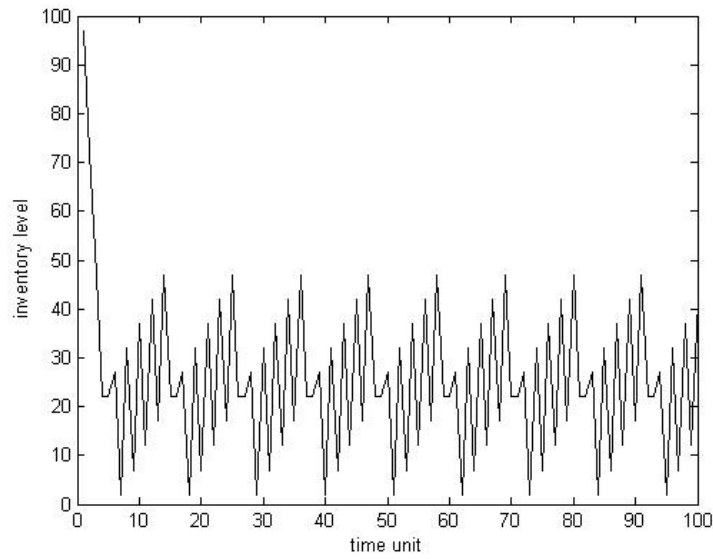


Figure 4.1. Cyclic pattern of on-hand inventory for constant demand = 25.

#### 4.3.2.2 Normally distributed demand with constant mean and varying standard deviation

In this subsection, different levels of variation to the average demand are added and the performances of the proposed  $ACO_R$ -based approaches as well as the corresponding theoretical approaches are compared. The effect of variation is evaluated by varying standard deviation at four different levels, i.e.,  $\sigma \in \{0, 5, 10, 20\}$ . Since the possibility of negative demand is not considered in this study, 20 is the maximum value of  $\sigma$  for average demand of 50 considered here. The benchmark problem 4.3.2.2 in Table 4.2 summarizes the test results.

Generally, the same trend can be observed in both  $ACO_R$ -based and theoretical approaches when taking varying degree of uncertainty in the lead-time demand into account. They both react by increasing  $R$  and  $Q$  as the value of  $\sigma$  is increased.

Table 4.2. Testing results for various benchmark problems in section 4.3.2

Benchmark problem 4.3.2.2										
		ACOR		Axsäter's model			ACOR		Rosling's model	
		( $R^*, Q^*$ ; TC/period)	$Afr$	( $R^*, Q^*$ ; TC/period)	S	%	( $R^*, Q^*$ ; TC/period)	( $R^*, Q^*$ ; TC/period)		%
1	N(50, 0)	(192, 70; 133.43)	1	(192.14, 78.57; 127.28)	0.9	4.83	(195, 75; 127.3)	(193.26, 74.16; 127.42)		-0.09
2	N(50.05, 5.089)	(192, 101; 136.48)	0.902	(193.54, 82.34; 131.4)	0.9	3.87	(203, 71; 154.24)	(195.03, 78.76; 147.18)		4.8
3	N(50.07, 9.965)	(198, 106; 151.07)	0.901	(198.48, 88.82; 144.17)	0.9	4.78	(202, 106; 174.67)	(200.88; 84.19; 169.59)		2.997
4	N(50.76, 19.85)	(205, 124; 182.41)	0.903128	(215.77, 102.6; 181.998)	0.9	0.22	(208, 121; 226.46)	(220.19, 95.6; 226.05)		0.16
Benchmark problem 4.3.2.3										
		ACOR		Axsäter's model			ACOR		Rosling's model	
		( $R^*, Q^*$ ; TC/period)	$Afr$	( $R^*, Q^*$ ; TC/period)	S	%	( $R^*, Q^*$ ; TC/period)	( $R^*, Q^*$ ; TC/period)		%
1	N(50.08, 18.72) p=0.856	(210, 112; 176.84)	0.900402	(211.32, 100.62; 176.64)	0.9	0.115	(212, 124; 218.59)	(215.51, 93.89; 218.58)		0.0047
2	N(50.76, 19.85) p=0.715	(205, 124; 182.41)	0.903128	(215.77, 102.6; 181.998)	0.9	0.22	(208, 121; 226.46)	(220.19, 95.6; 226.05)		0.16
3	N(50.84, 19.86) p=0.419	(211, 124; 190.54)	0.900503	(216.09, 102.67; 182.12)	0.9	4.63	(217, 121; 228.48)	(220.52, 95.67; 226.24)		0.9899
4	N(50.69, 19.26) p=0.117	(207, 134; 190.38)	0.901606	(214.52, 101.79; 179.47)	0.9	6.08	(211, 127; 228.2)	(218.83, 94.92; 222.47)		2.58
Benchmark problem 4.3.2.4										
		ACOR		Axsäter's model			ACOR		Rosling's model	
		( $R^*, Q^*$ ; TC/period)	$Afr$	( $R^*, Q^*$ ; TC/period)	S	%	( $R^*, Q^*$ ; TC/period)	( $R^*, Q^*$ ; TC/period)		%
1	Weibull 1 ~N(50.2, 19.18) with p=0.021	(203, 136; 187.24)	0.9	(212.53, 101.31; 178.66)	0.9	4.8	(215, 123; 230.27)	(216.82, 94.47; 221.47)		3.97
2	Weibull 2 ~N(49.64, 19.14) with p<0.005	(203, 134; 191.84)	0.901	(210.33, 100.83; 177.94)	0.9	7.8	(203, 130; 231.09)	(214.61, 94.02; 220.63)		4.74
3	Logistic ~N(51.39, 20.65) with p<0.005	(214, 136; 201.18)	0.903	(219.5, 104.09; 185.97)	0.9	8.18	(210, 127; 247.57)	(224.09, 96.91; 231.59)		6.9
4	Exponential ~N(49.6, 52.13) with p<0.005	(252, 180; 321.87)	0.903	(273.87, 134.5; 332.81)	0.9	-3.29	(223, 137; 416.6)	(284.12, 122.84; 436.57)		-4.57
Benchmark problem 4.3.2.5										
		ACOR		Axsäter's model			ACOR		Rosling's model	
		( $R^*, Q^*$ ; TC/period)	$Afr$	( $R^*, Q^*$ ; TC/period)	S	%	( $R^*, Q^*$ ; TC/period)	( $R^*, Q^*$ ; TC/period)		%
1	M3-N1879 ~N(7529, 1601) with p=0.299	(32893, 8821; 15036.66)	0.923611	(33189.33, 2224.64; 9018.82)	0.9	66.73	(32503, 8742; 18879.08)	(33453.44, 2016.95; 12086.92)		56.19
2	M3-N1894 ~N(4124, 495.7) with p<0.005	(17348, 4572; 6425.46)	0.902778	(17227.02, 1242.76; 3136.04)	0.9	104.89	(17280, 4719; 6982.69)	(17323.73, 1134.35; 4119.98)		69.48
3	M3-N1882 ~N(6022,881) with p<0.005	(25660, 7056; 12162.4)	0.9375	(25583.41, 1698.64; 5216.87)	0.9	133.14	(26048, 7119; 12751.32)	(26048, 7119; 12751.32)		83.98

The proposed  $ACO_R$  generally gives a higher  $Q^*$  value and slightly costly solution than the theoretical approach due to the difference in modeling as mentioned earlier. One major deviation from this general pattern is noted when  $\sigma = 20$ . In this particular case, the reorder point of  $ACO_R$ -based approaches is relatively lower than that of the theoretical approach. This is probably because the increase in ordering quantity in this situation has already been sufficient to cover the fluctuation in the lead-time demand and there is no need to hold unnecessary high

stock. In a way,  $ACO_R$  brilliantly balances between holding and ordering cost and picks the solution that minimizes the total cost per unit time.

#### **4.3.2.3 Demand with imperfect fit to normal distribution**

Since both theoretical models are originally designed to handle normal demand distribution, four data sets with varying degree of fit to normal distribution,  $\sim N(50, 20)$ , are generated and tested in order to compare the performances of both  $ACO_R$ -based approaches and theoretical approaches. The testing data sets are arranged in descending order of fit to normal distribution (in terms of  $p$ -value), i.e, starting with well fit, fair fit, and finally lousy fit. The first data set resembles normal distribution the most, with a  $p$ -value of 0.856. This means that it is very likely (85.6%) to observe a value of the test statistic at least as extreme as that which has been observed, if  $H_0$  is true ( $H_0$ : the data set is normally distributed). The second data set resembles normal distribution with a  $p$ -value of 0.715, followed by the third data set with a  $p$ -value of 0.419. The fourth data resembles normal distribution the least, with  $p$ -value of only 0.117. Since 0.117 is still strictly larger than 0.05, there is no evidence to reject the normal hypothesis with a confidence of 95%. All four approaches, Rosling (2002), Axsäter (2006) and two versions of  $ACO_R$ -based algorithms, are applied to the above-mentioned four data sets. The paired comparison results are provided in Table 4.2, under the heading “benchmark problem 4.3.2.3”.

As mentioned earlier, the first data set resembles normal distribution the most; therefore, it fits the assumption of the theoretical model the best. For this dataset, the  $ACO_R$  results are very close to the theoretical results; the percent differences in total cost per unit time between  $ACO_R$

and both theoretical approaches are only 0.115% and 0.0047%, respectively. The results indicate that the new approach is comparable to Rosling (2002) and Axsäter (2006) under the normal distribution assumption. As the distribution of the demand data deviate more away from the assumption, the percent difference increases from 0.115% up to 6.08% for the fill rate constrained case and from 0.0047% up to 2.58% for the backorder penalty case. This is expected due to the fact that the theoretical model is defined under the assumption of normal distribution while the proposed  $ACO_R$  approach is not affected by the assumption at all and is handling all cases approximately equally well.

There seems to be a trend in the increase of total cost per unit time as the  $p$ -value decreases. The proposed  $ACO_R$  approach deals with decreasing  $p$ -value by recalculating the  $R^*$  and  $Q^*$  values, while the theoretical approach seems to ignore the  $p$ -value changes (note that their  $R^*$  and  $Q^*$  for different cases are similar because there is little changes in the  $\mu$  and  $\sigma$  values). To minimize the total cost per unit time, the  $ACO_R$  approach balances the cost components by taking into account the relative magnitude of unit holding cost and ordering cost ( $h=2$  vs.  $A=100$ ). Note that the order quantity is adjusted higher to reduce the need to order frequently as  $p$ -value decreases in proper consideration of the relative lower holding cost and higher ordering cost. Such an adjustment is reasonable and common in most practical inventory systems. On the other hand, the theoretical models do not have any mechanism to adjust their optimal solutions in response to the worsening fit.

The theoretical model is proven to be strictly correct in the mathematical sense only under ideal assumptions. Another assumption made is that after receipt of an order all outstanding backorders are satisfied and consequently the probability of a shortage immediately after receipt is negligible. To reveal the inadequacy of theoretical model to handle real-world data, the inventory level is analyzed by applying the optimal policy values found by the theoretical model to the  $ACO_R$ -based approach with fill rate constraint. The results shown in Figure 4.2 indicate that the inventory system based on theoretically found values missed orders in succession for all 4 datasets. Since the stock on hand is too tight, starting from the first stock-out, the inventory system is always one unit time behind to meet the current demand, leading to unacceptable low fillrate. If the order quantity is insufficient to cushion the following demand, the probability of a shortage after immediate order receipt cannot be ignored. As shown above, all fill rates attained by theoretical  $(R^*, Q^*)$  in Figure 4.2 are below the specified value of 0.9. In other words, the optimal solution found by the theoretical model for each dataset is actually infeasible, as shown through the evaluation module of the proposed approach.

What theoretical approaches suggest is an ideal circumstance. It requires strong conforming to the assumption. Under the required fillrate constraint, holding more stocks than that indicated by the theoretical value is necessary; if it is not complied, then the inventory system may run the risk of missing orders successively. The same applies to the case when considering backorder penalty cost rather than fillrate (refer to the right hand side of the table that shows the comparison between Rosling and  $ACO_R$ ). Note that the unit backorder penalty is ten times more

expensive than unit holding cost ( $b=20$  vs.  $h=2$ ). When minimizing the total cost, it is reasonable for  $ACO_R$  to favor holding more stock over being penalized. The ordering frequency is accordingly reduced by the same reason.

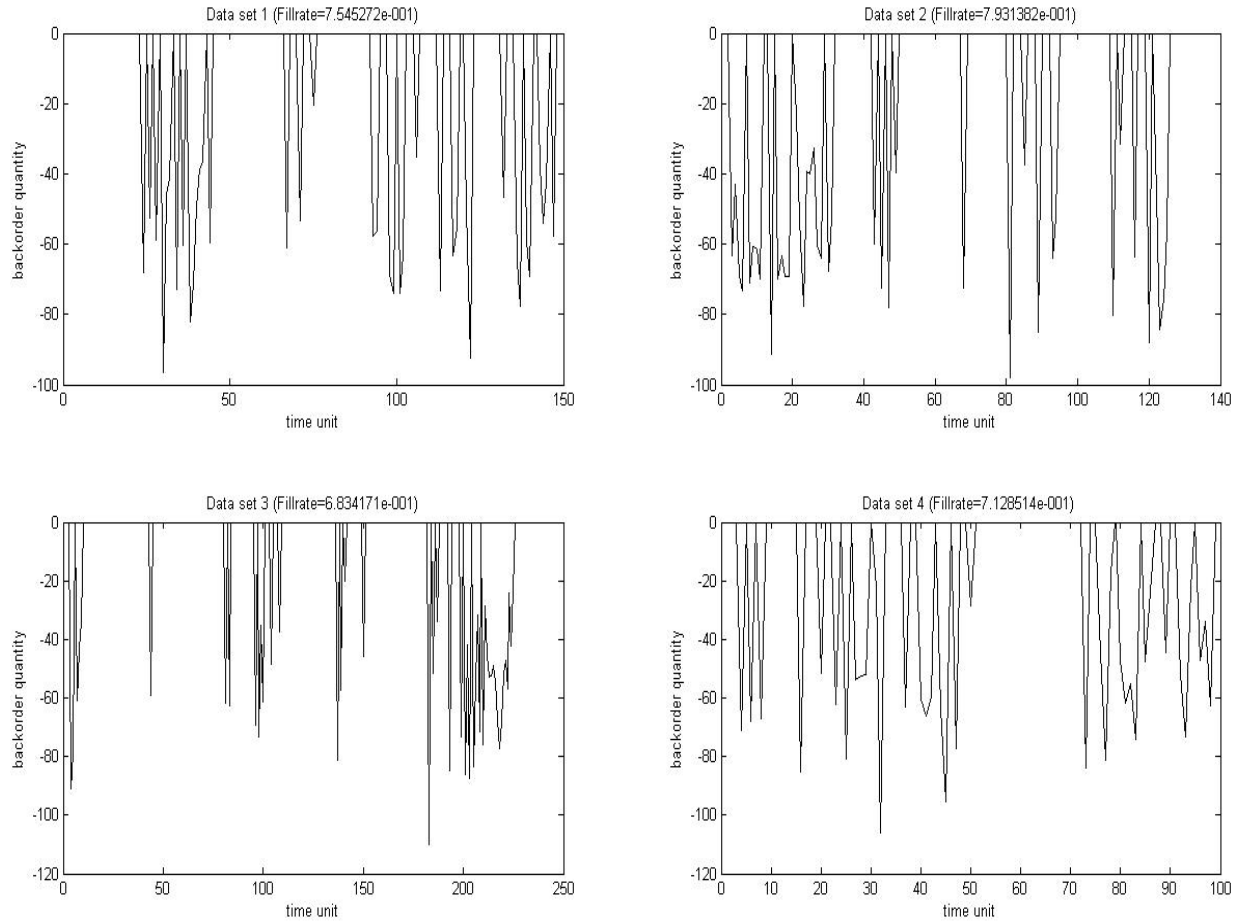


Figure 4.2. Backlog quantity profiles (or orders missed) due to out-of-stock based on theoretically found  $(R^*, Q^*)$  values for all four datasets corresponding to Benchmark problem 4.3.2.3.

#### 4.3.2.4 Non-normally distributed demand

In this subsection, all four approaches are tested with non-normally distributed datasets generated from three distributions, specifically Weibull, logistic distribution and exponential



distribution. All these data sets clearly violate the assumption of normality made in Rosling (2002) and Axsäter (2006). For sake of comparison, the generated data is fit into normal distribution anyway and try to estimate their most appropriate  $\mu$  and  $\sigma$  values to be used in the theoretical models. This would happen if one blindly applies these theoretical models without paying attention to the normal demand assumption.

Data sets 1 and 2 are generated from Weibull distribution rather than normal distribution and they resemble normal distribution very less. Weibull distribution is chosen over other types of distribution due to the fact that Tadikamalla (1978) has shown that Weibull distribution can adequately be used to present the lead time demand. Weibull distribution can present unimodal demand distributions ranging from monotonically decreasing to heavily skewed to normal type distributions. It is a versatile and widely-used distribution and that it can take on the characteristics of other types of distributions, based on the value of the shape parameter chosen. The corresponding  $p$ -value of the normal probability test for data sets 1 and 2 are 0.021 and  $<0.005$ , respectively. Data set 1 can be regarded as a borderline set. Its  $p$ -value is 0.021, which means we can decide to reject  $H_0$  or not based on the significance level chosen. For those who require stronger evidence against  $H_0$  (say  $\alpha = 0.01$ ), they may fail to reject  $H_0$  and conclude that the distribution still resembles normal distribution in some way; but for those prefer somewhat higher significance level ( $\alpha = 0.05$ ), they may reject  $H_0$  and conclude that the distribution cannot be regarded as normal distribution. For data set 2, it is virtually not normally distributed due to its extreme small  $p$ -value ( $<0.005$ ). The  $p$ -values of probability distribution testing for the last

two distributions are both  $<0.005$ , which means they both deviate from normal distribution to a large extent. These data sets are tested to show the versatility of the proposed  $ACO_R$  approach, i.e., its applicability to demand data from any type of distribution in order to find optimal  $(R, Q)$  policy. Table 4.2 summarizes the corresponding results obtained, under heading “benchmark problem 4.3.2.4”.

There is no point to investigate the percent difference here since the difference between the results of  $ACO_R$  and its corresponding theoretical model is expected to be significant. The  $ACO_R$  approach treats each of these datasets as it is while the theoretical approaches still treat it as normal. Therefore, the results obtained by the two theoretical approaches are not expected to be trustworthy. They are presented for the sake of completeness. Obviously, those theoretical approaches become useless when facing various types of non-normal demand distributions.

