

LAPORAN PENELITIAN

SOLVING NON-LINEAR DECREASING DEMAND INVENTORY POLICIES CONSIDERING SHORTAGE BACKORDERS USING PARTICLE SWARM OPTIMIZATION



The Jin Ai, S.T., M.T., D.Eng.
Ririn Diar Astanti, S.T., M.M.T., D.Eng.

**PROGRAM STUDI TEKNIK INDUSTRI
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ATMA JAYA YOGYAKARTA
2013**

LEMBAR PENGESAHAN LAPORAN PENELITIAN INTERNAL KELOMPOK

1	Judul Proposal Penelitian	Solving Non-Linear Decreasing Demand Inventory Policies Considering Shortage Backorders Using Particle Swarm Optimization	
2	Kata Kunci (Inggris)	Inventory Policy, Non-Linear Decreasing Demand, Shortage Backorders, Particle Swarm Optimization	
3	Jenis Kegiatan	Laboratorium/ Lapangan /Pustaka	
4	Nama Ketua Peneliti	The Jin Ai, S.T., M.T., D.Eng.	
	Pangkat/Golongan	Lektor Kepala/IVA	
	NPP/NIDN	01.99.662/0520117501	
	Jabatan	Ketua Program Studi Teknik Industri	
	Unit/Fakultas/Jurusan	Fakultas Teknologi Industri / Teknik Industri	
	Alamat	Jl. Babarsari 43 Yogyakarta 55281	
	No. Telp/Faks	0274 – 487711	
	Email	jinai@mail.uajy.ac.id	
		Anggota-1	Anggota-2
	Nama Lengkap	Ririn Diar Astanti, S.T., M.M.T., D.Eng.	
	Pangkat/Golongan	Asisten Ahli/IIIB	
	NPP/NIDN	11.03.767/0518127701	
	Jabatan	Koordinator Pengelola Kelas Internasional Teknik Industri	
	Unit/Fakultas/Jurusan	Fakultas Teknologi Industri / Teknik Industri	
5	Lokasi Penelitian	Laboratorium Sistem Produksi Fakultas Teknologi Industri UAJY	
6	Jarak dari Kampus UAJY	0 (Km)	0 (Jam)
7	Waktu Pelaksanaan	Periode: Januari – Juni 2013	Waktu Efektif: 90 (Jam)
8	Berbebani SKS	Ketua: 0 (sks)	Anggota: 0 (sks)
9	Biaya	Dana UAJY	Dana Pribadi Peneliti
		Rp. 4.140.000	Rp.
	Jumlah Total	Rp. 4.140.000	
	Terbilang	Empat Juta Seratus Empat Puluh Ribu Rupiah	

Mengetahui dan Menyetujui
Pimpinan Unit,

Yogyakarta, 31 Juli 2013
Ketua Peneliti,

Ir. B. Kristyanto, M.Eng., Ph.D.
NPP: 05.91.343/NIDN: 0510036001

The Jin Ai, S.T., M.T., D.Eng.
NPP: 01.99.662/NIDN: 0520117501

Mengetahui dan Menyetujui
Ketua LPPM,

Dr. Ir. Y. Djarot Purbadi, MT
NPP: 08.87.217/NIDN: 0516065701

ABSTRACT

This research considering a non-linear decreasing demand inventory policies considering shortage backorders problem, in which the demand is known and decreases exponentially, the instantaneous replenishment is made at time t_i ($i=1,2,...,n$) where $t_1=0$, the quantity received at t_i is used partly to meet accumulated shortages in the previous cycle from time s_{i-1} to t_i ($s_{i-1} < t_i$) ($i=2,...,n$), and no shortages at the beginning ($t_1=0$) and the end of the planning horizon ($s_n=H$).

An algorithm based on particle swarm optimization is proposed in this research for solving the defined problem. The algorithm is consist of an enumerative procedure to find optimal number of replenishment (n^*) over a particle swarm optimization to find the optimal set of shortage starting point of each cycle ($\{s_i^*\}$) and the set of replenishment time ($\{t_i^*\}$). The particle representation of the problem for fixed value of n is a random key of $(n-1)$ elements, in which particle is consists of $(n-1)$ dimensions and each dimensions position is limited from 0 to 1.

Computational testing results indicate that the proposed algorithm is outperformed well known method in the past for solving the same problem, i.e. Wee's method (1995) in which the proposed method is able to provide smaller total cost across various testing problem.

Keywords: Inventory Policy, Non-Linear Decreasing Demand, Shortage Backorder, Particle Swarm Optimization, Solution Methodology, Heuristics

ACKNOWLEDGEMENT

The authors acknowledge Prof. Dr. Voratas Kachitvichyanukul and his High Performance Computing Research Group in Asian Institute of Technology, Thailand, for providing the ET-Lib, the evolutionary computing library that being used as the basis of the particle swarm optimization programming part in this research.

The authors also acknowledge Prof. Dr. Hui-Ming Wee and his Supply Chain Re-engineering Research Group in Chung Yuan Christian University, Taiwan, for intense discussion of the decreasing demand inventory problem.

TABLE OF CONTENTS

Title Page	i
Identification Page	ii
Abstract	iii
Acknowledgement	iv
Table of Contents	v
CHAPTER 1 INTRODUCTION	1
CHAPTER 2 LITERATURE REVIEW	5
CHAPTER 3 METHODOLOGY	9
CHAPTER 4 MATHEMATICAL AND ALGORITHM DEVELOPMENT	10
CHAPTER 5 RESULTS AND DISCUSSION	20
CHAPTER 6 CONCLUSION AND RECOMMENDATIONS	24
LIST OF REFERENCES	25
APPENDICES	27

CHAPTER 1

INTRODUCTION

1.1. TITLE

This research is entitled “Solving Non-Linear Decreasing Demand Inventory Policies Considering Shortage Backorders Using Particle Swarm Optimization”

1.2. RATIONALE

Product can be categorized into two types. They are functional product such as rice, sugar and mineral water, and innovative product such as electronics products, automotive product and apparels product. Some of the innovative products demand may decrease with time. This might be due to the shift in customer preference or the newer developed technology that may replace the older products. An example for this case is the tube-television which is being replaced by LCD or LED technology. As a result, the demand of the tube-television is decreasing from time to time. To the best of authors’ knowledge there is limited research for inventory policy problem with decreasing demand pattern (see Chapter 2 for detail literature reviews).

In practical cases, there is a situation when customer order arrives but the finishes good are not available. This situation is called shortage. Several researchers in the past have proposed inventory model considering the shortage and its solution methodology for several specific cases using various optimization approaches (see Chapter 2 for detail literature reviews).

Particle Swarm optimization (PSO) is a population based search method proposed by Kennedy and Eberhart (1995), which were motivated by the group organism behavior such as bee swarm, fish school, and bird flock. PSO imitated the physical movements of

the individuals in the swarm as a searching method, altogether with its cognitive and social behavior as local and global exploration abilities. One of PSO advantage is its simplicity of its iteration step which only consists of updating two set of equations. Due to its simplicity and unexplored potential in the inventory policy research area, this research is trying to apply PSO to solve the non-linear decreasing demand inventory policies considering shortage backorders.

