Swarm and Evolutionary Computation 8 (2014) 26–32

Contents lists available at SciVerse ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

A hybrid intelligent algorithm by combining particle swarm optimization with chaos searching technique for solving nonlinear bilevel programming problems

Zhongping Wan [a], Guangmin Wang [b,*], Bin Sun [a]

[a] School of Mathematics and Statistics, Wuhan University, Wuhan 430072, PR China
[b] School of Economics and Management, China University of Geosciences, Wuhan 430074, PR China

ARTICLE INFO

Article history:
Received 11 October 2012
Received in revised form
29 June 2013
Accepted 8 August 2013
Available online 25 September 2013

Keywords:
Nonlinear bilevel programming problems
Hybrid intelligent algorithm
Particle swarm optimization
Chaos search technique

ABSTRACT

In this paper, a hybrid intelligent algorithm by combining the particle swarm optimization (PSO) with chaos searching technique (CST) is presented for solving nonlinear bilevel programming problems. The bilevel programming is transformed into a single level programming problem by use of the KKT conditions of the lower level problem. Then, the hybrid intelligent algorithm is proposed to solve the transformed problem. Our approach embeds the CST into PSO. Firstly, the algorithm is initialized by a set of random particles which travel through the search space. Secondly, an optimization problem is solved by CST to judge whether the particle is feasible or not. In each iteration, all the feasible particles are ranked in ascending order. Particles in the front of list are updated by PSO, while particles in the end of list are updated by CST. The CST used here is not only to enhance the particles but also to improve the diversity of the particle swarm so as to avoid PSO trapping the local optima. Finally, the hybrid intelligent algorithm is commented by illustrating the numerical results on several benchmark problems from the references.

1. Introduction

The bilevel programming problem (BLP) is a nested optimization problem with two levels (namely the upper and lower level) in a hierarchy. The decision maker at the upper level (the leader) firstly optimizes his/her objective function independently. After the leader chooses the decision, the decision maker at the lower level (the follower) makes his/her decision. The leader knows the objective and constraint functions of the follower who may or may not know the objective and (or) constraint functions of the leader. However, the leaders decision is influenced by the reaction of the follower. Since many practical problems, such as engineering design, management, economic policy and traffic problems, can be formulated as hierarchical problems, BLP has been studied and received increasing attention in the literatures. During the last three decades, some surveys and bibliographic reviews were given by several authors [1–3]. Reference books on bilevel programming and related issues have emerged [4–6].

The bilevel programming problem is a nonconvex problem, which is extremely difficult to solve. Firstly, Jeroslow [7] pointed out, then Ben-Ayed and Blair [8] and Bard [9] proved sequentially that the bilevel programming problem is a NP-Hard problem. Vicente et al. [10] also showed that even the search for the local optima to the bilevel linear programming is NP-Hard. See Ref. [11] for more detailed discussion of the complexity issues in linear bilevel programming problem. Therefore, many researchers are devoted themselves into developing the algorithms for solving BLP. Traditional approaches for solving BLP can be roughly classified into the following categories [6]: vertex enumeration methods, decent algorithm, approaches based on Kuhn–Tucker condition and penalty functions, etc. The properties such as differentiation and continuity are necessary when proposing the traditional algorithms. Unfortunately, the bilevel programming problem is nonconvex. Thus, many researchers tend to propose the heuristic algorithms for solving the bilevel programming problem because of their key characteristics of minimal problem restrictions such as differentiation.

Mathieu et al. [12] firstly developed a genetic algorithm (GA) for solving bilevel linear programming problem because of its good characteristics such as simplicity, minimal problem restrictions, global perspective and implicit parallelism. Motivated by the same reason, other kinds of genetic algorithm for solving bilevel programming were also proposed in Refs. [13–19]. Because of the prominent advantage that neural computing can converge to the equilibrium point (optimal solution) rapidly, the

* Corresponding author.
E-mail addresses: zpwan-whu@126.com (Z. Wan), wgm97@163.com (G. Wang).

