Optimal inventory replenishment and pricing for a single-manufacturer and multi-retailer system of deteriorating items

Zhixiang Chen

Department of Management Science,
School of Business,
Sun Yat-Sen University
135, West Xingang Road,
Guangzhou 510275, China
Fax: +86-20-84036924
Email: mnsycz@mail.sysu.edu.cn

Abstract: This paper studies an integrated optimisation model of inventory replenishment and pricing decisions in a single-manufacturer multi-retailer system of deteriorating items. Two meta-heuristic algorithms, i.e., improved particle swarm optimisation incorporated with SA mechanism (SAPSO), and a continuous domain ant colony optimisation algorithm (CDACO) are developed. Experiments show that two algorithms are effective for solving the model; however, SAPSO has higher effectiveness, efficiency and accuracy than CDACO. Sensitivity analysis shows that price elasticity coefficient significantly impacts decisions of production and retailing, giving managerial implication that reducing price fluctuation can increase profit. However, the effect of deterioration of items is not significant both for cost and profit.

Keywords: production-marketing coordination; meta-heuristics; deteriorating items; particle swarm optimisation; ant colony optimisation; ACO.


Biographical notes: Zhixiang Chen is a Professor and Area Coordinator of the Department of Management Science in Business School of Sun Yat-sen University, China. He obtained his PhD in Management Science and Industrial Engineering from Huazhong University of Science and Technology, China. His research interests include production planning and control; inventory and supply chain optimisation, and JIT/lean production system. He has published papers in International Journal of Production Research (IJPR), Journal of Operational Research Society (JORS), International Journal of Productivity and Quality Management (IJPQM), International Journal of Operations & Production Management (IJOPM), International Journal of Manufacturing Technology and Management (IJMTM), International Journal of Industrial and System Engineering (IJISE), etc.
1 Introduction

Nowadays, customer centricity strategy is playing more and more important role in market competition (Walters, 2014). This strategy requires firms to have the ability of effectively integrating and coordinating the marketing and production operations decisions. In the last decade, the concept of integration has opened a host of research ideas and created a lot of attentions in industry and academic research community. Through integration, partners of supply chains can obtain synchrohal operations and quickly respond to customer demand. In this paper, we discuss an integrated optimisation problem of pricing, inventory replenishment decisions of deteriorating items for a single-vendor multi-buyer diverging system comprising of a single manufacturer and multiple retailers, and design meta-heuristic algorithms to efficiently solve the problem.

In literature, some authors have discussed the integrated optimisation issue of single manufacturer and multiple retailers (distributors) for deterministic demand and non-deteriorating items. For example, Siajadi et al. (2006) presented a new methodology to obtain joint economic lot size in the single-vendor multi-buyer inventory system. Chan and Kingsman (2007) also studied one single-vendor multi-buyer supply chain by synchronising delivery and production cycle. Darwish and Odah (2010) studied a vendor managed inventory model for single-vendor multi-retailer supply chains, an efficient algorithm was developed to alleviate the complexity of the model and render the mathematics tractable. Jha and Shanker (2013) studied a single-vendor multi-buyer integrated production-inventory model with controllable lead time and service level constraints. A Lagrangian multiplier technique-based algorithm approach was proposed. Sue-Ann et al. (2012) studied a single-vendor multi-buyer supply chain under vendor-managed inventory (VMI) mode of operations, and they used PSO and GA-AIS to solve the model.

From these studies, we know that optimal integrated decision can render benefits to the whole supply chain partners (vendors and buyers). However, all these integrated optimisation models are concerned about non-deteriorating items and no pricing decision is considered.

In reality, there are many items that deteriorate over time, such as fresh food (e.g., milk products, vegetables, meat or fish items), cosmetic and paper. Deterioration is a natural phenomenon for many products, such as volatile liquids, agricultural productions, radioactive substances, films, drugs, blood, fashion goods, electronic components and etc. These items are subject to depletion by some natural phenomena rather than demand, for example, spoilage, shrinkage, decay and obsolescence, etc. (Chen and Chang, 2010). Deterioration can take place in many forms, such as chemical changes, physical changes, and biological changes. Since deteriorating items are usually fresh and are usually short-life-cycle products, their demand is price-sensitive. In this situation, when making any decisions for inventory, it is necessary to consider the impact of pricing on demand. This leads to the research interests in joint decision of pricing and inventory in academia. Motivated by this reality background, the objective of this paper is to establish a joint decision model for retail pricing and production-inventory optimisation for deteriorating items for a single manufacturer and multiple retailers, and develop meta-heuristic algorithms to solve the model, investigate the impact of market and production factors on system performance.
The decisions of production and retailing for these kinds of deteriorating items, for example, inventory and delivery are different from the other non-deteriorating items. Although there are some authors who studied the integrated optimisation models of single-vendor single-buyer system for deteriorating items (Yan et al., 2011), or models of single-vendor multi-buyer system for deteriorating items (Yang and Wee, 2002; Wu and Sarker, 2013), none of them is concerned with pricing decision.

Recently, integrated model of pricing and inventory decision has also attracted academic attention. Most of the literature are concerned about pricing and inventory replenishment decision for non-deteriorating demand in single stage system. Also, there are some literature are concerned about pricing and inventory replenishment for deteriorating item in single stage inventory system, such as Saha and Basu (2010), Jaggi et al. (2010), Madhavi et al. (2011), Guria et al. (2012), Maihami and Kamalabadi (2012), Maihami and Karimi (2014), Mahata et al. (2014), Chang et al. (2015). None of these studies are the integration of supplier and buyer; all of them are single echelon inventory optimisation of deteriorating items.

Some authors have studied the single vendor single buyer inventory of deteriorating items, for example, Lee and Kim (2014) studied an optimal policy for a single-vendor single-buyer integrated production-distribution model with both deteriorating and defective items. Shah et al. (2014) studied the determination of optimal pricing, shipment and payment policies for an integrated supplier-buyer deteriorating inventory model in buoyant market with two-level trade credit. However, little attention has been paid to the decision of pricing and delivery of single-vendor and multi-buyer (diverging) system for deteriorating items. As our best knowledge, there is no research on the problem of single vendor multiple buyer inventory of deteriorating items considering pricing and shortage. Based on this background, we are motivated to study a single vendor multiple buyer system of deteriorating items, considering the pricing and shortage constraint. Since the model proposed in this paper is complex nonlinear programming model, traditional closed form solution procedure by differentiation method can not be effectively applied to solve the model, in this paper, we try to develop meta-heuristics method to solve the model, this will rich the literature body of this area.

The main contribution of this paper lies in four aspects. First, this study reveals the possibility and the value of joint pricing and inventory replenishment for deteriorating item in single-vendor multi-buyer (diverging) system. Second, we extend present joint replenishment model to a model with consideration of pricing and delivery policy for multiple retailers, embodying the practice requirement of synchronous supply chains. Third, we investigate the application of meta-heuristic algorithms in solving complicated nonlinear programming problems. Fourth, some characteristics and managerial implications for practice are summarised.

The rest of this paper is organised as follows. Section 2 formulates the problem of single-manufacturer and multi-retailer inventory with deteriorating items. Section 3 describes the two algorithms of SAPSO and continuous domain ant colony optimisation (CDACO). Section 4 uses numerical examples to illustrate the application of the two algorithms in solving the model, and conducts sensitivity analysis for some parameters to highlight the general characteristics of the system. Section 5 summarises the major conclusions and future research directions.
2 Formulation of the problem

The system of the proposed integrated inventory model is shown in Figure 1, in which, one manufacturer produces products and delivers them to different retailers’ warehouses. The products deteriorate in production, transportation and warehousing.