1.3. PROBLEM STATEMENT

The main problem in this research is to find an effective and efficient way on applying Particle Swarm Optimization to solve a non-linear decreasing demand inventory policies considering shortage backorders problem that is defined below.

The problem is developed based following assumptions:

- a) Demand is known and decreases exponentially
- b) The replenishment is made at time t_i ($i = 1, 2, \dots, n$) where $t_1 = 0$
- c) Replenishment is instantaneous
- d) The quantity received at t_i is used partly to meet accumulated shortages in the previous cycle from time s_{i-1} to t_i ($s_{i-1} < t_i$); ($i = 2, \dots, n$)
- e) No shortages at the beginning ($t_1 = 0$) and the end of the planning horizon ($s_n = H$)

The notations used are defined below:

- H length of the planning horizon under consideration
- A initial demand rate
- α parameter of the decreasing rate of demand rate

$f(t)$ instantaneous demand rate at time t , $f(t) = A \cdot e^{-\alpha t} \ t \geq 0$

$I(t)$ inventory level at time $t \in [t_i, t_{i+1}]$ which is evaluated after the replenishment arrives at time t_i and before the replenishment arrives at time t_{i+1} in the i^{th} cycle $[t_i, t_{i+1}]$

c_1 ordering cost per order

c_2 holding cost per unit per unit time

c_3 shortage cost per unit per unit time

n number of replenishments

t_i the i^{th} replenishment time ($i = 1, 2, \dots, n$)

s_i the shortage starting point of cycle i , which is the time at which the inventory level reaches zero in the i^{th} cycle $[t_i, t_{i+1}]$; ($i = 1, 2, \dots, n-1$)

Graphical representation of behavior of the inventory system with decreasing demand and shortage backorders is shown in Figure 1.

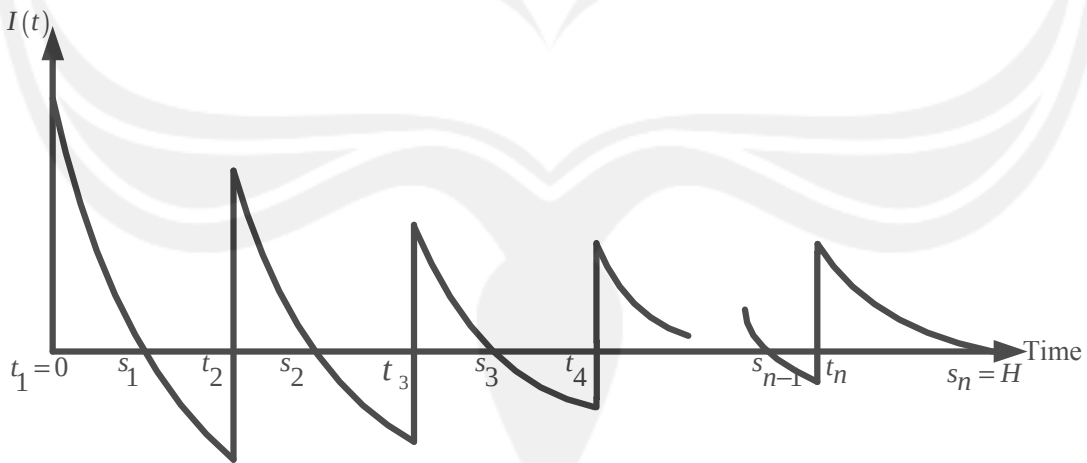


Figure 1. Graphical Representation of the Inventory System with Decreasing Demand and Shortage Backorders

1.4. OBJECTIVES AND OUTCOME

The overall objective of this research is to develop effective and efficient particle swarm optimization algorithms for solving the non-linear decreasing demand inventory policies considering shortage backorders problem which are able to provide high quality solutions in reasonable amount of computational time. This objective is achieved through the following sub-objectives:

- to develop a PSO algorithm for solving the non-linear decreasing demand inventory policies considering shortage backorders problem,
- to implement the computer program of the PSO algorithm for solving the non-linear decreasing demand inventory policies considering shortage backorders problem,
- to evaluate the performance of proposed method on a benchmark problems and compare with other methods.

The expected outcome of this research is a research paper to be published in international journal. The target journal is “International Journal of Swarm Intelligence” published by Inderscience publisher.

1.5. SIGNIFICANCE OF THIS RESEARCH

Significance of this research:

1. Propose a new technique for solving the non-linear decreasing demand inventory policies considering shortage backorders problem
2. Broaden the applicability of particle swarm optimization

CHAPTER 2

LITERATURE REVIEW

2.1. INVENTORY POLICY MODEL

The first inventory policy model with linear decreasing demand was proposed by Zhao et al. (2001). They also developed heuristic approach for variable replenishment interval policy to solve the model. This research was followed by Goyal and Giri (2003) who also developed heuristic approach for variable replenishment interval policy and stated that their heuristic approach perform better than that of Zhao et al. (2001).

As demand might follow non-linear pattern, several researches have proposed inventory policy problem for non-linear decreasing demand patterns. Yang et al. (2004) developed the eclectic approach for solving inventory policy problem with non-linear decreasing demand $(f(t) = a + bt + ct^2)$ under the condition that $a \geq 0, b < 0, c > 0$ and $b + 2ct \leq 0$. Their approach allows each cycle length to be variable (i.e. variable replenishment interval policy).

In practical cases, there is a situation when customer order arrives but the finishes good are not available. This situation is called shortage. Several researchers have proposed inventory model considering the shortage. Wee (1995) developed an exact solution method for inventory policy problem by considering exponentially decreasing demand pattern $(f(t) = A \exp^{-\alpha t})$ and shortage situation. The method allows each cycle length to be constant (i.e. fixed replenishment interval policy). The deterioration factor (θ) was incorporated in Wee's model (1995). Even though the solution procedure proposed by Wee initially applied for deteriorating cases, the method can also be applied for non deteriorating cases by setting the deterioration factor equal to zero $(\theta = 0)$. Benkherouf

(1998) investigated the method of Wee (1995) and stated that “...*the procedure suggested by Wee is independent of the demand rate.*”

Benkherouft (1995) developed the optimal procedure under the condition that demand $f(t)$ are decreasing over a known and finite planning horizon and $f(t)/f'(t)$ are non-increasing in t . Unlike Wee's (1995) method, Benkherouft (1995) method was dealing with variable replenishment interval policy.

Basically there are 2 types of inventory policy which are: fixed interval replenishment policy and variable replenishment interval policy. Those policies can be applied to several inventory models. According to Zhou et al. (2004) inventory model considering shortage can be group into two. They are: Inventory Followed by Shortage (IFS) Model and Shortage Followed by Inventory (SFI) Model. Each group consist of two models where the difference between two models in each group is that whether the shortage is allowed to be happened in the last cycle or not. (Teng et al. 1997). This research is dealing with IFS Model.

2.2. PARTICLE SWARM OPTIMIZATION

Particle Swarm is a population based stochastic optimization technique developed by Eberhart and Kennedy (1995), inspired by social behavior of bird flocking or fish schooling.