2210-6502/$ - see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.swevo.2014.08.001

neural network approach was used to solve bilevel programming problem in Refs. [20–23]. Tabu search [24–27], fuzzy [28–30], simulated annealing [31], interactive fuzzy [32–35], rule sets [36], ant colony optimization [37] and interactive fuzzy goal [38,39] are also typical intelligent algorithms for solving bilevel programming problem. Recently, particle swarm optimization (PSO) [40], as a new algorithm of evolutionary computation, was also applied to solve the bilevel programming problem [41,42]. The intelligent algorithms have different characteristics as well as advantages and disadvantages. To deal with complicated optimization problem, hybridizing these techniques is a natural choice to make best of their advantages and avoid their disadvantages. Therefore, what techniques to use and how to hybridize them are two major problems to solve when designing a hybrid algorithms. Hybridizing different search methods (to combine global search and local search methods or to combine the search operators of different algorithms) has been widely used to solve the optimization problems [43–48]. Furthermore, the hybrid algorithms are also proposed to solve the bilevel programming problems. Yaakob and Watada [49] integrate genetic algorithm and neural network to produce a hybrid intelligent algorithm for solving bilevel programming models. Kuo and Han [50] propose a hybrid of genetic algorithm and particle swarm optimization to solve bilevel linear programming problem. Wong et al. [51] apply a decision system, based on an artificial neural network (ANN) and modified ant colony optimization (ACO) to solve the stochastic dynamic lot-sizing problem. In the methodology, ANN is used to learn the simulation results, followed by the application of a real-valued modified ACO algorithm to find the optimal decision variables. It is well-known that PSO has the advantage of good convergence performance and the disadvantage of easily trapping in local minima. While, chaos searing techniques (CST) have the advantage of easier jumping local optimal solution with the property of nonrepeatedly traversal all the states according to its own "rules" in a certain range. Based on the above the fact, we propose a hybrid intelligent algorithm for solving nonlinear bilevel programming problems by combining global search method (PSO) and local search method (CST). In our hybrid algorithm, CST is embedded in the PSO to improve the worse particles for overcoming the disadvantage that PSO may be trapped in local minima. As the same time, CST is also used to judge the point is feasible or not by solving problem (4). The aim of our proposed algorithm is to combines the PSO's advantage of good convergence performance with the CST's advantage of easier jumping local optimal solution to overcome the limitations resulted from the noncontinuity and nondifferentiability of the nonlinear programming problems.

The remaining of this paper is organized as follows. Section 2 introduces the problem definition and properties of nonlinear bilevel programming problems. Section 3 proposes a hybrid algorithm by combining particle swarm optimization algorithm and chaos searching technique for solving nonlinear bilevel programming problems. Some illustrative examples are provided in Sections 4 and 5 concludes the paper.

2. Problem definition and properties

The nonlinear bilevel programming problems(NBLP) consist of two levels, namely, the upper and lower levels each having its nonlinear objective function. NBLP are formulated as follows:

$$\begin{aligned}&(NBLP)\quad \min_{x,y}\quad F(x,y)\\ &\text{s.t.}\quad g(x,y)\le 0\quad \text{where } y \text{ solves the following problem}\\ &\min_{y}\quad f(x,y)\\ &\text{s.t.}\quad h(x,y)\le 0\end{aligned}\tag{1}$$

where $F(x,y)$, $f(x,y)$ are object functions of the upper and lower level problems, respectively. $g(x,y)$ and $h(x,y)$ are the constraint functions of the upper and lower level problems, respectively. $x\in R^{n_1}, y\in R^{n_2}$ are the decision variables under the control of the upper lower level problems, respectively.

Next we give the following definitions of the NBLP [4]:

- The constraint region of NBLP
  $\Omega=\{(x,y)\,|\,g(x,y)\le 0,\ h(x,y)\le 0\}$
- The projection of Ω onto the upper level's decision space
  $\Omega(X)=\{x\,|\,\text{there exists } y \text{ such that } (x,y)\in\Omega\}$
- For each fixed $x\in\Omega(X)$, the constraint region of the lower level problem
  $\Omega(x)=\{y\,|\,h(x,y)\le 0\}$
- For each fixed $x\in\Omega(X)$, the rational reaction set of the lower level problem
  $M(x)=\{y\,|\,y\in\arg\min\{f(x,y),y\in\Omega(x)\}\}$
- The inducible region of NBLP
  $IR=\{(x,y)\,|\,(x,y)\in\Omega, y\in M(x)\}$

