A fuzzy lifetime-based particle swarm optimisation with varying swarm size to solve a production inventory model

Partha Guchhait*
Department of Mathematics, Vidyasagar University, Midnapore, Paschim-Medinipur, West Bengal, 721102, India
Email: parthaguchhait@gmail.com
*Corresponding author

Manas Kumar Maiti
Department of Mathematics, Mahishadal Raj College, Mahishadal, Purba-Medinipur, West Bengal, 721628, India
Email: manasmaiti@yahoo.co.in.

Abstract: Here, a modified particle swarm optimisation (MPSO) algorithm with varying swarm size for constrained optimisation problem is proposed. In this MPSO, a life time is assigned to each particle at the time of generation depending on its fitness. After completion of a generation, if no movement is made by the particle, its age is increased by unity. When age of a particle exceeds the lifetime, it is discarded from the swarm. Diversity in the swarm is maintained using information entropy theory. A fuzzy possibility/necessity-based fitness evolution is proposed to deal with fuzzy optimisation problems using this MPSO. Efficiency of the algorithm is tested against a list of crisp valued standard benchmark nonlinear test functions. This algorithm is used to solve a production inventory model with fuzzy costs, where lifetime of the product is random in nature. At the beginning of planning horizon price discount is offered to the customers for few cycles to boost the demand. Demand also depends on stock and selling price. The model is illustrated with numerical examples and some sensitivity analyses have been made.

Keywords: modified PSO algorithm; fuzzy lifetime; varying swarm size; possibility/necessity measure; EPQ model; price discounted promotional demand; random planning horizon.

Reference to this paper should be made as follows: Guchhait, P. and Maiti, M.K. (2016) 'A fuzzy lifetime-based particle swarm optimisation with varying swarm size to solve a production inventory model', Int. J. Computational Complexity and Intelligent Algorithms, Vol. 1, No. 1, pp.68–98.

Biographical notes: Partha Guchhait is a Teacher of Mathematics at Dherua Anchal Satabala High School, Paschim-Medinipur, West Bengal, India. He received his BSc in Mathematics and MSc in Applied Mathematics from Vidyasagar University, Midnapore, Paschim-Medinipur, West Bengal, India. He received his PhD (2013) in Applied Mathematics from Vidyasagar...
A fuzzy lifetime-based PSO with varying swarm size

Manas Kumar Maiti is working in the Department of Mathematics, Mahishadal Raj College, Mahishadal, Purba-Medinipur, West Bengal, India. He received his MSc and MPhil in Applied Mathematics from Calcutta University, Midnapore, Paschim-Medinipur, West Bengal, India, and MTech in Computer Science from I.I.T Kharagpur, West Bengal, India. He received his PhD (2006) in Applied Mathematics from Vidyasagar University, India. He has more than 40 publications in reputed research journals. His fields of interest are inventory control, transportation problem, travelling salesman problem, portfolio management, fuzzy mathematics, soft computing techniques, etc.

1 Introduction

The particle swarm optimisation (PSO) is a exhaustive search algorithm based on the motion of a flock of birds searching for food. A PSO starts with a set of potential solutions which is called swarm of the decision-making problem under consideration. Individual solutions are called particles and food is analogous to optimal solution. Since the introduction of the PSO, it is modified and the process is still going on. Here, this algorithm is modified with the introduction of life-time for each particle of a swarm at the time of their birth and it is done depending on their fitness. After completion of a generation, if no movement is made by a particle, its age is increased by unity. It is discarded from the swarm when age exceeds its life-time. At the time of generation of initial swarm, diversity in the swarm is maintained using information entropy theory. Diversity in the swarm is checked after a fixed number of generations, called period. After each period, swarm size may change depending on the diversity of the swarm. If diversity level drops below the fixed threshold, some new (child) particles are included in the swarm. Child particles are obtained using fuzzy life-time-based crossover and mutation operations on the particles of the swarm. Similarly, if the diversity exceeds a predefined upper limit, some particles are discarded. In the algorithm, crossover probability of a pair of parents is a function of their age-type (young, middle-aged, old, etc.) and is obtained using a fuzzy rule-base and possibility theory (cf., Appendix A). Inertia weight \(w\) and probability of mutation \(p_m\) are initially taken very high and gradually decreases to a certain limit with iteration using a particular law [cf., Section 4.1 (h)]. Due to these improvements, this algorithm [called modified PSO i.e., modified particle swarm optimisation (MPSO)] has more global search ability at the beginning of the run and has more local search ability near the end of the run. Performance of this algorithm is tested against a set of well established benchmark nonlinear test functions (TFs) of which local and global extrema are known. These TFs (cf., Appendix C) and their global/local optima obtained by the proposed MPSO are presented. This algorithm is used to find optimal decision of a production inventory model with stock and price discounted promotional demand in a random planning horizon.

In this paper, a production inventory model of an item is developed, where lifetime of the product is assumed as random in nature and follows normal distribution with known
mean and standard deviation. It is also assumed that producer offers a price discount period to his customers in few cycles at the beginning of the planning horizon. During these period, demand increases in each cycle depending on the discount rate. Demand also depends on stock and selling price. After withdrawals of price discount period, demand depends on stock and price for the rest of the cycles. There is only one stochastic constraint – sum of the production cycle length is less than the length of random planning horizon. Models are formulated for both the crisp and fuzzy inventory costs. For crisp inventory parameters, total profit under the above mentioned constraint is maximised using the proposed MPSO to take optimal decisions. When some inventory parameters are fuzzy in nature, total profit through out the system is fuzzy in nature too. In this case, optimal decisions are made using the proposed MPSO with fuzzy objective function. Again, the model is also solved by another soft computing technique i.e., genetic algorithm (GA) (proposed by Guchhait et al., 2010). Obtained results with these two techniques are compared and presented. The models are illustrated with numerical examples and some sensitivity analyses have been made.

2 Literature review

The PSO algorithm has been introduced by Kennedy and Eberhart (1995) and is inspired by the emergent motion of a flock of birds searching for food. It is not only a recently invented high performance optimiser that is easy to understand and implement but, it also requires little computational bookkeeping and generally only a few lines of code (Boeringer and Werner, 2004). Since its introduction, PSO has seen many improvements and applications. Like other heuristics, most of the improvements on PSO are directed towards improving the convergence and increasing the diversity of the swarm. A brief discussion on its improvement in different directions and applications are given in Engelbrecht (2005). Some researchers tried to improve the performance of PSO by variable parameters (Shi and Eberhart, 1998; Zheng et al., 2003; Ratnaweera et al., 2004), some other change the updating equation (Ueno et al., 2005), or adapted operators of the GA (Michalewicz, 1992) or evolutionary strategy (ES) such as crossover, mutation, and sharing (Shi et al., 2005). Similar to other heuristics, PSO also has the problem of converging to undesired local optima because of the diversity of swarm decreasing in latter evolutionary periods and excessive combination with other evolutionary methods may weaken the powerful advantages of PSO. Chen and Zhao (2009) proposed a PSO that uses an adaptive variable swarm size method. In their work, the setting generation is divided into several equal periods; each period is called a ‘ladder’ (a fixed number of generations). This heuristic algorithm is developed with periodic partial increasing or declining of individuals according to diversities in the end of every ladder. Also, in each ladder, the current swarm maintains the same size and the diversity are estimated only at the end of this ladder. If the current diversity is larger than the threshold, the swarm size will be decreased and particles with small score in the ladder will be removed, if the current diversity values is smaller than the threshold, some new individuals produced by the crossover operator will appended to the swarm, otherwise, the swarm size is maintained in current level. So, the swarm size of the ladder is determined by the diversity of the swarm in the terminal of their front ladder (previous ladder).

Now-a-days, it is found (in the subcontinent countries) that some manufacturers offer price discount in the form of putting additional unit(s) in every pack of that item. This
process of boosting a product is commonly practiced by the manufacturer especially when a product is newly launched in the market. Pal et al. (2009) developed an EPQ model incorporating price discounted promotional demand in a fuzzy planning horizon. They assume that producer offers price discount in every production cycle for some time, but ignore the effect of product availability in stimulating demand, though this effect is well established (Chung et al., 2000; Maiti and Maiti, 2006). Liang and Zhou (2011) published a two-warehouse inventory model for deteriorating items under conditionally permissible delay in payment. Recently, Molamohamadi et. al. (2014) presented a review article of inventory models under trade credit which really helps the young researchers in this direction.

Production-inventory/inventory models are normally developed under the common assumption that product lifetime is infinite and models are developed under infinite planning horizon (Abad, 2000; Sarkar et. al., 2011; Guchhait et. al., 2010, 2013; De and Sana, 2013; De et. al., 2014). According to this assumption, product specification remains unchanged for ever. But, in reality it is observed that rapid development of technology leads to rapid change in product specifications with new features, new packets and name. Naturally, lifetime of a product is finite and normally it is imprecise (stochastic or fuzzy) in nature. In the literature, there are number of papers with this assumption of stochastic parameters (Gurnani, 1983; Moon and Yun, 1993; Roy et al., 2007). In fact estimation of stochastic parameters is made on sufficient amount of past data. On the other hand, estimation of fuzzy parameters is done by expert's opinion. So when past data is insufficient (especially for the newly launched products) one has to depend on fuzzy parameters. But, only few number of production inventory models have been developed incorporating this realistic feature (Taleizadeh et. al., 2013; Guchhait et. al., 2012; Chakraborty et. al., 2013; De and Sana, 2013). Accordingly, the problem is developed with fuzzy parameters.