#### **4.3.2.5 Auto-correlated demand**

To further show the value of the proposed approach in practice, three time-series data are taken from the industrial category of the M3 Competition data (Makridakis and Hibon, 2000), i.e., N1879, N1882 and N1894, for testing. The M3 Competition data organizes real-world data into various subcategories (Micro, Industry, Macro, Finance, Demographic and Other), which was originally created to evaluate the performance of forecasting method submitted to the competition. Datasets are selected from the industrial category because it better fits the application. All three datasets were found to be auto-correlated after being tested with the autocorrelation functionality available in Minitab®. The demand stream of N1879 is detected to

have a quadratic trend:  $Y_t = 9820 - 78.7t + 0.4888t^2$ . Likely, N1882 is found to have a quadratic trend with  $Y_t = 4832.4 + 8.22t + 0.08502t^2$  and N 1894 for trend  $Y_t = 4303 - 13.24t + 0.1118t^2$ . All these three datasets are auto-correlated with 5% significance limits. For these three datasets, the assumption of i.i.d. in Rosling(2002) and Axsäter (2006) is clearly violated and we cannot rely on those theoretical approaches to obtain reliable solutions. The test results are given in Table 4.2, under the heading “benchmark problem 4.3.2.5”. The theoretical results tend to underestimate the order quantity and hence the total cost per unit time. On the other hand, the proposed  $ACO_R$  approaches offer themselves as an efficient and simple alternative to both theoretical approaches.

The huge percent difference of results between  $ACO_R$  and its corresponding theoretical approach is not unexpected. The first time-series data set, N1879, somewhat follows normal distribution with a  $p$ -value of 0.229. There is no coincidence that it happens to have the smallest percent difference, among all three tested. However, even this smallest percent difference is too large to be acceptable. As shown above, both theoretical approaches fail to account for the auto-correlation in the data sets. There are too many restrictions on the data for the theoretical approaches to function well, which severely impairs their practical usage in industry.

#### 4.4 Discussion

As a meta-heuristic, the performance of  $ACO_R$  depends on the complexity of the problem search space and its parameter settings. Too complicated problem search space and poor parameter settings will affect the speed of convergence and the quality of final results eventually

found. In this section, discussion will primarily focus on five important issues. A design experiment for investigating the effects of ACO<sub>R</sub>-related algorithmic parameters is first carried out, followed by examining the effect of inventory-related algorithmic parameters. Thirdly, the computational effort of different approaches is discussed and the convergence profile taken by the ACO<sub>R</sub>-based algorithm in determining the optimal inventory policy is investigated, aiming to provide an in-depth view of how it works. Fourthly, the neighborhood of the optimal solution for a selected data set is explored to understand the interaction between search space complexity and algorithm's search ability. Lastly, the effect of allowing the ACO<sub>R</sub>-based algorithms to take on real  $(R, Q)$  values, instead of integer values, is studied.

#### **4.4.1 Effect of ACO<sub>R</sub>-related algorithmic parameters**

The search power of an Ant-based algorithm is directly determined by its algorithmic parameters. Generally speaking, the higher the values of *NumAnt*, *NumRun* and *Maxeval*, the more powerful search it will perform. On the other hand, the longer CPU time it will take to find the optimal solution. Values of  $q$  and  $\xi$  are actually trying to balance between intensification vs. diversification. There is no a global trend for the choice of  $q$  and  $\xi$  since some problem sets need more intensification to find the optimal solution while others may rely on more diversification. Among those parameters, one parameter may have more significant effect than another on improving the search process. Therefore, it will be helpful to understand the effect of the algorithmic parameters and reach a good compromise. To this end, the effects of five ACO<sub>R</sub> parameters are tested according to the Taguchi design. The Taguchi experimental design is a

useful tool for choosing a good set of parameter values. In this optimization problem, the objective is to minimize total cost per unit time, a performance characteristic which is the smaller the better. Therefore, a higher Signal-to-Noise (S/N) ratio as defined below corresponds to a better parameter combination.

$$S/N_i = -10 \times \log \left( \sum_{j=1}^n \frac{y_{i,j}^2}{n} \right) \quad (4.14)$$

In Eq. (4.14),  $n$  is the number of trials for each combination;  $y_{i,j}$  is the result of the  $j$ -th trial in the  $i$ -th combination. As discussed below, in the experimental design,  $n=3$  and  $i=1,2,3,\dots, 27$ .

Table 4.3 lists the levels of each factor. According to the convergence profile presented in Section 4.4.3, it is observed that the original empirical values for the algorithmic parameters ( $NumRun=30$ ,  $NumAnt=30$ ,  $q=0.7$ ,  $\xi=0.7$ ,  $Maxeval=1,000$ ) are already sufficient to find the optimal solutions.

Table 4.3. Signal levels and codes of factors

Factor	Level 1	Level 2	Level 3
$q$	0.7	0.8	0.9
$\xi$	0.7	0.8	0.9
$NumRun$	10	20	30
$NumAnt$	10	20	40
$Maxeval$	100	500	1000

The ACO<sub>R</sub> algorithm converges stably to the optimal solutions for all problem sets. From this point on, raising their corresponding values too much does not seem necessary. Each parameter is designed to have three levels to capture the nonlinear behavior. For each level, those

values chosen should be far apart from each other enough so that the search power of different combination of parameters could be distinguished. Based on this principle, the specific values for each level have been decided as presented. Since there are five factors (parameters) and three levels for each factor, the L27 orthogonal array is chosen; the details are omitted to save space. Accordingly, the corresponding combinations of parameters were tested. Each experimental condition is repeated three times and computed the average S/N ratio. After calculating the S/N ratio for each combination, the average S/N value is calculated for each level of each factor. Tables 4.4 and 4.5 summarize the results for the two proposed ACO<sub>R</sub>-based approaches: with fill rate constraint and with backlog penalty cost, respectively.

Table 4.4. S/N ratios of five three-level factors of ACO<sub>R</sub> with fill-rate constraint.

Level	q	$\xi$	NumRun	NumAnt	Maxeval
1	35.699	<b>35.806</b>	33.872	32.826	9.146
2	34.507	35.182	35.201	<b>37.095</b>	44.352
3	<b>36.276</b>	35.494	<b>37.41</b>	36.561	<b>52.985</b>
Delta	1.769	0.624	3.538	4.269	43.839
Rank	4	5	3	2	1

Table 4.5. S/N ratios of five three-level factors of ACO<sub>R</sub> with backorder penalty cost.

Level	q	$\xi$	NumRun	NumAnt	Maxeval
1	44.37	44.37	44.37	42.35	22.13
2	43.39	43.39	43.39	44.63	<b>55.67</b>
3	<b>45.71</b>	<b>45.71</b>	<b>45.71</b>	<b>46.50</b>	<b>55.67</b>
Delta	2.32	2.32	2.32	4.15	33.55
Rank	3	3	3	2	1

The best combination of parameter settings for each version of ACO<sub>R</sub> is, as highlighted in bold in the table,  $q(3)-\xi(1)-NumRun(3)-NumAnt(2)-Maxeval(3)$  for the fillrate case and

$q(3)-\xi(3)-NumRun(3)-NumAnt(3)-Maxeval(2/3)$  for the back order penalty cost case. These codes represent  $q=0.9$ ,  $\xi=0.7$ ,  $NumRun=30$ ,  $NumAnt=20$ ,  $Maxeval=1000$  for the fillrate case and  $q=0.9$ ,  $\xi=0.9$ ,  $NumRun=30$ ,  $NumAnt=40$ ,  $Maxeval=500$  or  $1000$  for the backorder penalty cost case. It can also be seen that the parameter that has the largest effect on the solution evolution is *Maxeval* in both cases.

A fine meta-heuristic design can balance the trade-off between intensification and diversification so that the search is guided towards the global optimum without getting stuck in local optimums. It is well-known that meta-heuristic is especially good at tackling NP-hard problems. Usually such problems are handled by heuristic methods (not exact methods) since it is not possible to find efficient (i.e., polynomial time) algorithms to solve them optimally. In those cases, when it is permitted to find approximate good enough solutions (probably not exactly optimal) in reasonable amount of time, metaheuristics like ACO<sub>R</sub> as employed here almost always live up to the expectation. However, it is important to emphasize the stochastic nature of ACO<sub>R</sub>, or any meta-heuristic in general. Unlike any deterministic algorithm, the result of a stochastic algorithm varies from run to run. The ability of a meta-heuristic to find the global optimal solution often depends on the complexity of the search space that is problem dependent as well as the algorithmic parameters chosen. Although there is no guarantee that the global optimal solution will be found, it has been shown in many studies that a well-designed meta-heuristic often finds the global optimal solution if known.

#### 4.4.2 Effect of inventory-related parameters

In this subsection, three additional sets of experiments are performed to investigate the effect of inventory-related parameters one variable at a time. The effects of relative weights of cost parameters are studied by varying the ordering cost at three levels ( $A=0.02$ , 2 and 100). In addition, the effects of lead time and fill rate are studied by using three order lead times ( $L=1$ , 10 and 100) and three fill rates ( $S=0.99$ , 0.85 and 0.6), respectively. For each set of parameters, the optimal values of  $R$  and  $Q$  as well as the long run total cost per unit time. are reported. The results are presented in Table 4.6.

The results indicate that (1) increasing lead time from 1 to 10 time units does not affect the performance of each algorithm much; however, (2) when lead time is set to be very long (100 time units), the percent difference between the theoretical model and the proposed algorithm is increased drastically, over 10%.

Table 4.6. Optimal ( $R$ ,  $Q$ ) policies found under different inventory-related parameters

Average demand $\sim N(50.76, 19.85)$ ; $h=2$ ; $A=100$ ; $b=20$ ; $S=0.9$								
	ACOR		Axsäter's model		%	ACOR		Rosling's model
	$(R^*, Q^*; TC/period)$	$Afr$	$(R^*, Q^*; TC/period)$	$S$		$(R^*, Q^*; TC/period)$	$(R^*, Q^*; TC/period)$	%
$L=1$	(50, 116; 147.99)	0.9	(48.82, 89.28; 144.82)	0.9	2.19	(57, 102; 175.45)	(51.21, 84.65; 170.2)	3.08
$L=10$	(544, 122; 238.57)	0.901	(541.01, 116.22; 233.35)	0.9	2.23	(525, 110; 282.79)	(547.7, 107.1; 298.78)	-5.35
$L=100$	(5147, 196; 679.4)	0.9	(5245.48, 166.66; 573.39)	0.9	18.49	(5042, 159; 889.47)	(5271.77; 151.3; 765.05)	16.26
Average demand $\sim N(50.76, 19.85)$ ; $h=2$ ; $L=4$ ; $b=20$ ; $S=0.9$								
	ACOR		Axsäter's model		%	ACOR		Rosling's model
	$(R^*, Q^*; TC/period)$	$Afr$	$(R^*, Q^*; TC/period)$	$S$		$(R^*, Q^*; TC/period)$	$(R^*, Q^*; TC/period)$	%
$A=0.02$	(238, 51; 114.89)	0.901	(251.12, 5.7; 105.78)	0.9	8.61	(259, 55; 153.69)	(253.65, 5; 143.19)	7.33
$A=2$	(236, 57; 116.64)	0.901	(241.65, 26.4; 111.29)	0.9	4.8	(259, 55; 155.5)	(244.89, 23.92; 149.28)	4.16
$A=100$	(205, 124; 182.41)	0.903	(215.77, 102.6; 181.998)	0.9	0.22	(208, 121; 226.5)	(220.19, 95.6; 226.1)	0.16
Average demand $\sim N(50.76, 19.85)$ ; $h=2$ ; $L=4$ ; $A=100$ ; $b=20$								
	ACOR		Axsäter's model		%			
	$(R^*, Q^*; TC/period)$	$Afr$	$(R^*, Q^*; TC/period)$	$S$				
$S=0.99$	(261, 92; 265.61)	0.99	(266.994, 89.87; 274.56)	0.99	-3.26			
$S=0.85$	(197, 129; 166.49)	0.851	(202.29, 107.81; 160.92)	0.85	3.46			
$S=0.6$	(111, 219; 115.77)	0.603	(137.02, 146.61; 99.4)	0.6	16.47			



In order to deal with the fluctuated demand during the long lead time,  $ACO_R$  orders more, leading to much higher total cost per period than the theoretically-calculated value.

The three ordering cost levels were selected to examine three scenarios: ordering cost is considerably smaller than holding cost; ordering cost is same as holding cost; ordering cost is much higher than holding cost. It is observed that the percent difference is gradually reduced when the relative weight of ordering cost going from low to high. The ordering quantity of those theoretical models is shown very sensitive to the change in ordering cost. When facing low ordering cost ( $A=0.02$ ), theoretical models tend to order very small quantity frequently. As the ordering cost goes up, theoretical models increase ordering quantity rapidly to reduce ordering frequency. It seems that  $ACO_R$  does not react as quickly and  $ACO_R$  has the tendency to order more as cushion to varying demand in order to minimize number of backorders.

Lastly, the results of varying fill rates indicate that (1) when fill rate is relatively high, i.e., over 0.85, the percent difference between different approaches is small; however, (2) when the target fill rate is set extreme low, say 0.6, the percent difference is magnified. For all the fill rate cases, the actual fill rates ( $Afr$ ) achieved by the proposed  $ACO_R$  algorithm are always slightly higher than the target fill rates ( $S$ ) because of the discrete nature of the proposed approach. The two cases where relatively large % differences occur, i.e. very long lead time and very low service level, are rare situations in practice. Nevertheless, a more in-depth examination of the relatively large % difference is needed.

#### 4.4.3 Computational effort of different approaches and convergence profiles of the proposed $ACO_R$ algorithms

Throughout the experiment, the CPU time needed by each algorithm to converge to optimal solution is recorded. Since both Rosling (2002) and Axsäter (2006) are originally designed for normally-distributed demand, only the CPU time tested on those normally-distributed demand problem sets are computed for the average CPU time. For the fill-rate case, the average CPU time needed for Axsäter (2006) is 35.50 seconds while the corresponding  $ACO_R$  algorithm takes 23.54 seconds (based on 1000 evaluations). For the backorder cost case, the average CPU time employed by Rosling (2002) is 11.64 seconds while the corresponding  $ACO_R$  algorithm requires 24.08 seconds (based on 1000 evaluations). The relatively long CPU time needed for Axsäter (2006) is due to the search for the right  $R$  to satisfy the required fill rate, referring to Eq. (13) in Axsäter (2006). The iterative technique proposed by Rosling (2002) employs the shortest CPU time on the problem set probably because the method only includes solving two equations iteratively. The proposed  $ACO_R$  algorithm needs to perform problem-dependent search each time and its efficiency is directly related to its algorithmic parameters. It has been shown in Section 4.4.1 that the parameter that has the largest effect on the solution evolution is *Maxeval*. Therefore, the convergence profiles of the proposed  $ACO_R$  algorithms are exported and presented next for a better understanding of the ant-based search.

The convergence profiles of the two proposed  $ACO_R$  approaches in searching for the optimal solution for Sample 2 in benchmark problem 4.3.2.3 are shown in Figures 4.3 and 4.4.

The algorithmic parameters used are  $q=0.7$ ,  $\xi =0.7$ ,  $NumRun=30$ ,  $NumAnt=30$ , and  $Maxeval=1000$  (these values are all empirical values and have not been tuned for any specific problem set). In each figure, the profiles of both the average and the best solution among all 30 runs are plotted to trace the search performed by  $ACO_R$  and to see how those solutions converge to the optimal solution as the number of evaluations increases. The two figures shown, and all other figures not shown, indicate that the proposed algorithms can converge very quickly (approximately 300 evaluations on average). Note that the above CPU statistics for  $ACO_R$  are based on 1000 evaluations; the CPU value reported has been actually overestimated. However, it is not common to set *Maxeval* of a metaheuristic to be the exact number of evaluation needed (it is also impossible to do that because no one really knows exactly how many evaluations will be needed to converge before hand). Almost all metaheuristics will allow a few more evaluations to ensure that the search converges stably to a global solution and no better solution will be found any more. By setting *Maxeval* to be 500 instead of 1000, the average CPU time needed for  $ACO_R$  algorithm is reduced to 11.59 seconds and still found the optimal solutions for all problem sets. Based on the above fact, it is believed that the computational effort needed by the proposed  $ACO_R$ -based approaches is modest in both cases.

The convergence profiles also reveal the fact that the considered inventory optimization problem cannot be regarded as a complicated optimization problem, comparing to other combinatorial optimization problems that  $ACO_R$ , or more generally metaheuristics, are coping with (most of them may need over thousands of evaluations before locate a good solution).