Firstly, we suppose that $\Omega\neq\emptyset$ is compact and $\Omega(X)\neq\emptyset$. For each $x\in\Omega(X)$, the lower level problem (LP) is formulated as follows:

$$\begin{aligned}&(LP)\quad \min_{y}\quad f(x,y)\\ &\text{s.t.}\quad h(x,y)\le 0\end{aligned}\tag{2}$$

To avoid situations where (2) is not well posed, it is natural to assume that $\Omega(x)\neq\emptyset$ and $M(x)\neq\emptyset$. Even so, NBLP may be not well defined when the rational reaction set, $M(x)$, is not single-valued [4]. Bard [9] used examples to illustrate the difficulties that often arise when $M(x)$ is multivalued and discontinuous. Here, we consider the situation that there is a unique solution to the lower level problem for each fixed $x\in\Omega(X)$. The reader can refer to [4,6] for how to do when that $M(x)$ is multivalued. Then, we can give the definitions of feasible solution and optimal solution to NBLP as follows:

Definition 1. A point (x,y) is called to be feasible to NBLP if $(x,y)\in IR$.

Definition 2. A feasible point (x^*,y^*) is called to be optimal to NBLP if $F(x^*,y^*)\le F(x,y),\forall(x,y)\in IR$.

From the definition of the feasible solution to NBLP, $(\overline{x},\overline{y})$ is a feasible solution means that $\overline{y}$ solves problem (2) for fixed $\overline{x}$. By applying Kuhn–Tucker conditions for problem (2), there exists a λ, such that

$$\begin{aligned}&\nabla_y f(\overline{x},\overline{y})+\lambda^T\nabla_y h(\overline{x},\overline{y})=0\\ &\lambda^T h(\overline{x},\overline{y})=0\\ &\lambda\ge 0\end{aligned}\tag{3}$$

where $\lambda\in R^m$ is a column variable. Obviously, Eq. (3) can be equivalently transformed into the optimization problem as follows:

$$\begin{aligned}&\min_{\lambda}\|\nabla_y f(\overline{x},\overline{y})+\lambda^T\nabla_y h(\overline{x},\overline{y})\|^2+\|\lambda^T h(\overline{x},\overline{y})\|^2\\ &\text{s.t.}\quad \lambda\ge 0\end{aligned}\tag{4}$$

Therefore, if $(\overline{x},\overline{y})$ is a feasible solution to NBLP, there exists an optimal solution to problem (4) and the optimal value equals zero. That is to say, we can solve Eq. (3) to judge whether the point $(\overline{x},\overline{y})\in S$ is feasible to NBLP.

Now, we can give the following definition:

Definition 3. Denote $w(x,y)=\min_{\lambda\geq 0}\|\nabla_y f(x,y)+\lambda^T\nabla_y h(x,y)\|^2+\|\lambda^T h(x,y)\|^2$ as the feasible weighting value of the point (x,y).

Obviously, the smaller the feasible weighting value is, the closer (x,y) is near the feasible region. (x,y) is a feasible solution if the feasible weighting value equals zero.

3. Design of the proposed algorithm

3.1. Brief introduction to PSO

The particle swarm optimization (PSO), which is a population-based algorithm, was inspired by the social behavior of animals such as fish schooling and bird flocking. Similar to other population-based algorithms, such as evolutionary algorithms, PSO can solve a variety of difficult optimization problems, and has shown a faster convergence rate than other evolutionary algorithms on some problems [40]. Another advantage of PSO is that it has very few parameters to adjust, which makes it particularly easy to implement.

In PSO, a number of simple entities the particles are placed in the search space of some problem or function, and each evaluates the objective function at its current location. Each particle then determines its movement through the search space by combining some aspect of the history of its own current and best (best-fitness) locations with those of one or more members of the swarm, with some random perturbations. The next iteration takes place after all particles have been moved. Eventually the swarm, like a flock of birds collectively foraging for food, is likely to move close to an optimum of the fitness function.