![Figure 1: Supply chain system of single-manufacturer multi-retailer](see online version for colours)

In order to formulate the integrated system as perceived in Figure 1, we first give out the assumptions and describe the notations used in the model.

2.1 Assumptions and notations

The following assumptions are held throughout the paper:
1. the supply chain comprises of one manufacturer and multiple retailers
2. demands at different retailers are functions of price
3. all products deteriorate over time
4. shortage is allowed, and all shortages are completely backlogged
5. all retailers’ replenishment cycles are equal to the manufacturer’s production cycle.

The notations used in the paper are listed as follows.

Indices:
- $i$ number of cycle, $i = 1, 2, \ldots, n$
- $j$ number of retailer, $j = 1, 2, \ldots, m$

Parameters:
- $H_v$ item holding cost of the manufacturer ($v$ in here denotes the vendor, i.e., manufacturer), ($$/unit/month)$
- $H_j$ item holding cost of retailer $j$ ($$/unit/month)$
$\text{Variables}$

$p_j$ price of the product at retailer $j$ ($/unit$)

$f_j$ fraction of shortage time to cycle time at retailer $j$

$T$ common replenishment and production cycle time for manufacturer and retailers.
2.2 Model formulation

The inventory levels of manufacturer and retailers are illustrated in Figure 2. Based on the assumptions, the problem is formulated as followings.

Figure 2  Inventory levels of manufacturer and retailers

2.2.1 The income of the system

The total income of the system depends on the price and demand. As assumed, the demand of retailers is the function of price, and usually, in literature, the demand is characterised by a linear or exponential function. For convenience, in this paper we apply linear demand function, i.e.,

\[ D(p_j) = a_j - b_j p_j, \]

where \( a_j \) is an initial demand for retailer \( j \) without price fluctuation (i.e., market scale factor of retailer \( j \)), and \( b_j \) is the price elasticity of retailer \( j \).
This kind of linear demand function has following assumptions:

1. \( D(p_j) > 0 \) is continuous for \( p_j > 0 \)

2. \( \frac{dD(p_j)}{dp_j} < 0 \), i.e., \( D(p_j) \) is a non-increasing function for all \( p_j \in (0, \infty) \)

3. the marginal revenue \( \frac{d\{p_jD(p_j)\}}{dp_j} = p_j + D(p_j) \frac{dD}{dp} \) is a strictly increasing function of \( p_j \) and thus \( \frac{1}{D(p_j)} \) is a convex function of \( p_j \).

Based on these assumptions, the total income function for all retailers is

\[
TI = TI = \sum_{j=1}^{m} (a_j - b_j p_j) p_j ,
\]

where \( p_j < a_j / b_j \).

### 2.2.2 Total cost of the system

#### 2.2.2.1 Total cost of retailers

The total cost of retailer \( j \) consists of the cost of inventory holding cost, deterioration cost, shortage cost and ordering cost.

The inventory of retailer \( j \) can be written as

\[
\frac{dI^+_{j}(t)}{dt} + \theta_b I^+_{j}(t) = -(a_j - b_j p_j), \quad (t_{i-1,j} \leq t \leq t_i, i = 1, 2, \ldots, n)
\]

and the shortage of retailer \( j \) is

\[
\frac{dI^-_{j}(t)}{dt} = -(a_j - b_j p_j), \quad (t_i \leq t \leq t_j, i = 1, 2, \ldots, n)
\]

where \( I^+_{j}(t) \) is the inventory level of retailer \( j \) at time \( t \), and \( I^-_{j}(t) \) is the shortage level of retailer \( j \) at time \( t \). Through integral computation of the above two equations, we can obtain the inventory and shortage of retailer \( j \) in period \( t \).

\[
I^+_{j}(t) = e^{\theta_b t} \int_{t_{i-1,j}}^{t_i} e^{-\theta_b u} \left( a_j - b_j p_j \right) du
\]

(4)

\[
I^-_{j}(t) = (a_j - b_j p_j) (t_i - t), \quad (t_i \leq t \leq t_j, i = 1, 2, \ldots, n)
\]

(5)

From equation (4) and (5) with the assumption that all retailers have the same delivery intervals, i.e., \( t_j - t_{i-1,j} = T \) (the common replenishment and production cycle time), we can obtain the total inventory and shortage of retailer \( j \).

\[
I^+_{j} = \int_{t_{i-1,j}}^{t_i} I^+_{j}(t) dt = \left( \frac{a_j - b_j p_j}{\theta_b} \right) \left[ e^{\theta_b T} (1-f_j) - 1 \right] - \left( \frac{a_j - b_j p_j}{\theta_b} \right) T (1 - f_j)
\]

(6)

\[
I^-_{j} = \int_{t_{i-1,j}}^{t_i} I^-_{j}(t) dt = -\left( \frac{a_j - b_j p_j}{2} \right) T^2 f_j^2.
\]

(7)
The inventory and shortage cost of retailer $j$ can then be calculated from equations (8) and (9).

$$TC_j = T^2 f_j^2$$

During each period of replenishment, each retailer will incur cost of losing deteriorated items. Assume the lost quantity of retailer $j$ in each cycle is $Q_j$. Therefore, the total cost for losing at retailer $j$ is expressed as

$$TC_j^f = C^f Q_j = C^f \left( I^j (t_{i,j}) \right) - \int_{t_{i,j}}^{t_j} (a_j - b_j p_j) du$$

$$= C^f \left[ e^{\theta t_j} - 1 \right] - (a_j - b_j p_j) T (1 - f_j)$$

So the total cost for $m$ retailers in each replenishment period is obtained as

$$TC^b = \frac{1}{T} \left[ \sum_{j=1}^{m} A_j + \sum_{j=1}^{m} H_j \left( \frac{a_j - b_j p_j}{\theta^2 b} \right) \left[ e^{\theta t_j} - 1 \right] - \left( \frac{a_j - b_j p_j}{\theta b} \right) T (1 - f_j) \right] \right] + \sum_{j=1}^{m} \pi_j \left( \frac{a_j - b_j p_j}{2} \right) T^2 f_j^2 + \sum_{j=1}^{m} \left[ \left( \frac{a_j - b_j p_j}{\theta b} \right) \left[ e^{\theta t_j} - 1 \right] \right]$$

### 2.2.2.2 Total cost of manufacturer

The instantaneous inventory level of manufacturer, $I^v(t)$, under production rate $P$ and deterioration rate $\theta$, can be described as follows.

$$\frac{dI^v(t)}{dt} + \theta I^v(t) = P \left( t_{i,j} \leq t \leq t_{i,j} \right), \quad j = 1, 2, \ldots, m$$

with the boundary conditions $I^v(t_{i,j-1}) = 0$.

