

# Hybrid particle swarm optimization and differential evolution algorithm for bi-level programming problem and its application to pricing and lot-sizing decisions

Weimin Ma · Miaomiao Wang · Xiaoxi Zhu

Received: 2 December 2012 / Accepted: 4 June 2013 / Published online: 22 June 2013  
© Springer Science+Business Media New York 2013

**Abstract** This paper proposes a hierarchical hybrid particle swarm optimization (PSO) and differential evolution (DE) based algorithm (HHPNODE) to deal with bi-level programming problem (BLPP). To overcome the shortcomings of basic PSO and basic DE, this paper improves PSO and DE, respectively by using a velocity and position modulation method in PSO and a modified mutation strategy in DE. HHPNODE employs the modified PSO as a main program and the modified DE as a subprogram. According to the interactive iterations of modified PSO and DE, HHPNODE is independent of some restrictive conditions of BLPP. The results based on eight typical bi-level problems demonstrate that the proposed algorithm HHPNODE exhibits a better performance than other algorithms. HHPNODE is then adopted to solve a bi-level pricing and lot-sizing model proposed in this paper, and the data is used to analyze the features of the proposed bi-level model. Further tests based on the proposed bi-level model also exhibit good performance of HHPNODE in dealing with BLPP.

**Keywords** Bi-level programming problem · Hierarchical · Particle swarm optimization · Differential evolution · Pricing and lot-sizing

## Introduction

Bi-level programming techniques aim to deal with decision problems involving two decision makers with a hierarchical structure (Calvete and Galé 2011). Both of these twodecision

makers seek to optimize their individual objective functions and control their own set of decision variables. The leader at the upper level of the hierarchy first specifies a strategy, and then the follower at the lower level of the hierarchy specifies a strategy so as to optimize the objective functions with full knowledge of the action of the leader. Bi-level programming techniques have been successfully applied to many different areas (Gao et al. 2011), such as mechanics, decentralized resource planning, electric power markets, logistics, civil engineering, and road network management and so on.

Bi-level programming problem (BLPP) has been proved to be a NP-hard problem (Jeroslow 1985) which is very difficult to solve. However, the bi-level programming is used so widely that there are many researchers focusing on BLPP, and several different kinds of methods have been developed to solve this problem. The main methods for solving linear bi-level programming problems can be divided into the following four categories (Hejazia et al. 2002): Methods based on vertex enumeration; methods based on Kuhn–Tucker conditions; fuzzy approach and methods based on meta-heuristics, such as genetic algorithm based approaches, simulated annealing based approaches, and so on. It needs to be pointed out that methods based on vertex enumeration and Kuhn–Tucker conditions have some limitations for solving BLPPs as they rely on the differentiability of the objective function, the convexity of search space, etc. Methods based on meta-heuristics, however, are independent of these limiting conditions and are suitable for solving highly complex nonlinear problems which traditional search algorithms cannot solve. In reality, most BLPPs are not merely linear problems. There are much more complex bi-level programming problems: nonlinear bi-level decision problems, multi-leader bi-level decision problems, multi-follower decision bi-level problems, multi-objective bi-level decision problems, fuzzy bi-level decision problems (Gao et al. 2011), and so on. Thus,

---

W. Ma · M. Wang (✉) · X. Zhu  
School of Economics and Management, Tongji University,  
Siping Road 1239, Shanghai, P. R. China  
e-mail: wangmiaomiaocome@163.com

it is important to develop effective and efficient methods to solve these problems.

Because of their inherent merits, meta heuristics have been widely applied to solve BLPPs. Genetic algorithms have been developed in Calvete et al. (2008), Hejazia et al. (2002), Li et al. (2010), tabu search is applied in Lukač et al. (2008), Rajesh et al. (2003), Wen and Huang (1996), simulated annealing is applied in Sahin and Ciric (1998) and neural network approach is proposed in Lan et al. (2007), Shih et al. (2004). Bi-level differential evolution algorithms are developed in Vincent et al. (2012). Particle swarm optimization algorithm is developed for solving bi-level programming problem in Gao et al. (2011), Kuo and Huang (2009), Li et al. (2006).

Supply chain is a complex hierarchical system, which contains many different interest participants, such as suppliers, manufacturers, distributors, retailers and end customers. These members compose a long and hierarchical chain. Each of the members in the chain independently controls a set of decision variables, disjoint from the others, and need to make decisions based on their own interests, while still considering the choice of the others, as the others' decisions will have an influence on their own interests. Therefore, it is especially suitable to adopt BLPP to deal with supply chain management (SCM) problems. As two of the most important problems in SCM, the decision of pricing and lot-sizing play important roles in optimizing profits in a supply chain. There are many studies that have focused on pricing and lot-sizing problems (Abad 2003; Guan and Liu 2010; Kébé et al. 2012; Li and Meissner 2011; Lu and Qi 2011; Raa and Aghezzaf 2005; Yildirmaz et al. 2009). The literature considers pricing and lot-sizing problems by using the traditional modeling methods, which may neglect the fact that a supply chain is a hierarchical system, and there is little literature on pricing and lot-sizing policies based on BLPP. Marcotte et al. (2009) considered the characterization of optimal strategies for a service firm acting in an oligopolistic environment, and the decision problem is formulated as a leader-follower game played on a transportation network. Dewez et al. (2008) and Gao et al. (2011) also established bi-level models for pricing problems.

This paper proposes a hierarchical hybrid particle swarm optimization (PSO) and differential evolution (DE) based algorithm (HHPSODE) to deal with BLPP. Unlike most problem-dependent algorithms designed for specific versions or based on specific assumptions, the proposed algorithm HHPSODE is a hierarchical algorithm, which solves BLPP iteratively by a modified PSO and a modified DE. On the one hand, in each iterative process of PSO, the particles sometimes may move too far away from the problem's feasible region, and this may lead to a local optimal solution. In the modified PSO, a velocity and position modulation method is applied to the movement of particles in order to guide them

within the region of interest, which can help the PSO maintain a faster convergence speed and global convergence. On the other hand, to overcome the shortcomings of basic DE, this paper adopts a modified mutation strategy. Combining the advantages of DE/rand and DE/best, the modified mutation strategy has the properties of fast convergence property and not falling easily into premature convergence. According to the interactive iterations of the modified PSO algorithm and DE algorithm, HHPSODE can solve the BLPP without any specific assumption or any transformation of the objective or constraints functions. Then eight benchmark bi-level problems from the related literature are employed to test the performance of the proposed algorithm HHPSODE, and the results demonstrate that the proposed algorithm HHPSODE shows better performance than other algorithms.

Because there is little literature on pricing and lot-sizing policies based on BLPP, this paper establishes a bi-level pricing and lot-sizing model. Then HHPSODE is employed to solve the bi-level model, and the features of the model are analyzed based on the data solved by HHPSODE. Finally, based on the proposed bi-level model, we do a further test to illustrate the performance of HHPSODE, and the results also exhibit good performance of HHPSODE in dealing with BLPP.

Following the introduction in "Introduction" section, the rest of this paper is organized as follows. "Background" section briefly describes the basic concept of BLPP, PSO and DE. "Algorithm design" section develops a hybrid algorithm HHPSODE for solving BLPP. Numerical experiments are shown in "Numerical experiments" section to demonstrate the performance of HHPSODE. "An application on pricing and lot-sizing decisions" section gives an application of BLPP on SCM and also analyzes the features of the proposed bi-level model. Finally, the conclusions are given in "Conclusion" section.

## Background

### Basic model of BLPP

In the BLPP, two decision makers are involved. The lower level decision maker optimizes his objective function under the given parameters from the upper level decision maker. General model of BLPP can be formulated as follows:

$$\begin{aligned} & \min_{x \in X} f_1(x, y) \\ & s.t. \quad G(x, y) \leq 0 \\ & \text{where, for given } x, \text{ the vector } y \text{ solves} \\ & \min_{y \in Y} f_2(x, y) \\ & s.t. \quad g(x, y) \leq 0 \end{aligned} \quad (1)$$

where,  $x \in X \subset R^{n1}$ ,  $y \in Y \subset R^{n2}$ .  $x$  is the vector of variables controlled by the leader (upper level variables), and  $y$  is the vector of variables controlled by the follower (lower level variables).  $f_1(x, y)$  and  $f_2(x, y)$  are the leader’s and the follower’s objective functions, respectively.  $G(x, y) \leq 0$  and  $g(x, y) \leq 0$  are corresponding constraints of the upper and lower level problems.

In formal terms, the bi-level programming problems are mathematical programs in which the subset of variables  $y$  is required to be an optimal solution of another mathematical program parameterized by the remaining variables  $x$ .