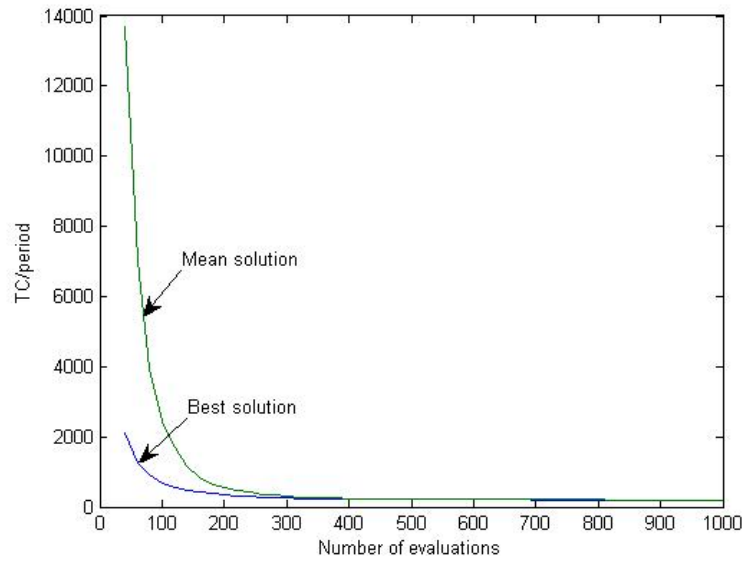


Figure 4.3. Converge profiles of best and mean solution for sample 2 in benchmark problem 4.3.2.3 (the  $ACO_R$  with fillrate constraint)

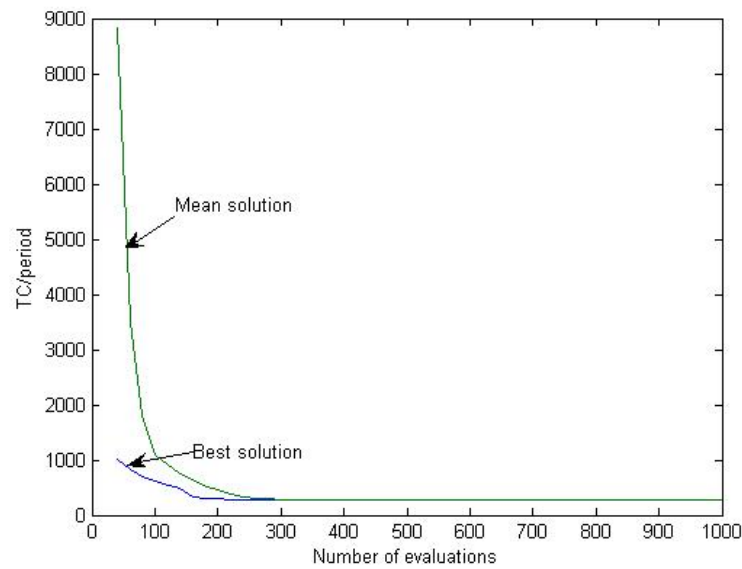


Figure 4.4. Converge profiles of best and mean solution for sample 2 in benchmark problem 4.3.2.3 ( $ACO_R$  with backorder penalty cost)

Another very important reason for recommending  $ACO_R$  or other metaheuristic is that it is best suited for integrating with the evaluation module that does not set any restrictions on lead-time demand. Theoretical approaches set assumptions on lead-time demand due to the fact

that analysis of stochastic model is hard and messy. The cost function will be extremely complex and difficult to work with analytically if the demand is erratic. The proposed  $ACO_R$ -based algorithms circumvent the need for demand parameter fittings, which enables them to handle wide scope of problems, especially those of practical relevance. Therefore, based on all the above experimental analyses, it is argued that the proposed  $ACO_R$ -based algorithms are very promising alternatives to the  $(R, Q)$  type of inventory control optimization.

#### **4.4.4 Neighborhood of optimal solutions**

The second dataset of benchmark problem 4.3.2.3 is used here to show the neighborhood of optimal solutions found both by  $ACO_R$ -based approaches and theoretical approaches. Through this example, it is aimed to explore the interaction between search space complexity and proposed algorithm's search ability in more detail. To this end, it is necessary to compute and compare the total cost per unit time for various  $(R, Q)$  combinations in the neighborhood of optimal solutions. As mentioned earlier, difference exists in the values of  $(R^*, Q^*)$  found by  $ACO_R$  and those theoretically computed (refer to Tables 4.1 and 4.2 for the  $R^*$  and  $Q^*$  values). To show that the  $ACO_R$  algorithm does indeed find the optimal solution and does not miss any other better solution, the neighborhood of the optimal solutions found by the proposed approach and the theoretical approach is meshed. For each combination of  $(R, Q)$  in that neighborhood, the total cost per unit time (TC/period) is computed using the evaluation module of the proposed  $ACO_R$  algorithm. Figure 4.5 presents the 3-D topography of the considered problem searched by the  $ACO_R$  algorithm with fill rate constraint. The topography is shown continuous but actually

discrete because that  $R$  and  $Q$  values are restricted only to integers in the proposed approach. A discussion of relaxing them to real values is given in Section 4.4.5.

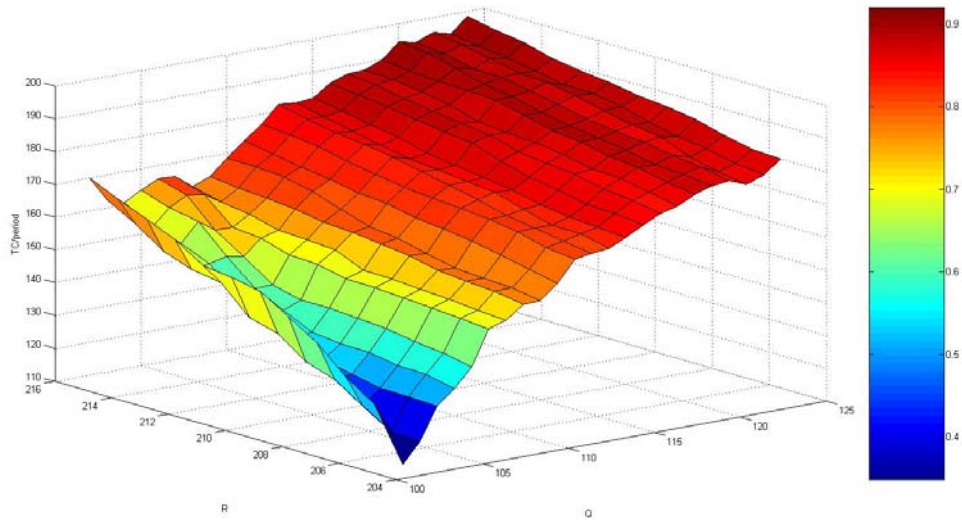


Figure 4.5. Neighborhood of optimal solution of sample 2 for the benchmark problem 4.3.2.3 using fill rate as color bar

In Figure 4.5, solutions attain high fill rate are shaded with deep red. Areas shaded other than deep red are infeasible even though the total cost per unit time maybe lower. Among all possible combinations of  $(R, Q)$  in the meshed area, only 19 combinations are feasible. It may not be easy to distinguish feasible/infeasible solutions from colors alone, Table 4.7 is thus prepared to list all 19 combinations, together with the corresponding TC/period and fill rate. It is clear that the solution found by the  $ACO_R$  algorithm is the best in the neighborhood. As reported in Section 4.3.2.3, the  $(R^*, Q^*)$  value found by the theoretical model is infeasible when checked by the evaluation module of the proposed approach.

Similarly, the 3-D plot of the topography of the considered problem when searched by the  $ACO_R$  algorithm with backorder penalty cost is presented in Figure 4.6. In the figure, the total

cost per period is presented by both z-axis and the color bar. Combinations with low cost are shaded with deep blue. Among all  $(R, Q)$  combinations in the meshed area, the combination of  $(208, 121)$  has the minimum cost, 226.4643. By no coincidence, it is reported by  $ACO_R$  as the optimal solution. Actually, the second least costly solution is  $(209, 121)$ , with total cost per period of 226.9426. The theoretical solution of  $R^*=220.1932$  and  $Q^*=95.6023$  is nowhere near to be the optimal because its assumption of normality is not perfectly met by the test dataset.

Table 4.7. Feasible solutions of the second data set of benchmark problem 4.3.2.3

$(R, Q)$	TC/period	Actual Fill Rate
(216, 119)	189.9618	0.90111
(214, 120)	191.3966	0.90111
(215, 120)	193.7157	0.9051463
(216, 120)	194.4883	0.9101917
(216, 122)	195.3332	0.9021191
(215, 123)	192.7338	0.9021191
(216, 123)	193.9489	0.9041372
<b>(205, 124)</b>	<b>182.4067</b>	<b>0.9031282</b>
(206, 124)	182.7988	0.9041372
(207, 124)	184.6557	0.9041372
(208, 124)	186.9982	0.9051463
(209, 124)	188.3473	0.9071645
(210, 124)	189.168	0.9091826
(211, 124)	190.6199	0.9122099
(212, 124)	191.7906	0.913219
(213, 124)	192.9786	0.9152371
(214, 124)	195.3264	0.9182644
(215, 124)	196.6918	0.9233098
(216, 124)	198.3246	0.9243189

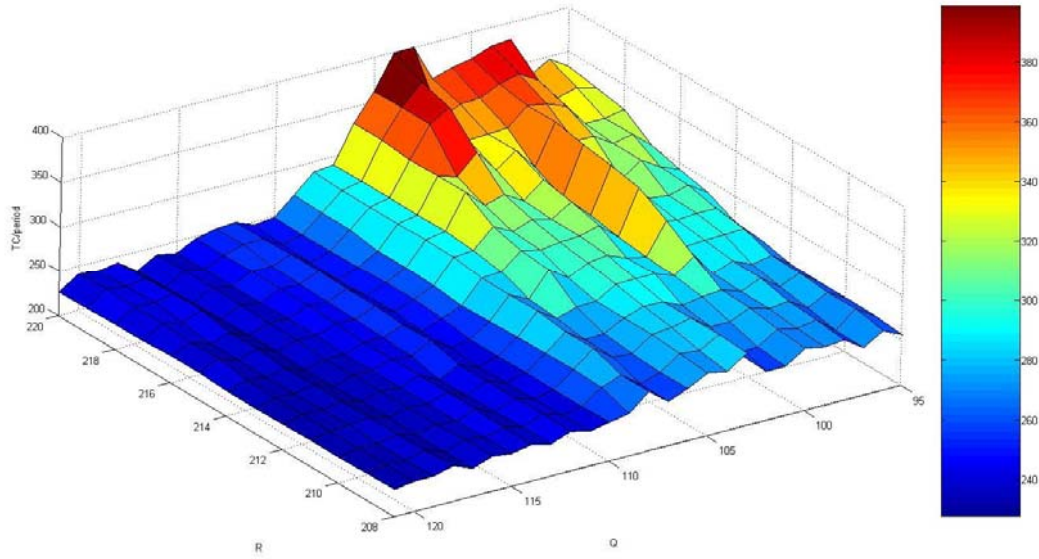


Figure 4.6. Neighborhood of optimal solution of sample 2 for the benchmark problem 4.3.2.3 using TC/period as color bar

In conclusion, in both the fill rate and backorder cost cases  $ACO_R$ -based algorithms are able to carry out effective and efficient search so that the optimal solution is always found in hundreds of evaluations. The same evaluation is applied to other data sets presented in the previous section. The search ability of the proposed algorithm is proven to be consistently reliable.

#### 4.4.5 Real $(R, Q)$ versus integer $(R, Q)$

In all of the above experiments, only integer  $(R, Q)$  values are considered because of their practical relevance: When dealing with inventory, the number of products is almost always counted in integer number (yet another unreasonable assumption made by theoretical models). Although considered impractical,  $ACO_R$  based on real  $(R, Q)$  is also tested for comparison with the theoretical results. For this investigation, four data sets used in Section 4.3.2.2 are tested by



ACO<sub>R</sub> allowing real ( $R$ ,  $Q$ ) values using the same algorithmic parameters as those used in Section 4.3. Recall that these four data sets represent demands following approximate normal distribution with the same mean but different variances. Table 4.8 summaries the results obtained.

Table 4.8. Computational results found by ACO<sub>R</sub> allowing real ( $R$ ,  $Q$ ) values on the same data sets used in Table 4.2.

Benchmark problem 4.3.2.2								
	ACOR		Axsäter's model			ACOR	Rosling's model	
	( $R^*$ , $Q^*$ ; TC/period)	$Afr$	( $R^*$ , $Q^*$ ; TC/period)	S	%	( $R^*$ , $Q^*$ ; TC/period)	( $R^*$ , $Q^*$ ; TC/period)	%
N(50, 0)	(196.09, 83.34; 132.98)	1	(192.14, 78.57; 127.28)	0.9	4.48	(199.99, 74.997; 127.43)	(193.26, 74.16; 127.42)	0.009
N(50.05, 5.089)	(194.07, 101.57; 137.39)	0.906	(193.54, 82.34; 131.4)	0.9	4.56	(198.09, 103.05; 152.29)	(195.03, 78.76; 147.18)	3.47
N(50.07, 9.965)	(192.11, 110.85; 149.43)	0.902	(198.48, 88.82; 144.17)	0.9	3.65	(195.51, 110.15; 172.27)	(200.88, 84.19; 169.59)	1.58
N(50.76, 19.85)	(207.22, 125.41; 182.95)	0.9111	(215.77, 102.6; 182)	0.9	0.52	(215.72, 124.89; 220.32)	(220.19, 95.6; 226.09)	-2.55

By setting the initial inventory level as the theoretical inventory level, ACO<sub>R</sub> based on real ( $R$ ,  $Q$ ) values constantly provides good solutions under different degree of uncertainty in lead-time demand. Comparing Table 4.2 with Table 4.8, it is not difficult to see that solutions with lower TC/period are found when allowing ( $R$ ,  $Q$ ) to take on real values rather than integers. Hence, the percent differences of TC/period for some data sets are shown smaller values than those using integer ( $R$ ,  $Q$ ) values. This may be due to the fact that allowing real ( $R$ ,  $Q$ ) values adds much more possible ( $R$ ,  $Q$ ) combinations to the search space, among them some possibly have lower total cost per period. Note that for the last data set, the solution found by ACO<sub>R</sub> costs even less than the theoretical value in the backorder cost case. Since ACO<sub>R</sub> always performs a problem-specific search while theoretical approaches only provide a general solution to one type of problem that fit certain parameters, it is possible for certain combination of real ( $R$ ,  $Q$ ) found by ACO<sub>R</sub> to fit the specific problem better than the general case.

Last but not least, it is important to note that the shift from integer values to real values enlarges the scope of search space to a great extent. It is natural that the search ability of  $ACO_R$  allowing real  $(R, Q)$  values should be enhanced accordingly in order to find the optimal solution when facing complicated problem spaces. This implies that it might be necessary to tune some of the algorithm-related parameters such as increasing number of ants, number of evaluations or number of runs and etc.

#### 4.5 Conclusion

A new computational approach that integrates an evaluation module based on a period-by-period updating mechanism and a proven meta-heuristic,  $ACO_R$ , has been presented for determining the optimal  $(R, Q)$  policy for single-echelon inventory system with complete backorder. The backorder is treated in two ways: one with fillrate constraint and another with penalty cost in order to make a comparison with two most recent theoretical approaches for tackling the same problem. The proposed  $ACO_R$ -based approaches were tested with five groups of benchmark problems. The test results indicate that the proposed approaches generally performed well comparing to its corresponding theoretical approaches for those datasets that fit the normal distribution assumption well. The proposed approaches are advantageous over those theoretical approaches when applied to datasets ill conforming to the normal and iid assumptions. The proposed approach extends the scope of inventory data a single algorithm can handle and fits the very need of most industrial practice. Moreover, the computational complexity of the algorithm is modest comparing to the other NP-hard problems that  $ACO_R$  is tackling. Therefore,

the proposed  $ACO_R$ -based computational approach offers an interesting alternative to existing theoretical approaches for practical use. The proposed approach allows more flexibility in modeling realistic inventory problems.

## CHAPTER 5 OPTIMIZATION OF SLOW-MOVING PERISHABLE ITEMS

### 5.1 The problem

In this section, the inventory system is extended to single-echelon single-product perishable inventory system. Both the backorder case and the lost sale case are considered. The inventory system is controlled by a continuous review  $(S-I, S)$  policy.  $S$  is the order-up-to-level, which is provided to protect against stockouts. If replenishment is instantaneous, the optimal  $S$  would be zero. Obviously, with a positive leadtime, the optimal  $S$  is usually positive. The  $(S-I, S)$  policy dictates that whenever an item is consumed by demand, a reorder is placed immediately for that unit. This restores inventory position (the total of stock on-hand plus stock on order minus backorders) to the spare stock level,  $S$ .

The proposed computational approach is designed to deal with demand generated by any stochastic process. For comparison purpose, of particular interest is demand generated by a stationary Poisson process with mean equaling to  $\lambda$ . Therefore, the time between demand arrivals is exponentially distributed with mean of  $1/\lambda$ . The time interval formed by exponential distribution can take on any nonnegative real value. The only input required by the proposed approach is the time interval data which indicates the interval of successive unit-demand arrivals. At each of these time points, one unit-demand is required by the customer and a unit is immediately triggered for replenishment. The order will arrive in a fixed lead time  $L$ .

Each perishable item has a fixed lifetime of  $M$  units of time ( $M > L$ ). This means that each item may retain for  $M$  units of time to satisfy demand after which it must be discarded. Since the aging of the product starts at the time order is placed, each item actually stays in stock for only  $M-L$  units of time. After  $M-L$  length of time, the unused item will be perished and trigger another order to the supplier. The items stayed in stock are subject to holding cost  $h$  per unit item and unit time. The cost for a unit to perish is  $p$ . In the backorder case, a backorder cost per unit  $\bar{b}$  will be charged for each unmet demand and ordering cost will not be considered (as in the case that items are considered to be so expensive that the ordering costs are negligible compared to the value of the items). In the lost sale case, a lostsale cost per unit  $\bar{L}$  will be charged for each unsatisfied demand and replenishment will be assessed by an ordering cost  $c$ . Note that the different treatments of ordering cost are dictated by the theoretical models to be compared.

## 5.2 Mathematical model for the considered problem

Expressions for costs of replenishing, shortage, outdating and carrying are built up based on an event-driven updating mechanism. The inventory level is continuously reviewed in which the review period is set to be the time advances by events of demand consumption, ordering receiving, ordering and outdating. The subscript  $e$  represents the corresponding event occurrence during the planning horizon. It should be emphasized here that they are not evenly distributed over the planning horizon but are driven by the customer demand, in which the time between demand arrivals is assumed to be exponentially distributed. Demand size is one unit per time. A customer demand will trigger a new order for sure in the  $(S-I, S)$  order-up-to policy. Therefore,

the following event of order receiving and outdating for this replenishment can be determined by adding the corresponding lead time/lifetime ( $L/M$ ).

<b>Subscripts</b>	
$e$	Event $e$ in the planning horizon, $e=1,2, \dots$
<b>Notations</b>	
$T_e$	Current time when event $e$ happens
$L$	Leadtime
$M$	Fixed lifetime
$c$	Fixed ordering cost
$h$	Holding cost per product unit and unit time
$\bar{b}$	Backorder penalty cost per unit
$\bar{L}$	Lost sale cost per unit
$l$	Target fill rate
$p$	Perish cost per unit
$S$	Order-up-to level
$TC/period$	Average total cost per unit time

When considering perishable items, a state description (age distribution) is required so that once replenishment decision has been taken, the state transitions and the associated cost during the next period are uniquely determined in stochastic sense. In this study, a vector  $P$  is used to record the time when each item on stock will perish.

$$P = \{P_1, P_2, P_3, \dots, P_i, \dots, P_J\}$$

In which  $J=x_e$ , is a dynamically updating index to record the total number items on hand at current time.

Therefore, each unit on stock has a tag,  $P_i$  to indicate when it will perish. This vector  $P$  is updated and sorted every time as the inventory state changes. Each time a customer demand arrives, the oldest unit ( $P_1$ ) is first retrieved to satisfy the demand, according to the

First-in-first-out (FIFO) policy. Obviously, if the unit is consumed before it becomes outdated it will not perish again so its corresponding perishing time is removed from the vector.