From equation (12), we obtain

$$I^v(t) = e^{-\theta t} \left( \int P e^{\theta u} du + C \right) = \frac{P}{\theta} + C e^{\theta t}.$$
After some algebraic manipulation, (14) can be rearranged as

\[
e^{\theta_b (v_{i,j-1} - v_{i,j})} = 1 - \frac{\theta_e (a_j - b_j p_j)}{\theta_b (1 - \theta_e)} \left[ e^{\theta_b (v_{i,j-1} - v_{i,j})} - 1 \right] + \frac{\theta_e (a_j - b_j p_j)}{\theta_b (1 - \theta_e)} p \left[ e^{\theta_b (v_{i,j-1} - v_{i,j})} - 1 \right]
\]

(15)

Because the production time of retailer \( j \) is \( T_j = t_j - t_{j-1} \), using equation (15), one can obtain:

\[
T_j = -\frac{1}{\theta_e} \ln \left( 1 - \frac{\theta_e (a_j - b_j p_j)}{\theta_b (1 - \theta_e)} \left[ e^{\theta_b (v_{i,j-1} - v_{i,j})} - 1 \right] + \frac{\theta_e (a_j - b_j p_j)}{\theta_b (1 - \theta_e)} p \left[ e^{\theta_b (v_{i,j-1} - v_{i,j})} - 1 \right] \right)
\]

(16)

The total production time of manufacturer in cycle \( i \), is

\[ T_i = T = \sum_{j=1}^{n} T_j \]

(17)

The amount of inventory held by manufacturer for retailer \( j \) in cycle \( i \) can be obtained from equation (13) as

\[
I_j^{'} = \frac{P}{\theta_v} \left( T_j + \frac{1}{\theta_v} (e^{-\theta_v T_j} - 1) \right)
\]

(18)

The inventory of manufacturer in cycle \( i \) is given by Wu and Sarker (2013) as

\[
I_v = \sum_{j=1}^{n} \frac{P}{\theta_v} \left( T_j + \frac{1}{\theta_v} (e^{-\theta_v T_j} - 1) \right)
\]

(19)

The amount number of deteriorated items of manufacturer in cycle \( i \) can be obtained from the total production quantity minus delivery quantity to all retailers, i.e.,

\[
W_v^{'} = TP - \sum_{j=1}^{n} I_j^{'} (t_j) = -\frac{P}{\theta_v} \sum_{j=1}^{n} \ln \left[ 1 - \frac{\theta_e (a_j - b_j p_j)}{\theta_b (1 - \theta_e)} \left[ e^{\theta_b (v_{i,j-1} - v_{i,j})} - 1 \right] + \frac{\theta_e (a_j - b_j p_j)}{\theta_b (1 - \theta_e)} p \left[ e^{\theta_b (v_{i,j-1} - v_{i,j})} - 1 \right] \right]
\]

\[
-\frac{P}{\theta_v} \sum_{j=1}^{n} (1 - e^{-\theta_v t_j})
\]

(20)

The total cost of manufacturer can be expressed as

\[
TC_v = \frac{1}{T} \left[ \frac{A_v}{\theta_v} + H_v \sum_{j=1}^{n} \left( \frac{P}{\theta_v} T_j + \frac{P}{\theta_v} (e^{-\theta_v T_j} - 1) \right) \right] - \frac{c_d P}{\theta_v} \sum_{j=1}^{n} \ln \left[ 1 - \frac{\theta_e (a_j - b_j p_j)}{\theta_b (1 - \theta_e)} \left[ e^{\theta_b (v_{i,j-1} - v_{i,j})} - 1 \right] + \frac{\theta_e (a_j - b_j p_j)}{\theta_b (1 - \theta_e)} p \left[ e^{\theta_b (v_{i,j-1} - v_{i,j})} - 1 \right] \right]
\]

\[
-\frac{c_d P}{\theta_v} \sum_{j=1}^{n} (1 - e^{-\theta_v t_j})
\]

(21)
2.2.2.3 Total cost of transportation

Transportation cost plays an important role in supply chain systems, and Swenseth and Godfrey (2002) studied the problems of inventory optimisation considering transportation cost. Unlike their model, in this paper, we assume each retailer carries out an independent transportation system to transport items from manufacturer to its warehouse. Thereby, the transportation cost depends on the transportation numbers and per unit transportation cost, from equation (19), we obtain:

\[
TC^T = \frac{1}{T} \sum_{j=1}^{m} \frac{P}{\theta_v} \left[ T_j + \frac{1}{\theta_v} \left( e^{-\theta_v T_j} - 1 \right) \right] F_0 / C_p,
\]

where \( F_0 \) is fixed transportation cost per shipment, \( C_p \) is capacity of the truck.

2.2.2.4 Constrained optimisation problem for profit maximisation

Combing the functional values from equations (1), (11) (21), and (22), the optimisation model is expressed as

\[
Max TP(p_j, f_j, T) = T \left( (TC^c + TC^b + TC^T) = \sum_{j=1}^{m} (a_j - b_j p_j) p_j \right)
\]

\[
\begin{align*}
A_c + H_c \sum_{j=1}^{m} \left[ \frac{P}{\theta_v} T_j + \frac{P}{\theta_v} \left( e^{-\theta_v T_j} - 1 \right) \right] \\
- \frac{1}{T} \sum_{j=1}^{m} \theta_v \left( \frac{a_j - b_j p_j}{\theta_v} \right) \left[ e^{\theta_v T (1 - f_j)} - 1 \right] \\
- \frac{1}{T} \sum_{j=1}^{m} \left( \frac{a_j - b_j p_j}{\theta_b} \right) T (1 - f_j) \\
+ \sum_{j=1}^{m} c_j \left( \frac{a_j - b_j p_j}{\theta_b} \right) \left( e^{\theta_b T (1 - f_j)} - 1 \right) \\
- \frac{1}{T} \sum_{j=1}^{m} \frac{P}{\theta_v} \left[ T_j + \frac{1}{\theta_v} \left( e^{-\theta_v T_j} - 1 \right) \right] F_0 / C_p
\end{align*}
\]
Subject to
\[
p_j \leq \frac{a_j}{b_j}, \quad j = 1, 2, \ldots, m \tag{23a}
\]
\[
1 > T > 0 \tag{23b}
\]
\[
1 > f_j > 0, \quad j = 1, 2, \ldots, m. \tag{23c}
\]
This is obviously a complex nonlinear function of \(p_j, f_j\) and \(T\), the closed form solutions for which are seemingly impossible, therefore, in next section we propose two meta-heuristics to solve the model.

3 Algorithms and solution methods

Since the problem TP in (23) is a complicated function and it is difficult to show its convexity or concavity, therefore, the traditional differential method cannot be used. As a result, we try to adopt meta-heuristic algorithms to solve this model. Firstly, an improved PSO algorithm, SAPSO is developed. Then, another improved ant colony optimisation (CDACO) for continuous domain is also developed to solve the model.

3.1 Principle of basic PSO algorithm

PSO is a population-based stochastic search technique developed by Kennedy and Eberhart (1995), inspired by social behaviour of bird flocking, fish schooling and swarm theory. The basic principle of standard PSO can be described as follows.

Suppose there is a D-dimensional space problem, and a solution to the problem is
\[
X = (x_1, x_2, \cdots, x_j, \cdots, x_D).
\]
PSO uses the position of each particle to represent one solution, and the \(i^{th}\) particle of the swarm is represented by
\[
X_i = (x_{i1}, x_{i2}, \cdots, x_{ij}, \cdots, x_{iD}).
\]
The flying of a particle from one position to another position represents a search step and iteration of solution updating. The position vector of the \(i^{th}\) particle at the \(k^{th}\) step search is denoted by
\[
X_i(k) = (x_{i1}(k), x_{i2}(k), \cdots, x_{ij}(k), \cdots, x_{iD}(k)),
\]
the velocity vector of the \(i^{th}\) particle at the \(k^{th}\) step search is
\[
V_i(k) = (v_{i1}(k), v_{i2}(k), \cdots, v_{ij}(k), \cdots, v_{iD}(k)),
\]
where \(i = 1, 2, \cdots, M\), denotes the number of particles, \(j = 1, 2, \cdots, D\), denotes the dimensions of the problem. During search, each particle is attracted towards the local best position (the fitness of local optimal solution) achieved by itself and the global best position (the fitness of global optimal solution) found by the whole population. After
completing one step of the search process, i.e., an iteration of the algorithm, each particle updates its position and travelling velocity. The core of this algorithm is the updating formulas of particle position and travelling velocity. The position and moving of a particle is shown in Figure 3.