Suppose that the search space is D-dimensional, then the ith particle of the swarm can be represented by a D-dimensional vector, $X_i=(x_{i1},x_{i2},\ldots,x_{iD})$. The velocity of this particle can be represented by another D-dimensional vector $V_i=(v_{i1},v_{i2},\ldots,v_{iD})$. The best previously visited position of the ith particle is denoted as $p_i^{best}=(p_{i1},p_{i2},\ldots,p_{iD})$. The best previously visited position of the swarm is denoted as $g^{best}=(g_1,g_2,\ldots,g_D)$. Change the velocity and position of the ith particle according to the following equation (see notes below):

$$v_i^{k+1}=v_i^k+c_1r_1(p_{best}-x_i^k)+c_2r_2(g_{best}-x_i^k) \tag{5}$$

$$x_i^{k+1}=x_i^k+v_i^{k+1} \tag{6}$$

where c_1 and c_2 are positive constant, called acceleration, and r_1 and r_2 are two random numbers, uniformly distributed in [0,1]. In order to prevent the particle from leaving far away out of the searching space, the constant V_{max} was implemented for limiting the velocity. Details about PSO can be referred to Refs. [53,54].

3.2. Brief introduction to CST

Chaos is a kind of nonperiodic moving style. It exists widely in the nonlinear system and is unique to the system. It appears stochastic but can be generated through deterministic means. Chaos is a kind of unshaped out-of-order state, which blends with specific forms relative to some "immobile points", "periodic points" [55]. Chaos has subtle internal structure and it is a kind of "strange attractor", which can attract the movement of system and confine it within the specified range.

The chaos searching technique (CST) is a new kind of searching method [55]. The basic idea of the algorithm is to transform the variable of problems from the solution space to chaos space and then perform search to find out the solution by virtue of the randomicity, orderliness and ergodicity of the chaos variable. Chaos searching technique includes two steps: firstly, search all the points in turn within the changing range of variables and taking the better point as the current optimum point; then regard the current optimum point as the center, a tiny chaos disturbance is imposed and more careful search is performed to find out the optimum point. The chaos search technique has many advantages such as not sensitive to the initial value, easy to skip out of the locally minimum value, fast searching velocity and global gradual convergence.

The following Logistic map is used to generate the chaos sequence because it is more convenient to use:

$$z_{i+1}=\mu z_i(1-z_i) \tag{7}$$

where $z_i\in[0,1]$ $(i=1,2,\ldots,)$ is the chaos variable, i $(i=1,2,\ldots,)$ is the times of iteration; and μ is the control parameter. It is easy to testify that the system is entirely in chaos situation when $\mu=4$ and the chaos space belongs to [0, 1].

3.3. The idea of the proposed algorithm

The main idea of our algorithm is to embed CST into PSO for solving the nonlinear bilevel programming problems so as to combine the their advantages and avoid their disadvantages. The algorithm is described in details as follows: firstly, the particles of the swarm are randomly initialed. Then, problem (4) is solved to judge whether the particle is feasible or not. If there exists a solution to problem (4) and the objective function value equals zero, then the particle is feasible, and add the particle to the feasible list and set the upper level's objective function value as the fitness value of the particle; otherwise, the particle is infeasible, and add the particle to the infeasible list and set the objective function value of problem (4) as the fitness value of the particle. Secondly, the particles in the feasible list are ranked in ascending order; after that, the particles in the infeasible list are ranked in ascending order. Then, the velocity of particle near the top and its new position will be assigned according to Eqs. (5) and (6); and the particles in the end of the list are updated by use of CST. After an iteration, the fitness values of the particles are computed again. And repeat the above steps until the terminal criterions are met.

3.4. Steps of the proposed algorithm

The flow chart of the proposed algorithm is shown in the following Fig. 1.

The steps of the proposed algorithm are listed in details as follows:

Step 1. Initialize the parameters. Population size (the number of particles) is set $M=m+n$, where m particles are updated by PSO and n particles are updated by CST. Maximal velocity, V_{max}, two learning factors, c_1 and c_2, and two random variables, r_1, $r_2\in[0,1]$, are initially set. The maximum number of iterations (T) is set up to be used as the termination conditions of the algorithm and set the counter of iteration $t=0$.

Step 2. Initialize the particles. Initialize the ith $(i=1,2,\ldots,M)$ particle randomly with initial position, X_i, within the pre-specified range and velocity, V_i, in the range of maximal speed, V_{max}. And the best previously visited position of the ith $(i=1,2,\ldots,M)$ particle, P_i^{best}, is initialized as X_i.