The implementation of the inventory model is based on an event updating mechanism, starting from the first event to the last event. During the whole planning horizon, a timing vector  $T = \{T_1, T_2, T_3, \dots, T_e\}$  is maintained as a dynamic memory that records any time point that may change the inventory state and incur some cost. First, it stores all the time points when unit-demand occurs, which is exactly the content of input data. At the same time, a demand vector  $D$  of the same size of  $T$  is kept in which  $D_e=1$  where  $T_e$  indicate the time point when a unit demand occurs. Secondly, the expected perishing time of on-hand inventory is also tracked. Each consuming or perishing event triggers a new order and the order arriving time is also registered in the dynamic memory. Obviously, it is sufficient to just check those time points kept in the memory since the inventory level does not change at other time points and no costs other than holding cost incurs. For each of those time points, the corresponding event (the arrival of a customer demand, the perishing of a unit in stock, or the arrival of an order issued earlier) is calculated and the resulting cost are added to the cost function. Units in stock between any consecutive time points are subject to holding cost.

The timing vector is dynamically updated throughout and only sufficient numbers of future time points (after the current time) are kept so that the timing vector can be controlled within a reasonably manageable size. The current time index keeps advancing until the last customer demand arrives at  $T_{end}$ .

At the current time  $T_e$ , the holding cost of on-hand inventory,  $x_e$  is first calculated. Let  $H(x_e)$  be the holding cost during the period between two successive events happen.

Obviously,

$$H(x_e) = h \cdot x_e \cdot (T_e - T_{e-1}). \quad (5.1)$$

Then three cases can be distinguished: whether the current time  $T_e$  represents an event of replenishing, consuming or outdating.

(1) If the current time  $T_e$  is the time when a due replenishment comes in, the inventory level will be updated by

$$x_e = x_{e-1} + (y_{e-L} - x_{e-L}) \quad (5.2)$$

The vector  $P$  is also updated simultaneously by adding new items and their corresponding expiration time.

(2) If the current time  $T_e$  indicates a customer walking in ( $D_e=1$ ), the company tries to satisfy with its available on-hand inventory,  $x_e$ . Each customer demand is satisfied by the oldest unit ( $P_l$ ) in stock based on FIFO policy.  $P_l$  is removed from the vector  $P$  once it is consumed. Any shortfall occurred will be backordered ( $\bar{b}$ ) or lost ( $\bar{L}$ ), depending on the case.

Let  $G(x_e)$  be the penalty costs when event  $T_e$  happens, we will have

$$G(x_e) = \begin{cases} 0, & \text{if } x_e \geq bx_{e-1} + D_e \\ -b \cdot bx_e, & \text{else} \end{cases} \quad (5.3.1)$$

If the shortage is backordered



$$G(x_e) = \begin{cases} 0, & \text{if } x_e \geq D_e \\ L \cdot (D_e - x_e) & \text{else} \end{cases} \quad (5.3.2)$$

If the shortage is lost

The backorder quantity up to current time  $T_e$ ,  $bx_e$  ( $bx_0 = 0$ ), is calculated as:

$$bx_e = \begin{cases} 0, & \text{if } x_e \geq bx_{e-1} + D_e \\ bx_{e-1} + D_e - x_e, & \text{else} \end{cases} \quad (5.4)$$

The fill rate achieved,  $fr$ , is calculated as:

$$fr = \frac{\sum_e \min\{D_e, (x_e - bx_{e-1})^+\}}{\sum_e D_e} \geq 1 \quad (5.5)$$

The inventory level after fulfilling customer demand is given by

$$x_e = (x_e - D_e)^+ \quad (5.6)$$

The vector  $P$  is updated accordingly by removing the consumed item.

(3) If the current time  $T_e$  is the time when outdating item is due to expiration ( $P_i = T_e$ ), the vector  $P = \{P_1, P_2, \dots, P_i, \dots, P_{x_e}\}$  will be updated by deleting the outdating item. Let  $I(x)$  be the indicator function as follows:

$$I(x) = \begin{cases} 1, & \text{if } x = T_e \\ 0, & \text{else} \end{cases} \quad (5.7)$$

Then the total number of items due to outdate at current time is  $\sum_i I(P_i)$ . The total number of items on hand will be changed accordingly. For example, the current time  $T_e = 1.15$  and the current inventory state is  $P = \{1.15, 2.23, 4.67, 5.89\}$  with  $x_e = 4$ , which indicate an item on hand is about to expire. Then the vector  $P$  is update as  $\{2.23, 4.67, 5.89\}$  with  $x_e = 3$ .

Let  $O(x_e)$  represents the corresponding outdated cost, then

$$O(x_e) = p \sum_i I(P_i) \quad (5.8)$$

Of course, perishing events can only occur when the item is still on hand. The inventory level after removing the outdated item is given by

$$x_e = x_e - \sum_i I(P_i) \quad (5.9)$$

A customer demand or an outdated item will trigger a new order for sure in the  $(S-I, S)$  order-up-to policy. The  $(S-I, S)$  policy dictates that whenever an item is consumed/expired, a reorder is placed immediately for that unit to raise the inventory level up to  $S$ .  $S$  is the decision variable. The inventory order-up-to levels,  $y_e, \forall e$ , are determined as follows:

$$y_e = \begin{cases} S, & \text{if } x_e < S \\ x_e, & \text{else} \end{cases} \quad (5.10)$$

In this study, the computational approach is compared to two previous work: Schmidt and Nahmias (1985) and Olsson and Tydesjö (2010). For comparison purpose, the objective function is tailored to each model.

For Schmidt and Nahmias (1985), in which they consider lost sale case and include ordering cost, the objective function is as follows:

Denote by  $v_e(x)$  the total cost of the company with inventory level  $x$  at  $T_e$ . The following optimality equation is obtained:

$$v_e(x_e) = c\delta(y_e - x_e) + H(x_e) + G(x_e) + O(x_e) \quad (5.11)$$

Given the initial system state, the optimal system cost over the time horizon is

$$\text{Min } TC / \text{period} = \frac{\sum_e v_e(x_e)}{T_{end}} \quad (5.12)$$

In which  $T_{end}$  is the time when the last customer demand arrives.

For Olsson and Tydesjö (2010), in which they consider backorder case and did not include ordering cost, the objective function is further distinguished by two cases:

Case 1. Consider backorder penalty cost and try to minimize the total cost:

$$v_e(x_e) = H(x_e) + G(x_e) + O(x_e) \quad (5.13)$$

Given the initial system state, the optimal system cost over the time horizon is

$$\text{Min } TC / \text{period} = \frac{\sum_e v_e(x_e)}{T_{end}} \quad (5.12)$$

Case 2. Using target fill rate as a constraint and try to minimize the sum of holding and outdating cost:

$$v_e(x_e) = H(x_e) + O(x_e) \quad (5.14)$$

Given the initial system state, the optimal system cost over the time horizon is

$$\text{Min } TC / \text{period} = \frac{\sum_e v_e(x_e)}{T_{end}} \quad (5.12)$$

$$\text{Subject to } fr = \frac{\sum_e \min\{D_e, (x_e - bx_{e-1})^+\}}{\sum_e D_e} \geq 1 \quad (5.5)$$

### 5.3 ACO<sub>R</sub> for optimizing ( $S$ -1, $S$ ) policies

The ACO<sub>R</sub> optimizer is detailed in section 4.2 so it is not repeated here. The evaluation module which implements the considered problem is presented as follows. When dealing with the fill rate constraint, the parameter-less constraint handling method mentioned earlier (Deb 2000) is employed, only those feasible solutions (solutions which satisfy the specified fill rate) will be qualified to compete for the final optimal solution. All the cost incurred throughout the finite planning horizon will be added up and then divided by the total length of the horizon to estimate the long-run total cost per unit time that is the criterion used by ACO<sub>R</sub> to find the optimal  $S$  value.

The pseudo-code of the evaluation module is given below.

---

#### **The evaluation module for slow-moving perishable items**

Input: Demand data, Lead time,  $S$  values, and cost parameters

Output: Total cost per unit time

- (1) Set the initial inventory equal to  $S$  and record the expected lifetime of the stock on hand;  
Initialize the timing memory ( $T$ ), the initial timing memory will include all unit demand arrival time, the order arriving time and the stock perish time triggered by those demand;  
Sort the timing memory;
- (2) While the current time( $T_e$ )  $\leq$  last time of demand arrival ( $T_{end}$ )
  - (2.1) If  $T_e$  is the perish time for certain item in stock( $P_i=T_e$ )
    - (2.1.1) The perished item is discarded from the stock, Eq. (5.9) together with its corresponding timing memory
    - (2.1.2) A new item is ordered to maintain the inventory level up to  $S$  and its corresponding order arriving time and its perish time is added to the

timing memory, Eq. (5.10)

(2.1.3) The perish cost is charged for the item discarded, Eq. (5.8)

End if

(2.2) If current time is the order arriving time for certain item

(2.2.1) If there is backorder

The coming order is used to fulfill backorder, Eq. (5.4)

Else

The coming order is added to the stock, Eq. (5.2)

Its corresponding perish time is added to the timing memory

End if

End if

(2.3) If current time is the time of unit demand arrival for certain customer( $D_e=1$ )

(2.3.1) If on-hand inventory  $\geq 1$

The oldest item in the stock is used to fulfill the demand, Eq. (5.6)

A new order is correspondingly triggered, Eq. (5.10)

The timing memory is updated to include the ordering arriving time and its expected perish time of the new order

Else

The unit demand is backordered/lost and its corresponding cost is charged Eq. (5.3) and Eq.(5.4)

A new order is therefore triggered, Eq. (5.10) and the timing memory is updated to include the ordering arriving time and its expected perish time of the new order

End if

End if

(2.4) calculate the total cost for current time  $T_e$ , Eq. (5.11), (5.13), (5.14) depends on the case

(2.5) The current time is removed from the timing memory

(2.6) Sort the timing memory

(2.7) The current time is set as the next earliest time point,  $T_{e+1}$

End while

(3) All costs incurred during the whole time period is added to the total cost function and the fill rate is computed if needed, Eq. (5.12) and (5.5).

---

## 5.4 Experimental details and results

In this section, the experimental details for evaluating the performance of ACO<sub>R</sub>-based inventory optimization algorithms are first presented (section 5.4.1), followed by the test results (section 5.4.2). The lost sale version of ACO<sub>R</sub>-based algorithm is tested using the corresponding problem sets in (Schmidt and Nahmias, 1985) and the two backorder versions are tested using problem sets in (Olsson and Tydesjö, 2010). The first and last row of Table 1 in Schmidt and Nahmias (1985) are chosen for comparison because these two rows deal with extreme cases: one with very short leadtime and the other with very short lifetime. By checking these two extreme cases, it is possible to cover a variety of cases in-between. The first and second cases in Olsson and Tydesjö (2010) are chosen to compare due to the fact that these two cases were solved exactly. The comparison is carried out to see how well the proposed approach performs relative to the theoretical approach. Lastly, the proposed approach is tested against data sets that violate the Poisson distribution assumption to provide a comprehensive overview of its performance.

### 5.4.1 Experimental details

In the lost sale case, Poisson intensity, the inventory holding cost per unit and the ordering cost per replenishment are fixed, i.e.,  $\lambda = 50$ ,  $h = 20$ ,  $c = 100$ . The lostsale penalty and outdating cost are varied, specifically  $\bar{L} \in \{150, 200, 300, 600\}$  and  $p \in \{10, 100\}$ . Four values of the product lifetime and two values of the leadtime are considered, specifically  $m \in \{0.01, 0.05, 0.1, 1\}$  and  $L \in \{0.01, 1\}$ . Since aging of the lifetime,  $m$ , in Schmidt and Nahmias (1985) is assumed to start when a unit arrives in the stock, which is different from the lifetime,  $T$ , defined in Olsson and Tydesjö (2010), it is important to find the relationship between  $T$  and  $m$  for the sake of consistency. According to the definition of  $T$  in Olsson and Tydesjö (2010),  $T=m+L$ .

In all the considered problem sets for the backorder case (Olsson and Tydesjö2010), both the inventory holding cost per unit item and unit time and lead time are set at one, i.e.,  $h=1$  and  $L=1$ . The first problem set deals with backorder cost per unit when the optimal values of  $S$  are relatively low. It covers selected combinations of  $\lambda \in \{1,4\}$ ,  $T \in \{2,4\}$ ,  $\bar{b} \in \{4,6\}$  and  $p \in \{5,10\}$ . The second problem set deals with backorder cost per unit when the optimal values of  $S$  are relatively high with selected combinations of  $\lambda \in \{0.5,10\}$ ,  $T \in \{2,10\}$ ,  $\bar{b} \in \{4,30\}$  and  $p \in \{10,50\}$ . As pointed out in Olsson and Tydesjö (2010), the second problem set deals with more extreme values so that the performance of the subject algorithm under a wide range of  $S$  values can be tested. The third problem set considers problems under a specified service level constraint, 1. The problem set covers selected combinations of  $\lambda \in \{0.5,1,4,10\}$ ,  $T \in \{2,4\}$  and  $l \in \{0.9,0.98\}$ .

The input of arrival time data to the  $ACO_R$  is generated by the Minitab® random number generator that follows exponential distribution without dependency among data points, that is, a process in which demand occur continuously and independently at a constant average rate. As mentioned earlier, if the demand is generated by a stationary Poisson process with intensity  $\lambda$ , the time between two successive arrivals of unit demand in this Poisson process follow an exponential distribution with mean equals to  $1/\lambda$ . Therefore, the mean of the input data set that follows exponential distribution is the inverse of the Poisson mean  $\lambda$ . However, the difference between this exponential random number generator and the theoretical assumption should be noted. It is impossible to generate a data set that strictly follows the theoretical assumption; some sampling error is expected. In other words, the data sets generated will never be ideally continuously exponentially distributed with mean exactly identical to the specified value. The same holds true for real world demand data, which can never be expected to strictly follow a certain theoretical distribution. The salient feature of the proposed  $ACO_R$ -based computational approach is that the input is simply the actual demand history, without the need of any assumption of demand distribution or any distribution parameter fitting. At the beginning of the  $ACO_R$ -based algorithm, raw unit-demand arrival-time data is imported and the algorithm itself explores and exploits the search space based on its real-time ants-like intelligence.

In order to approximate the proposed method to its corresponding theoretical models as much as possible, sufficiently large data sets are generated, i.e., 10,000 data points, to simulate the long-run total cost per unit time. For each unit-demand arrival-time data set generated,



probability testing is performed to make sure that the  $p$ -value of the hypothesis test is large enough so that  $H_0$  will not be rejected under some chosen degree of confidence level ( $H_0$ : the subject data set follows exponential distribution with  $\mu = 1/\lambda$ ). The data sets used to run the computational experiments all have a  $p$ -value strictly larger than 0.5, i.e., they all resemble exponential distribution to a large extent. Those measures are needed to ensure that the proposed approach and its corresponding theoretical approach are compared on relatively good equal footing.

The notations of algorithmic parameters and inventory-related parameters used in the proposed ACO<sub>R</sub>-based algorithms are presented as follows:

<b>Algorithmic parameters</b>	
<i>NumAnt</i>	Number of ants
<i>N</i>	Number of dimensions of the considered problem
<i>q</i>	Parameter ranging [0,1] that controls intensification vs. diversification
$\xi$	The higher the value of $\xi$ , the lower the convergence speed of the algorithm
<i>Maxeval</i>	Maximum number of evaluations for stopping each run
<i>NumRun</i>	Number of runs
<b>Inventory-related parameters</b>	
$\lambda$	The mean of Poisson demand process
<i>L</i>	Replenishment lead time
<i>m</i>	Product lifetime in lost sale case
<i>T</i>	Product lifetime in backorder case
<i>c</i>	Fixed ordering cost in the lost sale case

<b>Inventory-related parameters (continued)</b>	
$h$	Holding cost per unit and unit time
$p$	Fixed perishing cost per unit
$\bar{b}$	Backorder penalty cost per unit
$\bar{L}$	Lost sale penalty cost per unit
$l$	Required minimum fillrate

All the following test problems are tested using the following set of parameter values:

$NumRun=10$ ,  $NumAnt=10$ ,  $N=1$  (one variable,  $S$ ),  $q=0.7$ ,  $\xi=0.7$ , and  $Maxeval=100$ .

## 5.4.2 Experimental results

### 5.4.2.1 ACO<sub>R</sub> for the lost sale case

In this subsection, the lost sale version of ACO<sub>R</sub> is tested with selected problems used in Schmidt and Nahmias (1985). The motivation behind this investigation is to compare the solutions of ACO<sub>R</sub> which is a computational approach with the solutions of Schmidt and Nahmias (1985), which uses a theoretical mathematical model. The analytical formulation and solution procedure in Schmidt and Nahmias (1985) must make assumptions about demand distribution, specifically Poisson, while the ACO<sub>R</sub>-based approach does not. Also, it has been mentioned that the unit-demand arrival-time data generated do not strictly follow the theoretical distribution and always contain some sampling error. Thus, there will be inherent differences between these two different approaches. The comparison is meant to provide some insight into how ACO<sub>R</sub> approximates the theoretical model for the problem considered in this study.

The test results are summarized in Table 5.1. For each set of parameters, the optimal  $S$ , denoted as  $S^*$ , and its long-run total cost per unit time found by  $ACO_R$ , denoted as  $C(S^*)$  are reported, and they are compared to its corresponding theoretical results. The percent difference of total cost per unit time,  $\%C(S^*)$  is computed as  $\frac{(C(S^*)_{ACO_R} - C(S^*)_{S\&N})}{C(S^*)_{S\&N}} \times 100\%$  to provide a numerical sense of discrepancy. A value of zero for  $S$  means that the system never orders and all upcoming demand are lost. Obviously, no perishing cost and holding cost will be assessed when  $S^*=0$  and the lostsale cost is all the cost that will incur.

As shown in Table 5.1, the proposed  $ACO_R$ -based computational approach found all  $S^*$  as reported by Schmidt and Nahmias (1985). In most cases, the  $S^*$  values found by  $ACO_R$  and the expected total cost calculated is very close to the theoretical ones. The maximum  $\%C(S^*)$  reported is 8.5083% and the minimum is 0. The statistical summary of all 64  $\%C(S^*)$  values are presented in Figure 5.1. The average percent difference between the two different approaches calculated over all test cases is 0.5061%, which validates the soundness of the proposed computational approach.