Basic PSO

As a typical model of swarm intelligence (SI) and a new evolutionary algorithm based on SI, PSO has gained wide attention among SI researchers since it was proposed by Kennedy and Eberhart (1995). Because PSO is easy to implement with only a few parameters to adjust, it has been applied successfully to many different areas, such as combinatorial problems (Belmecheri et al. 2012), electric power systems (Sadeghierad et al. 2010), neural network systems (Gaitonde and Karnik 2012), fuzzy systems control (Bingül and Karahan 2011), industry fields (Chu and Hsieh 2012) and many other various applications (Chan and Tiwari 2007).

Particle swarm optimization consists of a swarm of particles without quality and volume. Each particle represents a candidate solution. In each iteration, the particle will track two extreme values: One is the best solution of each particle gained so far, which represents the each particle’s cognition level; the other is the overall best solution gained by any particle in the population so far, which represents society’s cognition level. PSO supposes that there are  $N$  particles in the  $D$  dimensions.  $x_i = (x_{i1}, x_{i2}, \dots, x_{iD})$  and  $v_i = (v_{i1}, v_{i2}, \dots, v_{iD})$  represent the particle’s position and velocity, respectively.  $p_i = (p_{i1}, p_{i2}, \dots, p_{iD})$  denotes the best position that the particle has visited.  $p_g = (p_{g1}, p_{g2}, \dots, p_{gD})$  denotes the best position that the swarm has visited. The particles are manipulated according to the equations below:

$$v_{id}(t + 1) = wv_{id}(t) + c_1r_1(p_{id}(t) - x_{id}(t)) + c_2r_2(p_{gd}(t) - x_{id}(t)) \tag{2}$$

$$x_{id}(t + 1) = x_{id}(t) + v_{id}(t + 1) \tag{3}$$

where,  $1 \leq d \leq D$ ,  $1 \leq i \leq N$ ,  $c_1$  and  $c_2$  are non-negative constants, which are called the cognitive and social parameter.  $r_1$  and  $r_2$  are two random numbers, which are uniformly distributed in the range (0,1).  $w$  denotes the inertia weight, which plays an important role in balancing global search ability and local search ability of PSO.

Shi and Eberhart (1999) proposed the linearly decreasing strategy as follows:

$$w(t) = w_{\min} + (w_{\max} - w_{\min}) \cdot \left( \frac{T_{\max} - t}{T_{\max}} \right) \tag{4}$$

where, the superscript  $t$  denotes the  $t$ th iteration, and  $T_{\max}$  denotes the iteration’s maximum number.  $w_{\min}$  and  $w_{\max}$ , respectively denote the minimum and maximum of original inertia weight.

Basic DE

Differential evolution algorithm, proposed by Storn and Price (1997), has become a popular and effective algorithm for its many attractive characteristics, such as compact structure, ease of use, fast convergence speed and robustness. Thanks to these features, DE has been applied to solve problems in many scientific and engineering fields (Plagianakos et al. 2008), such as technical system design (Storn 1999), scheduling problem (Vincent et al. 2012), pattern recognition (Ilonen et al. 2003) and so on.

Assume that the population of the standard DE algorithm contains  $ND$ -dimensional vectors. Then the  $i$ th individual in the population could be presented with a  $D$ -dimensional vector. The basic operators in classical DE include the following five steps (Price et al. 2005):

1. Initialization: Input the population size  $N$ , the scaling factor  $F \in [0, 2]$  and the crossover rate  $CR \in [0, 1]$ . Then individuals in first generation are generated randomly:
2. Evaluation: For each individual  $y_i(t)$ , evaluate its fitness value  $fit(y_i(t))$ .
3. Mutation: The most frequently used mutation strategies implemented in the basic DE are (Zhao et al. 2011):

$$\begin{aligned} \text{DE/rand/1} : v_{yid}(t) &= y_{r1d}(t) + F \cdot (y_{r2d}(t) - y_{r3d}(t)) \\ \text{DE/best/1} : v_{yid}(t) &= y_{bestd}(t) + F \cdot (y_{r1d}(t) - y_{r2d}(t)) \\ \text{DE/rand - to - best/1} : v_{yid}(t) &= y_{r1d}(t) + F \cdot (y_{bestd}(t) - y_{r2d}(t)) + F \cdot (y_{r3d}(t) - y_{r4d}(t)) \\ \text{DE/best/2} : v_{yid}(t) &= y_{bestd}(t) + F \cdot (y_{r1d}(t) - y_{r2d}(t)) + F \cdot (y_{r3d}(t) - y_{r4d}(t)) \\ \text{DE/rand/2} : v_{yid}(t) &= y_{r1d}(t) + F \cdot (y_{r2d}(t) - y_{r3d}(t)) + F \cdot (y_{r4d}(t) - y_{r5d}(t)) \end{aligned}$$

where,  $d = 1, 2, \dots, D$ . The indices  $r_1, r_2, r_3, r_4, r_5 \in \{1, 2, \dots, N\}$  are mutually exclusive integers randomly generated, which are also different from the index  $i$ .

4. Crossover: Crossover operator is designed to increase the diversity of the population. First, an integer  $d_{rand} \in \{1, 2, \dots, D\}$  is generated randomly. Then, we have the trail vector  $u_{yi}(t) = (u_{yi1}(t), u_{yi2}(t), \dots, u_{yiD}(t))$  by the following equation:

$$u_{yi}(t) = \begin{cases} v_{yid}(t), & \text{if } \text{rand}[0, 1] \leq CR \text{ or } d = d_{rand} \\ y_{id}(t), & \text{otherwise} \end{cases} \quad (5)$$

$$d = 1, 2, \dots, D$$

5. Selection: Selection operates by comparing the individuals' fitness value to generate the next generation population:

$$y_i(t+1) = \begin{cases} y_i(t), & \text{if } \text{fit}(y_i(t)) < \text{fit}(u_{yi}(t)) \\ u_{yi}(t), & \text{otherwise} \end{cases} \quad (6)$$

### Algorithm design

Similar to other intelligent algorithms, both basic PSO and DE have some shortcomings: stagnation and premature convergence. Therefore, one of the most important topics is to design more effective search strategies to enhance the performance of basic PSO and DE. Many researchers have made many attempts to solve these problems. For PSO, [Janson and Middendorf \(2005\)](#) arranged the particles in a dynamic hierarchy according to the quality of their so-far best-found solution, and the result was improved performance on most of the benchmark problems considered. [García-Nieto and Alba \(2011\)](#) proposed a velocity modulation method and a restarting mechanism to enhance scalability of PSO. [Belmecheri et al. \(2012\)](#) proposed a PSO with a local search strategy, and the approach has shown its effectiveness on several combinatorial problems. For DE, [Zhao et al. \(2011\)](#) proposed an improved self-adaptive DE to solve large-scale continuous optimization problems. [Vincent et al. \(2012\)](#) developed different variants of the bi-level differential evolution (BiDE) algorithms. [Cai et al. \(2012\)](#) proposed a learning-enhanced DE (LeDE) that encourages individuals to exchange information systematically.

In this section, we introduce a modified velocity and position modulation of PSO and a modified mutation strategy of DE to improve performance of basic PSO and DE. Then, a constraint handling mechanism is presented. Lastly, the framework of HHPSODE for solving BLPP is given. Note that all the pseudo-codes used in the following algorithms are designed for the minimization problem.

#### A modified PSO

This section employs a velocity and position modulation method to guide the particles within the region of interest ([García-Nieto and Alba 2011](#)). In each iterative process of

PSO, the particles sometimes move too far away from the problem's feasible region, and this may lead to a local optimal solution. To enhance scalability and ensure global convergence of PSO, this paper restricts the particle's movement to the feasible region of the problem by using velocity and position modulation as shown in Algorithm 1.

---

#### Algorithm 1 Pseudo-code of velocity and position modulation procedure

---

1. Calculate the new velocity value  $v_{idnew}$  according to the following equations:

$$v_{idnew} = w(t)v_{sid}(t) + c_1r_1(p_{si}(t) - x_{id}(t)) + c_2r_2(p_{sg}(t) - x_{id}(t))$$

$$w(t) = w_{\min} + (w_{\max} - w_{\min}) \cdot \left( \frac{T_{\max} - t}{T_{\max}} \right)$$

2. if  $v_{idnew} \leq x_{dlow}$  then  
 3.  $v_{id}(t+1) = x_{dlow}$   
 4. else if  $v_{idnew} \geq x_{dupp}$  then  
 5.  $v_{id}(t+1) = x_{dupp}$   
 6. end if  
 7.  $x_{idnew} = x_{id}(t) + v_{id}(t+1)$   
 8. if  $x_{dlow} \leq x_{idnew} \leq x_{dupp}$  then  
 9.  $x_{id}(t+1) = x_{idnew}$   
 10. else  
 11.  $x_{id}(t+1) = x_{id}(t) - v_{id}(t+1)$   
 12. end if
- 

In Algorithm 1,  $x_{dlow}$  and  $x_{dupp}$  denote the  $d$ th dimension lower and upper bound of the decision variable  $x$ .  $v_{idnew}$  and  $x_{idnew}$  denote the new velocity value and the new position of  $i$ th particle in  $d$ th dimension.