Production cost of a manufacturing system mainly depends on the cost of raw materials and labour. Normally, raw material costs are imprecise in nature. In the existing literature of production/inventory control problems, labour cost is usually assumed as constant. However, in many realistic situations, because of the firms and employees perform the same task repeatedly; they learn how to perform rapidly. Therefore, processing cost of per unit product decreases to a certain limit in every cycle. At the same time, part of ordering cost may also decreases to a certain limit in every cycle. This phenomenon is known as the ‘learning effect’, in the literature. Although, different types of learning effects have been studied in various areas (Kuo and Yang, 2006), it has rarely been studied in the context of inventory control problems (Pal et al., 2009).

3 Mathematical prerequisite

The following notations and assumptions are used in developing the model.

- **Notations**
  
  - $T$: length of one cycle
  - $K$: production rate
  - $Q_i$: maximum inventory level in $i^{th}$ cycle
  - $c_h$: holding cost per unit in $
\( q(t) \)  inventory level at any time \( t \) in \( i^{th} \) cycle

\( Z \)  total profit from \( N \) cycles

\( \alpha \)  probability level for the stochastic constraint on the planning horizon.

- **Assumptions**

1. Inventory system involves only one item.

2. The time horizon \( H \) is random in nature and follows normal distribution with known mean \( m_H \) and standard deviation \( \sigma_H \) and \( h \) is real planning horizon such that \( P(H \leq 0) \) is negligible.

3. The time horizon accommodates \( N \) full cycles, i.e., \( NT \leq h \).

4. Price discount is offered to the customers in the first \( M \) cycles.

5. Production cost per unit in \( i^{th} \) cycle \( c_{pi} = c_r + L_i \), where \( c_r \) is raw material cost and \( L_i \) is labour cost. \( L_i \) decreases in each cycle due to learning effect and is of the form

\[
L_i = L_0 + L_1 / i^{\beta_i}
\]

where \( L_0, L_1 \) and \( \beta_i \) are constants so chosen to best fit the labour cost function.

6. Setup cost in \( i^{th} \) cycle \( c_{si} \) (in $) is partly constant and partly decreases in each cycle due to learning effect of the employees and is of the form:

\[
c_{si} = c_0 + c_1 / i^{\beta_s}
\]

where \( c_0, c_1 \) and \( \beta_s \) are constants so chosen to best fit the set-up cost function.

7. Selling price in \( i^{th} \) cycle \( s_{pi} \) is a mark-up of production cost \( c_{pi} \). \( m_1 \) and \( m_2 \) are mark-ups during price discount period and normal period respectively, i.e.,

\[
s_{pi} = \begin{cases} 
  m_1 c_{pi} & \text{for } 0 \leq t \leq MT; \quad m_1 > 1, \\
  m_2 c_{pi} & \text{for } MT \leq t \leq NT; \quad m_2 > m_1,
\end{cases}
\]

8. Demand of the item in \( i^{th} \) cycle \( D_i(t) \) is of the form:

\[
D_i(t) = \begin{cases} 
  \frac{A + Bq_i - CR(m_2 - m_1)MT}{s_{pi}^{\gamma}} & \text{for } 0 \leq q_i \leq Q_0 \quad (0 \leq t \leq MT, s_{pi} = m_1 c_{pi}) \\
  \frac{A + BQ_0 - CR(m_2 - m_1)MT}{s_{pi}^{\gamma}} & \text{for } Q_0 \leq q_i \leq Q \quad (MT \leq t \leq NT, s_{pi} = m_2 c_{pi})
\end{cases}
\]

where \( A, B(0 < B < 1), C, \) and \( R(0 < R < 1) \) are four parameters so chosen to best fit the demand function. \( Q_0 \) is the maximum inventory level in \( i^{th} \) cycle and displayed inventory has impact on demand up to level \( Q_0 \).
Here, $T, N, M, m_1$ and $m_2$ are decision variables. In the development of the model, it is assumed that the life time of the product, $\hat{H}$, is random in nature and this is taken as planning horizon of the model. ‘$N$’ cycles are completed during the real time horizon ‘$h$’, here length of each cycle is ‘$T$’ and then $NT \leq h$ clearly. At the beginning of $i^{th}$ cycle, i.e., at $t = (i-1)T$, item is produced at a rate ‘$K$’ and inventory is built up at a rate $K - D_i(t)$. When $t = (i-1)T + t_0$, inventory level reaches $Q_0$. Production stopped when $t = (i-1)T + t_1$, and at that time inventory level reaches $Q_i$. After that, inventory is depleted due to the demand rate $D_i(t)$ and its level reaches $Q_0$ again at $t = (i-1)T + t_2$.

Finally, inventory vanishes at $t = iT$ and production for next cycle starts. Inventory levels over time are depicted in Figures 1 and 2.

**Figure 1** Inventory levels in $i^{th}$ cycles when $Q_0 < Q_i$

**Figure 2** Inventory levels in $i^{th}$ cycle when $Q_0 > Q_i$

### 3.1 Formulation for first $M$ cycles

Depending upon the values of $Q_0$ and $Q_i$, two cases may arise.

**Case 1** ($Q_0 \leq Q_i$): according to the above assumptions, instantaneous state $q_i(t)$ of the item in $i^{th}$ ($i \leq M$) cycle ($(i-1)T \leq t \leq iT$) is given by
\[
\frac{dq_i(t)}{dt} = \begin{cases} 
K - \frac{A + Bq_i - CR(m_2-m_1)t}{m_c p_i'} & (i-1)T \leq t \leq (i-1)T + t_0 \\
K - \frac{A + BQ_0 - CR(m_2-m_1)T}{m_c p_i'} & (i-1)T + t_0 \leq t \leq (i-1)T + t_f \\
A + Bq_i - CR(m_2-m_1)MT & (i-1)T + t_f \leq t \leq (i-1)T + t_3 \\
A + BQ_0 - CR(m_2-m_1)MT & (i-1)T + t_3 \leq t \leq iT 
\end{cases} 
\]  

where \( q_i((i-1)T) = 0, \quad q_i((i-1)T + t_0) = Q_1, \quad q_i((i-1)T + t_f) = Q_0, \quad q_i(iT) = 0. \)

Solving the equation (1) and using the boundary conditions we have

\[
t_0 = \left(\frac{m_c p_i}{B}\right) \log \left[ \frac{K (m_c p_i)' - A + CR(m_2-m_1)T}{K (m_c p_i)' - A - BQ_0 + CR(m_2-m_1)T} \right] 
\]  

\[
t_f = T - \left(\frac{m_c p_i}{B}\right) \log \left[ \frac{A + BQ_0 - CR(m_2-m_1)T}{A - CR(m_2-m_1)T} \right] 
\]  

\[
t_f = t_0 + (t_f - t_0) \left[ \frac{A + BQ_0 - CR(m_2-m_1)T}{K (m_c p_i)'} \right] 
\]  

\[
Q_0 = Q_0 + (t_f - t_0) \left[ \frac{A + BQ_0 - CR(m_2-m_1)T}{K (m_c p_i)'} \right] 
\]

So, the holding cost for \( i \)th cycle, \( c_i \)HOC = \( c_h \) (\( H_1 + H_2 + H_3 + H_4 \)), where

\[
H_1 = \int_{(i-1)T}^{(i-1)T + t_0} q_i(t) dt = \int_{0}^{Q_1} \left[ \frac{(m_c p_i)'}{K (m_c p_i)'} q_i(t) dt \right] 
\]

\[
= \left(\frac{m_c p_i}{B^2}\right) \left[ K (m_c p_i)' - A + CR(m_2-m_1)T \right] 
\]

\[
\times \log \left[ \frac{K (m_c p_i)' - A + CR(m_2-m_1)T}{K (m_c p_i)' - A + BQ_0 - CR(m_2-m_1)T} \right] - BQ_0 
\]
A fuzzy lifetime-based PSO with varying swarm size

\[ H_2 = \sum_{(i-1)T+t_0}^{iT+t_1} q_i(t) dt = \int_{(i-1)T+t_0}^{iT+t_1} \left[ \frac{(m_c p_i)^Y q_i(t) dq_i}{Q_B (m_c p_i)^Y - \{A + B Q_0 - CR^{(m_2 - m_1)T}\}} \right] \]

\[ = \frac{2 (Q_0^2 - Q_0^2) (m_c p_i)^Y}{K (m_c p_i)^Y - \{A + B Q_0 - CR^{(m_2 - m_1)T}\}} \]

\[ H_3 = \sum_{(i-1)T+t_0}^{iT+t_2} q_i(t) dt = \int_{(i-1)T+t_0}^{iT+t_2} \frac{-(m_c p_i)^Y q_i(t) dq_i}{A + B Q_0 - CR^{(m_2 - m_1)T}} = \frac{(Q_0^2 - Q_0^2) (m_c p_i)^Y}{2 \{A + B Q_0 - CR^{(m_2 - m_1)T}\}} \]

\[ H_4 = \sum_{(i-1)T+t_0}^{iT+t_2} q_i(t) dt = \int_{(i-1)T+t_0}^{iT+t_2} \frac{-(m_c p_i)^Y q_i(t) dq_i}{A + B Q_0 - CR^{(m_2 - m_1)T}} = \frac{(m_c p_i)^Y}{B^2} \left[ B Q_0 - \{A - CR^{(m_2 - m_1)T}\} \times \log \left\{ \frac{A + B Q_0 + CR^{(m_2 - m_1)T}}{A - CR^{(m_2 - m_1)T}} \right\} \right] \]

Case 2 (Q_0 > Q_1): in this case, instantaneous state \( q_i(t) \) of the item in \( i^{th} \) \((i <= M)\) cycle \((i-1)T <= t <= iT\) is given by

\[ \frac{dq_i(t)}{dt} = \begin{cases} \frac{K - A + B Q_0 - CR^{(m_2 - m_1)T}}{m_c p_i} & (i-1)T \leq t \leq (i-1)T + t_i \\ \frac{-A + B Q_0 - CR^{(m_2 - m_1)T}}{m_c p_i} & (i-1)T + t_i \leq t \leq iT \end{cases} \]

where \( q_i((i-1)T) = 0, q_i((i-1)T + t_i) = Q_i, q_i(iT) = 0 \).