Step 3. Compute the fitness values of the particles. Eq. (4) is solved by use of CST. If there exists a solution to problem (4) and the objective function value equals zero, then the particle is feasible, and add the particle to the feasible list and set the upper level's objective function value as the fitness value of the particle; otherwise, the particle is infeasible, and add the particle to the

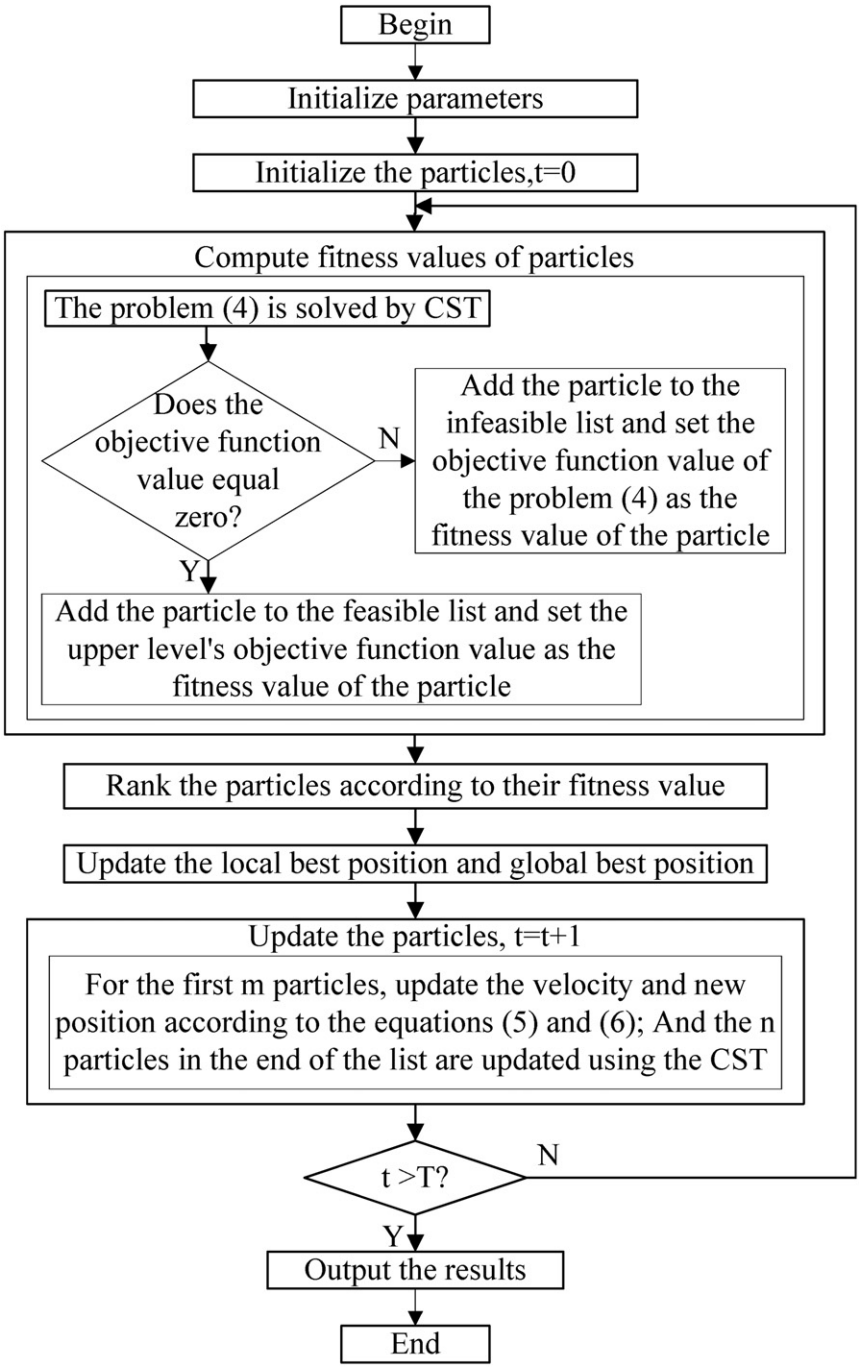

Fig. 1. The flow chart of the proposed hybrid intelligent algorithm.

infeasible list and set the objective function value of problem (4) as the fitness value of the particle.