Clerc (2006) states that the basic principle of PSO is a set of moving particles is placed inside the search space. Each particle has the following characteristics:

- It has a position (θ) and velocity (ω). Its positions are multi-dimensional, and can be translated or decoded into the solution, which will be evaluated with the objective function (minimization or maximization).

- It knows its position and the objective function value of this position, but it also remember its best previous position or pbest (ψ).
- It know its neighbor's best previous position and objective function value.

Particles have several choices to move in each period of time or iteration elapsed, there are:

- Follow its own way,
- Go towards its best previous position,
- Go towards its neighbor's best previous position

The movement of particles is called velocity, that can be formulated as follow:

$$\omega_{li}(\tau+1) = w(\tau+1)\omega_{lh}(\tau) + c_p u(\psi_{lh}-\theta_{lh}(\tau)) + c_g u(\psi_{gh}-\theta_{lh}(\tau)) \quad (1)$$

$$\theta_{lh}(\tau+1) = \theta_{lh}(\tau) + \omega_{lh}(\tau+1) \quad (2)$$

where τ is iteration index, l is particle index, h is dimension index, u is uniform random number in interval $[0,1]$, $w(\tau)$ is inertia weight in the τ^{th} iteration, $\omega(\tau)$ is velocity of l^{th} particle at the h^{th} dimension in the τ^{th} iteration, $\theta(\tau)$ is position of l^{th} particle at the h^{th} dimension in the τ^{th} iteration, ψ_{lh} is personal best position (pbest) of l^{th} particle at the h^{th} dimension, ψ_{gh} is global best position (gbest) of l^{th} particle at the h^{th} dimension, c_p is personal best acceleration constant, and c_g is global best acceleration constant.

Equation (2) shows that the particle position of next period is obtained from the sum of the current position with the velocity of the next period. Equation (1) showed that the velocity of next period obtained from the sum of the times of social or cognitive weights (w , c_p , and c_g) with current velocity, pbest, and gbest. Generally, the algorithm of PSO will be:

- Initialization: Determine the number of particles (N), the particle's position (can be random or defined), and particle's velocity.
- Decode particles into solution.

- Evaluate the particles, based on the objective function.
- Update pbest value,
- Update gbest value,
- Update velocity and position for each particle,
- If the stopping criterion is reached, stop. Otherwise return to decoding step.

The PSO algorithm had been proposed for solving various optimization problems such as Traveling Salesman Problem (Clerc, 2004), Protein Motif Discovery (Chang et al, 2004), as well as the Vehicle Routing with Simultaneous Pickup and Delivery by Ai and Kachitvichyanukul (2009).

CHAPTER 3

METHODOLOGY

Following steps are taken to accomplished the research:

1. Literature Survey

- Survey state-of-the-art research related to non-linear decreasing demand inventory policies considering shortage backorders problem
- Survey state-of-the-art research related to application of particle swarm optimization, especially on inventory policy problem
- Select some basic ideas that can be utilized for developing an application of particle swarm optimization to non-linear decreasing demand inventory policies considering shortage backorders problem

2. Algorithm Development

- Develop the elements of particle swarm optimization algorithm (solution representation, decoding method, etc.) for solving the non-linear decreasing demand inventory policies considering shortage backorders problem
- Implement the algorithm into computer program

3. Computational Experiment

- Select some benchmark problem on non-linear decreasing demand inventory policies considering shortage backorders problem
- Run the computer program on the selected benchmark problem
- Compare the result of proposed method with existing method in the past

CHAPTER 4

MATHEMATICAL AND ALGORITHM DEVELOPMENT

4.1. OPTIMIZATION PROBLEM

The behavior of the inventory system with decreasing demand and shortage backorders considered in this research is represented graphically in Figure 1. The total inventory cost of this system can be expressed as:

$$TC(n, \{s_i\}, \{t_i\}) = nc_1 + c_2 \sum_{i=1}^n I_i + c_3 \sum_{i=1}^{n-1} S_i \quad (3)$$

where

I_i cumulative holding inventory during cycle i ($i = 1, 2, \dots, n$), and is determined as:

$$I_i = \int_{t_i}^{s_i} I(t) dt = \int_{t_i}^{s_i} \int_t^{s_i} f(\tau) d\tau dt \quad (4)$$

S_i cumulative shortage during cycle i ($i = 1, 2, \dots, n-1$), and is determined as:

$$S_i = \int_{s_i}^{t_{i+1}} (-I(t)) dt = \int_{s_i}^{t_{i+1}} \int_t^{s_i} f(\tau) d\tau dt \quad (5)$$

Eq. (4) and (5) are expanded and the expansions are substituted into Eq.(3). The total inventory cost become:

$$TC(n, \{s_i\}, \{t_i\}) = nc_1 + c_2 \sum_{i=1}^n \left\{ (s_i - t_i) F(s_i) - \int_{t_i}^{s_i} F(t) dt \right\} + c_3 \sum_{i=1}^{n-1} \left\{ (s_i - t_{i+1}) F(s_i) + \int_{s_i}^{t_{i+1}} F(t) dt \right\} \quad (6)$$

The optimization problem is to find number of replenishment (n), the set of shortage starting point of each cycle ($\{s_i\}$), and the set of replenishment time ($\{t_i\}$) in order to minimize Total Cost (TC) expressed in the Eq. (6).

4.2. OBTAINING SHORTAGE STARTING POINT OF A CYCLE

The shortage starting point s_i of a cycle, except the last cycle, is determined based on the reduction cost concept which is proposed by Astanti and Luong (2009). The reduction cost is defined as the difference between the reduction in holding cost when shortage is allowed and the shortage cost that is occurred. It can be formulated as follows:

$$\text{Maximize } RC_2^{(i)} = c_2(RH^{(i)}) - c_3(RS^{(i)}) \quad (7)$$

Subject to

$$t_i \leq s_i \leq t_{i+1}$$

in which:

$RC_2^{(i)}$ Reduction cost in cycle i

$RH^{(i)}, RS^{(i)}$ the reduction of cumulative inventory and cumulative shortage in cycle i as shown in figure 2.