Solving the equation (10) and using the above boundary conditions we have

\[ Q_i = \frac{e^{TB/(m_c p_i)^Y} - 1}{B} \left[ \left\{A - CR^{(m_2 - m_1)T}\right\} \left\{K - A + CR^{(m_2 - m_1)T}\right\} \right] \]

\[ t_i = \frac{(m_c p_i)^Y}{B} \log \left[ \frac{(m_c p_i)^Y \left\{K - A + CR^{(m_2 - m_1)T}\right\}}{(m_c p_i)^Y \left\{K - A - B Q_0 + CR^{(m_2 - m_1)T}\right\}} \right] \]

So, the holding cost for \( i^{th} \) cycle, \( c_i HOC = c_i (H_1 + H_2) \), where

\[ H_1 = \int_{(i-1)T}^{iT} q_i(t) dt \] and \[ H_2 = \int_{(i-1)T+t_0}^{iT} q_i(t) dt \]

Can be obtained easily as in Case 1 of Section 3.1
3.2 Formulation for last $N-M$ cycles

Depending on the values of $Q_0$ and $Q_i$, again two cases may arise.

Case 1 ($Q_0 \leq Q_i$): according to the above assumptions, instantaneous state $q(t)$ of the item in $i^{th}$ ($i > M$) cycle ($i-1)T = t = iT$ is given by

$$
\frac{dq_i(t)}{dt} = \begin{cases} 
K - A + BQ_i - CR^{(m_2-m_1)} \frac{m_2 c_{pi}}{m_2 c_{pi}'} & (i-1)T \leq t \leq (i-1)T + t_{0i} \\
K - A + BQ_i - CR^{(m_2-m_1)} \frac{m_2 c_{pi}}{m_2 c_{pi}'} & (i-1)T + t_{0i} \leq t \leq (i-1)T + t_{0i} \\
A + BQ_i - CR^{(m_2-m_1)}MT \frac{m_2 c_{pi}}{m_2 c_{pi}'} & (i-1)T + t_{1i} \leq t \leq (i-1)T + t_{2i} \\
A + BQ_i - CR^{(m_2-m_1)}MT \frac{m_2 c_{pi}}{m_2 c_{pi}'} & (i-1)T + t_{2i} \leq t \leq iT 
\end{cases}
$$

(13)

where $q((i-1)T) = 0$, $q((i-1)T + t_{0i}) = Q_0$, $q((i-1)T + t_{1i}) = Q_i$, $q((i-1)T + t_{2i}) = Q_0$, $q(iT) = 0$.

Solving the equation (13) and using the boundary conditions we have,

$$
t_{0i} = \frac{(m_2 c_{pi})'}{B} \log \left[ \frac{K}{K(m_2 c_{pi})'} - A + CR^{(m_2-m_1)}MT \frac{m_2 c_{pi}}{m_2 c_{pi}'} \right]
$$

(14)

$$
t_{2i} = T - \frac{(m_2 c_{pi})'}{B} \log \left[ \frac{A + BQ_i - CR^{(m_2-m_1)}MT}{A - CR^{(m_2-m_1)}MT} \right]
$$

(15)

$$
t_{1i} = t_{0i} + (t_{2i} - t_{0i}) \left[ \frac{A + BQ_i - CR^{(m_2-m_1)}MT}{K(m_2 c_{pi})'} \right]
$$

(16)

$$
Q_i = Q_0 + (t_{2i} - t_{0i}) \left[ \frac{A + BQ_i - CR^{(m_2-m_1)}MT}{K(m_2 c_{pi})'} \right]
$$

(17)

So, the holding cost for $i^{th}$ cycle, $c_i \text{HOC}_i = c_i(H_1 + H_2 + H_3 + H_4)$, where

$$
H_1 = \int_{(i-1)T}^{(i-1)T + t_{0i}} q_i(t) dt, \quad H_2 = \int_{(i-1)T + t_{0i}}^{(i-1)T + t_{1i}} q_i(t) dt, \quad H_3 = \int_{(i-1)T + t_{1i}}^{(i-1)T + t_{2i}} q_i(t) dt, \quad H_4 = \int_{(i-1)T + t_{2i}}^{iT} q_i(t) dt 
$$

Can be obtained easily as in Case 1 of Section 3.1
Case 2 ($Q_{0} > Q_{i}$): in this case, instantaneous state $q_{i}(t)$ of the item in $i^{th}$ ($i > M$) cycle \((i-1)T <= t <= iT) is given by

\[
\frac{dq_{i}(t)}{dt} = \begin{cases} 
-K + A + Bq_{i} - CR^{(m_{2} - m_{1})MT} & (i-1)T \leq t \leq (i-1)T + t_{i} \\
-A + Bq_{i} - CR^{(m_{2} - m_{1})MT} & (i-1)T + t_{i} \leq t \leq iT 
\end{cases}
\]

\[ (18) \]

where $q_{i}((i-1)T) = 0, q_{i}((i-1)T + t_{i}) = Q_{0}, q_{i}(iT) = 0$.

Solving the equation (18) and using the above boundary conditions we have

\[
Q_{i} = \frac{e^{TB/[(m_{2}c_{pl})^{2}]} - 1}{B} \left[ (m_{2}c_{pl})^{2} K - A + CR^{(m_{2} - m_{1})MT} \right] \left[ (m_{2}c_{pl})^{2} K - A + BQ_{0} + CR^{(m_{2} - m_{1})MT} \right] \]

\[ (19) \]

\[
t_{i} = \frac{m_{2}c_{pl}}{B} \log \frac{(m_{2}c_{pl})^{2} K - A + CR^{(m_{2} - m_{1})MT}}{(m_{2}c_{pl})^{2} K - A + BQ_{0} + CR^{(m_{2} - m_{1})MT}} \]

\[ (20) \]

So, the holding cost for $i^{th}$ cycle, $c_{i}HOC_{i} = c_{i}(H_{1} + H_{2})$, where

\[
H_{1} = \int_{(i-1)T}^{iT} q_{i}(t)dt \quad \text{and} \quad H_{2} = \int_{(i-1)T}^{iT} q_{i}(t)dt
\]

Can be obtained easily as in Case 1 of Section 3.1

3.3 Formulation of objective function

Incorporating all the cases, the total profit from $N$ cycles can be represented by

\[
Z = SP - PC - HOC - OC
\]

where

\[
SP = \text{total selling price} = \sum_{i=1}^{M} K_{ti}m_{1}c_{pl} + \sum_{i=M+1}^{N} K_{ti}m_{2}c_{pl}
\]

\[
PC = \text{Total production cost} = \sum_{i=1}^{N} K_{ti}c_{pl}
\]

\[
HOC = \text{Total holding cost} = \sum_{i=1}^{N} HOC_{i}
\]

\[
OC = \text{Total setup cost} = \sum_{i=1}^{N} c_{si} = \sum_{i=1}^{N} \left( c_{0i} + \frac{c_{1i}}{i^{\beta}} \right)
\]

A fuzzy lifetime-based PSO with varying swarm size
3.4 Model with crisp objective

When all the inventory parameters are crisp, then the problem reduces to determine $T$, $N$, $M$, $m_1$ and $m_2$ to

Maximize $Z$

subject to $P(NT \leq \hat{H}) > \alpha$ (22)

Using the optimisation process with stochastic constraints by Charnes and Cooper (1959) (cf., Appendix B), the above problem reduces to determine $T$, $N$, $M$, $m_1$ and $m_2$ to

Maximize $Z$

subject to $NT \leq m_H - \varepsilon\sigma_H$, where $\alpha = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\varepsilon} e^{-t^2/2} dt$ (23)

For different values of ‘$\alpha$’, values of ‘$\varepsilon$’ are taken from standard normal distribution table and the problem is solved using a modified heuristic technique MPSO (cf., §4).

3.5 Model with fuzzy objective

Here, it is assumed that the holding cost ‘$c_h$’, the setup cost coefficients ‘$c_0$’ and ‘$c_1$’ as triangular fuzzy number (TFN), $\tilde{c}_h = (c_{h1}, c_{h2}, c_{h3})$, $\tilde{c}_0 = (c_{01}, c_{02}, c_{03})$, $\tilde{c}_1 = (c_{11}, c_{12}, c_{13})$ respectively. Then, the total holding cost $HOC$ becomes a TFN, $HOC_F = (HOC_1, HOC_2, HOC_3)$ and the total setup cost $OC$ also becomes a TFN, $OC_F = (OC_{F1}, OC_{F2}, OC_{F3})$ where

$$H_{Fj} = c_{Fj} \sum_{i=1}^{N} HOC_i$$ and $$OC_{Fj} = \sum_{i=1}^{N} \left( c_{ij} + \frac{c_{ij}}{i^p} \right), \quad j = 1, 2, 3.$$ 

Hence, the total profit $Z$ also becomes a TFN, $\tilde{Z} = (Z_1, Z_2, Z_3)$ where $Z_1 = SP - PC - OC_{F1} - H_{F1}$, $Z_2 = SP - PC - OC_{F2} - H_{F2}$ and $Z_3 = SP - PC - OC_{F3} - H_{F3}$. So, the problem reduces to

Determine $T$, $N$, $M$, $m_1$ and $m_2$ to

Maximize $\tilde{Z}$

subject to $P(NT \leq \hat{H}) > \alpha$ (24)

Using the optimisation process with stochastic constraints by Charnes and Cooper (1959) (cf., Appendix B), the above problem reduces to

Determine $T$, $N$, $M$, $m_1$ and $m_2$ to

Maximize $\tilde{Z}$

subject to $NT \leq m_H - \varepsilon\sigma_H$, where $\alpha = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\varepsilon} e^{-t^2/2} dt$ (25)