The  $S^*$  value increases as  $\bar{L}$  increases and it decreases as  $m$  decreases, which is reasonable and consistent with intuition. When the lost sale penalty is very high, the system tends to employ a high  $S$  since all the other relative costs are negligible compared to the value of lostsale penalty. In this circumstance, high  $S$  level decreases the possibility of stockout and helps to reduce cost.

Table 5.1 Both the results of Schmidt and Nahmias (1985) and ACO<sub>R</sub> for the lostsale case,  $\lambda = 50, h = 20, c = 100$ .

L=0.01 p=10																				
Lbar	150					200					300					600				
	ACOR		Sand N		%C(S*)	ACOR		Sand N		%C(S*)	ACOR		Sand N		%C(S*)	ACOR		Sand N		%C(S*)
	S*	C(S*)	S*	C(S*)		S*	C(S*)	S*	C(S*)		S*	C(S*)	S*	C(S*)		S*	C(S*)	S*	C(S*)	
0.01	0	7500	0	7500	0	0	10000	0	10000	0	1	13119	1	14339	-8.508	2	19401	2	20703	-6.289
0.05	1	6373.5	1	6213	2.583	2	6236.8	2	6556	-4.869	2	6688.9	2	7038	-4.960	3	7399	3	7739	-4.393
0.1	2	5175.5	2	5357	-3.388	3	5417.9	3	5493	-1.367	3	5489	3	5562	-1.312	3	5674	3	5770	-1.664
1	4	5195.5	4	5074	2.395	4	5153.8	4	5078	1.493	4	5163.9	4	5086	1.532	5	5189	5	5094	1.865
L=0.01 p=100																				
Lbar	150					200					300					600				
	ACOR		Sand N		%C(S*)	ACOR		Sand N		%C(S*)	ACOR		Sand N		%C(S*)	ACOR		Sand N		%C(S*)
	S*	C(S*)	S*	C(S*)		S*	C(S*)	S*	C(S*)		S*	C(S*)	S*	C(S*)		S*	C(S*)	S*	C(S*)	
0.01	0	7500	0	7500	0	1	10050	1	10000	0.5	0	15000	0	15000	0	1	25568	1	25788	-0.853
0.05	1	6346.2	1	6473	-1.959	1	7219.8	1	7355	-1.838	2	7796	2	7892	-1.216	2	8603	2	9338	-7.871
0.1	2	5497.9	2	5463	0.639	2	5600.7	2	5660	-1.048	3	5671.8	3	5868	-3.344	3	5885	3	6076	-3.144
1	4	5148.7	4	5074	1.472	4	5153.8	4	5078	1.493	4	5163.9	4	5086	1.532	4	5194	4	5094	1.963
L=1 p=10																				
Lbar	150					200					300					600				
	ACOR		Sand N		%C(S*)	ACOR		Sand N		%C(S*)	ACOR		Sand N		%C(S*)	ACOR		Sand N		%C(S*)
	S*	C(S*)	S*	C(S*)		S*	C(S*)	S*	C(S*)		S*	C(S*)	S*	C(S*)		S*	C(S*)	S*	C(S*)	
0.01	0	7500	0	7500	0	0	10000	0	10000	0	15	15389	15	14905	3.247	107	25325	107	23731	6.717
0.05	20	7136	20	6897	3.465	45	8412.9	45	7937	5.996	62	9903	62	9222	7.385	84	11826	84	11173	5.844
0.1	42	6229.2	42	6152	1.255	51	6778.9	51	6703	1.132	60	7567.3	60	7371	2.663	72	8558	72	8371	2.234
1	58	5354.2	58	5273	1.540	62	5408.6	62	5323	1.608	65	5462	65	5371	1.694	68	5524.7	68	5429	1.763
L=1 p=100																				
Lbar	150					200					300					600				
	ACOR		Sand N		%C(S*)	ACOR		Sand N		%C(S*)	ACOR		Sand N		%C(S*)	ACOR		Sand N		%C(S*)
	S*	C(S*)	S*	C(S*)		S*	C(S*)	S*	C(S*)		S*	C(S*)	S*	C(S*)		S*	C(S*)	S*	C(S*)	
0.01	0	7500	0	7500	0	0	10000	0	10000	0	0	15000	0	15000	0	47	28728	47	28297	1.523
0.05	18	7345.9	18	7216	1.800	32	9097.4	32	8667	4.966	48	11232	48	10614	5.822	70	13982	70	13824	1.143
0.1	34	6553.9	34	6405	2.325	43	7324.4	43	7185	1.940	52	8314.2	52	8206	1.319	64	10025	64	9856	1.715
1	58	5329.1	58	5273	1.064	62	5378.8	62	5323	1.048	65	5423.1	65	5371	0.970	68	5471.8	68	5430	0.770

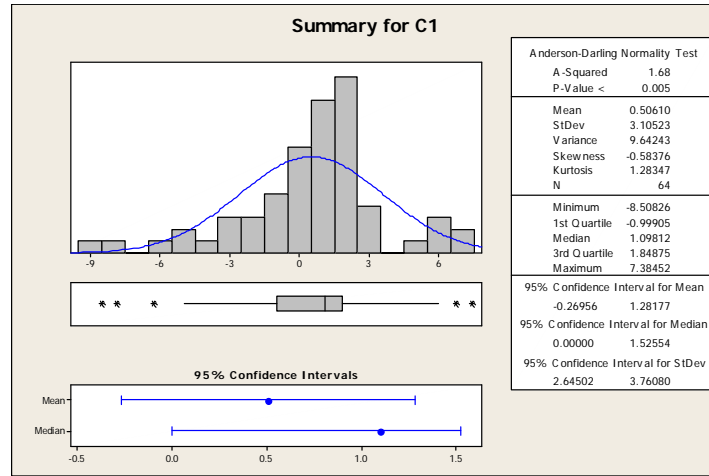


Figure 5.1. Statistical summary for all  $\%C(S^*)$  values in the lost sale case

When the product lifetime is very short, the system tends to not ordering since units will expire almost as soon as they arrive and have no actual utility in the system. As the product lifetime increases, the system tends to stock more. When the leadtime is very long comparing to the product lifetime, high level of  $S^*$  is employed since a large fraction of  $S^*$  units will always be on order at any point in time. This is especially true when the perish cost is low and the lostsale penalty is very high; in this situation the system requires the highest level of  $S^*$ , 107, as shown in Table 5.1. Since failing to meet one unit-demand will cost as high as \$600 and units will perish very soon after they arrive, the system has to keep a high level of  $S^*$  so as to meet demand as much as possible. In general,  $ACO_R$  can be considered as a reliable and effective computational approach as shown above. One substantial advantage is that it requires neither any prior knowledge of demand distribution nor parameter fitting, which implies its potential for wider applications, especially in those highly variable and irregular demand data that do not fit any theoretical models.

#### 5.4.2.2 ACO<sub>R</sub> for two backorder cases

In this subsection, the results obtained by ACO<sub>R</sub> for the backorder case is compared against the results given in Olsson and Tydesjö (2010). Tables 5.2 and 5.3 summarize all the test results for the case of backorder cost per unit for the first and second problem set, respectively. In each table, the optimal value of  $S$ ,  $S^*$ , obtained by both approaches and the associated total cost per unit time,  $C(S^*)$ , are listed. By the same token, the corresponding percent difference of total cost per unit time,  $\%C(S^*)$ , is computed for each test problem. Note again that, for some test problems, both optimization procedures lead to  $S^*=0$ . This simply implies that no item is kept in stock and all customer demand is backordered. The company does not keep any stock and places an order whenever a unit-demand arrives. When  $S^*=0$ , no perishing cost and holding cost will incur and the only cost levied is the backorder cost (recall that the ordering cost is assumed zero). The results shown in Tables 5.2 and 5.3 demonstrate the effectiveness and accuracy of the proposed ACO<sub>R</sub>-based computational approach. ACO<sub>R</sub> found the same  $S^*$  as in Olsson and Tydesjö (2010) for all test problems. Furthermore, the absolute percent difference observed lies between a maximum of 2.716% and a minimum of 0.052%. ACO<sub>R</sub> approximates the counterpart theoretical model very well with the additional advantage that neither distribution assumption nor parameter fitting is needed. What ACO<sub>R</sub> needs is the actual demand history only and with this input data, it can efficiently find the optimal  $S$  tailored to each item's unique demand pattern for each company.

In Table 5.4, the service-level constrained backorder case is considered. Similarly, the optimal  $S$  values obtained by both approaches and the associated optimal costs are listed together with the actual service levels ( $\beta$ ) achieved. The percent difference between the total costs of both approaches is also computed for each test problem. ACO<sub>R</sub> once again live up to the expectation with the maximum absolute percent difference of 2.492% and minimum of 0.126%. The proposed ACO<sub>R</sub> approach successfully find the same  $S^*$  as in Olsson and Tydesjö (2010) for all test problems. The high accuracy achieved by ACO<sub>R</sub> justifies it as a reliable alternative to the theoretical approach for optimizing perishable inventory problems considered in this study.

The actual service level ( $\beta$ ) obtained by ACO<sub>R</sub> is very close to the one obtained by Olsson and Tydesjö (2010), both are slightly larger than the target service level ( $l$ ) due to the discrete nature of  $S$ . In a nutshell, ACO<sub>R</sub> constantly produces accurate results in all problem sets.

The overall performance of the proposed ACO<sub>R</sub> approach is very satisfying for the backorder cases. It provides comparable results to its theoretical counterpart, Olsson and Tydesjö (2010). Especially in service-level constrained backorder case, the proposed ACO<sub>R</sub> approach continues to provide superior results in which their actual service level ( $\beta$ ) are all slightly higher than the target service level ( $l$ ). The proposed approach offers advantage over Olsson and Tydesjö (2010) that it can handle a wide range of demand data because it makes no assumption on demand distribution assumption. Given any series of unit demand arrival time data, ACO<sub>R</sub> can efficiently compute and find the optimal  $S$  without much computational efforts.

Table 5.2 Both the results of Olsson and Tydesjö (2010) and  $ACO_R$  for the backorder cost per unit case,  $\bar{b}$ ,  $h=1$  and  $L=1$ .

Problem No.	$\lambda$	$T$	$\bar{b}$	$p$	Olsson and Tydesjö		$ACO_R$		%C( $S^*$ )
					$S^*$	$C(S^*)$	$S^*$	$C(S^*)$	
1	1	2	4	5	1	3.98	1	3.91	-1.741
2	1	2	4	10	0	4	0	4.05	1.365
3	1	2	6	5	1	5.44	1	5.35	-1.579
4	1	2	6	10	0	6	0	6.08	1.365
5	1	4	4	5	2	2.6	2	2.55	-1.773
6	1	4	4	10	2	3	2	2.93	-2.370
7	1	4	6	5	2	3.18	2	3.13	-1.456
8	1	4	6	10	2	3.59	2	3.51	-2.265
9	4	2	4	5	6	8.3	6	8.27	-0.311
10	4	2	4	10	5	10.34	5	10.18	-1.516
11	4	2	6	5	7	9.84	7	9.89	0.527
12	4	2	6	10	8	12.7	8	12.37	-2.576
13	4	4	4	5	7	4.91	7	4.90	-0.232
14	4	4	4	10	7	4.96	7	4.90	-1.238
15	4	4	6	5	8	5.37	8	5.45	1.555
16	4	4	6	10	8	5.49	8	5.45	-0.665



Table 5.3 Both the results of Olsson and Tydesjö (2010) and  $ACO_R$  for the backorder cost per unit case,  $\bar{b}$ ,  $h=1$  and  $L=1$ .

Problem No.	$\lambda$	$T$	$\bar{b}$	$p$	Olsson and Tydesjö		$ACO_R$		%C( $S^*$ )
					$S^*$	$C(S^*)$	$S^*$	$C(S^*)$	
17	0.5	2	4	10	0	2	0	1.97	-1.370
18	0.5	2	4	50	0	2	0	1.97	-1.370
19	0.5	2	30	10	1	12.62	1	12.80	1.460
20	0.5	2	30	50	0	15	0	14.79	-1.372
21	0.5	10	4	10	1	1.43	1	1.47	2.657
22	0.5	10	4	50	1	1.57	1	1.55	-1.573
23	0.5	10	30	10	2	3.09	2	3.11	0.528
24	0.5	10	30	50	2	3.79	2	3.89	2.507
25	10	2	4	10	14	12.62	14	12.86	1.922
26	10	2	4	50	12	20.23	12	19.85	-1.884
27	10	2	30	10	18	23.84	18	24.14	1.277
28	10	2	30	50	16	53.4	16	53.43	0.052
29	10	10	4	10	16	8	16	8.22	2.716
30	10	10	4	50	16	8	16	8.22	2.716
31	10	10	30	10	20	11.04	20	11.23	1.724
32	10	10	30	50	20	11.04	20	11.23	1.724

Table 5.4 Both the results of Olsson and Tydesjö (2010) and  $ACO_R$  for the service level constraint case,  $h=1$ ,  $L=1$  and  $p=10$ .

Problem No.	$\lambda$	$T$	$l$	Olsson and Tydesjö			$ACO_R$			%C( $S^*$ )
				$S^*$	$\beta$	$C(S^*)$	$S^*$	$\beta$	$C(S^*)$	
33	1	2	0.9	5	0.93	19.44	5	0.92	19.39	-0.267
34	1	2	0.98	7	0.98	29.88	7	0.98	29.78	-0.341
35	1	4	0.9	4	0.97	6.11	4	0.972	6.24	2.074
36	1	4	0.98	5	0.99	8.74	5	0.984	8.65	-1.084
37	4	2	0.9	8	0.91	13.25	8	0.9009	12.92	-2.492
38	4	2	0.98	11	0.98	26.42	11	0.984	26.68	0.992
39	4	4	0.9	8	0.95	4.25	8	0.9449	4.27	0.362
40	4	4	0.98	10	0.99	6.81	10	0.994	6.86	0.684
41	0.5	2	0.9	4	0.91	18.04	4	0.9087	17.98	-0.313
42	0.5	2	0.98	7	0.99	34.12	7	0.995	34.02	-0.304
43	0.5	4	0.9	3	0.96	6.27	3	0.955	6.32	0.737
44	0.5	4	0.98	4	0.99	9.18	4	0.9853	9.22	0.428
45	10	2	0.9	15	0.91	9.01	15	0.9085	9.12	1.196
46	10	2	0.98	19	0.99	21.29	19	0.978	21.26	-0.126
47	10	4	0.9	15	0.92	5.1	15	0.9235	5.17	1.318
48	10	4	0.98	18	0.99	8.02	18	0.975	7.89	-1.586

### 5.4.2.3 ACO<sub>R</sub> applied to non-Poisson demand data

The two theoretical perishable inventory models compared in this study are both based on the assumption that demand follows Poisson distribution. Using these theoretical models on demand data deviating from the assumed Poisson distribution can produce inaccurate optimal inventory policy, leading to undesirable results. The issues and complications that typically arise in practice is that the demand data do not perfectly fitted into an assumed distribution.

In this subsection, the proposed ACO<sub>R</sub> are tested with two non-Poisson demand data. Each of them is generated as follows: randomly generate  $\lambda$  data points within one time unit by manual selection to represent the occurrence of  $\lambda$  customer arrivals in one time unit. Repeat the pattern for 10,000 times to simulate the long run time period. The data generated in this manner all cannot be regarded as Poisson distributed according to the probability test ( $p$ -value  $< 0.05$ ) but they are assumed Poisson-distributed with mean equal to  $\lambda$  anyway for the sake of comparison (one data set with  $\lambda$  of 4 for the backorder case and the other data set with  $\lambda$  of 50 for the lostsale case). Table 5.5 reports both the results obtained by ACO<sub>R</sub> and its counterpart theoretical models.

The differences, both in the values of  $S^*$  and the over 10% discrepancy in  $\%C(S^*)$ , are expected. Deviation from the assumed distribution will directly affect the accuracy of the results obtained by the theoretical models. The results could be underestimating the real cost as the backorder case or overestimating as the loss sale case and the service level constrained backorder case. Traditional theoretical approaches become useless when the demand data is highly variable

and irregular and does not fit well with the assumed Poisson distribution. On the other hand, the proposed ACO<sub>R</sub>-based computational approach offers an alternative when addressing this type of demand data. The proposed approach hence opens a new avenue to develop an effective and all-encompassing inventory solution. The proposed approach can handle various types of real world demand data without the need to derive new models.

Table 5.5 Test results of ACO<sub>R</sub> applied to non-Poisson demand data, the approximation of theoretical models are also provided for comparison

Backorder case											
$L$	$\lambda$	$T$	$\bar{b}$	$p$	$h$	$c$	Olsson and Tydesjö		ACO <sub>R</sub>		%C( $S^*$ )
							$S^*$	$C(S^*)$	$S^*$	$C(S^*)$	
1	4	2	4	5	1	0	6	8.3	4	9.47	14.08
$L$	$\lambda$	$T$	$l$	$p$	$h$	$c$	Olsson and Tydesjö		ACO <sub>R</sub>		%C( $S^*$ )
							$S^*$	$C(S^*)$	$S^*$	$C(S^*)$	
1	4	2	0.9	10	1	0	8	13.25	7	10.88	-17.883
Lost sale case											
$L$	$\lambda$	$T$	$\bar{L}$	$p$	$h$	$c$	Schmidt and Nahmias		ACO <sub>R</sub>		%C( $S^*$ )
							$S^*$	$C(S^*)$	$S^*$	$C(S^*)$	
0.1	50	0.2	150	10	20	100	6	5861	5	5069	-13.511

## 5.5 Discussion

The effectiveness and robustness of ACO<sub>R</sub> for solving perishable inventory models has been validated in the above experimental results. ACO<sub>R</sub> is effective and robust because for the considered perishable inventory problem it consistently produces optimal solutions for all sorts of test problems considered. Since the test problems cover a wide range of  $S$  values, the evidence is strong that ACO<sub>R</sub> is capable of finding good solutions for a variety of instances.