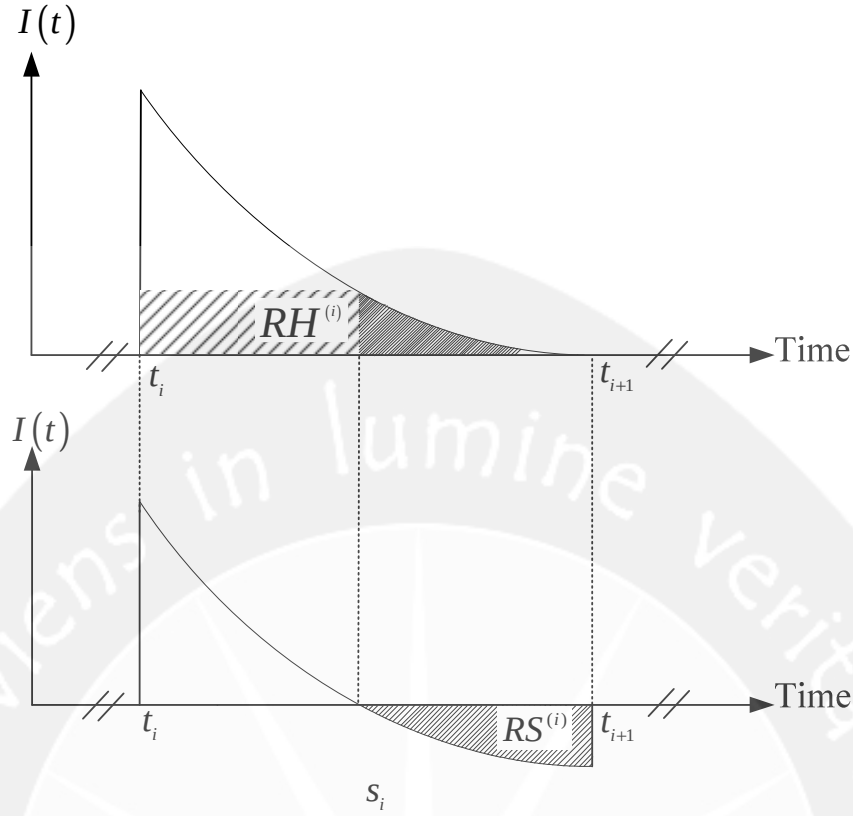


Figure 2. Reduction in Holding Cost by Incorporating Shortage

Shortage is allowed in the corresponding cycle if the maximum reduction cost is positive.

The detail expression of $RC_2^{(i)}$ is briefly discussed below (Astanti and Luong, 2009):

$$\begin{aligned}
 RC_2^{(i)} &= c_2 \left(RH^{(i)} \right) - c_3 \left(RS^{(i)} \right) \\
 &= c_2 \left\{ (s_i - t_i) \int_{s_i}^{t_{i+1}} f(t) dt + \int_{s_i}^{t_{i+1}} I(t) dt \right\} - c_3 \left\{ \int_{s_i}^{t_{i+1}} S(t) dt \right\} \\
 &= c_2 \left\{ (t_{i+1} - t_i) F(t_{i+1}) + (t_i - s_i) F(s_i) - \int_{s_i}^{t_{i+1}} F(t) dt \right\} \\
 &\quad - c_3 \left\{ (s_i - t_{i+1}) F(s_i) + \int_{s_i}^{t_{i+1}} F(t) dt \right\}
 \end{aligned} \tag{8}$$

Proposition: For each cycle i , the optimal value of s_i always exist and it can be determined as:

$$s_i^* = \frac{c_2 t_i + c_3 t_{i+1}}{c_2 + c_3} \quad (9)$$

Proof:

We have

$$\begin{aligned} \frac{dRC_2^{(i)}}{ds_i} &= 0 \\ \Leftrightarrow c_2 \{0 + t_i f(s_i) - F(s_i) - s_i f(s_i) + F(s_i)\} - c_3 \{F(s_i) + s_i f(s_i) - t_{i+1} f(s_i) - F(s_i)\} &= 0 \\ \Leftrightarrow c_2 t_i f(s_i) - c_2 s_i f(s_i) - c_3 s_i f(s_i) + c_3 t_{i+1} f(s_i) &= 0 \\ \Leftrightarrow -s_i f(s_i) [c_2 + c_3] + f(s_i) [c_2 t_i + c_3 t_{i+1}] &= 0 \\ \Leftrightarrow f(s_i) [c_2 t_i + c_3 t_{i+1} - (c_2 + c_3) s_i] &= 0 \\ \Leftrightarrow s_i^* = \frac{c_2 t_i + c_3 t_{i+1}}{c_2 + c_3} \end{aligned}$$

It is noted that the value of s_i defined above is valid, i.e., $t_i < s_i < t_{i+1}$. And

when $s_i > s_i^*$, $\frac{dRC_2^{(i)}}{ds_i} < 0$, and $\frac{dRC_2^{(i)}}{ds_i} > 0$ when $s_i < s_i^*$. Therefore, $RC_2^{(i)}$ will be maximized at s_i^* .

4.3. PARTICLE REPRESENTATION OF THE PROBLEM WITH FIXED N

As mentioned before, PSO is a population based search method that imitated the physical movements of the individuals in the swarm as a searching method. In PSO, a swarm of particles served as searching agent for a specific problem solution. A particle position is representing a solution of the problem. This section discusses how the particle can represent the problem.

From the structure of the problem represented in Figure 1, it is clearly seen that $t_1 = 0$ and $s_n = H$. Therefore, for fixed value of n , the number of independent variable is $(n - 1)$, which are $t_2, t_3, \dots, t_n$ and the boundary of the decision variable is $0 < t_2 < t_3 < \dots < t_n < H$. Therefore, for fixed value of n , particle representation is random key of $(n - 1)$ elements, so the particle is consists of $(n - 1)$ dimensions, and each dimensions position is limited from $0 < \theta_{li} < 1$. The decoding step from particle position into the set of shortage starting point of each cycle ($\{s_i\}$) and the set of replenishment time ($\{t_i\}$) is explained in Algorithm 1.

Algorithm 1. Decoding method

Step 1. Sort the particle position from the smallest to the largest one.

Step 2. Calculate $t_i = \theta_{[i]} \cdot H$

Step 3. Calculate $s_i = \frac{c_2 t_i + c_3 t_{i+1}}{c_2 + c_3}$

Example 1. Particle Representation and Decoding Method

Consider the problem with $c_2 = 10$, $c_3 = 40$, and $H = 4$. Given $n = 5$, therefore the particle consists of $n - 1 = 4$ dimensions. Suppose there is a particle with position value of $[0.8; 0.4; 0.1; 0.6]$. Following algorithm 1, after sorting the position value we have:

$$\theta_{[1]} = 0.1$$

$$\theta_{[2]} = 0.4$$

$$\theta_{[3]} = 0.6$$

$$\theta_{[4]} = 0.8$$

Therefore, the value of t_i can be determined consequently:

$$t_2 = \theta_{[1]} \cdot H = 0.1 \cdot 4 = 0.4$$

$$t_3 = \theta_{[2]} \cdot H = 0.4 \cdot 4 = 1.6$$

$$t_4 = \theta_{[3]} \cdot H = 0.6 \cdot 4 = 2.4$$

$$t_5 = \theta_{[4]} \cdot H = 0.8 \cdot 4 = 3.2$$

It is noted that $t_1 = 0$ and $s_5 = H = 5$. Finally, the value of s_i can be determined accordingly:

$$s_1 = \frac{c_2 t_1 + c_3 t_2}{c_2 + c_3} = \frac{10 \cdot 0 + 40 \cdot 0.4}{10 + 40} = 0.32$$

$$s_2 = \frac{c_2 t_2 + c_3 t_3}{c_2 + c_3} = \frac{10 \cdot 0.4 + 40 \cdot 1.6}{10 + 40} = 1.36$$

$$s_3 = \frac{c_2 t_3 + c_3 t_4}{c_2 + c_3} = \frac{10 \cdot 1.6 + 40 \cdot 2.4}{10 + 40} = 2.24$$

$$s_4 = \frac{c_2 t_4 + c_3 t_5}{c_2 + c_3} = \frac{10 \cdot 2.4 + 40 \cdot 3.2}{10 + 40} = 3.04$$

Therefore, it can be concluded that particle with value of [0.8; 0.4; 0.1; 0.6] is corresponding with solution of the problem with $\{t_i\} = \{0.0; 0.4; 1.6; 2.4; 3.2\}$ and $\{s_i\} = \{0.32; 1.36; 2.24; 3.04; 4.00\}$.