This problem is solved using a modified MPSO algorithm (which is discussed below in Section 4) for fuzzy objective.
4 Modified particle swarm optimisation

PSOs are exhaustive search algorithms based on the emergent motion of a flock of birds searching for food (Eberhart and Kennedy, 1995; Kennedy and Eberhart, 1995) and have been extensively used/modified to solve complex decision-making problems in different field of science and technology (Engelbrecht, 2005; Esmin et al., 2002; Feng, 2005; Liang et al., 2006). A PSO normally starts with a set of potential solutions (called swarm) of the decision-making problem under consideration. Individual solutions are called particles and food is analogous to optimal solution. In simple terms the particles are flown through a multi-dimensional search space, where the position of each particle is adjusted according to its own experience and that of its neighbours. Each particle ‘i’ has a position vector $X_i(t)$, a velocity vector $V_i(t)$, the position at which the best fitness ($pbest_i$) encountered by the particle so far, and the best position of all particles ($gbest$) in current generation $t$. In generation ($t + 1$), the position and velocity of the particle are changed to $X_i(t + 1)$ and $V_i(t + 1)$ using following rules:

$$V_i(t + 1) = wV_i(t) + c_1 r_1 (X_{pbest_i} - X_i(t)) + c_2 r_2 (X_{gbest} - X_i(t))$$

$$X_i(t + 1) = X_i(t) + V_i(t + 1)$$

The parameters $c_1$ and $c_2$ are set to constant value, which are normally taken as 2, $r_1$ and $r_2$ are two random values, uniformly distributed in $[0, 1]$, $w(0 < w < 1)$ is inertia weight which controls the influence of previous velocity on the new velocity. Global search performance is good with large inertial weight while a small inertia weight facilitates a local search. As a result, in the beginning of searching, $w$ is set to high value and decreasing to a lower limit at the end of evolutionary processing. Due to these improvements/changes, PSO have more global search ability at the beginning of the run and have more local search ability near the end of the run. Better performance is derived in many function optimisation problems (Chen and Zhao, 2009). The second part in equation (26) is called cognition character and the third part is social character of particles. The position of each particle is updated every generation. When a particle discovers a pattern that is better than any that it has found previously, it will track the better position by updating equations (26) and (27).

Similar to other heuristics, in PSO, poor parameter settings usually leads to several problems such as premature convergence. PSO also has the problem of converging to undesired local optima because of the diversity of swarm decreasing in latter periodic of evolutionary. Again excessive combination with other evolutionary methods may weaken the powerful advantages of PSO. Overcoming these drawbacks, Chen and Zhao (2009) presented a PSO algorithm that uses an adaptive variable swarm size and periodic partial increasing or declining of particles in the form of ladder function. The aim was to enhance the overall performance of PSO. Their algorithm adjusts the swarm size automatically according to the value of diversity of the swarm in ultimate time of current ladder. For increasing the particles they use simple crossover and mutation operation of GA and for decreasing, particles with lower fitness are removed.

In this paper, the PSO algorithm is modified (in this case it is called as modified PSO i.e., MPSO), where at the time of generation of each particle, a lifetime is assigned depending upon its fitness. Fitness of a particle depends on the objective function value due to it. After completion of a generation (iteration), if no movement is made by the
particle, its age is increased by unity. When age exceeds lifetime, particles are discarded from the swarm at the beginning of every iteration. It avoids the local convergence of the algorithm. At the time of generation of initial swarm, diversity in the swarm is maintained using information entropy theory. Every time a new particle $X_i$ is generated, the entropy between this one and previously generated individuals is calculated. If this information quantity is higher than a threshold $E_T$, fixed at the beginning, $X_i$ is included in the swarm otherwise $X_i$ is again generated until diversity exceeds the threshold $E_T$. This method induces a good distribution of initial swarm. Diversity in the swarm is also checked after a fixed number of generations called period. After each period, swarm size may change depending on the diversity of the swarm. If diversity level drops threshold level $E_T$, some new (child) particles are included in the swarm. Child particles are obtained using fuzzy lifetime-based crossover and mutation operations on the swarm. Similarly, if the diversity exceeds predefined upper limit, some particles are discarded. For crossover operation particles are classified into young, middle age and old (in fuzzy sense) according to their age and lifetime. Following comparison of fuzzy numbers using possibility theory (cf., Appendix A), here crossover probability is measured as a function of parent’s age interval (a fuzzy rule base on parents age limit is also used for this purpose). In this MPSO, a subset of better child particles are included with the parent swarm for next period and maximum size of this subset is a percentage of the size of its parent set. To control memory overflow at the runtime of the algorithm, an upper limit of swarm size is imposed (Maxsize). Similarly, a lower limit (Minsize) of swarm size is maintained. Algorithm terminates when difference between maximum fitness (Maxfit) and average fitness (Avgfit) of the swarm becomes negligible (less than a predefined small value $\epsilon$). Here inertia weight ($\omega$) and probability of mutation ($p_m$) are initially very high and follows a decreasing law in function of the generation number. General structure of this MPSO for a maximization problem is presented below. The same algorithm can be used for minimisation problem, by negating the objective value or changing the inequalities of step 22 and step 23 of the algorithm.

In the algorithm $B_l$ and $B_r$ are lower and upper limit vector respectively of variation of search space. Check constraint ($X_i$) checks whether the solution $X_i$ satisfies the constraints of the problem or not. If $X_i$ satisfies the constraints of the problem, it returns a value 1 otherwise return 0.

Proposed MPSO algorithm

1. Initialize period length $T$, $E_T$, Maxsize, Minsize, $\epsilon$, $\omega$ and $p_m$
2. Set iteration counter $t = 1$ and randomly generate initial swarm $S(t)$, where diversity in the swarm is maintained using entropy originating from information theory.
3. Find lifetime and set velocity $V_i(t)$ of each particles $X_i(t)$ of initial swarm $S(t)$.
4. Do
5. For $j = 1: T$ do
6. For $i = 1: pop_size$ do
7. \[ V_i(t + 1) = \omega V_i(t) + c_1 r_1 (X_{pbest_i}(t) - X_i(t)) + c_2 r_2 (X_{gbest}(t) - X_i(t)) \]
8. If ($V_i(t + 1) > V_{max}$) then set $V_i(t + 1) = V_{max}$
9. If ($V_i(t + 1) < -V_{max}$) then set $V_i(t + 1) = -V_{max}$
10. $X_i(t + 1) = X_i(t) + V_i(t + 1)$
11. If ($X_i(t + 1) > B_r$) then set $X_i(t + 1) = B_r$
A fuzzy lifetime-based PSO with varying swarm size

If ($X_i(t+1) < B_i$) then set $X_i(t+1) = B_i$
If check_constraint ($X_i(t+1)$) = 0
Increase age of $X_i(t)$ by 1
If age of $X_i(t) >$ life_time
Remove $X_i(t)$ from swarm.
else
Set $X_i(t+1) = X_i(t)$, $V_i(t+1) = V_i(t)$
End If
Else
Set age of $X_i(t+1) = 0$
if $f(X_i(t+1)) > f(X_{pbest_i})$ then set $X_{pbest_i} = X_i(t+1)$
if $f(X_i(t+1)) > f(X_{gbest})$ then set $X_{gbest} = X_i(t+1)$
End If
End For
Set $t = t + 1$
Set $Div =$ diversity of $S(t)$.
If $Div < E_T$
Include child particles in $S(t)$ using fuzzy crossover and mutation such that its size does not exceeds Maxsize.
End If
If $Div > E_T$
Remove some particles from $S(t)$ using q-tournament method Chen and Zhao (2009) such that its size does not drops Minsize.
End If
Decrease value of w and $p_m$.
Set $Maxfit =$ Maximum fitness in $S(t)$ and $Avgfit =$ Average fitness of $S(t)$.
While $(Maxfit - Avgfit \geq \alpha)$
Output: Best particle of $S(t)$
End Algorithm

Algorithm for inclusion of particles into $S(t)$:

1 For each pair of particles in $S(t)$ do
2 Determine probability of crossover $\tilde{p}_c$ or the selected pair of parents using fuzzy rule-base and possibility theory.
3 Perform crossover with probability $\tilde{p}_c$.
4 If crossover occurs and child particles satisfy the constraints of the problem store them into offspring set.
5 End For
6 For each particles $X_i(t)$ in $S(t)$ do
7 Perform mutation with probability $p_m$.
8 If mutation occurs and child particle satisfies the constraints of the problem store
them into offspring set.
9 End For
10 Rank the particles of offspring set using q-tournament method (Chen and Zhao, 2009).
11 Select a percent of better offsprings from the offspring set and insert into \( S(t) \), such that
maximum size of \( S(t) \) does not exceed Maxsize.
12 Remove all offsprings from the offspring set.
13 End Algorithm

4.1 Different steps of MPSO algorithm

a Representation of particles: a ‘\( n \) dimensional real vector’, \( X_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \), is
used to represent \( i^{th} \) solution, where \( x_{i1}, x_{i2}, \ldots, x_{in} \) represent \( n \) decision variables of
the decision-making problem under consideration. \( Xi \) is called \( i^{th} \) chromosome and \( x_{ij} \) is
called \( j^{th} \) gene of \( i^{th} \) chromosome.

b Initialisation: \( N \) solutions \( X_i = (x_{i1}, x_{i2}, \ldots, x_{in}), i = 1, 2, \ldots, N \) are randomly generated
by random number generator within the boundaries of each variable \([B_{il}, B_{ir}]\),
\( j = 1, 2, \ldots, n \). Initialise \( (S(1)) \) sub-function is used for this purpose.

c Constraint checking: for constrained optimisation problems, at the time of generation
of each individuals \( X_i \) of \( S(1) \), constraints are checked using a separate sub-function
\text{check\_constraint}(X_i), which returns 1 if \( X_i \) satisfies the constraints otherwise returns
0. If \( \text{check\_constraint}(X_i) = 1 \), then \( X_i \) is included in \( S(1) \) otherwise \( X_i \) is again
generated and it continues until constraints are satisfied.

d Diversity preservation: at the time of generation of \( S(1) \) diversity is maintained using
entropy originating from information theory. Following steps are used for this
purpose.