The efficiency of  $ACO_R$  for solving perishable inventory models is shown by its convergence profile summed over 10 runs. Since  $(S-I, S)$  policy deals with only one variable,  $S$ ,  $ACO_R$  for optimizing perishable inventory is expected to be relatively simple, compared to other combinational optimization problems that metaheuristics are able to solve. The convergence profiles of the proposed  $ACO_R$  approach applying to all cases considered in the previous section, as shown in Figures 5.2-5.5, respectively, strongly support that. Figure 5.2 depicts the convergence profiles of  $ACO_R$  applying to the lostsale case, specifically for the problem of  $L=0.01$ ,  $m=0.1$ ,  $\bar{L}=600$ , and  $p=10$ . Figure 5.3 plots the convergence profiles of  $ACO_R$  in solving problem No. 17 in the backorder cost per unit case. Figure 5.4 shows the convergence profiles of problem No. 33 in the service-level constrained backorder case. Figure 5.5 depicts the convergence profiles of  $ACO_R$  applying to non-Poisson demand data, specifically for the service-level constrained backorder case. These four figures are illustrated here as examples and all other figures are omitted to save space. The algorithmic parameters are consistently set at  $q=0.7$ ,  $\xi=0.7$ ,  $NumRun=10$ ,  $NumAnt=10$ , and  $Maxeval=100$ . In each figure, the profiles of both the mean and the best solution among all runs are plotted to trace the search performed by  $ACO_R$  and to see how those solutions converge to the optimal solution. A flat best solution profile shown in Fig. 5.5 indicates that the optimal solution is found from the start in at least one run.

Based on the following four figures, it can be easily observed that the  $ACO_R$  optimization of  $(S-I, S)$  policy is definitely not a complicated optimization procedure; the convergence to the optimal solution occurs very quickly within 100 evaluations on average (actually for some runs

the convergence occurs even after fewer than 100 evaluations, as indicated by the best solution profiles). This is clearly a very simple iterative optimization process.

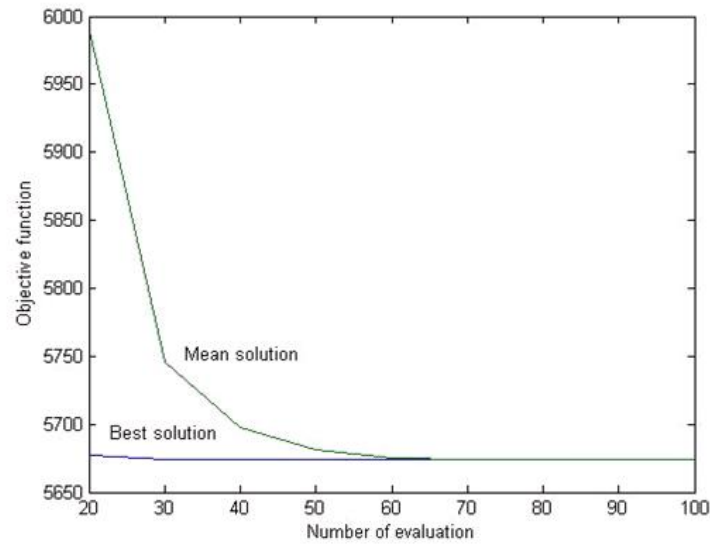


Figure 5.2. Convergence profiles of the best and the mean solutions over all runs for test problem ( $L=0.01$ ,  $m=0.1$ ,  $\bar{L}=600$ ,  $p=10$ ) in Table 5.1 ( $ACO_R$  with lost sale)

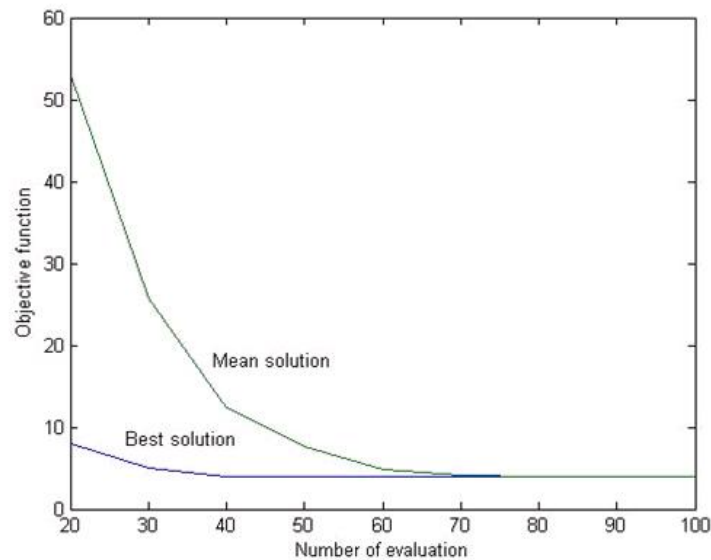


Figure 5.3. Convergence profiles of the best and the mean solutions over all runs for test problem 17 in Table 5.3 ( $ACO_R$  with backorder cost per unit)

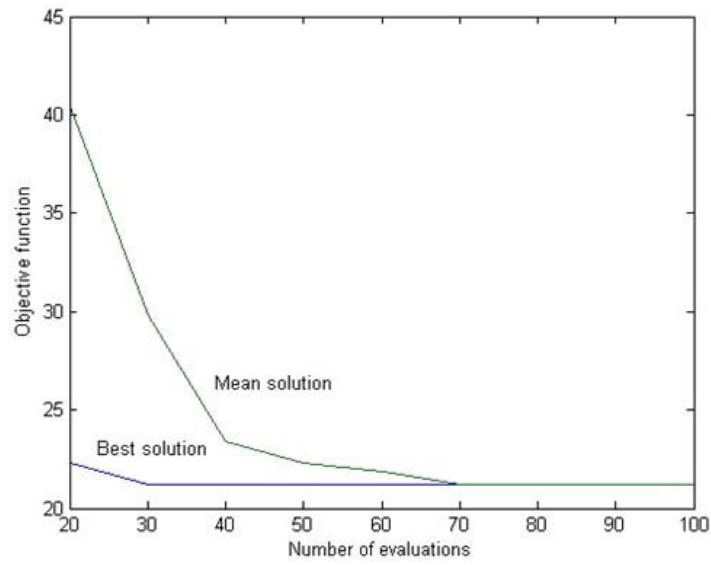


Figure 5.4. Convergence profiles of the best and the mean solutions over all runs for test problem 33 in Table 5.4 (ACO<sub>R</sub> with service level constraint)

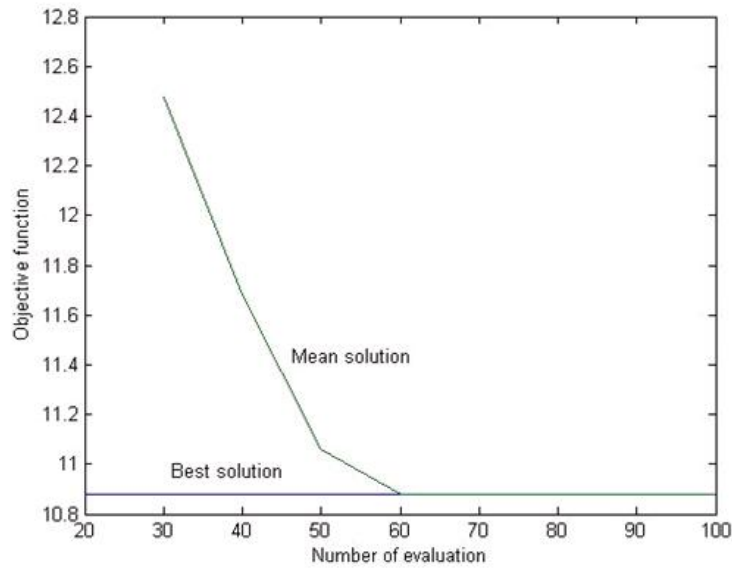


Figure 5.5. Convergence profiles of the best and the mean solutions of all runs for the second test problem in Table 5.5 (ACO<sub>R</sub> with service level constraint)

Since the considered ACO<sub>R</sub> approach utilizes an event-driven algorithm as its evaluation module, it is noticed that it may take relatively more computational effort to obtain a result

comparing to the traditional period-driven algorithm used in the chapter 4. The corresponding CPU time is therefore of particular interest. It is noticed that the CPU time for evaluating 10,000 data points in the experimental section is too long for practical use. Specifically, it took almost 400 seconds for the algorithm to find the optimal solution. In order to save CPU time without sacrificing the accuracy, an accuracy pretest is therefore conducted for all problem sets to see exactly how large the data size is sufficient for the algorithm to simulate long-run average and obtain accurate enough solutions. During the pretest, the data size is reduced gradually and the testing results regarding  $S^*$ ,  $\beta$  and  $C(S^*)$  are monitored closely. For a better illustration, Figure 5.6 based on the grand average  $\%C(S^*)$  of all problem sets is used here to show how the accuracy change as the data points vary from as small as 10 to 10,000. It is clear from Figure 5.6 that for all problem sets, the  $\%C(S^*)$  becomes within reasonable range, 0.258% comparing to  $ACO_R(10,000)$  and 0.325% comparing to Olsson and Tydesjö, when the data size is increased to 500. Therefore, it is concluded that demand data size of 500 points is at least required for a reliable result. Table 5.6 illustrates one of these testing results (problem set of service level constraint case). Other testing results are basically similar and are omitted here to prevent repetition.

By reducing the data size from 10,000 points to 500 points (the value in the parentheses representing the data size), the testing results are still close enough as shown in the above table. In Table 5.6,  $S^*$ ,  $\beta$  and  $C(S^*)$  of  $ACO_R(500)$  are reported in together with its  $\%C(S^*)$  comparing to Olsson and Tydesjö and  $ACO_R(10,000)$ , respectively.  $ACO_R(500)$ +Tabu is an



improved version of  $ACO_R$  (500) aiming to reduce repeated computations. It gave exactly the same results of  $ACO_R$  (500) with less CPU time. The detailed description of  $ACO_R$  (500)+Tabu will be discussed in the next paragraph. For the  $S^*$ ,  $\beta$  and  $C(S^*)$  of Olsson and Tydesjö and  $ACO_R$  (10,000), one can refer to Table 5.4 for detailed results.

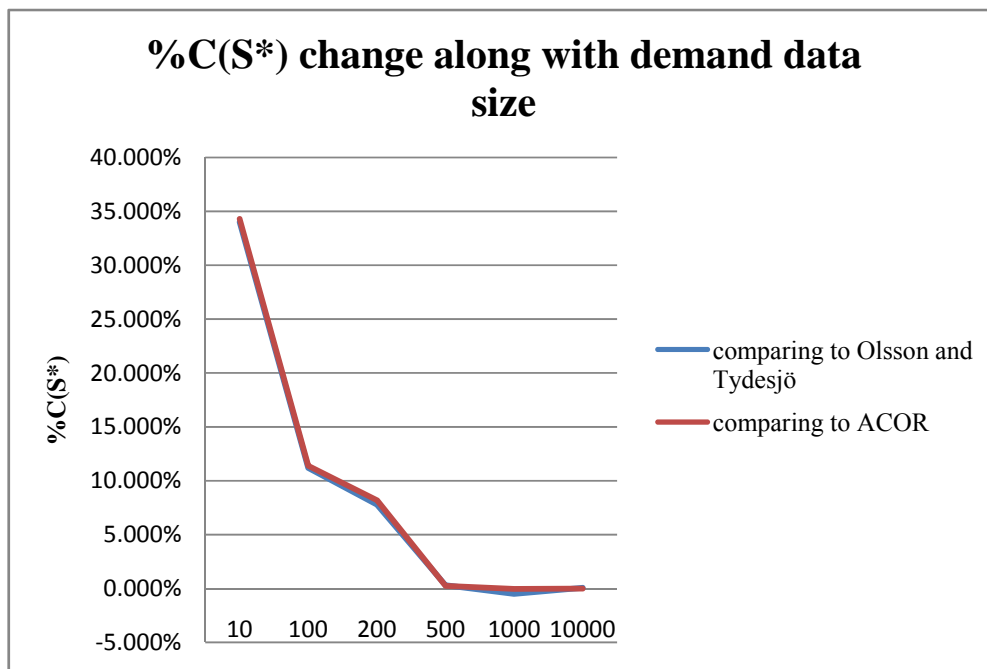


Figure 5.6.  $\%C(S^*)$  change along with different demand data size

In Table 5.6, the optimal  $S$  value remains the same and the percent difference of total cost is small, maximum 3.074% and minimum 0.007% comparing to  $ACO_R$  (10,000) and maximum 2.273% and minimum 0.136% comparing to Olsson and Tydesjö (2010), but the CPU time is controlled within a reasonable range. Based on this fact, the demand data of size 500 is used to test for CPU time in the following analysis, aiming to provide some useful insights for practical management. In addition, a demand data of size 500 should be also a more reasonable size of

demand data that a real company may be able to collect. The CPU results of service level constraint case are reported in detail as follows, other equivalent tables are just omitted to save space.

Table 5.6. Selected problem set illustration for accuracy pretest

Problem No.	ACO <sub>R</sub> (500)			ACO <sub>R</sub> (10,000)	Olsson and Tydesjö
	ACO <sub>R</sub> (500)+Tabu				
	<i>S</i> <sup>*</sup>	$\beta$	C( <i>S</i> <sup>*</sup> )	%C( <i>S</i> <sup>*</sup> )	%C( <i>S</i> <sup>*</sup> )
33	5	0.952	19.39	0.007%	-0.260%
34	7	0.996	29.79	0.046%	-0.296%
35	4	0.972	6.18	-0.958%	1.096%
36	5	0.998	8.77	1.495%	0.395%
37	8	0.906	13.29	2.859%	0.295%
38	11	0.986	26.82	0.506%	1.503%
39	8	0.958	4.26	-0.064%	0.298%
40	10	0.988	6.66	-2.816%	-2.151%
41	4	0.934	18.01	0.134%	-0.177%
42	7	0.994	34.01	-0.023%	-0.328%
43	3	0.948	6.33	0.180%	0.918%
44	4	0.988	9.19	-0.291%	0.136%
45	15	0.936	8.84	-3.074%	-1.914%
46	19	0.984	21.77	2.403%	2.273%
47	15	0.942	5.21	0.782%	2.116%
48	18	0.996	8.13	2.942%	1.309%

Table 5.7 presented the CPU time used by ACO<sub>R</sub> (500) for solving each problem, from problem 33 to problem 48, together with an improved version (ACO<sub>R</sub>+Tabu search). The average CPU time for solving this problem set for each approach is concluded at the bottom of each column. Noticed that the problem search space (possible  $S$  values for the perishable inventory system) is actually relatively limited in the real practice. The algorithmic parameters are set low at  $q=0.7$ ,  $\xi=0.7$ , NumRun=3, NumAnt=4, and Maxeval=30 for both ACO<sub>R</sub> (500) and ACO<sub>R</sub> (500)+Tabu, as long as the search power is enough to find optimal solutions for all problems. It

is observed that pure  $ACO_R$  algorithm employs longer CPU time than  $ACO_R (500)+Tabu$  due to the fact that purely employing  $ACO_R$  may end up to many helpless repetitive searches when dealing with the limited scope of problem space. Therefore, Tabu search is embedded into the subject  $ACO_R$  algorithm as  $ACO_R+Tabu$  in Table 5.7 so as to reduce unnecessary search time for the same solution. Tabu search uses a short term memory to escape from local minima and to avoid cycles. The short term memory is implemented as a tabu list that keeps tracking of the most recently visited solutions and forbids moves towards them. The neighborhood of the current feasible solution is thus restricted to the solutions that do not belong to the tabu list, which forms the allowed set. At each iteration, only solutions from the allowed set can be selected as the new current solution. Additionally, this solution is added to the tabu list until it reaches its maximum allowable length (the length here is set equal to the possible range of  $S$  values). After that, the tabu list is updated on a FIFO order. The oldest solution that was already in the tabu list is removed each time a new solution is added. The algorithm stops when the termination condition is met. By using this dynamic neighborhood search technique, it prevents from the possibility of repetitive search and as shown above reduces the CPU time by around 2 seconds on average.

Also note that the *NumRun*, *NumAnt* and *Maxeval* in this  $ACO_R$  for optimizing perishable inventory policy algorithm are all kept to be very low, maximum 10, 10, and 100, respectively. This suggests, without tuning the algorithmic parameters, the  $ACO_R$  for perishable inventory system is already effective enough to obtain good solution. Since  $ACO_R$  does not employ much search power in this considered problem, the investigation of algorithmic parameter setting

seems not necessary here. On the other hand, it also implies that, there is potential for the proposed ACO<sub>R</sub> approach to solve more complicated inventory problem involving higher number of parameters such as multi-echelon inventory models. In summary, the performances of the ACO<sub>R</sub> implementations in terms of efficiency and effectiveness are all very encouraging.

Table 5.7. CPU results of service level constraint case using two different strategies

Problem No.	CPU Time	
	ACO <sub>R</sub> (500)	ACO <sub>R</sub> (500)+Tabu
33	18.9	17.86
34	23.07	22.29
35	12.21	10.41
36	12.71	11.63
37	10.57	10.26
38	11.98	11.94
39	9.98	9.85
40	9.95	9.59
41	45.21	29.76
42	46.69	44.93
43	21.85	17.52
44	22.42	17.45
45	9.87	9.69
46	10.21	9.92
47	9.82	9.43
48	9.84	9.79
Grand Average	17.83	15.77

## 5.6 Conclusion

This chapter extends the computational approach to the determination of the optimal ( $S-I$ ,  $S$ ) policy for single-echelon single-product perishable inventory system. Different versions of ACO<sub>R</sub>-based optimization algorithm have been implemented: one for lostsale case and two for

backorder cases with one with backorder cost per unit and the other with service level constraint. All three versions of the proposed computational approach were tested against their counterpart theoretical models: i.e., Schmidt and Nahmias (1985) for the lost sale version and Olsson and Tydesjö (2010) for the two backorder versions. In fact the average % difference of total inventory costs per unit time obtained by both cases is within 3 percent. Furthermore, two salient features of the proposed approach make it really innovative: (1) It circumvents the infinite-dimension problem of the state space that theoretical/analytical approaches must face; and (2) It can be designed to handle a wide range of demand data because it makes no assumption on demand distribution assumption, and hence no need for distribution parameter fitting. Given any series of unit demand arrival time data,  $ACO_R$  can efficiently compute and find the optimal  $S$  without much computational efforts. The experimental results indicate that  $ACO_R$  can serve as a very reliable alternative to theoretical approach to deal with this type of perishable inventory problem. Similar conclusions were reached in chapter 4, in which  $(R, Q)$  policies were optimized using a slightly different computational approach.