**Figure 3**  Position moving of particle of basic PSO algorithm

In Figure 3, $X_i(k)$ is the position of particle $i$ at search step $k$, $X_i(k+1)$ is the new position of particle $i$ at search step $(k + 1)$, $pbest_i(k)$ is the best-so-far position of the individual particle $i$ at the search step $k$, $gbest(k)$ is the global best position of the whole population. $V_i(k)$ is the velocity of particle $i$ at search step $k$. The velocity vector updating and position vector updating of particle $i$ can be represented following the two formulas:

$$v_{ij}(k + 1) = w v_{ij}(k) + c_1 \phi_1 \left[ pbest_{ij}(k) - x_{ij}(k) \right] + c_2 \phi_2 \left[ gbest_j(k) - x_{ij}(k) \right]$$

(24)

$$x_{ij}(k + 1) = x_{ij}(k) + v_{ij}(k + 1),$$

(25)

where the first parameter, $w$ is called inertia weight which shows the effect of previous velocity vector on the new velocity vector, $c_1$ and $c_2$ are two positive acceleration constants used to the contribution of the cognitive and social components, respectively (Engelbrecht, 2005); these constants are also called cognitive learning factor and social learning factor, respectively. $\phi_1$ and $\phi_2$ are two uniformly distributed random numbers on the range $(0,1)$. In the velocity update formula (24), the inertia weight has important impact on the convergence and search ability of the algorithm.

### 3.2 Improved algorithm of particle swarm optimisation

We improve the algorithm basic PSO algorithm in two aspects; first, we adopt the dynamic inertia weight updating strategy to update the inertia weight. Second we incorporate simulated annealing strategy in local search to increase the convergence ability.

#### 3.2.1 Dynamic inertia weight strategy

In order to reflect the impact of solution history to the new solution, by giving a weight factor to the latest solutions to improve the performance of the algorithm, we adopt dynamic inertia weight strategy. The higher inertia weight will benefit the global search; on the contrary, the lower inertia weight will benefit the local search and speed up the convergence. In this paper, we use a linear inertia weight updating strategy, which is expressed as:
\[
\begin{align*}
    w &= w_{\text{max}} - (k - 1) \times (w_{\text{max}} - w_{\text{min}})/(k_{\text{max}} - 1) \\
\end{align*}
\]

where \( w_{\text{max}} \) and \( w_{\text{min}} \) are the maximum and minimum inertia weights, respectively (in this paper, they are set as 0.9 and 0.4 respectively), and \( k_{\text{max}} \) is the maximum iteration number, i.e., \( k = 1, 2, \ldots, k_{\text{max}} \). The basic idea of equation (26) is that as the iteration increases, the inertia weight decreases from maximum \( w_{\text{max}} \) to minimum \( w_{\text{min}} \).

### 3.2.2 Local search strategy incorporating SA mechanism

The disadvantage of basic PSO is its ability of local search. In large size problem, PSO can not guarantee to reach global optimal solution but easily reaches a local optimal point. In order to improve the local search ability of basic PSO, we propose a local search strategy incorporating SA (Simulated Annealing) mechanism.

SA searches new solution in the state space according to a probability, this equals the annealing process from high energy state to low energy state. The transition rule is that from high energy state to low energy state takes probability is 1, and from low energy state to high energy state takes probability as (Kirkpatrick et al., 1983):

\[
    p = \exp\left(-\frac{\Delta E}{K T}\right)
\]

where \( T \) is temperature parameter, \( K \) is Boltzmann’s constant, all are control parameters. \( \Delta E \) is the energy changes from current point to the next point, it equals to the function change of the problem, i.e., \( \Delta E = \phi_1 - \phi_0 \), where, \( \phi_1 \) and \( \phi_0 \) are the objective function of state 1 and 0. for minimisation problem, when \( \Delta E = \phi_1 - \phi_0 \leq 0 \), \( p = 1 \), otherwise,

\[
    p = \exp\left(-\frac{\Delta E}{K T}\right).
\]

In this paper, we set \( k = 1 \).

In this paper, we adopt a strategy called randomly walking to realise local search. This strategy generates \( n \) dimensions independent standard normal variables, \( Y_1, Y_2, \ldots, Y_n \) and computes the components of \( U \), \( U_i = Y_i / (Y_i^2 + Y_{i+1}^2 + \cdots + Y_n^2)^{1/2} \), \( (i = 1, 2, \ldots, n, \) is the dimension of variable), and then based on the components, the updating process of solution used in this paper is:

\[
    X(t+1) = X(t) + \Delta \cdot U^* \left(1 - T_{\text{min}} / T(t)\right)
\]

where \( \Delta \) is step length of search direction change (it is a small value, we adopt as 0.5 in this research), \( T(t) \) is the current temperature in temperature \( t \). \( T_{\text{min}} \) is the designed lowest temperature.

The algorithm steps of this improved PSO algorithm (SAPSO) can be described as following.
Algorithm 1  SAPSO procedure

Step 1: Initialise the parameters: number of particle, M; maximum iteration number $k_{\text{max}}$; learning factors $c_1$ and $c_2$; inertia weights $w_{\text{max}}$ and $w_{\text{min}}$; variable number of the models, $D$ and the maximum velocity of particle, $v_{\text{max}}$.

Step 2: Generate initial solution (initial position and velocity) for each particle.
Set $k = 1$ (initial iteration number), randomly generate the initial position of each particle, $X_i(k)$, and initial velocity $V_i(k)$, and calculate the initial fitness of each particle: $F(X_i(k))$.

(a) Find the best individual position for particle $i$, $p_{\text{best}}(k = 1) = X_i(1)$ with $F(p_{\text{best}}(k = 1) = F(X_i(1))$.

(b) Find the best position for all particles:
$g_{\text{best}}(k = 1) = \arg\max_{1 \leq i \leq M} (F(p_{\text{best}}(k = 1)))$ for maximisation problem.

While ($k \leq k_{\text{max}}$)

Step 3: Update inertia weight, particle position and velocity:

\[
\begin{align*}
\omega(k + 1) &= \omega_{\text{max}} - (k - 1) * (\omega_{\text{max}} - \omega_{\text{min}}) / (k_{\text{max}} - 1) \\
V_i(k + 1) &= V_i(k) + c_1 \phi_1 [p_{\text{best}}(k) - x_i(k)] + c_2 \phi_2 [g_{\text{best}}(k) - x_i(k)] \\
x_i(k + 1) &= x_i(k) + V_i(k + 1).
\end{align*}
\]

Step 4: Evaluate the fitness of new solution, and find the individual best solution of each particle and global best solution of the whole population, For maximisation:

If $F(X_i(k)) > F(p_{\text{best}}(k))$

$p_{\text{best}}(k) = X_i(k); g_{\text{best}}(k) = \arg\max_{1 \leq i \leq M} (F(p_{\text{best}}(k)))$

Step 5: Enter the local search using SA mechanism

$t = T_{\text{max}}$ (Initial temperature, the parameter of SA)

While $t > T_{\text{max}}$ (temperature cooling loop)

Generate new solution direction in the current solution using

\[
U_i = Y_i / (Y_1^2 + Y_2^2 + \cdots, Y_i^2)^{1/2}
\]

$X(t + 1) = X(t) + \Delta \cdot U_i$ ($1 - T_{\text{min}} / T(t)$)

If $X(t + 1) \in \Omega$

$\Delta \phi = \phi(t + 1) - \phi(t)$

$p = \exp \left( \frac{\Delta \phi}{T} \right)$

If $\Delta \phi < 0 \mid p > \text{random}$

$\phi^* = \phi(x(t + 1))$

Endif

Endif

Update temperature (annealing)

Endwhile

Update global optimal

Step 6: If $k \leq k_{\text{max}}$, $k = k + 1$: Go to Step 3.