4.4. PARTICLE SWARM OPTIMIZATION ALGORITHM FOR THE PROBLEM WITH FIXED N

The details of the PSO algorithm for solving the problem are explained below in Algorithm 2. In this algorithm, the particles are initialized in step 1. The iteration of particles movement is described by steps 2-8, in which the particles' fitness value are evaluated in steps 2-3, their cognitive and social information are updated in steps 4-7, and

their positions are updated in step 8. Step 9 is the controlling step to repeat or stop the iteration. It is noted that $\theta^{\min} = 0$ and $\theta^{\max} = 1$.

Notation

τ	: Iteration index; $\tau = 1 \dots T$
l	: Particle index, $l = 1 \dots L$
h	: Dimension index, $h = 1 \dots H$
u	: Uniform random number in the interval $[0,1]$
$w(\tau)$	: Inertia weight in the τ^{th} iteration
$\omega_{lh}(\tau)$	: Velocity of the l^{th} particle at the h^{th} dimension in the τ^{th} iteration
$\theta_{lh}(\tau)$	: Position of the l^{th} particle at the h^{th} dimension in the τ^{th} iteration
ψ_{lh}	: Personal best position (pbest) of the l^{th} particle at the h^{th} dimension
ψ_{gh}	: Global best position (gbest) at the h^{th} dimension
ψ_{lh}^L	: Local best position (lbest) of the l^{th} particle at the h^{th} dimension
ψ_{lh}^N	: Near neighbor best position (nbest) of the l^{th} particle at the h^{th} dimension
c_p	: Personal best position acceleration constant
c_g	: Global best position acceleration constant
c_l	: Local best position acceleration constant
c_n	: Near neighbor best position acceleration constant
$\theta^{\max}$	: Maximum position value
$\theta^{\min}$	: Minimum position value

- Θ_l : Vector position of the l^{th} particle, $[\theta_{l1} \ \theta_{l2} \ \cdots \ \theta_{lH}]$
 Ω_l : Vector velocity of the l^{th} particle, $[\omega_{l1} \ \omega_{l2} \ \cdots \ \omega_{lH}]$
 Ψ_l : Vector personal best position of the l^{th} particle, $[\psi_{l1} \ \psi_{l2} \ \cdots \ \psi_{lH}]$
 Ψ_g : Vector global best position, $[\psi_{g1} \ \psi_{g2} \ \cdots \ \psi_{gH}]$
 Ψ_l^L : Vector local best position of the l^{th} particle, $[\psi_{l1}^L \ \psi_{l2}^L \ \cdots \ \psi_{lD}^L]$
 $Z(\Theta_l)$: Fitness value of Θ_l
 FDR : Fitness-distance-ratio

Algorithm 2: PSO Algorithm for Non-Linear Decreasing Demand Inventory Policies Considering Shortage Backorders

1. Initialize L particles as a swarm, generate the l^{th} particle with random position Θ_l in the range $[\theta^{\min}, \theta^{\max}]$, velocity $\Omega_l = 0$ and personal best $\Psi_l = \Theta_l$ for $l=1 \dots L$. Set iteration $\tau = 1$.
2. For $l=1 \dots L$, decode $\Theta_l(\tau)$ to a set of solution R_l - each R_l is consists of $\{s_i\}$ and $\{t_i\}$ based on Algorithm 1.
3. For $l=1 \dots L$, compute the objective function of R_l , and set this value as the fitness value of Θ_l , represented by $Z(\Theta_l)$.
4. Update pbest: For $l=1 \dots L$, update $\Psi_l = \Theta_l$, if $Z(\Theta_l) < Z(\Psi_l)$.
5. Update gbest: For $l=1 \dots L$, update $\Psi_g = \Psi_l$, if $Z(\Psi_l) < Z(\Psi_g)$.
6. Update lbest: For $l=1 \dots L$, among all pbest from K neighbors of the l^{th} particle, set the personal best which obtains the least fitness value to be Ψ_l^L .

7. Generate nbest: For $l=1...L$, and $h=1...H$, set $\psi_{lh}^N = \psi_{oh}$ that maximizing fitness-distance-ratio (FDR) for $o=1...H$. Where FDR is defined as

$$FDR = \frac{Z(\Theta_l) - Z(\Psi_o)}{|\theta_{lh} - \psi_{oh}|} \text{ which } l \neq o \quad (3.4)$$

8. Update the velocity and the position of each l^{th} particle:

$$w(\tau) = w(T) + \frac{\tau - T}{1 - T} [w(1) - w(T)] \quad (3.5)$$

$$\begin{aligned} \omega_{lh}(\tau+1) = & w(\tau) \omega_{lh}(\tau) + c_p u(\psi_{lh} - \theta_{lh}(\tau)) + c_g u(\psi_{gh} - \theta_{lh}(\tau)) \\ & + c_l u(\psi_{lh}^L - \theta_{lh}(\tau)) + c_n u(\psi_{lh}^N - \theta_{lh}(\tau)) \end{aligned} \quad (3.6)$$

$$\theta_{lh}(\tau+1) = \theta_{lh}(\tau) + \omega_{lh}(\tau+1) \quad (3.7)$$

If $\theta_{lh}(\tau+1) > \theta^{\max}$, then

$$\theta_{lh}(\tau+1) = \theta^{\max} \quad (3.8)$$

$$\omega_{lh}(\tau+1) = 0 \quad (3.9)$$

If $\theta_{lh}(\tau+1) < \theta^{\min}$, then

$$\theta_{lh}(\tau+1) = \theta^{\min} \quad (3.10)$$

$$\omega_{lh}(\tau+1) = 0 \quad (3.11)$$

9. If the stopping criterion is met, i.e. $\tau = T$, stop. Otherwise, $\tau = \tau + 1$ and return to step 2.

4.5. FINDING THE OPTIMAL NUMBER OF REPLENISHMENT (N*)

It is noted that the optimization problem considered here is to find number of replenishment (n), the set of shortage starting point of each cycle ($\{s_i\}$), and the set of replenishment time ($\{t_i\}$) in order to minimize Total Cost (TC) expressed in the Eq. (6).

PSO Algorithm presented in Algorithm 2 (Section 4.5) is able to find the best value of set of shortage starting point of each cycle ($\{s_i^*\}$), and the set of replenishment time ($\{t_i^*\}$) given the fixed value of n .

Therefore, it is still needed to find the optimal number of replenishment (n^*). This research proposes an enumeration technique over PSO Algorithm to tackle this situation. The optimal value of n can be obtained by evaluating the $TC(n, \{t_i\}, \{s_i\})$ from Eq. (6) using enumeration technique starting from $n=1$. The detail enumerative procedure for obtaining optimal value of n is presented in Algorithm 3.