The procedure can be explained as follows:

After calculating the new velocity value  $v_{idnew}$  according to Eqs. (2) and (4) in "Basic PSO" section, the velocity vector magnitude is bounded (line 2–6), which prevents the given particle from moving far from the area of interest. Then, once the new velocity  $v_{id}(t+1)$  is obtained, the new particle position  $x_{idnew}$  is calculated according to Eq. (3) in "Basic PSO" section, and the new position  $x_{idnew}$  should not exceed the problem limits ( $x_{dlow}, x_{dupp}$ ). If  $x_{idnew}$  exceeds ( $x_{dlow}, x_{dupp}$ ), the new position is recalculated by subtracting the new velocity from the old particle position (line 7–12).

#### A modified mutation in DE

The most frequently used mutation strategies implemented in the basic DE have some shortcomings. For example, DE/rand may have slower convergence than DE/best, while DE/best may fall easily into premature convergence. For this reason, this paper adopts a modified mutation strategy named "DE/rand-to-best" by combining the advantages of DE/rand and DE/best, which has a fast convergence property and does not easily fall into premature convergence. The proposed modified mutation strategy first establishes an archive in the operation process of DE ([Zhao et al. 2011](#)). The archive is initiated as empty, then, after each generation, the parent solutions that fail in the selection process are added to the archive. If the archive size exceeds the population size  $N$ , then some solutions are randomly removed from the archive to keep the

archive size at  $N$ . The archive provides information about the evolution path and improves the diversity of the population. Compared with DE/rand and DE/best, DE/rand-to-best benefits from its fast convergence and strong global search ability by incorporating best solution information in the evolutionary search and keeping the diversity of the population. The proposed mutation strategy is generated as follows:

$$v_{yid}(t) = y_{r1d}(t) + F \cdot (y_{bestd}(t) - y_{r2d}(t)) + F \cdot (y_{r3d}(t) - \tilde{y}_{r4d}(t)) \tag{7}$$

where,  $y_{r1d}$ ,  $y_{r2d}$  and  $y_{r3d}$  are individuals randomly selected from the current population,  $y_{bestd}$  is the best individual of the current population.  $\tilde{y}_{r4d}$  is randomly chosen from the union of the current population and the archive. The pseudo-code of the modified mutation strategy is below:

---

```

Pseudo-code of modified mutation strategy of DE
Initiate archive to be empty.
if fit(yi(t)) > fit(uyi(t)) then
    Add yi(t) to the archive.
    else if fit(yi(t)) < fit(uyi(t)) then
        Add uyi(t) to the archive.
end if
if size(archive) > N then
    Randomly remove size(archive)-N solutions from the archive to keep size(archive)=N.
end if
for i=1:N do
    Randomly select yrd, yrd and yrd from the population.
    Record the best individual ybestd of the population according to each individual's fitness value.
    Randomly selected  $\tilde{y}_{rd}$  from the population ∪ archive.
end for
    
```

---

### Constraint handling mechanism

For BLPP (1), both the upper and the lower level programming problems are standard constraint optimization problems without considering the information interaction between the leader and the follower. However, the constraint handling mechanism is very important for the constraint optimization problem. To this end, we adopt a penalty function based technique to deal with the constraints.

Considering the constraint optimization problem below:

$$\min f(x)$$

$$s.t. g_i(x) \leq 0, \quad i = 1, 2, \dots, p$$

where,  $S$  denotes search space,  $x \in S, S \subseteq R^n$ .

By using a penalty factor, the above problem can be transformed into the following problem:

$$\min F(x) = f(x) + M \cdot \sum_{i=1}^p (\max\{g_i(x), 0\})^2$$

where,  $M$  is a pre-set and sufficiently large positive constant called penalty factor.

Without loss of generality, we take the lower level programming problem as a single independent constraint optimization problem to describe the constraint handling technique. We suppose that there are  $p$  inequality constraints in the lower level programming problem, and the decision variable of the upper level programming problem,  $x$ , is given. In the search space, a particle which satisfies the constraints is called a feasible particle, otherwise it is called an infeasible particle. In this condition, we can calculate all particles' fitness according to the following equations

$$\text{fit}(x, y) = \begin{cases} f(x, y), & \text{if } y \in \Omega(x) \\ F(x, y), & \text{if } y \in S \setminus \Omega(x) \end{cases} \tag{8}$$

$$F(x, y) = f(x, y) + M \cdot \sum_{i=1}^p (\max\{g_i(x, y), 0\})^2 \tag{9}$$

where,  $S$  is the search space, and  $\Omega(x)$  is the feasible set of the lower level programming problem.

### The steps of HHPSODE to solve BLPP

This section presents the solution procedure of HHPSODE for solving BLPP. HHPSODE is consisted of a modified PSO algorithm based main program and a modified DE algorithm based subprogram. Algorithm 2 and Algorithm 3 are the main program and the subprogram. In addition, Fig. 1 is a flow chart of the main program to illustrate HHPSODE intuitively. Details are shown below.

---

#### Algorithm 2: Main program-PSO

1. Initialize the parameters: population size  $N_1$  and  $N_2$ ; maximum iteration  $T_{max1}$  and  $T_{max2}$ ; learning factors  $c_1$  and  $c_2$ ; maximum inertia weight  $w_{max}$  and minimum inertia weight  $w_{min}$ ; scaling factor  $F$ ; crossover rate  $CR$ ; penalty factor  $M$ ; upper level decision variable's upper and lower bounds:  $x_{d,upp}$  and  $x_{d,low}$ ; lower level decision variable's upper and lower bounds:  $y_{d,upp}$  and  $y_{d,low}$ .
  2. for  $t=1:T_{max1}$  do
  3. for  $i=1:N_1$  do
  4. for  $d=\{1,2,\dots,D_1\}$ , initiate the velocity  $v_{sid}(t) = 0$ ; randomly generate the  $i$ th particle's position  $x_{id}(t)$  according to the following equation:
 
$$x_{id}(t) = x_{d,low} + rand * (x_{d,upp} - x_{d,low})$$

$$v_{sid}(t) = 0.1 * x_{id}(t)$$
  5. Initiate  $p_{sid} = p_{sgd} = x_{id}(t)$ .
  6. For each given  $x_i$ , solve the lower level programming problem by using Algorithm 3 and output  $y^*$  as the best-found solution of the lower level problem:
 
$$y^* = \text{Subprogram-DE}(x_i, y_i)$$
  7. Calculate the  $i$ th particle's fitness value according to:
 
$$\text{fit}_i(x_i(t), y^*) = f_i(x_i(t), y^*) + M \cdot \sum_{i=1}^p (\max\{G_i(x_i(t), y^*), 0\})^2$$
  8. for  $d=\{1,2,\dots,D_1\}$ , update the  $i$ th particle's position according to Algorithm 1.
  9. Update  $p_{si}$  and  $p_{sg}$ :
 
$$\text{if } \text{fit}_i(p_{si}(t), y^*) > \text{fit}_i(x_i(t), y^*), \text{ then } p_{si}(t) = x_i(t)$$

$$\text{if } \text{fit}_i(p_{sg}(t), y^*) > \text{fit}_i(p_{si}(t), y^*), \text{ then } p_{sg}(t) = p_{si}(t)$$
  10. for  $d=\{1,2,\dots,D_1\}$ , update the  $i$ th particle's position according to Algorithm 1.
  11.  $t=t+1$
  12. end for
  13. end for
-