1 Probability, \( pr_{jk} \), that the value of the \( i^{th} \) variable of the \( j^{th} \) particle is different
from the \( i^{th} \) variable of the \( k^{th} \) particle is calculated using the formula

\[
pr_{jk} = 1 - \frac{|x_{ij} - x_{ik}|}{B_{ir} - B_{il}}
\]

where \([B_{il}, B_{ir}]\) is the variation domain of the \( i^{th} \) variable.

2 Entropy of the \( i^{th} \) variable, \( E_i(M) \), \( i = 1, 2, \ldots, n \) is calculated using the formula.

\[
E_i(M) = \sum_{j=1}^{M-1} \sum_{k=j+1}^{M} -pr_{jk} \log pr_{jk}
\]

where \( M \) is the size of the swarm.

3 Average entropy of the current swarm is calculated by the formula

\[
E(M) = \frac{1}{n} \sum_{i=1}^{n} E_i(M)
\]
Incorporating the above three steps a separate sub-function check_diversity($X_i$) is developed. Every time a new particle $X_i$ is generated, the entropy between this one and previously generated individuals is calculated. If this information quantity is higher than a threshold, $E_T$, fixed at the beginning, $X_i$ is included in the swarm otherwise $X_i$ is again generated until diversity exceeds the threshold, $E_T$. This method induces a good distribution of initial swarm.

\textit{e} Determination of fitness and lifetime: fitness $Z(X_i)$ of the particle $X_i$ is determined using the following formula:

\begin{equation}
Z(X_i) = \begin{cases} 
  f(X_i) & \text{for maximization problem,} \\
  L - f(X_i) & \text{for minimization problem}
\end{cases}
\end{equation}

where $f(x)$ is objective function and $L$ is a sufficiently large positive real to make fitness positive. Following Michalewicz (1992), at the time of birth life-time of $X_i$ is computed using the following formula:

If $\text{Avgfit} \geq Z(X_i)$, lifetime($X_i$) = \[
\frac{\text{Minlt} + \frac{K(Z(X_i) - \text{Minfit})}{\text{Avgfit} - \text{Minfit}}}{2}
\]

If $\text{Avgfit} < Z(X_i)$, lifetime($X_i$) = \[
\frac{\text{Minlt} + \frac{K(Z(X_i) - \text{Avgfit})}{\text{Maxfit} - \text{Avgfit}}}{2}
\]

where Maxlt and Minlt are maximum and minimum allowed lifetime of a particle, $K = (\text{Maxlt} - \text{Minlt})/2$. Maxfit, Avgfit, Minfit represent the best, average and worst fitness of the current population. To optimise different TFs it is assumed that Maxlt = 7 and Minlt = 1, N = 10. According to the age, a particle can belongs to any one of age intervals – young, middle-age or old, whose membership functions are presented in Figure 3.

\textit{f} Crossover:

- Determination of probability of crossover ($\tilde{p}_c$): probability of crossover $\tilde{p}_c$, for a pair of parents $(X_i, X_j)$ is determined as below:

  1. At first age intervals (young, middle-age, old) of $X_i$ and $X_j$ are determined by making possibility measure of fuzzy numbers – young, middle-age, old with respect to their age using Lemma 4 (cf., Appendix A). If age of $X_i$ is $a_i$, then possibility measures of fuzzy numbers young, middle-age and old with respect to $a_i$ are calculated. Then age interval having maximum possibility measure is taken as age interval of $X_i$. Similarly, age interval of $X_j$ is determined,

  2. After determination of age intervals of the parents their crossover probability ($\tilde{p}_c$) is determined as a linguistic variable (low, medium or high) using a fuzzy rule-base as presented in Table 1. Membership function of these linguistic variables is presented in Figure 4.
3 Crossover process: for each pair of parent solutions $X_i$, $X_j$ a random number $c$ is generated from the range $[0, 1]$ and if $\text{nes}(c < \tilde{p}_i) > \beta$ (cf., Lemma 1 in Appendix A), crossover operation is made on $X_i$, $X_j$, where $\beta(0 < \beta < 1)$ is a predefined necessity level. For the proposed model it is assumed that $\beta = 0.5$. To made crossover operation on each pair of coupled solutions $X_i$, $X_j$, a random number $c_1$ is generated from the range $[0, 1]$ and their offsprings $Y_1$ and $Y_2$ are determined by the formula:

$$Y_1 = c_1 X_i + (1 - c_1) X_j, \quad Y_2 = c_1 X_j + (1 - c_1) X_i$$

For constrained optimisation problems, if a child solution satisfies the constraints of the problem then it is included in the offspring set otherwise it is not included in the offspring set.

g Mutation:

1 Selection for mutation: for each offspring generate a random number ‘$r$’ from the range $[0, 1]$. If $r < p_m$ then the solution is taken for mutation, where $p_m$ is the probability of mutation.

2 Mutation process: random mutation process is used for this purpose. According to this process a randomly selected variable of the solution is replaced by its regenerated value with its search boundary. So to mutate a solution $X = (x_1, x_2, \ldots, x_n)$ a random integer $i$ in the range $[1, n]$ has to be selected. Then replace $x_i$ by randomly generated value within the boundary $[B_{il}, B_{ir}]$ of $i^{th}$ variable of $X$. New solution (if satisfies constraints of the problem) included in the offspring set. Constraint checking of a child solution $C_i$ is made using check_constraint($C_i$) function.

h Reduction process of $\omega$ and $p_m$: let $\omega(0)$ and $p_m(0)$ is the initial value of $\omega$ and $p_m$ respectively. Then the value of $\omega$ and $p_m$ in $T^{th}$ generation $\omega(T)$ and $p_m(T)$ are calculated by the following formulae:

$$\omega(0) = \omega(0)\exp(-T/\alpha_1)$$ and $$p_m(T) = p_m(0) \exp(-T/\alpha_2),$$ where $\alpha_1$ is calculated so that the final value of $\omega$ is small (0.2 in our case). So

$$\alpha_1 = \frac{\text{Maxgen}}{\log \left[ \frac{\omega(0)}{0.2} \right]}.$$

where Maxgen is the expected number of generations that the GA can run for convergence. Similarly, $\alpha_2$ is calculated so that the final value of $p_m$ is small enough ($10^{-2}$ in our case). So

$$\alpha_2 = \frac{\text{Maxgen}}{\log \left[ \frac{p_m(0)}{10^{-2}} \right]}.$$ 

i Inclusion of particles in $S(t)$:

- Following Chen and Zhao (2009) maximum number of offsprings to be included in $S(t)$ is $N_i = N(t) \times \lambda \times \text{Div}(S(t))$, where $N(t)$ represents the current size of $S(t)$, $\text{Div}(S(t))$ is the diversity of $S(t)$ and $\lambda$ is adding factor. The common value of $\lambda$ belongs to the interval $[0.2, 0.3]$. 

A fuzzy lifetime-based PSO with varying swarm size

• Particles in the offspring set are at first ranked using q-tournament process (Chen and Zhao, 2009).
• Particles in the offspring set are arranged in descending order of their rank.
• First \( N_t \) particles are included in \( S(t) \). After inclusion, excess particles are removed if the number of particles in \( S(t) \) exceeds Maxsize.

(j) Deletion of particles from \( S(t) \):
• Again, following Chen and Zhao (2009), maximum number of offsprings to be deleted from \( S(t) \) is \( N_D = N(t) \times \eta \times \text{Div}(S(t)) \), where \( N(t) \) is the current size of \( S(t) \), \( \text{Div}(S(t)) \) is the diversity of \( S(t) \) and \( \eta \) is deletion factor, whose value is less than \( \lambda \). The common value belongs to the interval \([0.1, 0.2]\).
• Particles in \( S(t) \) are at first ranked using the q-tournament process (Chen and Zhao, 2009).
• \( S(t) \) is arranged in descending order of rank of particles.
• Last \( \min\{N_D, O(S(t)) - \text{Minsize}\} \) particles are deleted from \( S(t) \), so that after deletion, the number of particles in \( S(t) \) does not drops below Minsize.

k Termination condition: algorithm terminates when difference between maximum fitness (Maxfit) of particle, i.e., fitness of the best solution of the swarm and average fitness (Avgfit) of the swarm becomes negligible.

l Implementation: with the above function and values the algorithm is implemented using C-programming language in a personal computer consist of Intel 3.07 GHZ processor, 448 MB RAM and Microsoft Windows XP operating system.

m Convergence of the MPSO algorithm: a set of TFs are used to test the efficiency of the proposed algorithm MPSO. It is observed that results obtained by MPSO for different TFs are equal to the global optimum in most of the cases and in other cases solutions are all most near to global optima. This implies that this algorithm can be used as decision-making tool for different constrained/unconstrained optimisation problems.

Figure 3 Membership functions of age intervals
Figure 4  Membership functions of crossover probabilities

Table 1  Fuzzy rule-base for crossover probability

<table>
<thead>
<tr>
<th>Parent 2</th>
<th>Parent 1</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Young</td>
<td>Young</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Middle-age</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Old</td>
<td>Old</td>
<td>Low</td>
<td>Medium</td>
</tr>
</tbody>
</table>

4.2 MPSO for fuzzy objective function

A general single-objective mathematical programming problem with fuzzy objective function can be written as:

Maximize $f(X, \xi)$, subject to $g_i(X) \leq 0, \quad i = 1, 2, \ldots, m,$ (29)

where $X = (x_1, x_2, \ldots, x_n)$ is a decision vector, $\xi$ is a vector of fuzzy parameters, $f(X, \xi)$ is the return function and $g(X)$ are constraints.