Algorithm 3: Enumerative Procedure to Find n^*

1. Set $n = 1$. Evaluate $TC(n=1)$ and set $TC^* = TC(n=1)$
2. Set $n = n + 1$. Use PSO algorithm (Algorithm 2) to determine the best value of $\{t_i\}$ and $\{s_i\}$ and evaluate $TC(n, \{t_i\}, \{s_i\})$
3. If $TC(n, \{t_i\}, \{s_i\}) < TC^*$ then set $TC^* = TC(n, \{t_i\}, \{s_i\})$ and go to Step 2. Otherwise, set $n^* = n - 1$ and Stop.

CHAPTER 5

RESULTS AND DISCUSSION

5.1. COMPUTATIONAL EXPERIMENTS

The numerical examples used in this research follow the example of Wee (1995). The demand rate decreases exponentially following $f(t) = 500 \cdot e^{-0.98t}$ and the ordering cost is \$250 per order (c_1), the carrying cost is \$40 per unit per year (c_2), the backlogged shortage cost is \$80 per unit per year (c_3) and the system operates during a prescribed period of 4 year (H).

In order to make generalization, the demand rate $f(t) = 500 \cdot e^{-0.98t}$ is considered as $f(t) = A \cdot e^{-\alpha t}$. Therefore, the cumulative demand rate can be calculated as

$$F(t) = \int_0^t A \cdot e^{-\alpha t} dt = \frac{A}{\alpha} (1 - e^{-\alpha t}) \quad (17)$$

and

$$\int \frac{A}{\alpha} (1 - e^{-\alpha t}) dt = \frac{Ae^{-\alpha t}}{\alpha^2} + \frac{At}{\alpha} \quad (18)$$

Substituting equations 17 and 19 to equation 6, the total cost expression for this numerical example can be stated as

$$\begin{aligned} TC(n, \{s_i\}, \{t_i\}) = & nc_1 + c_2 \sum_{i=1}^n \left\{ (s_i - t_i) \frac{A}{\alpha} (1 - e^{-\alpha s_i}) - \frac{Ae^{-\alpha s_i}}{\alpha^2} - \frac{As_i}{\alpha} + \frac{Ae^{-\alpha t_i}}{\alpha^2} + \frac{At_i}{\alpha} \right\} + \\ & c_3 \sum_{i=1}^{n-1} \left\{ (s_i - t_{i+1}) \frac{A}{\alpha} (1 - e^{-\alpha s_i}) + \frac{Ae^{-\alpha t_{i+1}}}{\alpha^2} + \frac{At_{i+1}}{\alpha} - \frac{Ae^{-\alpha s_i}}{\alpha^2} - \frac{As_i}{\alpha} \right\} \end{aligned} \quad (19)$$

In the computational experiments, various parameter values of c_1 , c_2 , and c_3 are incorporated in order to observed the performance of proposed method. Nine (9) problems are then generated with each corresponding parameters presented in Table 1.

Table 1. Problem Parameters

Problem	Problem Parameters		
	c_1	c_2	c_3
1	250	10	40
2	250	10	80
3	250	10	120
4	250	20	40
5	250	20	80
6	250	20	120
7	250	40	40
8	250	40	80
9	250	40	120

5.2. DETAIL RESULT OF PROBLEM 1

The proposed method is run with following algorithm parameters: number of particles (L) = 30, number of iteration (τ) = 1000, number of neighbor (K) = 5, parameter $w(1) = 0.9$, parameter $w(T) = 0.4$, parameter $c_p = 1$, parameter $c_g = 1$, parameter $c_l = 1$, parameter $c_n = 1$.

After running 10 replications of algorithm, it is found that the best solution is found at $n^* = 6$ with $TC^* = 2420.0$. The details of s_i and t_i are presented in Table 2.

Table 2. Detail Results of Problem 1

i	s_i	t_i
1	0.000000	0.248616
2	0.310769	0.605979
3	0.679781	1.043771
4	1.134768	1.611264
5	1.730388	2.428165
6	2.602610	4.000000

5.3. RESULTS AND COMPARISON

Using the same setting as Problem 1, the remaining problems are solved. The results in terms of the best number of replenishment and its corresponding total cost are presented in Table 3. For comparison purpose, the result of the same problem solved using Wee's method (1995) is presented. The detail of s_i and t_i are presented in Appendix 2.

Table 3. Computational Result in terms of (n , TC)

Problem	Wee's method (1995)	PSO
1	(5; 2720.4)	(6; 2420.0)
2	(5; 2844.5)	(6; 2505.4)
3	(5; 2890.7)	(6; 2536.9)
4	(7; 3596.8)	(7; 3220.1)
5	(8; 3901.6)	(8; 3448.4)
6	(8; 4017.3)	(8; 3539.2)
7	(9; 4484.9)	(9; 4031.7)
8	(10; 5112.8)	(10; 4543.8)
9	(11; 5398.3)	(10; 4778.1)

It is shown from Table 3 above that this proposed PSO method is outperformed Wee's method (1995) in which this proposed method is able to provide smaller total cost across all nine testing problem.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

In this research, an algorithm based on particle swarm optimization is successfully develop for solving the non-linear decreasing demand inventory policies considering shortage backorders problem. The algorithm is consist of an enumerative procedure to find optimal number of replenishment (n^*) over a particle swarm optimization to find the optimal set of shortage starting point of each cycle ($\{s_i^*\}$) and the set of replenishment time ($\{t_i^*\}$). It is noted that the particle representation of the problem for fixed value of n is a random key of $(n - 1)$ elements, in which particle is consists of $(n - 1)$ dimensions and each dimensions position is limited from 0 to 1.

When the proposed algorithm is compared with the well known method in the past for solving the same problem, i.e. Wee's method (1995), it is shown that the proposed PSO method is outperformed Wee's method (1995) in which this proposed method is able to provide smaller total cost across various testing problem.

To broaden the application of particle swarm optimization, it is suggested in the future that this approach is tested over various problem in the inventory management area and being evaluated its performance.

LIST OF REFERENCES

- Ai, T.J., dan Kachitvichyanukul, V., (2009) *A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery*, *Computers & Operations Research* 36, 1693-1702.
- Astanti, R.D., dan Luong, H.T. (2009) A heuristic technique for inventory replenishment policy with increasing demand pattern and shortage allowance. *International Journal of Advanced Manufacturing Technology* 41, 1199-1207.
- Benkherouft, L. (1995) On an inventory model with deteriorating items and decreasing time-varying demand and shortage. *European Journal of Operational Research* 86 (1995):293-299.
- Benkherouft, L. (1998) Note on a deterministic lot-size inventory model for deteriorating items with shortages and a declining market. *Computers Ops Res* 25(1): 63-65.
- Chang, B.C.H., Ratnaweera, A., dan Halgamuge, S.K., (2004) *Particle swarm optimisation for protein motif discovery*, *Genetic Programming and Evolvable Machines* 5, 203-214.
- Clerc, M. (2006). *Particle Swarm Optimization*. London: ISTE.
- Clerc, M., (2004) *Discrete particle swarm optimization illustrated by travelling salesman problem*, In: *New Optimization Techniques in Engineering*, Springer, Berlin.
- Goyal S.K., and Giri B.C. (2003) A simple rule for determining replenishment intervals of an inventory item with linear decreasing demand rate. *Int. J. Production Economics* 83 (2003): 139-142
- Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, Volume 4, 1942–1948.