Step 4. Rank the particles. The particles in the feasible list are ranked in ascending order; After that, the particles in the infeasible list are ranked in ascending order.

Step 5. Update the local best position, P_i^{best}, and global best position, g^{best}. For the ith ($i=1,2,\ldots,M$) particle, compare particle's fitness evaluation with its P_i^{best}. If current value is better than P_i^{best}, then set the current value to P_i^{best}. Compare the first particle's fitness evaluation with the global best position, g^{best}. If current value is better than g^{best}, then set the current value to g^{best}.

Step 6. Update the particles. For the first m particles, the velocity of particle and its new position will be assigned according to Eqs. (5) and (6); And the n particles in the end of the list are updated using the CST with the initial chaos variable X_i.

Step 7. Terminal conditions. $t=t+1$, If the number of iterations is larger than the maximum number of iterations (T), goto Step 8, otherwise goto Step 3.

Step 8. Output the results. Output the optimal particle, compute and output the upper level and lower level's objective function values.

Notes: Firstly, the CST is not only used to solve problem (4) but also embedded in the PSO to improve the worse particles. This way not only enhances the particles but also improves the diversity of the particle swarm to avoid PSO trapping the local optima. Secondly, the upper level and lower level's decision variables are all randomly

generated and updated by PSO or CST in our algorithm, which is different from the way that only the upper level's decision variable is encoding and the lower level's decision variable is computed according to the upper level's decision variable [14,42]. And, the feasible weighting value is introduced to replace the fitness value of the particle when it is infeasible, which can force the infeasible particle to be feasible. Thirdly, in the early stage of the algorithm, the feasible weighting value is used to rank the infeasible particles, which can avoid the situation that the infeasible particle is denoted as the best particle although it is the best of all when only the fitness value is used by all particles.

4. Computational experiments

In this section, the problems from references are presented to illustrate the feasibility and efficiency of the adaptive genetic algorithm. The parameters are set as follows: population size (the number of particles) $M=45$, where $m=40$ and $n=5$; maximal velocity $V_{max}=2$, two learning factors, $c_1=c_2=2$ and the maximum number of iterations $T=8$.

Firstly, we consider the following problem from Example 1 in Ref. [56] and solve it by our proposed algorithm:

$$\min_x \quad F(x,y)=(x_1-30)^2+(x_2-20)^2-20y_1+20y_2$$
$$\text{s.t.} \quad x_1+2x_2\geq 30, \quad x_1+x_2\leq 25, \quad x_2\leq 15$$
$$\min_{0\leq y\leq 10} \quad f(x,y)=(x_1-y_1)^2+(x_2-y_2)^2$$

We execute the proposed algorithm in 10 independent runs, the best solution is $(x^*,y^*)=(19.99984.7573,9.9501,4.959)$ and the upper level's objective function $F(x^*,y^*)=232.5219$ as well as the lower level's objective function $f(x^*,y^*)=101.0372$ at the best solution (x^*,y^*), which are all near the exact values $(x,y)=(20,5,10,5)$, $F(x,y)=225$ and $f(x,y)=100$. The variety of fitness value in the algorithm is demonstrated in the following figure (Fig. 2).

For further test, we execute the proposed algorithm 50 independent runs on each problem. The best solution (x^*,y^*) and the upper level's objective function $F(x^*,y^*)$ as well as the lower level's objective function $f(x^*,y^*)$ at the best solution (x^*,y^*) are recorded. The comparison of the results in our paper with those in Ref. [14] are listed in Tables 1 and 2. And the best solution in Ref. [14] is denoted by (x,y), and the upper level's objective function