The above MPSO algorithm can be used to determine optimal decision of the above problem. To deal with such a problem at first a fuzzy goal ($\tilde{F}_g$) of the objective function ($\tilde{f}$) is determined using the above MPSO algorithm. Then to make decision of the fuzzy optimisation problem, possibility/necessity (optimistic/pessimistic sense) (cf., Appendix A) of the fuzzy objective ($\tilde{f}$) with respect to the fuzzy goal ($\tilde{F}_g$) is taken as fitness of a particle (solution) of the swarm. The process is discussed below. Other functions and values are same as discussed above.

- Determination of fuzzy goal: fuzzy goal $\tilde{F}_g$ of the fuzzy objective function $\tilde{f}(X, \xi)$ is considered as a linear fuzzy number (LFN) ($F_{g1}, F_{g2}$) and values of $F_{g1}, F_{g2}$ can be determined in different ways. Here the following formulae are proposed and are used for numerical illustrations for the fuzzy models. In the formula $S$ denotes the feasible search space of the problem.

$$F_{g1} = \inf_{x_0 \in S} \inf_{\xi_0 \in \xi} \tilde{f}(x_0, \xi_0), \quad F_{g2} = \sup_{x_0 \in S} \sup_{\xi_0 \in \xi} \tilde{f}(x_0, \xi_0)$$

- Fitness evaluation: for optimistic decision maker (DM),
A fuzzy lifetime-based PSO with varying swarm size

\[ \prod \hat{f}_{j(X, \xi)}(\tilde{F}_g) \]

can be taken as fitness of a solution \( X \) and for pessimistic DM, \( \hat{N}_{j(X, \xi)}(\tilde{F}_g) \) can be taken as fitness. If analytical form of membership function of \( \hat{f}(X, \tilde{\xi}) \) (obtained by the formula defined in equation (34) is a TFN \( (F_1(X), F_2(X), F_3(X)) \), then Lemma 2 gives

\[ \prod \hat{f}_{j(X, \xi)}(\tilde{F}_g) = U_1 / E_1, \]

where \( U_1 = F_3(X) - F_{g1} \) and \( E_1 = F_3(X) - F_2(X) + F_{g2} - F_{g1} \). As MPSO search plays a role to maximise the fitness of the chromosomes in the swarm, so maximisation of

\[ \prod \hat{f}_{j(X, \xi)}(\tilde{F}_g) \]

implies maximisation of \( F_2(X) \) (most feasible equivalent of \( \hat{f}(X, \tilde{\xi}) \) and \( F_3(X) \) (least feasible equivalent of \( \hat{f}(X, \tilde{\xi}) \) together. Again,

\[ \prod \hat{f}_{j(X, \xi)}(\tilde{F}_g) = 1 \]

implies \( F_2(x) \geq F_{g2} \), i.e., most feasible objective value achieves the highest level of profit goal \( (F_{g2}) \). Thus if DM is optimistic and allows some risk, he/she will find the fitness of a solution depending on the possibility measure. On the other hand, Lemma 3 gives \( \hat{N}_{j(X, \xi)}(\tilde{F}_g) = U_2 / E_2, \) where \( U_2 = F_2(X) - F_{g1} \) and \( E_2 = F_2(X) - F_1(X) + F_{g2} - F_{g1} \). So in this case maximisation of \( \hat{N}_{j(X, \xi)}(\tilde{F}_g) \) implies maximisation of \( F_1(X) \) (worst possible equivalent of \( \hat{f}(X, \tilde{\xi}) \) and \( F_2(X) \) most feasible equivalent of \( \hat{f}(X, \tilde{\xi}) \) together. \( \hat{N}_{j(X, \xi)}(\tilde{F}_g) = 1 \) implies \( F_1(X) \geq F_{g2} \), i.e., worst possible objective value achieves the highest level of profit goal \( (F_{g2}) \). If DM is pessimistic and does not allow any risk, he/she will determine the fitness of solution using necessity measure approach. Set \( Z(X) \) as fitness of \( X \).

5 Numerical illustration

5.1 Testing of algorithm

Here, different models are solved using the proposed MPSO algorithm. Different parameter values of the MPSO algorithm used for this purpose are given below:

Number of variables in a solution \( n = 5 \). A five dimensional real vector \( X = (x_1, x_2, x_3, x_4, x_5) \) is used to represent a solution. In which \( x_1 \) represents \( T \), \( x_2 \) represents \( N \), \( x_3 \) represents \( M \), \( x_4 \) represents \( m_1 \) and \( x_5 \) represents \( m_2 \). Integral parts of \( x_2 \) and \( x_3 \) are taken as values of \( N \) and \( M \) respectively. Minimum swarm size \( \text{Minsize} = 10 \), Maximum swarm size \( \text{Maxsize} = 100 \). Initial value of probability of mutation \( p_m(0) = 0.9 \) and inertia weight \( w(0) = 0.9 \). Lower limit of inertia weight=0.2. Value of ‘q’ in q-tournament method is 5. Expected value of maximum number of generation to converge the MPSO, Maxgen =
200. MPSO algorithm is run for the TFs (cf., Appendix C) using different seeds of random number generators for 40 times each and number of success of finding optimal solutions for each function are noted. If optimal solution is found in a run of the heuristic algorithm using a seed, we say that the run is successful. % of successful runs for different TFs due to that technique is listed in Table 2. It is observed that the performance of MPSO is good enough.

Table 2  Results of TFs following the proposed MPSO algorithm

<table>
<thead>
<tr>
<th>TF</th>
<th>Results obtained</th>
<th>Maximum iterations required for convergence</th>
<th>% of success for 40 runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF1</td>
<td>ES(3.1432, 3.1438) = −0.9998</td>
<td>164</td>
<td>96%</td>
</tr>
<tr>
<td>TF2</td>
<td>MZ(2.2025, 1.5744) = −1.8004</td>
<td>180</td>
<td>92%</td>
</tr>
<tr>
<td>TF3</td>
<td>F(1.0001, 0.9998) = 1.0002</td>
<td>159</td>
<td>98%</td>
</tr>
<tr>
<td>TF4</td>
<td>F2(0.0481, 0.0001) = −205.8479</td>
<td>176</td>
<td>98%</td>
</tr>
<tr>
<td>TF5</td>
<td>SH(x_1) = −186.7306</td>
<td>176</td>
<td>94%</td>
</tr>
<tr>
<td>TF6</td>
<td>RC(x_2) = 0.39773</td>
<td>98</td>
<td>96%</td>
</tr>
<tr>
<td>TF7</td>
<td>F_7(1.0013, 0.9998) = 0.0</td>
<td>178</td>
<td>94%</td>
</tr>
<tr>
<td>TF7</td>
<td>F_7(0.9996, 0.9998, 0.9997, 0.9996) = 0.0</td>
<td>Does not converge for all runs</td>
<td>90%</td>
</tr>
<tr>
<td>TF8</td>
<td>BH(0.0000, 0.0001) = 0.0</td>
<td>125</td>
<td>98%</td>
</tr>
<tr>
<td>TF9</td>
<td>F(0.5, 0.2499) = 0.25</td>
<td>102</td>
<td>98%</td>
</tr>
<tr>
<td>TF10</td>
<td>Z_7(0.0000, 0.0001, 0.0001) = 0.0</td>
<td>144</td>
<td>98%</td>
</tr>
</tbody>
</table>

Notes: For TF-5, 18 global minima are x_1 = [(-0.8004, -1.4262), (-1.4252, -0.8003), (4.8584, -7.0835), (-7.0836, 4.8579), (-7.7083, -7.0835), (-7.0839, -7.7083), (5.4826, 4.8582), (4.8581, 5.4828), (-1.4254, -7.0839), (-7.0835, -1.4253), (4.8583, -0.8001), (-0.8005, 4.8583) (5.4827, -7.7082), (-7.7086, 5.4830), (-7.7085, -0.8001), (-0.8001, -7.7086), (5.4821, -1.4250), (-1.4252, 5.4823)]. For TF-6, three global minima are x_2 = [(-3.1423, 12.2504), (3.1431, 2.2517), (9.4253, 2.2474)].

Figure 5  \( p_m(0) \) vs. % of success (see online version for colours)
A fuzzy lifetime-based PSO with varying swarm size

Performance of the proposed MPSO algorithm is improved by setting the appropriate values of the parameters of the algorithm against the above mentioned TFs (Here, TF-1 to TF-5 are taken). For different values of initial probability of mutation $p_m(0)$ and threshold $E_T$, performance of the proposed heuristic algorithm for these TFs are evaluated and plotted graphically (cf., Figure 5 and Figure 6). Figure 5 and Figure 6 reflected that the % of success of the MPSO is good enough for $p_m(0) = 0.9$ and $E_T = 0.12$ taking reasonable number of iterations for convergence. This study is made to fix-up the values of $p_m(0)$ and $E_T$ for which optimum results can be found in most of the runs of the algorithm and numerical experiments of the proposed models are made with these values of $p_m(0)$ and $E_T$.