- Teng, J.T., Chern, M.S. and Yang, H. L. (1997) 'An optimal recursive method for various inventory replenishment models with increasing demand and shortages', *Naval Research Logistics*, Vol. 44, No. 8, pp.791–806.
- Wee, H.M. (1995) A deterministic lo-size inventory model for deteriorating items with shortages and a declining market. *Computers Ops Res* 22(3): 345-356.
- Yang, J., Zhao, G. Q., & Rand, G. K. (2004) An eclectic approach for replenishment with non-linear decreasing demand. *Int. J. Production Economics* 92(2004): 125-131
- Zhao, G. Q., Yang, J., & Rand, G. K. (2001) Heuristics for replenishment with linear decreasing demand. *Int. J. Production Economics* 69 (2001): 339-345
- Zhou, Y.W., Lau, H.S. and Yang, S.L. (2004) 'A finite horizon lot-sizing problem with time-varying deterministic demand and waiting-time-dependent partial backlogging', *International Journal of Production Economics*, Vol. 91, No. 2, pp.109–119.

APPENDICES



Appendix 1. C# Program

```
/*
 * Created by SharpDevelop.
 * User: Jin Ai
 * Date: 3/22/2013
 * Time: 4:24 AM
 */
using System;
using System.IO;
using ETLib_GLNPSO;

namespace PSO_IFS
{
    class spPSO : PSO
    {
        // this part is the problem specific code
        public spPSO(int nPar, int nIter, int nNB, double dwmax, double dwmin, double dcp, double dcg, double
dcl, double dcn, int nN):
            base(nIter, nNB, dwmax, dwmin, dcp, dcg, dcl, dcn)
        {
            base.SetDimension(nPar, nN-1); //set the swarm size and dimension of each particle
        }

        public override void DisplayResult(TextWriter t)
        { //updated on 22 March
            t.WriteLine("");
            t.WriteLine("Result:");
            t.WriteLine("-----");
            //write the position of the best particle
            for(int i=0; i<this.sSwarm.pParticle[this.sSwarm.posBest].Dimension; i++)
                t.WriteLine("x({0}) = {1}", i, this.sSwarm.pParticle[this.sSwarm.posBest].BestP[i]);
            t.WriteLine("f(x) = {0}", this.sSwarm.pParticle[this.sSwarm.posBest].ObjectiveP);
        }
    }
}
```

```

public override double Objective(Particle P)
{
    //updated on 22 March
    //problem specific information
    double H = 4;
    double A = 500;
    double alpha = 0.98;
    double c1 = 250;
    double c2 = 20;
    double c3 = 120;

    double []X = new double[P.Dimension];
    double []t = new double[P.Dimension + 1];
    double []s = new double[P.Dimension + 1];

    double obj = 0;
    Array.Copy(P.Position, X, P.Dimension);
    Array.Sort(X);

    t[0] = 0;
    for (int i = 0; i < P.Dimension; i++)
        t[i+1] = X[i]*H;

    for (int i = 0; i < P.Dimension; i++)
        s[i] = (c2*t[i] + c3*t[i+1])/(c2 + c3);

    s[P.Dimension] = H;

    //evaluate the objective value
    obj = c1 * (P.Dimension + 1);
    for (int i = 0; i < P.Dimension + 1; i++)
    {
        obj += c2 * ((s[i] - t[i]) * A / alpha * (1 - Math.Exp(-alpha*s[i])));
        obj += c2 * (A / alpha / alpha * (Math.Exp(-alpha*t[i]) - Math.Exp(-alpha*s[i])));
        obj += c2 * (A / alpha * (t[i] - s[i]));
    }
    for (int i = 0; i < P.Dimension; i++)
    {
        obj += c3 * ((s[i] - t[i+1]) * A / alpha * (1 - Math.Exp(-alpha*s[i])));
    }
}

```

```

        obj += c3 * (A / alpha / alpha * (Math.Exp(-alpha*t[i+1]) - Math.Exp(-alpha*s[i])));
        obj += c3 * (A / alpha * (t[i+1] - s[i]));
    }

    return obj;
}

public override void InitSwarm()
{
    for(int i=0; i<sSwarm.Member; i++)
    {
        for (int j = 0; j < sSwarm.pParticle[i].Dimension; j++)
        {
            sSwarm.pParticle[i].Position[j] = rand.NextDouble(); //generate random number from 0 to 1
            sSwarm.pParticle[i].Velocity[j] = 0;
            sSwarm.pParticle[i].BestP[j] = sSwarm.pParticle[i].Position[j];
            sSwarm.pParticle[i].PosMin[j] = 0; //set the lower bound of a position
            sSwarm.pParticle[i].PosMax[j] = 1; //set the upper bound of a position
        }
        sSwarm.pParticle[i].ObjectiveP = 1.7E308; //initialize the personal best objective value
    }
    sSwarm.posBest = 0;
}

}

class Program
{
    public static void Main(string[] args)
    {
        //set the PSO's parameters
        int noPar = 30;
        int noIter = 1000;
        int noNB = 5;
        double wMax = 0.9;
        double wMin = 0.4;
        double cP = 1;
        double cG = 1;
        double cL = 1;
    }
}

```

```

double cN = 1;
string oFile = "PSOResult.xls";
int noRep = 10; //number of replications

// problem specific parameters
int n = 0;
double TC_old = 0;
double TC_new = 0;

// starting time and finish time using DateTime datatype
DateTime start, finish;

// elapsed time using TimeSpan datatype
TimeSpan elapsed;

// opening output file
TextWriter tw = new StreamWriter(oFile);
tw.WriteLine("Number of Particle \t {0} ", noPar);
tw.WriteLine("Number of Iteration \t {0} ", noIter);
tw.WriteLine("Number of Neighbor \t {0} ", noNB);
tw.WriteLine("Parameter wmax \t {0} ", wMax);
tw.WriteLine("Parameter wmin \t {0} ", wMin);
tw.WriteLine("Parameter cp \t {0} ", cP);
tw.WriteLine("Parameter cg \t {0} ", cG);
tw.WriteLine("Parameter cl \t {0} ", cL);
tw.WriteLine("Parameter cn \t {0} ", cN);
tw.WriteLine("Output File Name {0} ", oFile);
tw.WriteLine("");

for(int i=0; i<noRep; i++)
{
    tw.WriteLine("Replication {0}", i+1);

    n = 1;
    TC_new = 1.7E308;
    do
    {
        TC_old = TC_new;
    }
}

```

```

n += 1;
tw.WriteLine("n = {0}", n);
//call PSO to obtain TC(n) --> keep as TC_new
PSO myPSO = new spPSO(noPar, noIter, noNB, wMax, wMin, cP, cG, cL, cN, n);
myPSO.Run(tw, true);
TC_new = myPSO.sSwarm.pParticle[myPSO.sSwarm.posBest].ObjectiveP;
for(int j=0; j<myPSO.sSwarm.pParticle[0].Dimension; j++)
{
    tw.WriteLine("position best particle at dimension {0} is
{1}", j, myPSO.sSwarm.pParticle[myPSO.sSwarm.posBest].BestP[j]);
}
} while(TC_new < TC_old);
//the minimum TC is TC_old at n = n - 1;
n -= 1;
tw.WriteLine("n optimal = {0}", n);
tw.WriteLine("TC optimal = {0}", TC_old);
tw.WriteLine("");
tw.WriteLine("");
}
tw.Close();
Console.ReadKey();
}
}
}

```

Appendix 2. Detail Result of Computational Problem

Problem 1.