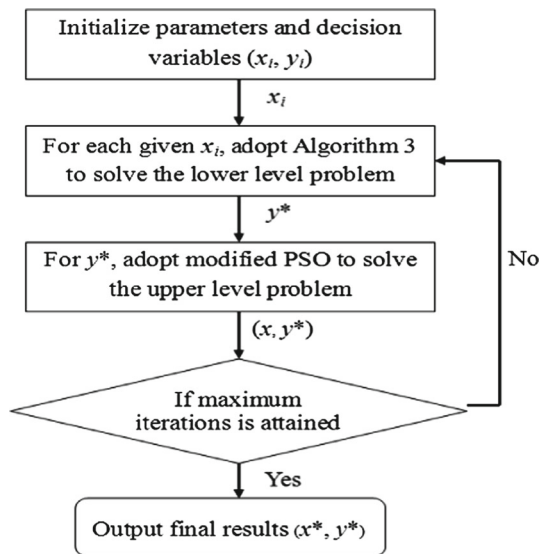


Fig. 1 Flow chart of HHPSODE

**Algorithm 3: Subprogram-DE**

1. for  $t=1:T_{max2}$  do
2. for  $i=1:N_2$  do
3. for  $d=\{1,2,\dots,D_2\}$ , randomly generate the  $i$ th individual's position  $y_{id}(t)$  according to the following equation:
 
$$y_{id}(t) = y_{dlow} + rand * (y_{dupp} - y_{dlow})$$
4. Initiate  $y_{best}(t) = y_{id}(t)$ .
5. For the given  $x$ , calculate the  $i$ th individual's fitness value according to:
 
$$fit_2(x, y_i(t)) = f_2(x, y_i(t)) + M \cdot \sum_{k=1}^q (\max\{g_k(x, y_i(t)), 0\})^2$$
6. Update the best individual position  $y_{best}$ :
 
$$\text{if } fit_2(x, y_{best}(t)) > fit_2(x, y_i(t)), \text{ then } y_{best}(t) = y_i(t)$$
7. for  $d=\{1,2,\dots,D_2\}$ , generate the next generation population according to the following equations:
 
$$v_{yid}(t) = y_{pid}(t) + F \cdot (y_{best}(t) - y_{pid}(t)) + F \cdot (y_{pid}(t) - \tilde{y}_{pid}(t))$$

$$u_{yid}(t) = \begin{cases} v_{yid}(t), & \text{if } rand[0,1] \leq CR \text{ or } d = d_{rand} \\ y_{id}(t), & \text{otherwise} \end{cases}$$

$$y_i(t+1) = \begin{cases} y_i(t), & \text{if } fit_2(x, y_i(t)) < fit_2(x, U_i(t)) \\ u_{yi}(t), & \text{otherwise} \end{cases}$$
8.  $t=t+1$
9. end for
10. end for
11. Output  $y^* = y_{best}$ .

1. Input the following parameters:  $N_1=N_2=20$ ;  $T_{max1}=T_{max2}=100$ ;  $c_1=c_2=2.0$ ;  $w_{max}=0.9$ ;  $w_{min}=0.4$ ;  $F=0.5$ ;  $CR=0.9$ ;  $M=10^5$ ;  $[x_{low}, x_{upp}] = [y_{low}, y_{upp}] = [0, 1000]$ .

2. for  $t=1:100$  do

3. Randomly generate 20 particles: for the  $i$ th particle, initiate its velocity  $v_{xid}(t) = 0$  and randomly generate its position  $x_{id}(t)$  ( $d=\{1,2\}$ ) according to:

$$x_{id}(t) = 0 + random * (1000 - 0) = 1000 * random$$

4. Initiate  $p_{xid}(t) = p_{ygd}(t) = x_{id}(t)$ .

5. For each given  $x_i$ , solve the lower level programming problem by using Algorithm 3:

(1) for  $t=1:100$ do

(2) Randomly generate 20 individuals: for the  $i$ th individual, randomly generate its position  $y_{id}(t)$  ( $d=\{1,2\}$ ):

$$y_{id}(t) = 0 + random * (1000 - 0) = 1000 * random$$

(3) Initiate  $y_{best}(t) = y_{id}(t)$ .

(4) For the given  $x$ , calculate the  $i$ th individual's fitness value according to:

$$fit_2(x, y) = 2x_1^2 + y_1^2 - 5y_2$$

$$\left[ \begin{aligned} & (\max(-x_1^2 + 2x_1 - x_2^2 + 2y_1 - y_2 - 3, 0))^2 \\ & + 10^5 * (\max(-x_2 - 3y_1 + 4y_2 - 4, 0))^2 \\ & + (\max(-y_1, 0))^2 + (\max(-y_2, 0))^2 \end{aligned} \right]$$

(5) Update the best individual  $y_{best}(t)$ :

$$\text{if } fit_2(x, y_{best}(t)) > fit_2(x, y_i(t)), \text{ then } y_{best}(t) = y_i(t)$$

(6) Generate the next generation population according to:

$$v_{yid}(t) = y_{pid}(t) - 0.5 * (y_{best}(t) - y_{pid}(t)) - 0.5 * (y_{pid}(t) - \tilde{y}_{pid}(t))$$

$$u_{yid}(t) = \begin{cases} v_{yid}(t), & \text{if } rand[0,1] \leq 0.9 \text{ or } d = d_{rand} \\ y_{id}(t), & \text{otherwise} \end{cases}$$

$$y_i(t+1) = \begin{cases} y_i(t), & \text{if } fit_2(x, y_i(t)) < fit_2(x, U_i(t)) \\ u_{yi}(t), & \text{otherwise} \end{cases}$$

(7)  $t=t+1$

(8) end for

(9) Output  $y^* = y_{best}$ .

6. Calculate the  $i$ th particle's fitness value according to the following equation:

$$fit_1(x, y^*) = -x_1^2 - 3x_2 - 4y_1^2 + (y_2^*)^2$$

$$+ 10^5 * \left[ \begin{aligned} & (\max(x_1^2 + 2x_2 - 4, 0))^2 \\ & + (\max(-x_1, 0))^2 + (\max(-x_2, 0))^2 \end{aligned} \right]$$

7. Update  $p_{xi}$  and  $p_{yg}$ :

$$\text{if } fit_1(p_{xi}(t), y^*) > fit_1(x_i(t), y^*), \text{ then } p_{xi}(t) = x_i(t)$$

$$\text{if } fit_1(p_{yg}(t), y^*) > fit_1(p_{xi}(t), y^*), \text{ then } p_{yg}(t) = p_{xi}(t)$$

8. Update the  $i$ th particle's position according to Algorithm 1:

(1) Calculate  $v_{xidnew}$  according to the following equations:

$$v_{xidnew} = w(t)v_{xid}(t) + 2.0 * r_1 * (p_{xid}(t) - x_{id}(t)) + 2.0 * r_2 * (p_{ygd}(t) - x_{id}(t))$$

$$w(t) = 0.4 + 0.5 * \left( \frac{100 - t}{100} \right)$$

(2) if  $v_{xidnew} \leq x_{dlow}$ , then  $v_{id}(t+1) = x_{dlow}$

(3) if  $v_{xidnew} \geq x_{dupp}$ , then  $v_{id}(t+1) = x_{dupp}$

(4) Calculate  $x_{xidnew}$  according to:  $x_{xidnew} = x_{id}(t) + v_{id}(t+1)$

(5) if  $x_{dlow} \leq x_{xidnew} \leq x_{dupp}$ , then  $x_{id}(t+1) = x_{xidnew}$

(6) else  $x_{id}(t+1) = x_{id}(t) - v_{id}(t+1)$

9.  $t=t+1$

10. end for

11. Output the best-found solution of T5:  $x^* = p_{yg} = (0.2508, 1.9628)$ ,  $y^* = (2.4796, 1.3465)$ . Then, calculating the objective functions of T5, we have:

$$f_1(x^*, y^*) = -(x_1^*)^2 - 3x_2^* - 4(y_1^*)^2 + (y_2^*)^2 = -14.0566$$

$$f_2(x^*, y^*) = 2(x_1^*)^2 + (y_1^*)^2 - 5y_2^* = -0.4584$$

**Example**

To explain the hybrid algorithm HHPSODE, we have chosen to apply HHPSODE on an example (take the test problem T5 for example) where the data are presented in Tables 6 and 7. The details of solving process are as follows:

**Numerical experiments**

To illustrate the performance of HHPSODE, we adopt eight typical bi-level problems including four bi-level linear pro-

**Table 1** Parameters setting of GA, PSO and HPSOBLP (Kuo and Huang 2009; Li et al. 2006)

Algorithm	GA	PSO	HPSOBLP
Parameter	Population: 20 Crossover rate: 0.8 Mutation rate: 0.1 Generations: 200	Population: 20 $v_{max}$ : 10 Inertial weight: 1.2 – 0.2 Iterations: 200	Population: 20, 40 $c_1 = c_2 = 2.0$ Inertial weight: 1.2 – 0.1 Generations: 120, 30

**Table 2** Comparisons of the best-found results of T1–T4 based on three different algorithms

Test problems	Upper level’s objective $f_1$			Lower level’s objective $f_2$		
	HHPSONDE	GA	PSO	HHPSONDE	GA	PSO
T1	85.6727	85.0551	85.0700	–48.9591	–50.16937	–50.1745
T2	11.0032	10.9968	10.9998	–11.0032	–10.9968	–10.9998
T3	15.9958	15.9917	16	–3.9996	–3.94636	–4
T4	29.2059	29.1480	29.1788	–3.3295	–3.1930	–3.1977

**Table 3** Comparisons of the average-found results of T1–T4 based on three different algorithms

Test problems	Upper level’s objective $f_1$			Lower level’s objective $f_2$		
	HHPSONDE	GA	PSO	HHPSONDE	GA	PSO
T1	85.2632	84.65781	84.85119	–49.9931	–50.03023	–50.0781
T2	10.9932	10.8082	10.9961	–10.9932	–10.8082	–10.9961
T3	15.9729	15.82567	15.98811	–3.9965	–3.946363	–3.99634
T4	27.2975	21.52948	24.81256	–3.1844	–3.39072	–3.1977

gramming problems (Kuo and Huang 2009) and four bi-level nonlinear programming problems (Li et al. 2006) to test the algorithm HHPSONDE, and then we compare HHPSONDE with other algorithms.