End while (stop condition)

Step 7: Output the best solution.
3.3 CDACO algorithm

Similar to PSO, another population-based meta-heuristic algorithm is also powerful in solving complicated optimisation problem; this is the ant colony optimisation (ACO) algorithm. ACO is one kind of random search techniques simulating an ant’s behaviour of searching food in nature. ACO algorithm was proposed by Dorigo et al. (1999). The basic principle of ACO is simple. In each iteration, each ant finds a path (solution), and then updates the information of search path—pheromone trail and the search probability according an updating rule; ultimately, all ants will travel to find food along the shortest path which has most pheromone, i.e., and find out the best solution.

Basic ACO has successfully been applied in solving combinatorial optimisation problem, like travelling salesman problem (TSP), scheduling, quadratic assignment problem (QAP), etc., but fewer applications have been conducted in continuous domain optimisation problems. The model proposed in this paper is a complicated continuous domain optimisation, so we propose an improved ACO to solve continuous domain problem, named as CDACO.

Figure 4  Solution space of CDACO

Since the basic ACO was originally developed for discrete domain problem, so the basic idea of solving continuous domain problem in this paper is to transfer continuous domain problem into discrete domain problem. The continuous variables \( x_i \) (like in our model, \( T, f_i \) and \( P_i \)) are divided into \( N \) sub-intervals such that the length of each interval is \( h_j = (x_j^U - x_j^L) / N \), \( j = 1, 2, \ldots, n \) \((n \) is the variable number). This means that each variable has \( N + 1 \) possible value points. The solution is \( x_i = x_i^L + \left( x_i^U - x_i^L \right) \frac{m_i}{N} \), and \( m_i \) is the value position of \( x_i \) in its solution space. Figure 4 shows the solution space of the problem.

The basic components of CDACO are transition probability and pheromone update.
3.3.1 Solution construction and transition probability

The transition probability $P_{ij}$ in the continuous domain optimisation is different from traditional transition probability in discrete optimisation. We use the formulas of (29) as transition probability of an ant from node $i$ (i.e., possible value point of variables) to the next node of variable $j$.

$$P_{ij} = \frac{\tau_{ij}}{\sum_{i \in \Omega} \tau_{ij}}$$  \hspace{1cm} (29)

where $\tau_{ij}$ is the pheromone trail in node $i$ for variable $j$ and $\Omega$ is the set of feasible nodes of variable $j$.

3.3.2 Update of pheromone trail

The updating of pheromone trail is formulated as:

$$\tau_{ij} \leftarrow (1 - \rho) \times \tau_{ij} + \Phi / f$$  \hspace{1cm} (30)

where $\rho$ is updating coefficient of pheromone trial, $\rho \in [0,1]$, $\Phi$ is adjustable parameter according to problem type and scale, and $f$ is the objective function value during an iteration.

3.3.3 Improvement strategy of algorithm convergence

In order to improve the convergence of the algorithm, we add a adaptable control policy for evaporation coefficient, $\rho$, by dynamically changing the value of evaporation coefficient $\rho$ during the search process, i.e., if objective function (or fitness) is not improved after some steps iteration (such as $N_{nc}$ iteration steps, then, the evaporation coefficient is updated as

$$\rho(t) = \begin{cases} 
0.95 \rho(t - N_{nc}), & \text{if } 0.95(t - N_{nc}) \geq \rho_{\text{min}} \\
\rho_{\text{min}}, & \text{otherwise.}
\end{cases}$$  \hspace{1cm} (31)

where $\rho(t)$ is the evaporation coefficient at iteration step $t$, $\rho(t - N_{nc})$ is the evaporation coefficient at iteration before $N_{nc}$ steps of $t$, and $\rho_{\text{min}}$ is the minim value of evaporation coefficient. The procedure of CDACO algorithm can be described as:

Step 1: Begin: input data and parameters initialisation
   a Set parameters of ACO and bound of variables,
   b Set $h_j = 1$ and $\epsilon = 0.00001$ (accuracy or stopping condition for continuous variable $y$)

Step 2. While $\max(h_j) > \epsilon$, $h_j = (x_j^U - x_j^L) / N$
   a Randomly generate 100 initial solutions and set pheromone trial to the first 30 best solutions among these 100 solutions.
   b $NC = 1$ (iterations counter)
   c While $NC \leq NC_{\text{max}}$ (maximum iteration number)
      Put ants at the first variable
Each ant select next node in each variable according equation (29)

Record the best solution of this iteration

Entering adaptable control policy of $\rho$ using (31):

If $NC > N_{nc}$ (if the solution is not improved after $N_{nc}$ iterations)

If $f_{\text{best}}(NC) = f_{\text{best}}(NC-N_{nc})$ ($f_{\text{best}}(NC)$ is the best fitness of iteration $NC$)

$\rho(NC) = 0.95*\rho(NC-N_{nc})$

Endif

If $\rho(NC) \leq \rho_{\text{min}}$: $\rho(NC) = \rho_{\text{min}}$

endif

endif

Update pheromone trail according equation (30)

$NC = NC + 1$

Find the best solution of this iteration

End while (c)

d Narrow the feasible zone of variable $x$:

$x^f_j = x^f_j + (m_j - \Delta)h_j$ ($m_j = \arg \max \tau_{ij}$, and $\Delta$ is coefficient)

$x^c_j = x^c_j + (m_j + \Delta)h_j$

End while (Step 2)

Step 3. Stop.

4 NUMERICAL analysis and sensitivity analysis

In this section, numerical examples are designed to demonstrate the application of the model. The two algorithms were coded in MATLAB and run on the same computer. The computer running environments are: Dell Pentium 4 CPU processor: 1.99 GHz, Memory: 512MB.

**Table 1** Parameters data of example 1

<table>
<thead>
<tr>
<th>Notations</th>
<th>Values</th>
<th>Notations</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_v$ ($/\text{unit/month}$)</td>
<td>2</td>
<td>$\theta_{vb}$</td>
<td>0.015</td>
</tr>
<tr>
<td>$H_j$ ($/\text{unit/month}$)</td>
<td>(3, 3, 3)</td>
<td>$c^d_j$ ($/\text{unit}$)</td>
<td>10</td>
</tr>
<tr>
<td>$A_j$ ($/\text{order}$)</td>
<td>(500, 500, 500)</td>
<td>$c^d_j$ ($/\text{unit}$)</td>
<td>(12, 12, 12)</td>
</tr>
<tr>
<td>$A_v$ ($/\text{batch}$)</td>
<td>2,000</td>
<td>$\pi_j$ ($/\text{unit/time-unit}$)</td>
<td>(12, 12, 12)</td>
</tr>
<tr>
<td>$a_j$</td>
<td>(3,000, 2,000, 2,500)</td>
<td>$\theta_b$</td>
<td>0.03</td>
</tr>
<tr>
<td>$b_j$</td>
<td>(10, 10, 20)</td>
<td>$\theta_r$</td>
<td>0.01</td>
</tr>
<tr>
<td>$P$ (units/month)</td>
<td>1,000</td>
<td>$F_0$ ($/\text{shipment}$)</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C_p$ (units/truck)</td>
<td>200</td>
</tr>
</tbody>
</table>
4.1 Examples

4.1.1 Example 1 (single manufacturer and three retailers)

The data of example for the proposed model are listed at Table 1. Computational results by respective algorithms are summarised in the subsequent tables.

4.1.1.1 SAPSO algorithm

Parameters chosen for SAPSO are: \( m = 200, c_1 = 1.5, c_2 = 1.5, w = 0.8, v_{\text{max}} = 0.5, \) and \( k_{\text{max}} = 200. \) Inertia weight factors: \( W_{\text{max}} = 0.9, W_{\text{min}} = 0.4. \) After ten runs, each run the SAPSO can obtain the same results. The convergence curve of SAPSO is shown in Figure 5(a).