5.2 Numerical illustration of the proposed model

5.2.1 Results of the model with crisp objective function

The following parameter values are used to illustrate the model:

\[
K = 2,500, Q_b = 1,000, d = 2,000, B = 0.25, C = 500, R = 0.7, c_r = 1.5,
L_0 = 0.5, L_1 = 1.25, \alpha = 0.90, \beta_1 = 0.4, c_0 = 30, c_1 = 35, \beta_2 = 0.5, \gamma = 2.5,
\]

\[
m_H = 20, \sigma_H = 1 \text{ and upper limit of } m_2 \text{ is } 2.
\]

As discussed earlier, to compare the results, the present crisp model (Section 3.4) is solved following the two algorithms, i.e., MPSO and GA (presented by Guchhait et al., 2010). For the above parametric values, the optimum cycle length (T), the total number of cycles (N), the number of cycles for which price discount is offered (M), the mark-ups of selling price ($m_1$ and $m_2$) and the profit (Z) are obtained by both the methods and presented. Obtained results for different values of ‘$\alpha$’ (degree of probability of satisfying the constraint on random planning horizon) are presented in Table 3.
As expected, profit decreases with increase of \( \alpha \). In fact, increase of \( \alpha \) decreases the effective length of planning horizon, which in turn decreases the profit. Also, it is observed that \( m_1 \) increases with \( \alpha \) to keep the profit high. In fact, increase of \( m_1 \) decreases the demand which decreases the profit, but \( m_1 \) directly increases the profit. As a result, though increase of \( m_1 \) decreases demand, increase in profit due to increase in \( m_1 \) dominated decrease in profit due to decrease in demand. So, the resultant effect keeps the total profit high. Again, from this table it is observed that after the certain increasing value of \( \alpha \), the number of cycles for price discount is more than the previous one, which in turn decreases the profit also. Similar trends of the results are observed for both the algorithms, but, the profit obtained by MPSO is better than that of GA. Of course, these better results do not contribute much in real sense for the present set of parametric values of the model. The differences are in the decimal places. As our problem is to maximise the profit, so we can say that the proposed algorithm performed better compared to the GA.

Again, for the above parameter values, results are also obtained by both the algorithms MPSO and GA for different values of \( R \) and presented in Table 4. It is observed that the profit decreases with \( R \). It happens because increase of \( R \) decreases the effect of price discount on demand, i.e., demand decreases with the increase of \( R \). As demand of the item decreases; the profit decreases automatically. It is also observed from the Table 4 that, to keep the demand high (with the increase of \( R \)), either \( m_1 \) decreases or \( M \) increases. Because, increasing demand reflects more profit. As the proposed model is to maximise the profit so, all these observations agree with reality. Here also, it is observed that the profit obtained by MPSO is more than that of GA.
5.2.2 Results of the model with fuzzy objective function

To illustrate the fuzzy model, the setup cost coefficients \( c_0, c_1 \) and the holding cost \( c_h \) are considered as TFNs, i.e., \( c_0 = (30, 35, 40), c_1 = (35, 40, 45) \) and \( c_h = (0.45, 0.5, 0.55) \) respectively. Other parameter values are same as in the crisp model. From the previous results (i.e., the results of crisp model as well as TFs) it can be concluded that the MPSO algorithm performed better compared to the other. For this reason the fuzzy model is solved using only MPSO algorithm. For the assumed parameter values, calculated value of \( F_{g1} = 447 \) and \( F_{g2} = 650.16 \). The results are obtained by MPSO algorithm for the fuzzy model (25) using both the measure of fuzzy number (i.e., possibility and necessity measures) and presented in Tables 5 and 6 respectively. It is observed from these tables (Tables 5 and 6) that maximum possible profit due to possibility measure \( (Z_3 = 649.25) \) is more than that of the maximum profit due to necessity measure \( (Z_3 = 647.65) \). But, there is some risk in possibility measure approach. Because, in that case minimum assured profit \( (Z_1 = 444.01) \) is less than that of the obtained following necessity measure \( (Z_1 = 445.29) \). It happens because possibility measure approach is followed by optimistic DMs. Using this approach actually \( Z_2 \) and \( Z_3 \) are optimised [cf., §4.1(2)]. On the other hand necessity measure approach is followed by pessimistic DMs. Using this approach, actually \( Z_1 \) and \( Z_2 \) are optimised [cf., §4.1(2)]. As a result, \( Z_3 \) in necessity measure approach may be less compared to the value of \( Z_3 \) in possibility measure approach. In this case, minimum possible profit (i.e., \( Z_1 \) in necessity measure) is optimised, so DM is free from risk.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( T )</th>
<th>( N )</th>
<th>( M )</th>
<th>( m_1 )</th>
<th>( m_2 )</th>
<th>( Z_1 )</th>
<th>( Z_2 )</th>
<th>( Z_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPSO</td>
<td>2.0794</td>
<td>9</td>
<td>6</td>
<td>1.6774</td>
<td>2</td>
<td>444.01</td>
<td>546.63</td>
<td>649.25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( T )</th>
<th>( N )</th>
<th>( M )</th>
<th>( m_1 )</th>
<th>( m_2 )</th>
<th>( Z_1 )</th>
<th>( Z_2 )</th>
<th>( Z_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPSO</td>
<td>2.0794</td>
<td>9</td>
<td>5</td>
<td>1.6627</td>
<td>2</td>
<td>445.29</td>
<td>546.47</td>
<td>647.65</td>
</tr>
</tbody>
</table>

6 Managerial implications and insights

In the present investigation, several cases of the models with the credit period are formulated with respect to retailer and presented. Manager of a retail shop can take the managerial decisions depending upon the actual prevailing situation which fits best with the given models for maximum profit.

1. This model suggests to a manager of a sector how to determine the number of cycles for which price discount is offered so that the profit can be maximised.

2. Results of the models with crisp and imprecise parameters are presented. Manager of a retail shop can choose their appropriate model depending upon the nature of parameters.
7 Conclusions

In this paper, a fuzzy lifetime-based MPSO algorithm is proposed which is efficient in solving constrained optimisation problem with crisp as well as fuzzy objective. The main features of the developed algorithm are summarised below:

- at the time of birth of a swarm, diversity is maintained using information entropy theory
- initial value of probability of mutation $p_m(0)$ and inertia weight $w(0)$ are taken very high and gradually decreases (towards a minimum limiting value) with increase of iteration counter
- crossover operations are made on the particles whose movements are not made using normal PSO functions
- moreover, if no movement of a particle is made during a finite number of generation (i.e., if age exceeds the lifetime), it is discarded from the swarm.

These realistic features make the algorithm more powerful in searching the global optima (if it exists). Performance of the MPSO algorithm (tested against a list of TFs) concludes that it is good enough. As the developed algorithm is more efficient, so that it can be used to solve different decision-making problems in different fields of science and technology. Here also, a production inventory model is developed incorporating the effect of stock and price on demand when price discount is offered to the customers for few cycles. Moreover, learning effects on production cost and setup cost are incorporated in this EPQ model. The aim of adaptation of the model is fourfold.

- study the effect of both the stock and price on demand when price discount is offered to the customers
- incorporate employees learning effects on production and set-up cost in a finite rate production model
- introduce lifetime of a product as random in nature in a EPQ model
- propose a MPSO algorithm which can deals with crisp as well as fuzzy objective function

Here, in the fuzzy model, production cost coefficients are not considered as fuzzy. It is due to the fact that if production cost is fuzzy then demand becomes fuzzy. If demand is fuzzy then cycle length will vary from cycle to cycle, as well as due to the impreciseness of demand, production quantities for different cycles may finish before the end of some cycles as well as may excess at the end of other cycles. So it is difficult to find optimal decision in that case following this approach. On the other hand, there are also some limitations in this algorithm which are given below.

- solutions obtained in this approach are normally near optimal, not exact
- near optimal solutions may not occur in some runs of the algorithm.

Further research work can be made on the optimisation problem with other environments like – rough, fuzzy-rough, fuzzy-random etc. Also, this model can be extended to a ‘multi-item inventory model’ by considering more than one item in the system. Again
due to generalised solution methodology, this approach can be applied to solve other inventory control problems along with the problems in other disciplines.

References


Appendix A

Let \( \tilde{a} \) and \( \tilde{b} \) be two fuzzy numbers with membership functions \( \mu_a(x) \) and \( \mu_b(x) \) respectively. Then according to Dubois and Prade (1980), Liu and Iwamura (1998) and Zadeh (1978),

\[
\text{pos}(\tilde{a} \ast \tilde{b}) = \sup \{ \min(\mu_a(x), \mu_b(y)) \}, x, y \in \mathbb{R} \}, x \ast y
\]

where the abbreviation \( \text{pos} \) represents possibility, \( \ast \) is any one of the relations \( >, <, =, \leq, \geq \) and \( \mathbb{R} \) represents set of real numbers.

\[
\text{nes}(\tilde{a} \ast \tilde{b}) = 1 - \text{pos}(\tilde{a} \ast \tilde{b})
\]

where the abbreviation \( \text{nes} \) represents necessity.

Similarly, possibility and necessity measures of \( \tilde{a} \) with respect to \( \tilde{b} \) are denoted by \( \Pi_{\mu_a}(\tilde{a}) \) and \( N_{\mu_b}(\tilde{a}) \) respectively and are defined as

\[
\Pi_{\mu_a}(\tilde{a}) = \sup \{ \min(\mu_a(x), \mu_b(x)) \}, x, y \in \mathbb{R} \}
\]

\[
N_{\mu_b}(\tilde{a}) = \min \{ \sup(\mu_a(x), 1 - \mu_b(x)) \}, x, y \in \mathbb{R} \}
\]

If \( \tilde{a}, \tilde{b} \subseteq \mathbb{R} \) and \( \tilde{c} = f(\tilde{a}, \tilde{b}) \) where \( f: \mathbb{R} \times \mathbb{R} \to \mathbb{R} \) is a binary operation then membership function \( \mu_{\tilde{c}} \) of \( \tilde{c} \) is defined as Dubois and Prade (1980)

For each \( z \in \mathbb{R} \), \( \mu_{\tilde{c}}(z) = \sup \{ \min(\mu_a(x), \mu_b(y)) \}, x, y \in \mathbb{R} \}, \text{ and } z = f(x, y) \)

**LFN:** a LFN \( \tilde{a} = (a_1, a_2) \) has two parameters \( a_1, a_2 \), where \( a_1 < a_2 \) and is characterised by the membership function \( \mu_a(x) \), given by

\[
\mu_a(x) = \begin{cases} 
0 & \text{for } x \leq a_1 \\
\frac{x - a_1}{a_2 - a_1} & \text{for } a_1 \leq x \leq a_2 \\
1 & \text{for } x \geq a_2 
\end{cases}
\]

**Triangular fuzzy number (TFN):** a TFN \( \tilde{a} = (a_1, a_2, a_3) \) has three parameters \( a_1, a_2, a_3 \), where \( a_1 < a_2 < a_3 \) and is characterised by the membership function \( \mu_a(x) \), given by

\[
\mu_a(x) = \begin{cases} 
\frac{x - a_1}{a_2 - a_1} & \text{for } a_1 \leq x \leq a_2 \\
\frac{a_3 - x}{a_3 - a_2} & \text{for } a_2 \leq x \leq a_3 \\
0 & \text{otherwise}
\end{cases}
\]

According to above definitions following lemmas can easily be derived.