Parameters setting of HHPSONDE: Iterations  $T_{max1} = T_{max2} = 100$ ; population size  $N_1 = N_2 = 20$ ; learning factors  $c_1 = c_2 = 2.0$ ; maximum inertia weight  $w_{max} = 0.9$  and minimum inertia weight  $w_{min} = 0.4$ ; scaling factor  $F = 0.5$ ; crossover rate  $CR=0.9$ ; penalty factor  $M = 100,000$ . The parameters setting of related algorithms proposed in the literature (Kuo and Huang 2009; Li et al. 2006) are given in Table 1.

GA and PSO are different algorithms mentioned in Kuo and Huang (2009). HPSOBLP, TRM (trust region method) and Original are different algorithms mentioned in Li et al. (2006). All computational experiences for the examples were implemented on the IBM E420 laptop with Windows 7 (Intel®Core™ i3 Duo CPU, 2 GB of RAM).

For each BLPP, we execute HHPSONDE 20 times independently, and record the best and the average results. Note that BLPPs always take the leader’s interest as a priority. In addition, the goals of the four linear BLPPs (T1–T4) are to maximize their objective functions, and the goals of the four non-linear BLPPs (T5–T8) are to minimize their objective functions. Thus for T1–T4, the larger realization of  $f_1$  (the upper level objective function) the better results we have,

and for T5–T8, the smaller realization of  $f_1$  (the upper level objective function) the better results we have.

Comparisons of the best and average-found results of T1–T4 based on three different algorithms (HHPSONDE, GA, PSO) are respectively given in Tables 2 and 3. Corresponding best and average decision variables’ solutions are given in Tables 4 and 5, respectively.

Tables 2 and 3 show that, compared with GA and PSO, HHPSONDE shows higher accuracy on the best and the average results of T1–T4. Take T1 for example, the best-found value of  $f_1$ (upper level objective) based on HHPSONDE, 85.6727, is larger than that of GA and PSO which are 85.0551 and 85.0700, respectively. In addition, the average-found values of  $f_1$  based on HHPSONDE are also larger than the average-found values of  $f_1$  based on GA and PSO. Similarly, for T2 and T4, the best values of  $f_1$  based on HHPSONDE are larger than the best values of  $f_1$  based on GA and PSO, and the average values of  $f_1$  based on HHPSONDE are larger than the average values of  $f_1$  based on GA. For T3, the best value of  $f_1$  solved by HHPSONDE is slightly less than the best value of  $f_1$  solved by PSO, but larger than that of GA.

Table 6 shows the best and average-found results of T5–T8 based on HHPSONDE. Table 7 gives corresponding decision variables’ solutions of T5–T8. Comparisons of the best-

**Table 4** Corresponding best-found solutions of decision variables based on three algorithms

Test problems	Best-found solutions of decision variables		
	HHPNODE	GA	PSO
T1	(16.5607, 10.7995)	(17.4582, 10.9055)	(17.4535, 10.9070)
T2	(15.8092, 11.0032)	(15.9984, 10.9968)	(15.9999, 10.9998)
T3	(3.9970, 3.9996)	(3.9994, 3.9974)	(4, 4)
T4	(0.0700, 0.8016, 0.0998, 0.5980, 0.4793)	(0, 0.898, 0, 0.599, 0.399)	(0.0004, 0.8996, 0, 0.5995, 0.3993)

**Table 5** Corresponding average solutions of decision variables based on three algorithms

Test problems	Average-found solutions of decision variables		
	HHPNODE	GA	PSO
T1	(17.3020, 10.8970)	(17.4329, 10.8659)	(17.4535, 10.9070)
T2	(15.7734, 10.9932)	(15.9041, 10.8082)	(15.9999, 10.9998)
T3	(3.9834, 3.9965)	(3.98658, 3.94636)	(4, 4)
T4	(0.0282, 0.8676, 0, 0.5462, 0.4374)	(0.15705, 0.86495, 0, 0.4719, 0.51592)	(0.0004, 0.8996, 0, 0.5995, 0.3993)

**Table 6** The best and average-found results of T5–T8 based on HHPNODE

Test problems	Upper level's objective $f_1$		Lower level's objective $f_2$	
	Best-found values	Average-found values	Best-found values	Average-found values
T5	-14.0566	-13.7010	-0.4584	-0.4999
T6	62.3814	62.4139	-24.4936	-24.4163
T7	2.0000	2.0019	24.0183	24.0135
T8	1.7980	2.1306	-0.3319	0.0586

**Table 7** Corresponding solutions of decision variables based on HHPNODE

Test problems	Best-found solutions of decision variables	Average-found solutions of decision variables
T5	(0.2508, 1.9628, 2.4796, 1.3465)	(0.1641, 1.8829, 2.4128, 1.2750)
T6	(1.4873, 2.2431)	(1.4806, 2.2397)
T7	(4.0000, 0)	(3.9995, 0)
T8	(1.2086, 0, 0.8873)	(1.5546, 0, 1.0259)

found results of T5–T8 based on HHPNODE and the other three algorithms are shown in Table 8.

From Table 8 we can see that, for T5, although there are slight gaps between HHPNODE and HPSOBLP for the best-found value of  $f_1$ , HHPNODE exhibits higher accuracy than TRM and Original. For T6 and T8, the best-found results of  $f_1$  solved by HHPNODE are much better than the results solved by HPSOBLP, TRM and Original. For T7, the best-found value of  $f_1$  based on HHPNODE, 2.000, is equal to that of the other three algorithms. Additionally, for the best-found values of  $f_2$  of T5–T8, HHPNODE also outperforms HPSOBLP, TRM and Original.

The test problems' average computation time based on HHPNODE are given in Table 9. From Table 9 we can see that the maximum average computation time of T1–T8 is

less than 28 s, and this indicates that the average time spent to find the solutions is acceptable. Note that we do not compare the computation time based on HHPNODE with other algorithms. This is because different authors may code their algorithms in different structures and implement their algorithms on different computers. In a word, for most of the test problems, HHPNODE outperforms other algorithms.

### An application on pricing and lot-sizing decisions

As an application of the proposed HHPNODE, we provide a bi-level pricing and lot-sizing model in a single manufacturer-single retailer system facing retail price-sensitive market demand in this section. The pricing and lot-



**Table 8** Comparisons of the best-found results based on four different algorithms

Test Problems	Upper level’s objective $f_1$				Lower level’s objective $f_2$			
	HHPSODE	HPSOBLP	TRM	Original	HHPSODE	HPSOBLP	TRM	Original
T5	-14.0566	-14.7578	-12.68	-12.68	-0.4584	0.2067	-1.016	-1.016
T6	62.3814	88.7757	88.79	88.79	-24.4936	-0.7698	-0.77	-0.77
T7	2.0000	2.0000	2	2	24.0183	24.0190	24.02	24.02
T8	1.7980	2.7039	2.75	2.75	-0.3319	0.5602	0.57	0.57

**Table 9** Average computation time based on HHPSODE

Test problems	Average computation time (s)
T1	27.565000
T2	26.435000
T3	27.768000
T4	27.333000
T5	26.286000
T6	25.459000
T7	26.255000
T8	27.222000

sizing model derives from the model developed by [Yildirmaz et al. \(2009\)](#), but differs from the model in the literature. In this system, the manufacturer, as the leader, first purchases raw materials from the supplier, and then the raw materials are transformed into end products. Finally, the end products are sold from the manufacturer to the retailer. As a follower, the retailer reacts, bearing in mind the selection of the manufacturer, making a decision after the manufacturer.

**Model formulation**

To formulate the mathematical bi-level model, some assumptions are given: The demand of the market is a monotonic decreasing function of the retail price; both manufacturer and retailer are aiming to maximize their own profits, and both of them have the right to make decisions; one unit of product is produced by one unit of raw material; and a 1 year planning horizon is considered.

The following symbols are used to model the problem:  
Decision variables are as follows:

- $p_m$ : Retailer’s unit purchase cost (or wholesale price).
- $p_r$ : Product selling price to end customer (or retail price).
- $\alpha$ : Number of retailer’s lot size.
- $Q$ : Retailer’s lot size.