The best solution obtained by SAPSO algorithm is: objective function \( TP = $388,046.42 \) with variables’ solutions \( T = 0.4656 \) (month), \( f_1 = 0.2246, f_2 = 0.2231, f_3 = 0.2238, P_1 = $151.97, P_2 = $101.41, P_3 = $64.17. \) The CPU time recorded is 12–13 (seconds).

Figure 5 Convergence curves of SAPSO and CDACO, (a) SAPSO algorithm (b) CDACO algorithm (see online version for colours)

Note: 3 retailers

4.1.1.2 CDACO algorithm

The parameters for CDACO were chosen as: \( \Phi = 1,000,000, m = 30 \) (ant number), \( NC_{\text{max}} = 50 \) (maximum iterations number), \( N = 10 \) (intervals of variable), \( \rho = 0.9, \rho_{\text{min}} = 0.5, N_{\text{nc}} = 5, \) After ten runs of CDACO program, seven runs obtain the optimal solution (70% success rate) as SAPSO, i.e., \( TP = $388,046.42 \) with solutions \( T = 0.4656 \) (month), \( f_1 = 0.2247, f_2 = 0.2230, f_3 = 0.2238, P_1 = $151.97, P_2 = $101.41, \) and \( P_3 = $64.17. \) The CPU time recorded is 34–35 (seconds). The convergence curve of CDACO is shown in Figure 5(b) (one of ten runs).
4.1.2 Example 2 (five retailers)

Here we use a little big size problem, which has five retailers. The model parameters are listed in Table 2.

Table 2 Parameters data of example 2

<table>
<thead>
<tr>
<th>Notations</th>
<th>Values</th>
<th>Notations</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>( H_v )  ($/unit/month)</td>
<td>5</td>
<td>( \theta_{\alpha} )</td>
<td>0.015</td>
</tr>
<tr>
<td>( H_j )  ($/unit/month)</td>
<td>(7, 5, 8, 6, 6)</td>
<td>( c_{\beta} ) ($/unit)</td>
<td>20</td>
</tr>
<tr>
<td>( A_j ) ($/order)</td>
<td>(500, 400, 600, 700, 300)</td>
<td>( c_{\gamma} ) ($/unit)</td>
<td>(22, 24, 25, 24, 23)</td>
</tr>
<tr>
<td>( A_v ) ($/batch)</td>
<td>2,000</td>
<td>( \pi_j ) ($/unit/time-unit)</td>
<td>(22, 25, 21, 26, 25)</td>
</tr>
<tr>
<td>( a_j )</td>
<td>(2,500, 2,000, 2,500, 1,000, 1,500)</td>
<td>( \theta_{\beta} )</td>
<td>0.03</td>
</tr>
<tr>
<td>( b_j )</td>
<td>(10, 12.5, 12.5, 10, 10)</td>
<td>( \theta_{\gamma} )</td>
<td>0.01</td>
</tr>
<tr>
<td>( P ) (units/month)</td>
<td>2,500</td>
<td>( F_0 ) ($/shipment)</td>
<td>500</td>
</tr>
<tr>
<td>( C_p ) (units/truck)</td>
<td>200</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Five retailers

The algorithms parameters of the two algorithms are the same as in example 1. The solutions of these two algorithms for this example are (convergence curves are similar to the example 1, for saving space, we do not show here again):

a  \( TP = $423,161.61, \) with solutions \( T = 0.4667\) (month), \( f_1 = 0.2606, f_2 = 0.1882, f_3 = 0.2964, f_4 = 0.2068, f_5 = 0.2128, P_1 = $126.59, P_2 = $81.28, P_3 = $101.64, P_4 = $50.99, P_5 = $76.17 \).

b  \( TP = $423,161.61, \) with solutions \( T = 0.4667\) (month), \( f_1 = 0.2606, f_2 = 0.1895, f_3 = 0.2964, f_4 = 0.2068, f_5 = 0.2130, P_1 = $126.59, P_2 = $81.28, P_3 = $101.64, P_4 = $50.99, P_5 = $76.17 \).

From the results, we find that both SAPSO and CDACO can efficiently solve continuous domain nonlinear programming problems, and they have different advantages and disadvantages: CDACO can obtain faster convergence efficiency, while SAPSO has higher success rate, also SAPSO has higher efficiency with shorter CPU time, it has its search ability of optimality is higher than CDACO.

4.2 Sensitivity analysis

In order to analyse the impact of parameters on the decisions, we conduct a sensitivity analysis. In this paper, we analyse the following parameters:

1. price elasticity coefficient \( b_j \)
2. deteriorating rate of distributors and manufacturer, \( \theta_{\alpha}, \theta_{\gamma} \)
3. ordering cost \( A_j \).

All the following analyses are based on example 1.
Table 3
Sensitivity analysis with respect to parameter $b_j$

<table>
<thead>
<tr>
<th>$\Delta b_j$ (%)</th>
<th>$b_j$</th>
<th>$T$ (month)</th>
<th>$f_1$</th>
<th>$f_2$</th>
<th>$f_3$</th>
<th>$P_1$ ($)</th>
<th>$P_2$ ($)</th>
<th>$P_3$ ($)</th>
<th>$TP$ ($)</th>
<th>$\Delta TP$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-100%*</td>
<td>0, 0, 0</td>
<td>0.2493</td>
<td>0.2247</td>
<td>0.2229</td>
<td>0.2238</td>
<td>300.00</td>
<td>200.00</td>
<td>125.00</td>
<td>1,584,523.62</td>
<td>308.33%</td>
</tr>
<tr>
<td>-80%</td>
<td>2, 2, 4</td>
<td>0.3047</td>
<td>0.2247</td>
<td>0.2229</td>
<td>0.2238</td>
<td>300.00</td>
<td>200.00</td>
<td>125.00</td>
<td>1,267,121.78</td>
<td>226.54%</td>
</tr>
<tr>
<td>-60%</td>
<td>4, 4, 8</td>
<td>0.3923</td>
<td>0.2247</td>
<td>0.2229</td>
<td>0.2238</td>
<td>300.00</td>
<td>200.00</td>
<td>125.00</td>
<td>949,737.34</td>
<td>144.75%</td>
</tr>
<tr>
<td>-40%</td>
<td>6, 6, 12</td>
<td>0.4627</td>
<td>0.2247</td>
<td>0.2230</td>
<td>0.2238</td>
<td>251.96</td>
<td>168.07</td>
<td>105.84</td>
<td>656,750.74</td>
<td>69.25%</td>
</tr>
<tr>
<td>-20%</td>
<td>8, 8, 16</td>
<td>0.4641</td>
<td>0.2247</td>
<td>0.2230</td>
<td>0.2238</td>
<td>189.46</td>
<td>126.41</td>
<td>79.80</td>
<td>488,804.84</td>
<td>25.97%</td>
</tr>
<tr>
<td>0%**</td>
<td>10, 10, 20</td>
<td>0.4656</td>
<td>0.2247</td>
<td>0.2230</td>
<td>0.2238</td>
<td>151.97</td>
<td>101.41</td>
<td>64.17</td>
<td>388,046.42</td>
<td>0</td>
</tr>
<tr>
<td>20%</td>
<td>12, 12, 24</td>
<td>0.4670</td>
<td>0.2247</td>
<td>0.2230</td>
<td>0.2238</td>
<td>126.97</td>
<td>84.74</td>
<td>53.75</td>
<td>320,881.75</td>
<td>-17.31%</td>
</tr>
<tr>
<td>40%</td>
<td>14, 14, 28</td>
<td>0.4684</td>
<td>0.2247</td>
<td>0.2230</td>
<td>0.2238</td>
<td>109.11</td>
<td>72.84</td>
<td>46.31</td>
<td>272,913.49</td>
<td>-29.67%</td>
</tr>
<tr>
<td>80%</td>
<td>18, 18, 36</td>
<td>0.4713</td>
<td>0.2247</td>
<td>0.2230</td>
<td>0.2238</td>
<td>85.31</td>
<td>56.97</td>
<td>36.38</td>
<td>208,970.98</td>
<td>-46.15%</td>
</tr>
<tr>
<td>100%</td>
<td>20, 20, 40</td>
<td>0.4727</td>
<td>0.2247</td>
<td>0.2230</td>
<td>0.2238</td>
<td>76.97</td>
<td>51.41</td>
<td>32.91</td>
<td>186,597.93</td>
<td>-51.91%</td>
</tr>
</tbody>
</table>