## **CHAPTER 6 OPTIMIZATION OF PERISHABLE ITEMS FOLLOWING COMPOUND POISSON DEMAND PROCESSES**

### **6.1 The problem**

In this chapter, the considered inventory problem is further extended to more general case. To be more specific, an intelligent computational approach is developed by combining the abovementioned two models. It is designed to account for fast-moving/slow-moving, nonperishable/perishable items. The optimization of inventory policies for perishable items following compound Poisson demand processes are studied in detail because the corresponding counterpart model (Baron et al. 2010) exists. Baron et al. 2010 illustrate two cases of compound Poisson demand processes exactly in their numerical results (exponential and unit demand sizes). It is focused on these two special cases here. In principle, the proposed approach can be extended to demand sizes of any type of distributions or even auto-correlated demand and it can consider both non-perishable and perishable cases. For comparison purpose, two cases of perishability considered in the counterpart model will be analyzed here: The first case is so-called “sudden deaths due to disasters” in which time perishability follows exponential distribution. It models items that are subject to a disaster e.g., spoilage because of extreme weather conditions or a malfunction of a refrigerator that stores the stock. The second case is time perishability to be a constant, which models items perish with an expiration date, e.g., produce, milk, medicines and etc. Following their assumptions, lead time is 0 and do not allow back orders. In cases where demand is larger than the inventory level, the next order should include the remaining portion of unsatisfied demand.

The aim is to minimize the total inventory costs which include holding cost, ordering cost and cost of the perishable items. The inventory policy considered is  $(s, S)$  policy. Baron et al. 2010 assume that the lead time is zero, which means the inventory level is raised to  $S$  immediately once it reaches 0. Therefore, the  $s$  is always set as 0. This simplifies the problem to determine only the optimal inventory order quantity,  $S$ .

## 6.2 Mathematical model for the considered problem

Similarly in chapter 5, the inventory level is continuously reviewed in which the review period is set to be the time advances by events of demand consumption, replenishing and outdating. The subscript  $e$  represents the corresponding event occurrence during the planning horizon. Customer demand arrives according to exponential distribution. Each customer can order any arbitrary size per time (not restrict to unit size). For comparison purpose, of particular interest is exponential demand size and unit demand size, as solved exactly in Baron et al. (2010). Note that the lifetime considered here is not necessarily constant any more. It can also be variable lifetime in which time perishability follows exponential distribution (“sudden death”).

For the case when items are subject to an expiration date,  $t_\theta$  is used to represent the fixed lifetime; for the case of “sudden death”, the lifetime  $M(\xi) \sim \exp(\xi)$

<b>Subscripts</b>	
$e$	Event $e$ in the planning horizon, $e=1,2, \dots$
<b>Notations</b>	
$T_{le}$	Current time when event $e$ happens
$\mu$	The mean of exponential demand size
$\xi$	The mean of exponential perishability
$\lambda$	The mean of Poisson arrival rate

<b>Notations (continued)</b>	
$t_0$	Fixed lifetime of constant perishability
$K$	Ordering cost
$h$	Holding cost per unit of inventory per unit time
$\pi$	Penalty cost of the perishability per unit of inventory
$S$	Inventory raise-up-to level
$TC/period$	Average total cost per unit time

The input data include a vector that records each demand arrival time and another vector that records the corresponding demand size ordered at each time.

An inventory state description (age distribution) is also required. Because there will be multiple items entering into the stock at the same time, a 2-row matrix  $P$  is needed to record the number of items, and their corresponding outdating time.

$$P^e = \begin{Bmatrix} P_{11}^e, P_{12}^e, P_{13}^e, \dots, P_{1i}^e, \dots, P_{1N_e}^e \\ P_{21}^e, P_{22}^e, P_{23}^e, \dots, P_{2i}^e, \dots, P_{2N_e}^e \end{Bmatrix}$$

$P^e$  is a dynamic matrix to record the beginning inventory state at time  $T_{le}$ . The size of  $P^e$ ,  $2 \times N_e$ , is changing along with the current time  $T_{le}$ , in which the first row records the number of items and the second row records their corresponding outdating time. For example, at  $T_{le}$ ,  $P_{1i}^e=3$  and  $P_{2i}^e=2.78$  shows that currently there are 3 items on stock which will perish at time point 2.78. Furthermore the matrix  $P^e$  is sorted each time the inventory state changes such that the first column corresponds to the oldest item(s) in the stock. Therefore, items on stock all have a tag,  $P_{ij}^e$ , to indicate when and how many will perish. This matrix  $P^e$  is updated and sorted every time as the inventory state changes. Each time a customer arrives, items are fulfilled following the FIFO policy. Obviously, items will be removed from the matrix as soon as they are consumed.



Following the event updating mechanism in chapter 5, each particular event is executed and the incurred cost is computed and recorded. During the whole planning horizon, a matrix  $T = \begin{Bmatrix} T_{11}, T_{12}, T_{13}, \dots, T_{1e}, \dots, T_{1end} \\ T_{21}, T_{22}, T_{23}, \dots, T_{2e}, \dots, T_{2end} \end{Bmatrix}$  is also required. The first row records any time point that may change the inventory state and incur some cost. If  $T_{1e}$  represents the time of demand arrival,  $T_{2e}$  will reflect the corresponding demand size. When  $T_{1e}$  represents the time of other event such as order arriving or item outdating, the second row value  $T_{2e}$  is set as zero.  $T_{1end}$  indicates the time point when the last customer walks in and  $T_{2end}$  is the corresponding demand size ordered by that last customer. As explained back into chapter 5, it is sufficient to just check those time points kept in the memory since the inventory level does not change at other time points and no costs other than holding cost incurs.

The timing matrix,  $T$ , is dynamically updated throughout and only sufficient numbers of future time points (after the current time) are kept so that the matrix can be controlled within a reasonably manageable size. The current time index keeps advancing until the last customer demand arrives at  $T_{1end}$ . At the beginning of current time  $T_{1e}$ , the holding cost of on-hand

inventory,  $\sum_{i=1}^{N_e} P_{li}^e$ , is first calculated.

Let  $H(P^e, T_{1e})$  be the holding cost during the period between two successive events happen.

Obviously,

$$H(P^e, T_{1e}) = h \cdot \sum_{i=1}^{N_e} P_{li}^e \cdot (T_{1e} - T_{1e-1}). \quad (6.1)$$

Then two cases can be distinguished: whether the current time  $T_{le}$  represents an event of consuming or outdating.

(1) If the current time  $T_{le}$  indicates a customer walking in ( $T_{2e} > 0$ ), the company tries to satisfy with its available on-hand inventory,  $\sum_{i=1}^{N_e} P_{li}^e$ . Each customer demand is satisfied based on FIFO policy, starting from the oldest items. Items are removed from the vector  $P^e$  accordingly once it is consumed. For example,  $P^e = \begin{matrix} 2, 3, 6, 8 \\ 1.23, 3.45, 5.6, 8.91 \end{matrix}$  with a customer demand of 4 items at  $T_e$ , the inventory transition can be obtained as  $P^{e+1} = \begin{matrix} 1, 6, 8 \\ 3.45, 5.6, 8.91 \end{matrix}$ . The first two items which will expire at time point 1.23 are used up and another two items which will expire at time point 3.45 are also used. In cases where demand is larger than the inventory level, the next order should include the remaining portion of unsatisfied demand.

The inventory level after fulfilling customer demand is given by

$$P_{li}^{e+1} = \begin{cases} (P_{li}^e - T_{2e})^+, & \text{if } i = 1 \\ (P_{li}^e - (T_{2e} - \sum_{j=1}^{i-1} P_{lj}^e)^+)^+, & \text{if } i > 1 \end{cases} \quad (6.2)$$

For any  $P_{li}^{e+1} = 0$ , there is no point to store their outdate time,  $P_{2i}^{e+1}$ , anymore because they are consumed by the demand. Therefore, the corresponding column(s) is/are removed from  $P^{e+1}$ .

(2) If the current time  $T_{le}$  is the time when outdating item is due to expiration ( $P_{2i}^e = T_{le}$ ),  $P^{e+1}$  will be updated by deleting the outdating item. As explained before, the first column of  $P^e$  corresponds to the oldest item(s) in the stock, which is subject to outdate at  $T_{le}$ .

Therefore the inventory state will be updated by deleting the first column of the  $P^e$  such that:

$$P^{e+1} = \left\{ \begin{matrix} P_{12}^e, P_{13}^e, \dots, P_{1N_e}^e \\ P_{22}^e, P_{23}^e, \dots, P_{2N_e}^e \end{matrix} \right\} \quad (6.3)$$

The corresponding outdating cost will apply:

$$O(P^e) = \pi \cdot P_{11}^e \quad (6.4)$$

At the end of  $T_{le}$ , the inventory state is transited from  $P^e$  to  $P^{e+1}$ . If inventory level after considering demand and outdating is zero ( $P^{e+1}$  is empty), according to the assumption in Baron et al. (2010), the inventory level is raised immediately up to  $S$ .  $S$  is the decision variable.

Mathematically, for the “sudden death” case, the  $P^{e+1}$  will be finalized as follows:

$$P^{e+1} = \begin{cases} \left\{ \begin{matrix} S \\ T_{le} + M(\xi) \end{matrix} \right\}, & \text{if } P^{e+1} \text{ is empty} \\ P^{e+1} & \text{else} \end{cases} \quad (6.5.1)$$

For the “constant expiration” case, the  $P^{e+1}$  will be finalized as follows:

$$P^{e+1} = \begin{cases} \left\{ \begin{matrix} S \\ T_{le} + t_0 \end{matrix} \right\}, & \text{if } P^{e+1} \text{ is empty} \\ P^{e+1} & \text{else} \end{cases} \quad (6.5.2)$$

Obviously, the  $P^{e+1}$  after replenishing will be at least of size  $2 \times I$  and the total inventory level,  $\sum_{i=1}^{N_{e+1}} P_{li}^{e+1}$ , will be at most up to  $S$ . The finalized  $P^{e+1}$  will serve as the initial inventory at the beginning of next event.

Denote by  $v_e(P^e, P^{e+1}, T_{le})$  the total cost of the company with initial inventory state  $P^e$  and ending inventory state  $P^{e+1}$  at  $T_{le}$ . The following optimality equations will be obtained:

$$v_e(P^e, P^{e+1}, T_{1e}) = K \cdot \delta(S - \sum_{i=1}^{N_{e+1}} P_{1i}^{e+1}) + H(P^e, T_{1e}) + O(P^e) \quad (6.6)$$

In which  $\delta(x)$  is an indicator function defined as:

$$\delta(x) = \begin{cases} 1, & x = 0 \\ 0, & x > 0 \end{cases} \quad (6.7)$$

Given the initial system state, the optimal system cost over the time horizon is

$$\text{Min } TC / \text{period} = \frac{\sum_e v_e(P^e, P^{e+1}, T_{1e})}{T_{1end}} \quad (6.8)$$

In which  $T_{1end}$  is the time when the last customer demand arrives.

The pseudo-code of the evaluation module for perishable items following compound Poisson processes is given below.

---

**The evaluation module for perishable items following compound Poisson processes**

Input: Demand arrival time data, demand size data,  $S$  values, and cost parameters

Output: Total cost per unit time

- (1) Set the initial inventory equal to  $S$  and record the expected lifetime of the stock on hand;  
Initialize the timing matrix ( $T$ ), the initial matrix will include all demand arrival time, its corresponding demand size and the perish time for each item on stock;  
Sort the matrix such that the first column corresponds to the earliest time point;
- (2) While the current time ( $T_{1e}$ )  $\leq$  last time of demand arrival ( $T_{1end}$ )
  - (2.1) If  $T_{1e}$  is the time of demand arrival for certain customer ( $T_{2e} > 0$ )
    - (2.1.2) The oldest item in the stock is used to fulfill the demand, Eq. (6.2)
End if
  - (2.2) If  $T_{1e}$  is the perish time for certain item in stock ( $P_{2i}^e = T_{1e}$ )

- (2.2.1) The perished item is discarded from the stock, together with its corresponding expiration time, according to Eq. (6.3)
- (2.2.2) The perish cost is charged for the item discarded, Eq. (6.4)
- End if
- (2.3) If the inventory state matrix after considering demand and outdating,  $P^{e+I}$ , is empty
- (2.3.1) Raise the inventory level to  $S$  according to Eq. (6.5)
- End if
- (2.4) Calculate the total cost for current time  $T_{le}$ , according to Eq. (6.6)
- (2.5) The current time is removed from the timing matrix  $T$
- (2.6) Sort the timing matrix  $T$  and inventory state matrix  $P$
- (2.7) The current time is set as the next earliest time point,  $T_{le+1}$
- End while
- (3) All costs incurred during the whole time period is added to the total cost function, Eq. (6.8).
- 

### 6.3 Barebones differential evolution for determining optimal inventory policy

For the considered inventory problem, the metaheuristic algorithm a relatively more recent one than ACO is chosen here as the optimizer. It is termed as “Barebones differential evolution” (BBDE) as in Omran et al. 2009. The barebones differential evolution capitalizes on the strengths of both the barebones Particle Swarm Optimization (Kennedy 2003) and self-adaptive differential evolution strategies. It is a newly developed, efficient hybrid optimization algorithm.

For each target vector  $x_i(t)$  of generation  $t$  in the BBDE, position updates are done as follows:

$$x_{i,j}(t+1) = \begin{cases} p_{i,j}(t) + r_{2,j} \times (x_{i_1,j}(t) - x_{i_2,j}(t)) & \text{if } U(0,1) > P_r \\ y_{i_3,j}(t) & \text{otherwise} \end{cases} \quad (6.9)$$

In which,

$$p_{i,j}(t) = r_{1,j}(t)y_{i,j}(t) + (1 - r_{1,j}(t))\hat{y}_j(t) \quad (6.10)$$

With  $i_1, i_2, i_3 \sim U(1, \dots, s)$ ,  $s$  is the population size and  $i_1 \neq i_2 \neq i$ ;  $j = 1, \dots, N_d$  refers to the specific dimension;  $r_{1,j}, r_{2,j} \sim U(0,1)$  and  $P_r$  is the probability of recombination.

In Eq. (6.10),  $p_i(t)$  represents the particle attractor, which is a stochastic weighted average of particle best and global best positions, borrowed from the Barebones PSO. Referring to Eq. (6.9), the mutation operator of DE is used to explore around the current attractor,  $p_i(t)$ , by adding a difference vector to the attractor. Crossover is done with a randomly selected particle best,  $y_{i_3}(t)$ , as these particle bests represent a memory of best positions found by individuals since the start of the search process. According to Eq. (6.9), for a  $(1-P_r)$  proportion of the updates, information from a randomly selected particle best,  $y_{i_3}(t)$ , is used (facilitating exploitation), while for a proportion of  $P_r$  of the updates step sizes are mutation of the particle attractor,  $p_i(t)$  (facilitating exploration). Mutation step sizes based on the difference vector between randomly selected particles,  $x_{i_1}(t)$  and  $x_{i_2}(t)$ . The BBDE is shown in Omran et al. (2009) to achieve a good balance between exploration and exploitation. It is noted that the exploitation of particle best position does not focus on a specific position. The particle personal best position,  $y_{i_3}(t)$ , is randomly selected for each position update.

**Key parameters used in BBDE algorithms**

$s$	The size of the population
$Maxeval$	Maximum number of evaluations for stopping each run

<b>Key parameters used in BBDE algorithms (continued)</b>
<i>NumRun</i> Number of runs

In this study, it is specified  $s=10$ ,  $Maxeval=300$  and  $NumRun=30$ . The pseudo code of the implemented BBDE algorithm is given below.

---

**BBDE for determining optimal inventory policy**

(1) Load the demand arrival data and the demand size data. Both algorithm-related and problem-dependent parameters are first initialized.

(2) Randomly generate  $s$  initial solutions within the bounds of the variables, evaluate and rank them in the archive according to the objective value. Each initial solution is evaluated by the evaluation module to compute its objective function value. Set the number of evaluations to be  $s$  (i.e.,  $eval = s$ ).

(3) While  $eval < Maxeval$  do

    (3.1) For  $i = 1: s$ ;

        (3.1.1) Randomly pick three vectors  $(x_{i_1}, x_{i_2}, x_{i_3})$  such that  $i_1 \neq i_2 \neq i_3 \neq i$

        (3.1.2) Update the position of the target vector  $x_i$  by following Eq. (6.9) .

        (3.1.3) The particle attractor  $p_i$  is calculated according to Eq. (6.10)

        (3.1.4) Check if any variable of the new solution is outside of the upper/lower bound. If so, the new trial solution is repaired by either randomly generating a new value within the bound or setting it to the bound value (with equal probability).

        (3.1.5) The new trial solution is evaluated by the evaluation module to compute its objective function value.

        (3.1.6) Update the particle best if the trial solution is better than the solution found so far

    End for

(3.2) Update the best solution found so far

End while

Output the result of the optimal solution and its objective value.

---

## 6.4 Computational results

Each set of demand series data inputted to the meta-heuristic approach is generated by a Minitab® random number generator designed for a given type of distribution without dependency among data points. It is chosen to generate 10,000 data points for each demand set so as to simulate the long-run total cost per unit time. The 10,000 demand points forms a demand pattern over 10,000 time units that follow a specified distribution. For the demand arrival timing data, a data set is generated by using the Minitab® exponential random number generator by specifying its target mean, which equals to  $1/\lambda$ . The demand size data will be generated by the similar procedure, following a specified distribution. Both data sets are imported into the meta-heuristic program. Each demand will occur according to certain Poisson arrival rate to simulate to simulate the compound Poisson demand process.

In the following experimental results section, the meta-heuristic approach is implemented into two special cases presented in Baron et al. 2010 since these two special cases are solved exactly by the author.

### 6.4.1 The control policy of exponential perishability with exponential demand size

In this section, the case where  $K=10$ ,  $h=1$ ,  $\pi=2$ ,  $\mu=3$  and  $\zeta=5$  with varying  $\lambda$  is considered. This represents the situation when demand arrives according to Poisson process ( $\lambda$ ) with



exponential demand size ( $\mu$ ). Each batch stored is subject to disasters which arrive according to a Poisson process with rate equals to 0.2. Table 6.1 summarized the experimental results obtained.  $S^*$  represented the optimal  $S$  found and  $C(S^*)$  records the corresponding total cost for both methods. Each value under the “% $C(S^*)$ ” column is percent deviation of BBDE from the Baron 2010 model, computed as  $\frac{TC/period(BBDE) - TC/period(Baron\ 2010)}{TC/period(Baron\ 2010)} \times 100$ . It can be observed that the percentage differences are all within reasonable range with maximum absolute percentage difference of 5.63% and the minimum of 0.8%. When  $\lambda$  is increasing (more customers are arriving),  $S^*$  and  $C(S^*)$  are increasing. The results shown in Table 6.1 nicely demonstrate that the proposed meta-heuristic approach is accurate and competitive to the theoretical model.