Parameter

- $f(t) = 500 \cdot e^{-0.98t}$
- $c_1 = \$250$ per order
- $c_2 = \$10$ per unit per year
- $c_3 = \$40$ per unit per year
- $H = 4$ year

Result

- $n = 6$ replenishments
- $TC = 2420.0$
- shortage starting points (s_i) and replenishment times (t_i):

i	s_i	t_i
1	0.000000	0.248616
2	0.310769	0.605979
3	0.679781	1.043771
4	1.134768	1.611264
5	1.730388	2.428165
6	2.602610	4.000000

Problem 2.

Parameter

- $f(t) = 500 \cdot e^{-0.98t}$
- $c_1 = \$250$ per order
- $c_2 = \$10$ per unit per year
- $c_3 = \$80$ per unit per year
- $H = 4$ year

Result

- $n = 6$ replenishments
- $TC = 2505.4$
- shortage starting points (s_i) and replenishment times (t_i):

i	s_i	t_i
1	0.000000	0.270175
2	0.303947	0.623965
3	0.663967	1.057270
4	1.106433	1.619000
5	1.683071	2.428668
6	2.521868	4.000000

Problem 3.

Parameter

- $f(t) = 500 \cdot e^{-0.98t}$
- $c_1 = \$250$ per order

- $c_2 = \$10$ per unit per year
- $c_3 = \$120$ per unit per year
- $H = 4$ year

Result

- $n = 6$ replenishments
- $TC = 2536.9$
- shortage starting points (s_i) and replenishment times (t_i):

i	s_i	t_i
1	0.000000	0.278259
2	0.301447	0.630743
3	0.658184	1.062409
4	1.096095	1.622036
5	1.665865	2.429053
6	2.492652	4.000000

Problem 4.

Parameter

- $f(t) = 500 \cdot e^{-0.98t}$
- $c_1 = \$250$ per order
- $c_2 = \$20$ per unit per year
- $c_3 = \$40$ per unit per year
- $H = 4$ year

Result

- $n = 7$ replenishments
- $TC = 3220.1$
- shortage starting points (s_i) and replenishment times (t_i):

i	s_i	t_i
1	0.000000	0.181634
2	0.272452	0.482651
3	0.587750	0.837336
4	0.962129	1.269627
5	1.423375	1.824794
6	2.025504	2.607626
7	2.898687	4.000000

Problem 5.

Parameter

- $f(t) = 500 \cdot e^{-0.98t}$
- $c_1 = \$250$ per order
- $c_2 = \$20$ per unit per year
- $c_3 = \$80$ per unit per year
- $H = 4$ year

Result

- $n = 8$ replenishments
- $TC = 3448.4$

- shortage starting points (s_i) and replenishment times (t_i):

i	s_i	t_i
1	0.000000	0.183752
2	0.229690	0.437422
3	0.489355	0.728398
4	0.788159	1.069888
5	1.140320	1.483847
6	1.569728	2.011235
7	2.121611	2.744615
8	2.900365	4.000000

Problem 6.

Parameter

- $f(t) = 500 \cdot e^{-0.98t}$
- $c_1 = \$250$ per order
- $c_2 = \$20$ per unit per year
- $c_3 = \$120$ per unit per year
- $H = 4$ year

Result

- $n = 8$ replenishments
- $TC = 3539.2$
- shortage starting points (s_i) and replenishment times (t_i):

i	s_i	t_i
1	0.000000	0.194682
2	0.227129	0.447017
3	0.483665	0.736422
4	0.778548	1.076034
5	1.125615	1.487728
6	1.548080	2.012402
7	2.089789	2.742904
8	2.851756	4.000000

Problem 6.

Parameter

- $f(t) = 500 \cdot e^{-0.98t}$
- $c_1 = \$250$ per order
- $c_2 = \$20$ per unit per year
- $c_3 = \$120$ per unit per year
- $H = 4$ year

Result

- $n = 8$ replenishments

- $TC = 3539.2$

- shortage starting points (s_i) and replenishment times (t_i):

i	s_i	t_i
1	0.000000	0.194682
2	0.227129	0.447017
3	0.483665	0.736422
4	0.778548	1.076034
5	1.125615	1.487728
6	1.548080	2.012402
7	2.089789	2.742904
8	2.851756	4.000000

Problem 7.

Parameter

- $f(t) = 500 \cdot e^{-0.98t}$
- $c_1 = \$250$ per order
- $c_2 = \$40$ per unit per year
- $c_3 = \$40$ per unit per year
- $H = 4$ year

Result

- $n = 9$ replenishments
- $TC = 4031.7$
- shortage starting points (s_i) and replenishment times (t_i):

i	s_i	t_i
1	0.000000	0.143440
2	0.215160	0.375525
3	0.455707	0.637531
4	0.728444	0.938371
5	1.043334	1.291669
6	1.415836	1.719849
7	1.871856	2.263955
8	2.460005	3.013049
9	3.289571	4.000000

Problem 8.

Parameter

- $f(t) = 500 \cdot e^{-0.98t}$
- $c_1 = \$250$ per order
- $c_2 = \$40$ per unit per year
- $c_3 = \$80$ per unit per year

- $H = 4$ year

Result

- $n = 10$ replenishments
- $TC = 4543.8$

- shortage starting points (s_i) and replenishment times (t_i):

i	s_i	t_i
1	0.000000	0.124147
2	0.186220	0.322991
3	0.391376	0.543650
4	0.619788	0.791568
5	0.877459	1.074540
6	1.173080	1.404320
7	1.519940	1.799915
8	1.939902	2.295272
9	2.472957	2.961446
10	3.205691	4.000000

Problem 9.

Parameter

- $f(t) = 500 \cdot e^{-0.98t}$
- $c_1 = \$250$ per order
- $c_2 = \$40$ per unit per year
- $c_3 = \$120$ per unit per year
- $H = 4$ year

Result

- $n = 10$ replenishments
- $TC = 4778.1$
- shortage starting points (s_i) and replenishment times (t_i):

i	s_i	t_i
1	0.000000	0.137721
2	0.183628	0.335216
3	0.385745	0.554341
4	0.610540	0.800498
5	0.863817	1.081432
6	1.153970	1.408845
7	1.493803	1.801701
8	1.904334	2.294028
9	2.423926	2.957621
10	3.135520	4.000000