Notes: Algorithm: SAPSO;
*there is no price elasticity;
**initial value
Table 4

Sensitivity analysis with respect to parameter $\theta_b$

<table>
<thead>
<tr>
<th>$\Delta \theta_b$ (%)</th>
<th>$\theta_b$</th>
<th>$T$ (month)</th>
<th>$f_1$</th>
<th>$f_2$</th>
<th>$f_3$</th>
<th>$P_1$ ($)</th>
<th>$P_2$ ($)</th>
<th>$P_3$ ($)</th>
<th>TP ($)</th>
<th>$\Delta TP$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-80%</td>
<td>0.006</td>
<td>0.4729</td>
<td>0.2051</td>
<td>0.2047</td>
<td>0.2049</td>
<td>151.96</td>
<td>101.40</td>
<td>64.16</td>
<td>388,241.73</td>
<td>0.050%</td>
</tr>
<tr>
<td>-60%</td>
<td>0.012</td>
<td>0.4710</td>
<td>0.2100</td>
<td>0.2094</td>
<td>0.2097</td>
<td>151.97</td>
<td>101.40</td>
<td>64.17</td>
<td>388,191.76</td>
<td>0.038%</td>
</tr>
<tr>
<td>-40%</td>
<td>0.018</td>
<td>0.4691</td>
<td>0.2150</td>
<td>0.2140</td>
<td>0.2145</td>
<td>151.96</td>
<td>101.41</td>
<td>64.17</td>
<td>388,142.56</td>
<td>0.025%</td>
</tr>
<tr>
<td>-20%</td>
<td>0.024</td>
<td>0.4673</td>
<td>0.2199</td>
<td>0.2185</td>
<td>0.2192</td>
<td>151.97</td>
<td>101.41</td>
<td>64.17</td>
<td>388,094.12</td>
<td>0.012%</td>
</tr>
<tr>
<td>0%*</td>
<td>0.030</td>
<td>0.4656</td>
<td>0.2247</td>
<td>0.2230</td>
<td>0.2238</td>
<td>151.97</td>
<td>101.41</td>
<td>64.17</td>
<td>388,046.42</td>
<td>0</td>
</tr>
<tr>
<td>20%</td>
<td>0.036</td>
<td>0.4638</td>
<td>0.2295</td>
<td>0.2275</td>
<td>0.2284</td>
<td>151.97</td>
<td>101.41</td>
<td>64.17</td>
<td>387,999.44</td>
<td>-0.012%</td>
</tr>
<tr>
<td>40%</td>
<td>0.042</td>
<td>0.4822</td>
<td>0.2342</td>
<td>0.2319</td>
<td>0.2330</td>
<td>151.97</td>
<td>101.41</td>
<td>64.17</td>
<td>387,953.16</td>
<td>-0.024%</td>
</tr>
<tr>
<td>60%</td>
<td>0.048</td>
<td>0.4605</td>
<td>0.2388</td>
<td>0.2362</td>
<td>0.2374</td>
<td>151.97</td>
<td>101.41</td>
<td>64.17</td>
<td>387,907.56</td>
<td>-0.036%</td>
</tr>
<tr>
<td>80%</td>
<td>0.054</td>
<td>0.4589</td>
<td>0.2434</td>
<td>0.2406</td>
<td>0.2419</td>
<td>151.97</td>
<td>101.41</td>
<td>64.17</td>
<td>387,862.83</td>
<td>-0.047%</td>
</tr>
<tr>
<td>100%</td>
<td>0.060</td>
<td>0.4574</td>
<td>0.2479</td>
<td>0.2448</td>
<td>0.2463</td>
<td>151.97</td>
<td>101.41</td>
<td>64.17</td>
<td>387,818.35</td>
<td>-0.059%</td>
</tr>
</tbody>
</table>

Notes: Algorithm: SAPSO;
*Initial value.
Table 5

Optimal inventory replenishment and pricing for a single-manufacturer

| $\Delta \theta_v (%)$ | $\theta_v$ | $T$(month) | $f_1$ | $f_2$ | $f_3$ | $P_1($) | P_2($) | P_3($) | TP($) | $\Delta TP(\%)$
|----------------------|-----------|------------|-------|-------|-------|---------|---------|---------|-------|----------------
| -80%                 | 0.002     | 0.4694     | 0.2246| 0.2230| 0.2238| 151.94  | 101.40  | 64.15   | 388,153.92| 0.028%        |
| -60%                 | 0.004     | 0.4684     | 0.2247| 0.2230| 0.2238| 151.95  | 101.40  | 64.16   | 388,127.01| 0.021%        |
| -40%                 | 0.006     | 0.4675     | 0.2247| 0.2230| 0.2238| 151.95  | 101.40  | 64.16   | 388,100.12| 0.014%        |
| -20%                 | 0.008     | 0.4665     | 0.2247| 0.2230| 0.2238| 151.96  | 101.41  | 64.16   | 388,073.26| 0.007%        |
| 0%*                  | 0.010     | 0.4656     | 0.2247| 0.2230| 0.2238| 151.97  | 101.41  | 64.17   | 388,046.42| 0.000%        |
| 20%                  | 0.012     | 0.4646     | 0.2247| 0.2230| 0.2238| 151.97  | 101.41  | 64.17   | 388,019.60| -0.007%       |
| 40%                  | 0.014     | 0.4637     | 0.2247| 0.2230| 0.2238| 151.98  | 101.41  | 64.18   | 387,992.80| -0.014%       |
| 60%                  | 0.016     | 0.4627     | 0.2247| 0.2230| 0.2238| 151.98  | 101.42  | 64.18   | 387,966.02| -0.021%       |
| 80%                  | 0.018     | 0.4618     | 0.2248| 0.2231| 0.2239| 151.99  | 101.42  | 64.18   | 387,939.27| -0.028%       |
| 100%                 | 0.020     | 0.4609     | 0.2248| 0.2231| 0.2239| 151.99  | 101.42  | 64.19   | 387,912.53| -0.035%       |

Notes: Algorithm: SAPSO;
*Initial value
Table 6

<table>
<thead>
<tr>
<th>$\Delta \theta_b$ (%)</th>
<th>$\theta_b$</th>
<th>$TC$ ($)</th>
<th>$\Delta TC$ (%)</th>
<th>$\Delta \theta_b$ (%)</th>
<th>$\theta_b$</th>
<th>$TC$ ($)</th>
<th>$\Delta TC$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-80%</td>
<td>0.006</td>
<td>14,769.69</td>
<td>-1.30%</td>
<td>20%</td>
<td>0.036</td>
<td>15,011.25</td>
<td>0.31%</td>
</tr>
<tr>
<td>-60%</td>
<td>0.012</td>
<td>14,819.48</td>
<td>-0.97%</td>
<td>40%</td>
<td>0.042</td>
<td>15,057.42</td>
<td>0.62%</td>
</tr>
<tr>
<td>-40%</td>
<td>0.018</td>
<td>14,868.53</td>
<td>-0.64%</td>
<td>60%</td>
<td>0.048</td>
<td>15,102.86</td>
<td>0.93%</td>
</tr>
<tr>
<td>-20%</td>
<td>0.024</td>
<td>14,916.86</td>
<td>-0.32%</td>
<td>80%</td>
<td>0.054</td>
<td>15,147.62</td>
<td>1.22%</td>
</tr>
<tr>
<td>0%*</td>
<td>0.030</td>
<td>14,964.43</td>
<td>0</td>
<td>100%</td>
<td>0.060</td>
<td>15,191.74</td>
<td>1.52%</td>
</tr>
</tbody>
</table>