Other related parameters are as follows:

- $C$ : Maximum load that can be placed in a truck.
- $D(p_r)$ : Annual demand for the product,  $D(p_r) = b - a \cdot p_r$ .

- $h_m$ : Manufacturer’s annual holding cost rate.
- $h_r$ : Retailer’s annual holding cost rate.
- $M_c$ : Production cost of unit product.
- $O_m$ : Manufacturer’s ordering cost per order.
- $O_r$ : Retailer’s ordering cost per order.
- $p_s$ : Manufacturer’s purchase cost.
- $R$ : Truckload charge

The manufacturer’s annual revenue, holding cost and ordering cost are denoted by  $R_m$ ,  $H_m$  and  $C_m$  as follows:

$$\begin{cases} R_m = (p_m - p_s - M_c)D \\ H_m = h_m \cdot (p_s + M_c) \cdot \frac{Q(\alpha-1)}{2} = \frac{(p_s+M_c)(\alpha-1)h_m Q}{2} \\ C_m = \frac{O_m}{\alpha Q} \end{cases}$$

The manufacturer’s annual profit is denoted by  $\Pi_m$  as follows:

$$\Pi_m = R_m - H_m - C_m = (p_m - p_s - M_c)D - \frac{(p_s + M_c)(\alpha - 1)h_m Q}{2} - \frac{O_m D}{\alpha Q} \tag{10}$$

The retailer’s annual revenue, holding cost and ordering cost are denoted by  $R_r$ ,  $H_r$  and  $C_r$  as follows:

$$\begin{cases} R_r = \left(p_r - p_m - \frac{F(Q)}{Q}\right) D \\ H_r = h_r \left(p_m + \frac{F(Q)}{Q}\right) \cdot \frac{Q}{2} = \frac{h_r(p_m Q + F(Q))}{2} \\ C_r = \frac{(O_r + F(Q))D}{Q} \end{cases}$$

The retailer’s annual profit is denoted by  $\Pi_r$  as follows:

$$\Pi_r = R_r - H_r - C_r = \left(p_r - p_m - \frac{F(Q)}{Q}\right) D - \frac{h_r(p_m Q + F(Q))}{2} - \frac{(O_r + F(Q))D}{Q} \tag{11}$$

We suppose that the retail price is a multiple of the wholesale price, that is

$$p_r = k p_m \tag{12}$$

where,  $k$  denotes the ratio of the retail price to the wholesale price, and  $k \geq 1$ .

**Table 10** Different results based on different combinations of demand coefficients ( $k^* = 2, p^* = 10$ )

	$a=2, b=1,000$	$a=4, b=1,000$	$a=6, b=1,000$	$a=8, b=1,000$	$a=10, b=1,000$
$p_m$	9.9989	9.9886	9.9979	9.9912	9.9901
$\alpha$	1	1	2	2	2
$k$	2	2	2	2	2
$p_r$	19.9978	19.9772	19.9958	19.9824	19.9802
$Q$	960	920	440	420	400
$\Pi_m$	2,799	2,590	2,397.2	2,192.3	1,992.1
$\Pi_r$	8,394	7,985.6	7,196.1	6,791.8	6,391.9
Average computation time (s)	75.770000	86.926000	71.877000	85.149000	70.508000

**Table 11** Different results based on different combinations of coefficients ( $k^* = 4, p^* = 10$ )

	$a=2, b=1,000$	$a=4, b=1,000$	$a=6, b=1,000$	$a=8, b=1,000$	$a=10, b=1,000$
$p_m$	9.9988	9.9987	9.9987	9.9995	9.9959
$\alpha$	1	2	1	2	2
$k$	4	4	4	4	4
$p_r$	39.9952	39.9948	39.9948	39.998	39.9836
$Q$	920	420	760	340	300
$\Pi_m$	2,598.9	2,198	1,799.2	1,398.9	997.6
$\Pi_r$	26,392	23,595	21,794	19,198	16,796
Average computation time (s)	80.651000	83.312000	79.956000	72.589000	74.469000

The relationship between the lot size and the number of deliveries is

$$Q = \frac{D}{\alpha} \tag{13}$$

To compute  $F(Q)$ , we assume that the transportation cost is determined in terms of truck loads; therefore freight cost is computed as (Yildirmaz et al. 2009):

$$F(Q) = \left\lceil \frac{Q}{C} \right\rceil R = \left\lceil \frac{D}{\alpha C} \right\rceil R \tag{14}$$

By combining Eqs. (10) and (11), and substituting Eqs. (12)–(14) into (10) and (11), we establish a bi-level joint pricing and lot-sizing model guided by the manufacturer as follows:

$$\begin{aligned} \max_{p_m} \Pi_m &= (p_m - p_s - M_c)D - \frac{(p_s + M_c)(\alpha - 1)h_m D}{2\alpha} - O_m \\ \text{s.t. } &p_s + M_c < p_m \leq p^* \\ \max_{\alpha, k} \Pi_r &= \left( kp_m - p_m - \frac{\alpha \left\lceil \frac{D}{\alpha C} \right\rceil R}{D} \right) D \\ &\quad - \frac{h_r (p_m D + \alpha) \left\lceil \frac{D}{\alpha C} \right\rceil R}{2\alpha} - \alpha \left( O_r + \left\lceil \frac{D}{\alpha C} \right\rceil R \right) \\ \text{s.t. } &\alpha \in N^+, 1 < k \leq k^* \end{aligned} \tag{15}$$

where,  $p^*$  and  $k^*$ , respectively represent upper bounds of  $p_m$  and  $k$ . Because the above bi-level model is a NP-hard problem, for which there is no solution in the classical method, we

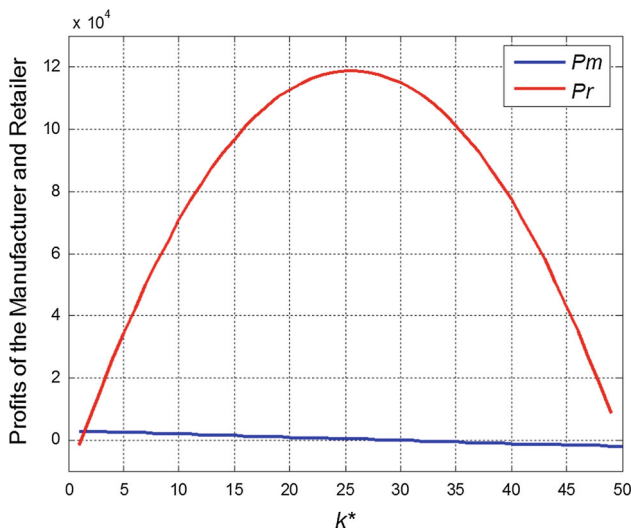
will adopt the proposed algorithm HHPSODE to deal with this problem in the next section.

### Model evaluation and analysis

In this section, we will adopt HHPSODE to solve the bi-level model (15) proposed in “Model formulation” section. The parameters in model (15) are set as follows:  $h_m = h_r = 0.01$ ;  $O_m = 2000$ ;  $O_r = 200$ ;  $p_s = 4$ ;  $M_c = 1$ ;  $C = 200$ ;  $R = 100$ . These parameter values are randomly produced and used to analyze features of the proposed model.

In reality, the wholesale price and retail price should not be too high, and they should have upper limits. To analyze the model, two upper bound groups of  $k^*$  (the upper bound of the ratio of the retail price to the wholesale price) and  $p^*$  (the upper bound of the wholesale price) are separately considered:  $k^* = 2, p^* = 10$ ;  $k^* = 4, p^* = 10$ . The parameter settings of HHPSODE are the same as in “Numerical experiments” section, and the results are shown in Tables 10 and 11.

The demand function proposed in this paper is a linear decreasing function of retail price.  $a$  denotes price elastic coefficient, and  $b$  represents maximum market demand. In the following numerical simulation, we fix  $b$  (the maximum market demand) and change  $a$  (the price elastic coefficient) from a lower price sensitivity to a higher price sensitivity to



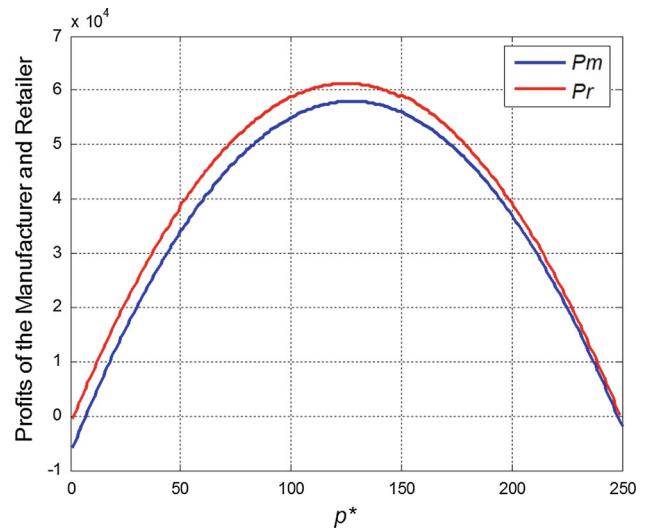
**Fig. 2** The profit curves of the manufacturer and the retailer under the condition of  $a=2, b=1,000; \alpha = 2; p_m = 10; k^* \in [1, 50]$  (Color figure online)

investigate variations of the manufacturer and the retailer’s profits when demand’s sensitivity to the retail price changes.