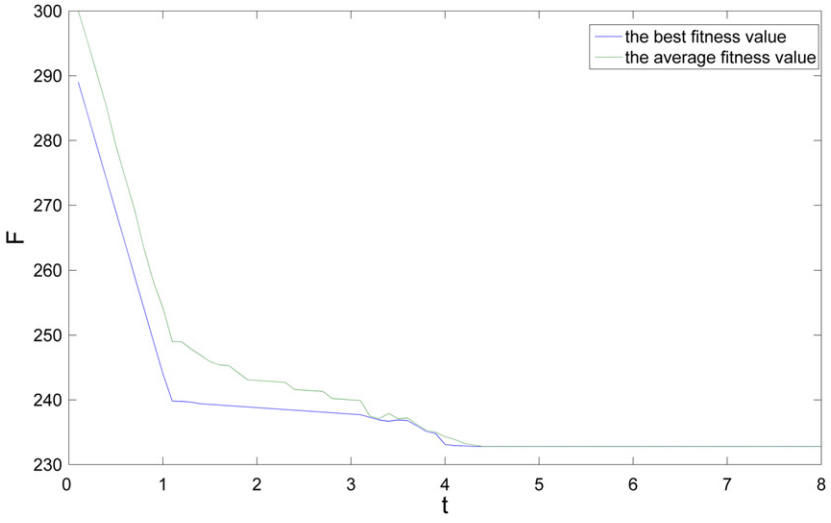

Fig. 2. The fitness values vary as the number of iteration.

Table 1
Comparison of the best solution in our paper with that in references.

No.	Results by PSO-CST (x^*,y^*)	Results in Ref. [14] (x,y)	Results in corresponding ref. $(\overline{x},\overline{y})$
1[3)] in Ref. [14]	(0.3844,1.6124,1.8690,0.8041)	(4.4e−7,2,1.875,0.9063)	(0,2,1.875,0.9063)
2[4)] in Ref. [14]	(0.1324,0.1754,0.6935,0.7327,0.2273)	(1.25e−13,0.9,0,0.6,0.4)	(0,0.9,0,0.6,0.4)
3[11)] in Ref. [14]	(0.1511,0.6256,0.369)	(1.4e−12,1,7.07e−13)	NA
4[13)] in Ref. [14]	(10.0020,9.9961)	(10.0,10.0)	(10.03,9.969)
5[14)] in Ref. [14]	(1.8602,0.9073,0.005)	(1.8888,0.8889,0)	NA
6[17)] in Ref. [14]	(7.0321,6.842047,5.9071,6.8312)	(7.0709,7.0713,7.0709,7.0703)	(7.0854,7.0291,7.0854,0)
7[21)] in Ref. [14]	(17.5039,29.8906,−2.4994,9.8894)	(0,30,−10,10)	NO
8[22)] in Ref. [14]	(12.4124,19.3109,−7.5859,−0.6899)	(0,30,−10,10)	NO
9[23)] in Ref. [14]	(17.2024,7.4665,7.2189,2.4251)	(20,5,10,5)	NO
10[24)] in Ref. [14]	(0.1946,14.9870,6.1019,7.9628)	(19.5629,5.2722,10,5.2722)	NO
11[25)] in Ref. [14]	(10.6084,10.0550,9.4545,5.1257)	(6.2048,12.8594,6.2048,10)	NO
12[26)] in Ref. [14]	(0.8606,1.4599,0.3138)	(1.8888,0.8889,0)	NO
13[27)] in Ref. [14]	(0.9099,1.5294,0.1762)	(0.6648,1.5746,0.0721)	NO
14[28)] in Ref. [14]	(0.9233,1.5083,0.1899)	(0.6648,1.5746,0.0721)	NO

Notes: "NA" means that the result is not available for the algorithm. "NO" means that problem from 10 to 18 only appear in Ref. [14].

Table 2
Comparison of the upper level's and the lower level's objective function values at the best solution in our paper with those in references.

No.	Results by PSO-CST		Results in Ref. [14]		Results in corresponding ref.	
	$F(x^*,y^*)$	$f(x^*,y^*)$	$F(x,y)$	$f(x,y)$	$F(\overline{x},\overline{y})$	$f(\overline{x},\overline{y})$
$1^{3)}$ in Ref. [14]	−14.7772	−0.2316	−12.68	−1.016	−12.68	−1.016
$2^{4)}$ in Ref. [14]	−29.2064	2.3641	−29.2	3.2	−29.2	3.2
$3^{11)}$ in Ref. [14]	640.7139	0.9946	1000	1	1000	1
$4^{13)}$ in Ref. [14]	100.0393	0.0000	100.0001	3.5e−11	100.58	0.001
$5^{14)}$ in Ref. [14]	−1.1660	7.4441	−1.2098	7.6168	3.57	2.4
$6^{17)}$ in Ref. [14]	1.9816	−1.9816	1.9802	−1.9802	1.9760	−1.9454
$7^{21)}$ in Ref. [14]	0.0527	0.0000	0	100	NO	
$8^{22)}$ in Ref. [14]	0.0004	0.0000	0	100	NO	
$9^{23)}$ in Ref. [14]	0.0075	125.0854	0	100	NO	
$10^{24)}$ in Ref. [14]	0.0000	84.2367	6.86e−15	91.45	NO	
$11^{25)}$ in Ref. [14]	0.0001	25.6292	1.47e−14	8.18	NO	
$12^{26)}$ in Ref. [14]	0.0082	2.5621	2.22e−16	7.62	NO	
$13^{27)}$ in Ref. [14]	0.0374	2.6969	1.22e−16	2.50	NO	
$14^{28)}$ in Ref. [14]	0.0337	2.7442	1.22e−16	2.50	NO	