Notes: Algorithm: SAPSO;

*Initial value
### Table 7: Sensitivity analysis with respect to parameter $\Delta A_i$

<table>
<thead>
<tr>
<th>$\Delta A_i /%$</th>
<th>$A_i$</th>
<th>$TC$</th>
<th>$\Delta TC$</th>
<th>$TP($)</th>
<th>$\Delta TP(%)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>–100%</td>
<td>0, 0, 0</td>
<td>11,340.89</td>
<td>–24.21%</td>
<td>391,718.84</td>
<td>0.95%</td>
</tr>
<tr>
<td>–80%</td>
<td>100, 100, 100</td>
<td>12,154.88</td>
<td>–18.77%</td>
<td>390,895.12</td>
<td>0.73%</td>
</tr>
<tr>
<td>–60%</td>
<td>200, 200, 200</td>
<td>12,916.41</td>
<td>–13.69%</td>
<td>390,123.84</td>
<td>0.54%</td>
</tr>
<tr>
<td>–40%</td>
<td>300, 300, 300</td>
<td>13,634.32</td>
<td>–8.89%</td>
<td>389,396.11</td>
<td>0.35%</td>
</tr>
<tr>
<td>–20%</td>
<td>400, 400, 400</td>
<td>14,315.31</td>
<td>–4.34%</td>
<td>388,705.34</td>
<td>0.17%</td>
</tr>
<tr>
<td>0%*</td>
<td>500, 500, 500</td>
<td>14,964.41</td>
<td>0</td>
<td>388,046.42</td>
<td>0</td>
</tr>
<tr>
<td>20%</td>
<td>600, 600, 600</td>
<td>15,585.74</td>
<td>4.25%</td>
<td>387,415.35</td>
<td>–0.16%</td>
</tr>
<tr>
<td>40%</td>
<td>700, 700, 700</td>
<td>16,182.29</td>
<td>8.24%</td>
<td>386,808.87</td>
<td>–0.32%</td>
</tr>
<tr>
<td>80%</td>
<td>900, 900, 900</td>
<td>17,312.07</td>
<td>15.69%</td>
<td>385,659.56</td>
<td>–0.62%</td>
</tr>
<tr>
<td>100%</td>
<td>1,000, 1,000, 1,000</td>
<td>17,849.16</td>
<td>19.28%</td>
<td>385,112.65</td>
<td>–0.76%</td>
</tr>
</tbody>
</table>

Notes: Algorithm: SAPSO;  
*Initial value
4.2.1 Effect of price elasticity coefficient $b_j$ on decisions

We use numerical simulation to test the change pattern of the total profit with respect to price coefficient $b_j$ and to find out the impact of pricing on profit. The results are shown in Table 3.

From Table 3, we know the following conclusions.

1. As price elasticity coefficient increases, total profit decreases. This means that price elasticity significantly impacts the profit.
2. With the increasing of price, production and delivery cycle time decreases.
3. Price elasticity and price change do not impact the inventory shortage proportion ratio $f_j$ ($j = 1, 2, 3$).

4.2.2 Effect of deterioration rate of retailers and manufacturer, $\theta_b, \theta_v$ on decisions

In order to observe the impact of $\theta_b, \theta_v$ on decisions, one can take partial derivative of $TP$ with respect to $\theta_b$ and $\theta_v$; however, since the derivative function is too complicated, it is not intuitively obvious, and thus we use numerical simulation to show the relationships. Tables 3 and 4 reflect the sensitivity analysis with respect to $\theta_b$ and $\theta_v$, respectively.

From the Tables 4 and 5, we find that deterioration rate does not significantly impact the total profit and decisions variables, although with the increase of deterioration rate, the total cost increases (see Table 6), and profit only decreases by a very small percentage. It is because the income is not impacted by deterioration rate (the income formula $\sum_{j=1}^{m}(a_j - b_j p_j) p_j$ can show this), and the total cost in the example only accounts for about 3% of the total income, so the effect of deterioration rate on total profit is very limited.

From Table 6, we also know that the impact of deterioration rate is also very limit for total cost. It reveals the effect of deterioration is not an important factor for computation of total cost. This implies that, in practice; the market decision and inventory decision does not need to consider the factor of deterioration rate. Our study shows that most literatures have enlarged the effect of deterioration rate in their research.

4.2.3 Effect of ordering cost $A_j$

Among all inventory parameters (holding cost and ordering), the numerical scale of ordering cost $A_j$ is the largest one, and it impacts the total cost of retailers, so we take this parameter as one factor for sensitivity analysis. Table 7 shows the result (the values of decision variables are not shown).

From the Table 7, it is obvious that the effect of ordering cost on total cost is larger than on total profit. This means that reduction in ordering cost can significantly reduce total cost; however, it can not significantly increase the system profit.
5 Conclusions

Inventory optimisation is a very important issue in supply chain and manufacturing operations. Traditionally, production and retail decisions are separately made by two companies. With the wide application of information technology, integrating the decisions of the supply chain is a new trend. This paper studies an integrated optimisation model of pricing, inventory decisions in a single manufacturer and multi-retailer system of deteriorating items. Two meta-heuristic algorithms are designed, i.e., SAPSO, CDACO. Numerical experiments show that two algorithms have advantage and disadvantage in solving the problem based on speed and convergence. Main conclusions can be drawn from this study are:

1. Integrated pricing and inventory replenishment is feasible. Price elasticity significantly impacts the retailers pricing and manufacturer production decision (production cycle time), if the product has high price elasticity, integration of pricing and production decision can benefit retailers and the manufacturer, increase the total profit of the supply chain.

2. Deterioration coefficient does not significantly impact the total cost and profit. This reveals that most literatures have exaggerated the role of deterioration rate in inventory optimisation. In practice, in a limit range of deterioration rate (<0.05), inventory replenishment and production decision can ignore the deterioration rate.

3. Compared with price factor, inventory parameter (ordering cost) significantly impacts the inventory cost, but does not significantly impacts the total profit of supply chain. This reveals that inside system factors of supply chain are not important as outside market factors (e.g., price), in increasing profit.

Although we have obtained some important conclusions and implications, some limitations of the study exist.

First, in this study, we only consider the price factor of demand. In practice, deteriorating items’ demand is influenced by other factors, such as, quality, freshness, inventory, etc. this model has simplified the factors of the demand function.

Second, this model does not reflect the competitive behaviour between retailers, only consider the independent demand. In practice, different retailers have different promotion strategies, such as discounting to increase demand. This model does not reflect these real operations of supply chain in practice.

In future, some extensions can be done based on this study. First, it is possible to extend it to the model with multiple products. Second, different retailers with different delivery cycles can also be considered to capture a more general perspective. Also, one can consider different transportation modes and delivery modes such as just-in-time delivery mode to further extend the system. Finally, different demand functions and pricing policies can be included to reflect the real operations of the industrial systems.
Acknowledgements

The author greatly appreciates the editors and the anonymous referees for their valuable comments and suggestions to improve the quality of the paper. This paper is funded by the NSFC of China (No#:71372154).

References