Tables 10 and 11 denote the profits of the manufacturer and the retailer under different upper bounds of the retail price and the wholesale price. From these two tables we can see that the higher demand’s sensitivity to the price, the less profits earned by the manufacturer and the retailer. In other words, demand’s sensitivity to price is negatively correlated with the profits of the manufacturer and the retailer, and this condition abides by the market’s rule.

Finally, we do a further test to verify the good performance of HHPSODE, using the following two steps:

- (1) On the one hand, without loss of generality, Figs. 2 and 3 are drawn based on condition of  $a=2, b=1,000$ . In these two figures,  $P_m$  (the blue curves) and  $P_r$  (the red curves), respectively represent the profits of the manufacturer and the retailer. In Fig. 2, we fix two parameters:  $p_m = 10$  and  $\alpha = 2$ . We then draw the profits curves of the manufacturer and the retailer when  $k^* \in [1, 50]$ . From Fig. 2 we can see that when  $k^*$  lies nearby 25, the retailer’s profit reaches its maximum value. In Fig. 3, we fix two parameters:  $k = 2$  and  $\alpha = 2$ . Then we draw the profits curves of the manufacturer and the retailer when  $p^* \in [1, 250]$ . From Fig. 3 we can see that when  $p^*$  is sited near 125, the manufacturer and the retailer’s profits reach their maximum values.
- (2) On the other hand, we employ HHPSODE to solve the model under the same condition as in Figs. 2 and 3. Firstly, we set  $p_m \in [5, 10]$  and  $k^* \in [1, 50]$ ; secondly, we set  $p^* \in [1, 250]$  and  $k \in [1, 2]$ . Then HHPSODE is executed to solve the proposed bi-level model under the two situations. The results of these two conditions



**Fig. 3** The profit curves of the manufacturer and the retailer under the condition of  $a=2, b=1,000; \alpha = 2; k = 2; p^* \in [1, 250]$  (Color figure online)

**Table 12** Best found-results based on HHPSODE ( $a=2, b=1,000$ )

	$5 < p_m \leq 10, k^* = 1:50$	$1 < k \leq 2, p^* = 1:250$
$p_m$	10	127.5041
$\alpha$	2	2
$k$	25.3514	2
$p_r$	253.514	255.0082
$\Pi_m$	464.2299	58,024
$\Pi_r$	118,840	61,259

are shown in Table 12. From Table 12 we can see that for the first condition, the best-found solution based on HHPSODE of  $p_m$  is 10, which is equal to the preset value of  $p_m$  in Fig. 2; the best-found solution based on HHPSODE of  $k^*$  is 25.3514, which is nearby 25. For the second condition, the best-found solution based on HHPSODE of  $k$  is 2, which is equal to the pre-set value of  $k$  in Fig.3; the best-found solution of  $p^*$  is 127.5041, which is nearby 125. This shows that the quasi-optimal solutions obtained by HHPSODE are very close to the optimal solution of the bi-level model. Therefore, it also illustrates that HHPSODE is effective and efficient in solving the model.

### Conclusion

This paper proposes a hierarchical hybrid PSO and DE based algorithm (HHPSODE) to deal with BLPP. To overcome the shortcomings of basic PSO and basic DE, this paper improve PSO and DE, respectively by using a velocity and position modulation method in PSO and a modified mutation strat-

egy in DE. HHPSODE employs the modified PSO as a main program and the modified DE as a subprogram. Unlike most problem-dependent algorithms designed for specific versions or based on specific assumptions, such as the gradient information of objective functions, the convexity of constraint regions and so on, HHPSODE has solved different classes of BLPPs directly. The performance of HHPSODE has been verified by eight benchmark test bi-level problems. According to results compared with other algorithms, HHPSODE exhibits a better performance.

Additionally, we employ HHPSODE to solve a bi-level pricing and lot-sizing model proposed in this paper. By analyzing the data, some managerial insights which abide by market rule are derived. This demonstrates that the proposed bi-level model is able to deal with pricing and lot-sizing problems in a manufacturer-guided supply chain system. Further tests of the proposed model also exhibit that HHPSODE is an effective and efficient algorithm to deal with BLPP. Therefore, HHPSODE is recommended for further application.

In reality, there many problems that are much more complex, such as multi-objective BLPP, multi-level programming problem and so on. Therefore, future work will focus on extending the proposed algorithm for solving multi-objective BLPP and multi-level programming problem. In addition, some improvements, such as combining PSO or DE with other intelligent algorithms, or modifying the constraint handling mechanism, could be applied to improve the calculation accuracy of the algorithm.

**Acknowledgments** The work was partly supported by the National Natural Science Foundation of China (71071113), a Ph.D. Programs Foundation of Ministry of Education of China (20100072110011), the Fundamental Research Funds for the Central Universities.

## Appendix

### Test problems

<p>T1</p> $\begin{aligned} \max f_1 &= -2x_1 + 11x_2 \text{ where } x_2 \text{ solves} \\ \max f_2 &= -x_1 - 3x_2 \\ \text{s.t. } x_1 - 2x_2 &\leq 4 \\ 2x_1 - x_2 &\leq 24 \\ 3x_1 + 4x_2 &\leq 96 \\ x_1 + 7x_2 &\leq 126 \\ -4x_1 + 5x_2 &\leq 65, \\ x_1 + 4x_2 &\geq 8 \\ x_1 \geq 0, x_2 &\geq 0 \end{aligned}$	<p>T2</p> $\begin{aligned} \max f_1 &= x_2 \text{ where } x_2 \text{ solves} \\ \max f_2 &= x_2 \\ \text{s.t. } -x_1 - 2x_2 &\leq 10 \\ x_1 - 2x_2 &\leq 6 \\ 2x_1 + x_2 &\leq 21 \\ x_1 + 2x_2 &\leq 38, \\ -x_1 + 2x_2 &\leq 18, \\ x_1 \geq 0, x_2 &\geq 0 \end{aligned}$
<p>T3</p> $\begin{aligned} \max f_1 &= 3x_2 \text{ where } x_2 \text{ solves} \\ \max f_2 &= -x_2 \\ \text{s.t. } -x_1 + x_2 &\leq 3 \\ x_1 - 2x_2 &\leq 12 \\ 4x_1 + x_2 &\leq 12 \\ x_1 \geq 0, x_2 &\geq 0 \end{aligned}$	<p>T4</p> $\begin{aligned} \max f_1 &= 8x_1 + 4x_2 - 4y_1 + 40y_2 + 4y_3 \\ &\text{where } y \text{ solves} \\ \max f_2 &= -x_1 - 2x_2 - y_1 - y_2 - 2y_3 \\ \text{s.t. } y_1 - y_2 - y_3 &\geq -1 \\ -2x_1 + y_1 - 2y_2 + 0.5y_3 &\geq -1 \\ -2x_1 - 2y_1 + y_2 + 0.5y_3 &\geq -1 \\ x_1, x_2, y_1, y_2, y_3 &\geq 0 \end{aligned}$
<p>T5</p> $\begin{aligned} \min f_1 &= -x_1^2 - 3x_2 - 4y_1y_2^2 \\ \text{s.t. } x_1^2 + 2x_2 &\leq 4, x_1 \geq 0, x_2 \geq 0 \\ &\text{where } y \text{ solves} \\ \min f_2 &= 2x_1^2 + y_1^2 - 5y_2 \\ \text{s.t. } x_1^2 - 2x_1 + x_2^2 - 2y_1 + y_2 &\geq -3 \\ x_2 + 3y_1 - 4y_2 &\geq -4, y_1 \geq 0, y_2 \geq 0 \end{aligned}$	<p>T6</p> $\begin{aligned} \min f_1 &= x^2 + (y - 10)^2 \\ \text{s.t. } x + 2y - 6 &\leq 0, -x \leq 0 \\ &\text{where } y \text{ solves} \\ \min f_2 &= x^3 + 2y^3 + x - 2y - x^2 \\ \text{s.t. } -x - 2y - 3 &\leq 0, -y \leq 0 \end{aligned}$
<p>T7</p> $\begin{aligned} \min f_1 &= (x - 5)^4 + (2y + 1)^4 \\ \text{s.t. } x + y - 4 &\leq 0, -x \leq 0 \\ &\text{where } y \text{ solves} \\ \min f_2 &= e^{-x+y} + x^2 + 2xy + y^2 + 2x + 6y \\ \text{s.t. } -x + y - 2 &\leq 0, -y \leq 0 \end{aligned}$	<p>T8</p> $\begin{aligned} \min f_1 &= (x_1 - y_2)^4 + (y_1 - 1)^2 + (y_1 - y_2)^2 \\ \text{s.t. } -x_1 &\leq 0 \\ &\text{where } y \text{ solves} \\ \min f_2 &= 2x_1 + e^{y_1} + y_1^2 + 4y_1 + 2y_2^2 - 6y_2 \\ \text{s.t. } 6x_1 + 2y_1^2 + e^{y_2} - 15 &\leq 0, -y_1 \leq 0, y_1 - 4 \leq 0 \\ 5x_1 + y_1^4 + y_2 - 25 &\leq 0, -y_2 \leq 0, y_2 - 2 \leq 0 \end{aligned}$