**Lemma 1:** if \( \tilde{a} = (a_1, a_2, a_3) \) be a TFN with \( 0 < a_1 \) and \( b \) is a crisp number then
Lemma 2: if \( \tilde{a} = (a_1, a_2, a_3) \) be a TFN and \( \tilde{b} = (b_1, b_2) \) be a LFN with \( 0 < a_1 \) and \( 0 < b_1 \) then
\[
\Pi_{\tilde{a}}(\tilde{b}) = \begin{cases} 
1 & \text{if } a_2 \geq b_1 \\
\frac{a_1 - b_1}{a_1 - a_2 + b_2 - b_1} & \text{if } a_2 \leq b_2 \text{ and } a_1 > b_2 \\
0 & \text{otherwise}
\end{cases}
\]

Lemma 3: if \( \tilde{a} = (a_1, a_2, a_3) \) be a TFN and \( \tilde{b} = (b_1, b_2) \) be a LFN with \( 0 < a_1 \) and \( 0 < b_1 \) then
\[
N_{\tilde{a}}(\tilde{b}) = \begin{cases} 
1 & \text{if } a_1 \geq b_2 \\
\frac{a_2 - b_2}{a_2 - a_1 + b_2 - b_1} & \text{if } a_2 > b_1 \text{ and } b_2 > a_1 \\
0 & \text{otherwise}
\end{cases}
\]

Lemma 4: if \( \tilde{a} = (a_1, a_2, a_3) \) be a TFN and \( b \) be a crisp number with \( 0 < a_1 \) and \( 0 < b \),
\[
\Pi_{\tilde{a}}(b) = N_{\tilde{a}}(b) = \begin{cases} 
\frac{b - a_1}{a_2 - a_1} & \text{if } a_2 \geq b \geq a_1 \\
\frac{a_3 - b}{a_3 - a_2} & \text{if } a_1 \geq b \geq a_2 \\
0 & \text{otherwise}
\end{cases}
\]

Appendix B

Optimisation with stochastic constraints (Charnes and Cooper, 1959): let \( x = (x_1, x_2, \ldots, x_n)^T \) be the decision vector, \( y = (y_1, y_2, \ldots, y_N)^T \) be the vector of \( N \) random variables with known mean and standard deviation, where \( y_1, y_2, \ldots, y_N \) represents \( N \) parameters of the problem, then a stochastic nonlinear programming problem (SNLP) can be stated in standard form as follows:

Find
\[
(x_1, x_2, \ldots, x_n)^T
\]
which minimize/maximize \( f(x, y) \)
subject to
\[
P(\phi_r(x, y) \geq 0) \geq p_r \quad (r = 1, \ldots, m)
\]
where \( P(\phi_r(x, y) \geq 0) \) represents probability of the event \( \phi_r(x, y) \).

According to Charnes and Cooper (1959), if all \( y_i (i = 1, 2, \ldots, N) \) follow independent normal distribution, the stochastic problem stated above is equivalent to following crisp nonlinear programming problem.
Find \( x = (x_1, x_2, \ldots, x_n)^T \) which minimize/maximize \( f(x, \bar{y}) \)

subject to

\[
\eta_r - \nu_r \left[ \sum \left( \frac{\partial \phi_r (x, \bar{y})}{\partial y_i} \right)^2 \right]^{\frac{1}{2}} \geq 0 \quad (r = 1, \ldots, m)
\]

where \( \eta_r \) and \( \sigma_r \) are mean and standard deviation of \( \phi_r (x, y) \) respectively. \((38)\)

Also, \( \nu_r \) is given by \( \nu_r = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\xi} e^{-t^2/2} dt \)

Appendix C

Following are the list of TFs which are used to test the efficiency of the proposed MPSO algorithm. It is observed that results obtained by MPSO for different TFs are equal to the global optimum in most of the cases and in other cases solutions are very near to global optima. This implies that this algorithm can be used as decision-making tool for different constrained/unconstrained optimisation problems.

TF-1 (taken from Bessaou and Siarry, 2001)

\[ ES(x_1, x_2) = -\cos(x_1) \times \cos(x_2) \times \exp \left\{ -\left( \left( x_1 - \pi \right)^2 + \left( x_2 - \pi \right)^2 \right) \right\}, \]

\[ -10 \leq x_1, x_2 \leq 10. \]

This TF has one global minima at \((x_1, x_2) = (\pi, \pi)\) and \( ES(\pi, \pi) = -1. \)

TF-2 (taken from Bessaou and Siarry, 2001)

\[ MZ(x_1, x_2, \ldots, x_n) = -\sum \sin(x_i) ln(i(x_i)^2 / \pi)^{2m}, \quad -\pi \leq x_1, x_2, \ldots, x_n \leq \pi \]

where \( m = 10. \) For \( n = 2, \) it has one global minima at \((x_1, x_2) = (2.25, 1.57)\) and \( MZ(2.25, 1.57) = -1.80. \)

TF-3 (taken from Michalewicz, 1992):

Minimize \( F(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 1)^2, \)

such that \( -x_1^2 + x_2^2 \geq 0, \ x_1 + x_2 \leq 2, \ -5 \leq x_1, x_2 \leq 5. \)

It has one global minima at \((x_1, x_2) = (1, 1), \) and \( F(1, 1) = 1. \)

TF-4 (taken from Bessaou and Siarry, 2001):

\[ F2(x_1, x_2) = 100 \times (x_2^2 - x_1^2) + (1 - x_1) \times (x_1 - 1)^2, \quad -2.048 \leq x_1, x_2 \leq 2.048. \]

It has one minima at \((x_1, x_2) = (2.048, 0)\) and \( F2(2.048, 0) = -205.8480. \)
TF-5 (taken from Bessaou and Siarry, 2001):

\[ SH(x_1, x_2) = \sum_{j=1}^{5} j \times \cos \left( (j+1) \times x_1 + j \right) \times \sum_{j=1}^{5} j \times \cos \left( (j+1) \times x_2 + j \right), -10 \leq x_1, x_2 \leq 10 \]

This problem has 760 local minima and 18 global minima. At global minima \((x_1, x_2), SH(x_1, x_2) = -186.7309\).

TF-6 (taken from Bessaou and Siarry, 2001):

\[ RC(x_1, x_2) = \left\{ x_2 - \left[ 5/\left( 4 \times \pi^2 \right) \right] \cdot x_1^2 + \left[ 5/\pi \right] \times x_1 - 6 \right\}^2 + 10 \times \left\{ 1 - \left[ 1/\left( 8 \pi \right) \right] \right\} \times \cos (x_1) + 10, -5 \leq x_1 \leq 10, 0 \leq x_2 \leq 15 \]

This problem has three global minima at \((x_1, x_2) = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)\) and \(RC(x_1, x_2) = 0.397887\) at any one of these minima.

TF-7 (taken from Bessaou and Siarry, 2001)

Minimize \( F_n(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n-1} \left[ 100 \times (x_1^2 - x_{j+1})^2 + (1-x_j)^2 \right] \),

\[-1 \leq x_1, x_2, \ldots, x_n \leq 5.\]

Two functions \( F_2 \) and \( F_4 \) are used. This problems has one global minima at \((x_1, x_2, \ldots, x_n) = (1, 1, \ldots, 1) \) and \( F_n(1, 1, \ldots, n) = 0 \).

TF-8 (taken from Bessaou and Siarry, 2001):

\[ BH(x_1, x_2) = x_1^2 + 2 \times x_2^2 - 0.3 \times \cos (3\pi x_1) \times \cos (4\pi x_2) + 0.3, -5 \leq x_1, x_2 \leq 5 \]

This problem has one global minima at \((x_1, x_2) = (0, 0)\) and \( BH(0, 0) = 0 \).

TF-9 (taken from Michalewicz, 1992):

Minimize \( F(x_1, x_2) = 100 \times (x_2 - x_1^2)^2 + (x_1 - 1), \)

such that \( x_1 + x_2^2 \geq 0, x_1^2 + x_2 \geq 0, \quad -0.5 \leq x_1 \leq 0.5, -1.0 \leq x_2 \leq 1.0 \)

It has one global minima at \((x_1, x_2) = (0.5, 0.25)\), and \( F(0.5, 0.25) = 0.25 \).

TF-10 (taken from Bessaou and Siarry, 2001):

\[ Z_n(x_1, x_2, \ldots, x_n) = \left[ \sum_{j=1}^{n} x_j^2 \right] + \left[ \sum_{j=1}^{n} 0.5 \times x_j \right]^2 + \left[ \sum_{j=1}^{n} 0.5 \times j \times x_j \right]^4, \]

\[-5 \leq x_1, x_2, \ldots, x_n \leq 5 \]

It has one global minima at \((x_1, x_2, \ldots, x_n) = (0, 0, \ldots, 0)\) and \( Z_n(0, 0, \ldots, 0) = 0 \).