Table 6.1. Optimal policy found by both methods for exponential demand size when  $\lambda$  is varied

$\lambda$	$S^*$	$C(S^*)$		% $C(S^*)$
		Baron 2010	BBDE	
0.5	0.92	4.11	4.14	0.80
1	1.40	5.11	4.92	-3.69
1.5	1.78	5.90	5.83	-1.20
2	2.11	6.58	6.21	-5.63
2.5	2.39	7.19	6.90	-4.04
3	2.65	7.74	7.41	-4.32
3.5	2.89	8.25	7.91	-4.12
4	3.11	8.73	8.45	-3.29
4.5	3.32	9.19	8.82	-3.95
5	3.52	9.61	9.24	-3.84
10	5.12	13.10	12.56	-4.13
20	7.39	18.07	17.81	-1.42
50	11.91	27.98	26.88	-3.92

#### 6.4.2 The control policy of constant perishability with unit demand size

In this section, the case where  $K=10$ ,  $h=1$ ,  $\pi=2$  and  $t_0=8$  with varying  $\lambda$  is considered. This represents the situation when demand arrives according to Poisson process ( $\lambda$ ) with unity demand size. Each batch stored is subject to constant expiration date. Table 6.2 records the corresponding testing results. Similarly, for each different  $\lambda$ ,  $S^*$ ,  $C(S^*)$  and  $\%C(S^*)$  are recorded. The proposed approach achieves maximum absolute percent difference of 3.69% and minimum of 0.01%. The high accuracy achieved by the metaheuristic approach justifies it as a reliable alternative to the theoretical approach for optimizing perishable inventory problems considered in this study. The meta-heuristic approach constantly provides accurate results when facing different arrival rates. The  $S^*$  and  $C(S^*)$  increases with the increase of  $\lambda$  (more customers are arriving), which is reasonable and similarly observed in the counterpart theoretical model.

Table 6.2. Optimal policy found by both methods for unit demand size when  $\lambda$  is varied

$\lambda$	$S^*$	$C(S^*)$		$\%C(S^*)$
		Baron 2010	BBDE	
0.5	3	3.88	4.02	3.69
1	4	5.03	5.11	1.71
1.5	5	6.00	6.00	-0.03
2	6	6.83	6.84	0.15
2.5	7	7.57	7.63	0.82
3	8	8.25	8.27	0.28
3.5	8	8.88	8.87	-0.01
4	9	9.44	9.53	0.94
4.5	10	10.00	9.96	-0.36
5	10	10.50	10.44	-0.62
10	14	14.64	14.65	0.06
20	20	20.50	20.75	1.20
50	32	32.13	32.12	-0.01

## 6.5 Conclusion

In this chapter, the proposed computational approach is extended to determine the optimal  $(s, S)$  policy for perishable items following compound Poisson demand process. The results showed that the metaheuristics (BBDE) integrated with the evaluation module are comparable to the counterpart theoretical models: Baron et al. (2010). From a methodological point of view, two salient features of the proposed approach make it really innovative: (1) It circumvents the infinite-dimension problem of the state space that theoretical/analytical approaches must face; and (2) It can be designed to handle a wide range of demand data because it makes no assumption on demand distribution assumption, and hence no need for distribution parameter fitting. Given any series of demand size data and demand arrival time data, the proposed meta-heuristic approach can efficiently compute and find the near-optimal policy. The experimental results verify that the proposed meta-heuristic approach can serve as a very reliable alternative to the theoretical approach.

## CHAPTER 7 SUMMARY AND CONCLUSION

For almost all mathematical inventory models in use today, it is common to assume the lead time demand (LTD) to follow a particular distribution (or distributional family). The inventory policy parameter, i.e., are then determined based on the assumed distribution (usually a number of other assumptions will also be made). Those assumptions are needed since the exact model may probably be too complicated and become mathematically intractable. Those assumptions reduce the exact model to a relatively simple model so that it can be solved optimally or near-optimally. However, those assumptions limit the scope of practical inventory problems the model can handle. In real-world practice, most demand data does not fit well to an assumed distribution and some may even be auto-correlated.

In this thesis, a totally new computational approach to the optimization of inventory policies for single company is proposed. It contains a meta-heuristic optimizer and an evaluation module. The meta-heuristic generates a candidate solution and supplies this candidate solution to the evaluation module. The evaluation module is tailored to each considered inventory problem to simulate the inventory movement within the system. It calculates the corresponding inventory cost and evaluates how good a solution is for the considered system. Based on the feedback from the evaluation module, the meta-heuristic optimizer iteratively improves the quality of the solution according to its searching mechanism. The meta-heuristic optimizer performs an intelligent search on the problem space. This cycle goes on and on until a near-optimal solution

is obtained (usually based on the user's preference for the amount of computational time devoted to the search).

The proposed approach has been applied to three distinct cases: (1) fast-moving nonperishable item (Chapter 4); (2) slow-moving perishable item (Chapter 5) and (3) fast-moving/slow-moving perishable/nonperishable items, in which the case of perishable items following compound Poisson demand process is examined in detail (Chapter 6). These three cases are selected to study because their corresponding theoretical/analytical model exists. By comparing the computational results from the new approach with those theoretical/analytical models, it is concluded that the new computational approach is capable of providing comparable results to those theoretical/analytical models when demand data conforms to their assumption. For demands not well conformed to their assumption, the proposed approach is able to handle it. On the other side, for such cases the validity of those theoretical/analytical approaches are questionable because some of the solutions can be very costly.

Traditional theoretical/analytical methods have potentially serious risks involved if used under uncertain demand conditions. Practitioner should be aware of the possibility of ill-fitting or autocorrelation of the demand while using these analytical inventory models. Practitioners need to understand the assumptions of these analytical models very well and determine when or when not to apply these models in practice. It is recommended that practitioners evaluate demand data first and gain some prior knowledge of the actual demand distribution before implementation. Considering the practical need of industry for which a comprehensive model is desirable to be

embedded in decision support system to assist in inventory management, it is concluded that the robustness of these analytical models with respect to changes in demand pattern cannot be regarded as adequate for real world implementation.

Hence, the advantage offered by the metaheuristic-based computational approach is substantial since it incorporates the stochastic character of the demand in the real world without making any assumptions on its distribution, which makes it applicable to a wide range of demand data. The main motivation behind developing such a computational approach is to bridge the existing gap between theory and practice so as to deliver a user-friendly and flexible model for the management of various inventories in a business environment. From this standpoint, it is believed that the proposed computational approach offers a promising alternative to existing inventory control methods. It can handle a wide scope of problems, especially those of practical relevance. By simply analyzing its unique historical demand data, a company will be able to gain fact-based insight and keep the inventory at a level that best tailored to their needs. Surplus stock will be cut down and the associated costs will be reduced significantly without impacting on its customer service level.

In principle, the inventory system can be tailored to more realistic or company-dependent form. As an illustration, when it comes to calculate inventory-related cost, certain cost components may be related to some realistic factors, such as size and time, in a more complicated manner. For example, in practice, some customers would like to wait for backlogging during the shortage periods, others would not. Consequently, the opportunity cost

due to lost sales should be taken into consideration in the cost function; for many time-sensitive products such as fashionable commodities and high-tech products, the length of customer waiting time becomes the main factor in determining whether backorder would be accepted or not; other complexities can include special procurement opportunities, on a one-time or repeating basis, cost change over time due to inflation, just to name a few. In those cases, cost components may be specified as functions of realistic factors. Those functions do not have to be convex for the meta-heuristic to find quality solutions.

Finally, it is worth mentioning that the successful use of ACO<sub>R</sub> and BBDE in this study suggests the more general use of metaheuristics into inventory problems. It is possible to apply various metaheuristics to more complicated inventory problems. Built upon the basic single-echelon problems as presented in this work, it is shown that intelligent computational approach can efficiently and effectively handle a wide scope of inventory problems, from single echelon to supply chain. When facing those multi-echelon supply chain problems, it is believed that the intelligent-search approach proposed here will show even more substantial advantages due to its salient distribution-free feature and its efficiency as an intelligent guided search. Exploring these possibilities will be the subject of the ongoing research.

## REFERENCES

1. Arrow, K. A., Karlin, S. and Scarf, H. E. 1958. *Studies in the Mathematical Theory of Inventory and Production*. Stanford University Press, Stanford, CA.
2. Axsäter, S., 2006. A simple procedure for determining order quantities under a fill rate constraint and normally distributed lead-time demand. *European Journal of Operational Research*, 174(1), 480-491.
3. Baron, O., Berman, O. and Perry, D. 2010. Continuous review inventory models for perishable items ordered in batches. *Mathematical Methods of Operations Research*, 72(2), 217-247.
4. Brown, R. G., 1967. *Decision Rules for Inventory Management*. Holt, Rinehart and Winston, New York.
5. Chazan, D. and Gal, S., 1977. Markovian model for a perishable product inventory. *Management Science*, 23(5), 512-521.
6. Chiu, H. N., 1995. An approximation to the continuous-review inventory model with perishable items and lead times. *European Journal of Operational Research*, 87(1), 93-108.
7. Cohen, M. A., 1976. Analysis of single critical number ordering policies for perishable inventories. *Operations Research*, 24(4), 726-741.
8. Costa, D., Hertz, A., 1997. Ants can colour graphs. *Journal of the Operational Research Society* 48(3), 295-305.
9. Deb, K., 2000. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering* 186(2-4), 311-338.
10. Dorigo, M., 1992. Optimization, Learning and Natural Algorithms (in Italian). Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy.
11. Dorigo, M., Gambardella, L.M., 1997. Ant colonies for the travelling salesman problem. *BioSystems* 43(2), 73-81.



12. Dorigo, M., Gambardella, L.M., 1997. Ant colony system: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation* 1(1), 53-66.
13. Dorigo, M., Caro, G.D., Gambardella, L.M., 1999. Ant algorithms for discrete optimization. *Artificial Life* 5(2), 137-172.
14. Duan, Q., Liao, T. W., 2010. Improved ant colony optimization algorithms for determining project critical paths, *Automation in Construction* 19(6), 676-693.
15. Ehrhardt, R. 1979. Power approximation for computing (s, S) inventory policies. *Management Science* 25(8), 777-786.
16. Ehrhardt, R. 1984. (s, S) policies for a dynamic inventory model with stochastic lead times. *Operations Research* 32(1), 121-132.
17. Federgruen, A., Zheng, Y.S., 1992. An efficient algorithm for computing an optimal (r, q) policy in continuous review stochastic inventory systems. *Operations Research* 40(4), 808-813.
18. Freeland, J. R., Porteus, E. L., 1980. Evaluating the effectiveness of a new method for computing approximately optimal (s,S) inventory policies. *Operations Research* 28(2), 353-364.
19. Fries, B. E., 1975. Optimal ordering policy for a perishable commodity with fixed lifetime. *Operations Research* 23(1), 46-61.
20. Gürler, U. and Özkaya, B. Y., 2008. Analysis of the (s, S) policy for perishables with a random shelf life. *IIE Transactions* 40(8), 759-781.
21. Hadley, G., Whitin, T. M., 1963. *Analysis of inventory systems*. Prentice-Hall, Englewood Cliffs, NJ.
22. Johansen, S. G., Hill, R. M. 2000. The (r,Q) control of a periodic-review inventory system with continuous demand and lost sales. *International Journal of Production Economics* 68(3), 279-286.
23. Liao, T.W., 2009. Improving the accuracy of computer-aided radiographic weld inspection by feature selection. *Ndt & E International* 42(4), 229-239.

24. Liao, T. W., Daftardar, S., 2009. Model based optimization of friction stir welding processes. *Science and Technology of Welding and Joining* 14(5), 426-435.
25. Liao, T. W., 2010. Two hybrid differential evolution algorithms for engineering design optimization. *Applied Soft Computing* 10(4), 1188-1199.
26. Makridakis, S., Hibon, M., 2000. The M3-Competetion: results, conclusions, and implications. *International Journal of Forecasting* 16(4), 451-476.
27. Matheus, P., Gelders, L., 2000. The (R, Q) inventory policy subject to a compound Poisson demand pattern. *International Journal of Production Economics* 68(3), 307-317.
28. Naddor, E. 1975. Optimal heuristic decisions for s,S inventory policy. *Management Science Series A-Theory* 21(9), 1071-1072.
29. Nahmias, S., Pierskal, W. P., 1973. Optimal ordering policies for a product that perishes in 2 periods subject to stochastic demand. *Naval Research Logistics* 20(2), 207-229.
30. Nahmias, S. 1975. Optimal ordering policies for perishable inventory—II. *Operations Research*. 23(4), 735-749.
31. Nahmias, S., 1976 a. On the equivalence of three approximate continuous review inventory models. *Naval Research Logistics*. 23(1), 31-36.
32. Nahmias, S., 1976 b. Myopic approximations for perishable inventory problem. *Management Science* 22(9), 1002-1008.
33. Nahmias, S., 1978. Fixed-charge perishable inventory problem. *Operations Research* 26(3), 464-481.
34. Nahmias, S., 1982. Perishable inventory-theory—a review. *Operations Research* 30(4), 680-708.
35. Nandakumar, P. and Morton, T. E., 1993. Near myopic heuristics for the fixed-life perishability problem. *Management Science* 39(12), 1490-1498.
36. Olsson, F. and Tydesjo, P., 2010. Inventory problems with perishable items: Fixed lifetimes and backlogging. *European Journal of Operational Research* 202(1), 131-137.

37. Omran, M. G. H., Engelbrecht, A. P., and Salman, A., 2009. Bare bones differential evolution. *European Journal of Operational Research*. 196(1), 128-139.
38. Porteus, E.L., 1979. Adjustment to the Norman-White approach to approximating dynamic programs. *Operations Research* 27(6), 1203-1208.
39. Ravichandran, N., 1995. Stochastic-analysis of a continuous-review perishable inventory system with positive lead time and Poisson demand. *European Journal of Operational Research* 84(2), 444-457.
40. Reimann, M., Doerner, K., Hartl, R.F., 2004. D-Ants: Savings Based Ants divide and conquer the vehicle routing problem. *Computers & Operations Research* 31(4), 563-591.
41. Roberts, D. M., 1962. Approximations to optimal policies in a dynamic inventory model. K.J. Arrow, S. Karlin, H.E. Scarf, eds. *Studies in Applied Probability and Management Science*, Stanford University Press, Stanford, CA.
42. Rosling, K., 2002. The square-root algorithm for single-item inventory optimization. Working Paper, Växjö University (Revised for Management Science).
43. Sahin, I, Sinha, D., 1987. On asymptotic approximations for (s, S) policies. *Stochastic Analysis and Applications* 5(2), 189-212.
44. Schmidt, C. P. and S. Nahmias., 1985. (S-1,S) policies for perishable inventory. *Management Science*. 31(6), 719-728.
45. Schneider, H. 1978. Methods for determining reorder point of an (s, S) ordering policy when a service level is specified. *Journal of the Operational Research Society* 29(12), 1181-1193.
46. Sivazlian, B.D., 1971. Dimensional and computational analysis in stationary (s, S) inventory problems with gamma distributed demand. *Management Science Series B-Application* 17(6), B307-B311.
47. Socha, K. and Dorigo, M., 2008. Ant colony optimization for continuous domains. *European Journal of Operational Research* 185(3), 1155-1173.
48. Stützle, T., Hoos, H.H., 2000. MAX-MIN Ant System. *Future Generation Computer Systems* 16(8), 889-914.

49. Tadikamalla, P.R., 1978. Applications of the Weibull distribution in inventory control. *Journal of Operational Research Society* 29(1), 77-83.
50. Tijms, H.C., Groenevelt, H., 1984. Simple approximations for the reorder point in periodic and continuous review (s,S) inventory systems with service level constraints. *European Journal of Operational Research* 17(2), 175-190.
51. Van Zyl, G. J. J. 1964. Inventory control for perishable commodities. Unpublished Ph.D. dissertation. Univeristy of North Carolina, U.S.A.
52. Veinott, A.F., Wagner, H.M., 1965. Computing optimal (s,S) inventory policies. *Management Science* 11(5), 525-552
53. Wagner, H. M., 1969. *Principles of Operations Research*, Prentice Hall, Englewood Cliffs, New Jersey.
54. Wagner, I.A., Linderbaum, M., Bruckstein, A.M., 2000. ANTS: Agents, networks, trees, and subgraphs. *Future Generation Computer Systems* 16(8), 915-926.
55. Weiss, H. J., 1980. Optimal ordering policies for continuous review perishable inventory models. *Operations Research*. 28(2), 365-374. Al-Rifai, M. H., Rossetti, M. D., 2007. An efficient heuristic optimization algorithm for a two-echelon (R, Q) inventory system. *International Journal of Production Economics* 109 (1-2), 195-213.
56. Yano, C. A., 1985. New algorithms for (Q, r) systems with complete backordering using a fill-rate criterion. *Naval Research Logistics* 32(4), 675-688.
57. Zheng, Y.S., Federgruen, A., 1991. Finding optimal (s, s) policies is about as simple as evaluating a single policy. *Operations Research* 39(4), 654-665.
58. Zheng, Y.S., 1992. On properties of stochastic inventory systems. *Management Science* 38(1), 87-103.

## **VITA**

Qinglin Duan was born to Jianping Duan (father) and Yingxiang Xu (mother) in Nanjing, People's Republic of China, in 1985. She obtained her Bachelor of Science degree in Industrial Engineering at Shanghai University in 2007. Later, Qinglin attended the University of New South Wales and received a Master of Science degree in Manufacturing Engineering and Management in 2008.

Qinglin joined the Doctor of Philosophy program in Engineering Science, College of Engineering, Louisiana State University at Baton Rouge, in 2009. Along the way to pursue her Doctor of Philosophy degree, Qinglin is expected to earn a Master of Science degree in Industrial Engineering at the Department of Mechanical and Industrial Engineering, Louisiana State University at Baton Rouge, in December 2012. Qinglin's ongoing Doctor of Philosophy research is in supply chain inventory optimization and risk management with metaheuristics. She is expected to graduate from Louisiana State University with a Doctor of Philosophy in Engineering Science degree in 2013.

Qinglin's research interest is in supply chain inventory optimization, risk management, metaheuristics algorithms and applications.