## References

- Abad, P. L. (2003). Optimal pricing and lot-sizing under conditions of perishability, finite production and partial backordering and lost sale. *European Journal of Operational Research*, *144*, 677–685.
- Belmecheri, F., Prins, C., Yalaoui, F., & Amodeo, L. (2012). Particle swarm optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows. *Journal of Intelligent Manufacturing*, doi:10.1007/s10845-012-0627-8.
- Bingül, Z., & Karahan, O. (2011). A Fuzzy Logic Controller tuned with PSO for 2 DOF robot trajectory control. *Expert Systems with Applications*, *38*(1), 1017–1031.
- Cai, Y. Q., Wang, J. H., & Yin, J. (2012). Learning-enhanced differential evolution for numerical optimization. *Soft Computing*, *16*, 303–330.
- Calvete, H. I., & Galé, C. (2011). On linear bi-level problems with multiple objectives at the lower level. *Omega*, *39*, 33–40.
- Calvete, H. I., Galé, C., & Mateo, P. M. (2008). A new approach for solving linear bi-level problems using genetic algorithms. *European Journal of Operational Research*, *188*, 14–28.
- Chan, F. T. S., & Tiwari, M. K. (2007). *Swarm intelligence, focus on ant and particle swarm optimization*. Vienna, Austria: I-Tech Education and Publishing.
- Chu, C. H., & Hsieh, H. T. (2012). Generation of reciprocating tool motion in 5-axis flank milling based on particle swarm optimization. *Journal of Intelligent Manufacturing*, *23*(5), 1501–1509.
- Dewez, S., Labbé, M., Marcotte, P., & Savard, G. (2008). New formulations and valid inequalities for a bi-level pricing problem. *Operations Research Letters*, *36*(2), 141–149.
- Gaitonde, V. N., & Karnik, S. R. (2012). Minimizing burr size in drilling using artificial neural network (ANN)-particle swarm optimization (PSO) approach. *Journal of Intelligent Manufacturing*, *23*(5), 1783–1793.
- Gao, Y., Zhang, G. Q., Lu, J., & Wee, H. M. (2011). Particle swarm optimization for bi-level pricing problems in supply chains. *Journal of Glob Optimization*, *51*, 245–254.
- García-Nieto, J., & Alba, E. (2011). Restart particle swarm optimization with velocity modulation: A scalability test. *Soft Computing*, *15*, 2221–2232.
- Guan, Y. P., & Liu, T. M. (2010). Stochastic lot-sizing problem with inventory-bounds and constant order-capacities. *European Journal of Operational Research*, *207*, 1398–1409.
- Hejazia, S. R., Memariani, A., Jahanshahloo, G., & Sepehri, M. M. (2002). Linear bi-level programming solution by genetic algorithm. *Computers & Operations Research*, *29*, 1913–1925.
- Ilonen, J., Kamarainen, J., & Lampinen, J. (2003). Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, *17*(1), 93–105.
- Janson, S., & Middendorf, M. (2005). A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on System, Man, and Cybernetics B*, *35*(6), 1272–1282.
- Jeroslow, R. G. (1985). The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming*, *32*, 146–164.
- Kébé, S., Sbihi, N., & Penz, B. (2012). A Lagrangean heuristic for a two-echelon storage capacitated lot-sizing problem. *Journal of Intelligent Manufacturing*, *23*(6), 2477–2483.
- Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, (pp. 1942–1948). Perth, Wa, Australia.
- Kuo, R. J., & Huang, C. C. (2009). Application of particle swarm optimization algorithm for solving bi-level linear programming problem. *Computers & Mathematics with Applications*, *58*, 678–685.
- Lan, K. M., Wen, U. P., Shih, H. S., & Lee, E. S. (2007). A hybrid neural network approach to bi-level programming problems. *Applied Mathematics Letters*, *20*, 880–884.
- Li, H. Y., & Meissner, J. (2011). Competition under capacitated dynamic lot-sizing with capacity acquisition. *International Journal of Production Economics*, *131*, 535–544.
- Li, M. Q., Lin, D., & Wang, S. Y. (2010). Solving a type of biobjective bi-level programming problem using NSGA-II. *Computers & Mathematics with Applications*, *59*, 706–715.
- Li, X. Y., Tian, P., & Min, X. P. (2006). A hierarchical particle swarm optimization for solving bi-level programming problems. *Lecture Notes in Computer Science, Artificial Intelligence and Soft Computing- ICAISC, 2006*(4029), 1169–1178.
- Lu, L., & Qi, X. T. (2011). Dynamic lot-sizing for multiple products with a new joint replenishment model. *European Journal of Operational Research*, *212*, 74–80.
- Lukač, Z., Šorić, K., & Rosenzweig, V. V. (2008). Production planning problem with sequence dependent setups as a bi-level programming problem. *European Journal of Operational Research*, *187*, 1504–1512.
- Marcotte, P., Savard, G., & Zhu, D. L. (2009). Mathematical structure of a bi-level strategic pricing model. *European Journal of Operational Research*, *193*, 552–566.
- Plagianakos, V., Tasoulis, D., & Vrahatis, M. (2008). A review of major application areas of differential evolution. In *Advances in differential evolution, Vol. 143*, (pp. 197–238). Springer, Berlin.
- Price, K. V., Storn, R. M., & Lampinen, J. A. (2005). *Differential evolution: A practical approach to global optimization*. Berlin: Springer.
- Raa, B., & Aghezzaf, E. H. (2005). A robust dynamic planning strategy for lot-sizing problems with stochastic demands. *Journal of Intelligent Manufacturing*, *16*(2), 207–213.
- Rajesh, J., Gupta, K., Kusumakar, H. S., Jayaraman, V. K., & Kulkarni, B. D. (2003). A Tabu search based approach for solving a class of bi-level programming problems in chemical engineering. *Journal of Heuristics*, *9*, 307–319.
- Sadeghierad, M., Darabi, A., Lesani, H., & Monsef, H. (2010). Optimal design of the generator of micro turbine using genetic algorithm and PSO. *Electrical Power and Energy Systems*, *32*, 804–808.
- Sahin, H. K., & Ciric, R. A. (1998). A dual temperature simulated annealing approach for solving bi-level programming problem. *Computers & Chemical Engineering*, *23*, 11–25.
- Shih, H. S., Wen, U. P., Lee, E. S., Lan, K. M., & Hsiao, H. C. (2004). A neural network approach to multi-objective and multilevel programming problems. *Computers & Mathematics with Applications*, *48*, 95–108.
- Shi, Y., & Eberhart, R. (1999). Empirical study of particle swarm optimization. In *International conference on evolutionary computation*, (pp. 1945–1950). IEEE press, Washington, USA.
- Storn, R. (1999). System design by constraint adaptation and differential evolution. *IEEE Transactions on Evolutionary Computation*, *3*(1), 22–34.
- Storn, R., & Price, K. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341–359.
- Vincent, L. W. H., Ponnambalam, S. G., & Kanagaraj, G. (2012). Differential evolution variants to schedule flexible assembly lines. *Journal of Intelligent Manufacturing*, doi:10.1007/s10845-012-0716-8.
- Wen, U. P., & Huang, A. D. (1996). A simple Tabu Search method to solve the mixed-integer problem bi-level programming problem. *European Journal of Operational Research*, *88*, 563–571.
- Yildirmaz, C., Karabatı, S., & Sayın, S. (2009). Pricing and lot-sizing decisions in a two-echelon system with transportation costs. *OR Spectrum*, *31*, 629–650.
- Zhao, S. Z., Suganthan, P. N., & Das, S. (2011). Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. *Soft Computing*, *15*, 2175–2185.