Notes: "NO" means that problem from 10 to 18 only appear in Ref. [14].

$F(x,y)$ as well as the lower level's objective function $f(x,y)$ at the best solution (x,y) are listed. The solution in corresponding reference is denoted by $(\overline{x},\overline{y})$, and the upper level's objective function $F(\overline{x},\overline{y})$ as well as the lower level's objective function $f(\overline{x},\overline{y})$ at the solution $(\overline{x},\overline{y})$ are listed.

The algorithm in Ref. [14] can guarantee that a global optimal solution is computed. We transform the bilevel programming into the single level programming using the KKT condition of the lower level problem, which is as same as that in Ref. [14]. But, our algorithm is more simple although our algorithm cannot guarantee a global optimal solution. Furthermore, in our algorithm, the upper level decision maker has predominance over the lower level decision maker, which is in accordance with the structure of bilevel programming. From Tables 1 and 2, we can obtain the following comments about our algorithm and the algorithm in Ref. [14]: the results by our algorithm are mostly in accordance with those by the NEA in Ref. [14] such as Examples 3, 4, 5, 6, 10 and 12. For some problems, such as Examples 2, 7 and 8, the upper level's objective function values by our algorithm is not worse than those by the algorithm in Ref. [14], while the lower level's objective function values by our algorithm is better than those by the algorithm in Ref. [14]. For Examples 1, 9, 11, 13 and 14, the upper level's objective function values by our algorithm is not worse than those by the algorithm in Ref. [14], while the lower level's objective function values by our algorithm is worse than those by the algorithm in Ref. [14]. The reason is that our algorithm prefers to the upper level's objective function.

5. Conclusions and future works

In this paper, we propose a hybrid intelligent algorithm to solve bilevel programming problems. In the proposed method, the CST is embedded in the PSO to enhance the worse particles and to improve the diversity of the particle swarm in order to avoid PSO trapping the local optima. The numerical results on several benchmark problems have shown that the proposed algorithm is feasible.

Generally speaking, the parameters of the intelligent algorithm have great influence on the convergence and performance. So in our future works, the following will be researched:

(1) Influence of the parameters in our algorithm on the performance and convergence, through which the appropriate parameters for the different problem can be obtained.
(2) Research to demonstrate the efficiency of the proposed algorithm by solving more and larger scale examples generated as the references.
(3) Comparison with other algorithms by solving more examples.

Acknowledgments

The authors would like to thank anonymous referees for their invaluable comments and suggestions. This research was partially funded by the National Natural Science Foundation of China (No. 71171150 and 71201146), the Social Science Foundation of Ministry of Education (No. 10YJC630233) and the Fundamental Research Funds for the Central Universities, China University of Geosciences Wuhan (No. CUG120410).

References

[1] L. Vicente, P.H. Calamai, Bilevel and multibilevel programming: a bibliography review, Journal of Global Optimization 5 (1994) 291–305.
[2] S. Dempe, Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints, Optimization 52 (2003) 333–359.
[3] B. Colson, P. Marcotte, G. Savard, Bilevel programming: a survey, A Quarterly Journal of Operations Research 3 (2005) 87–107.
[4] J.F. Bard, Practical Bilevel Optimization: Algorithm and Applications, Kluwer Academic Publishers, Dordrecht, 1998.
[5] A. Migdalas, P.M. Pardalos, P. Varbrand, Multilevel Optimization: Algorithms and Applications, Kluwer Academic Publishers, Dordrecht, 1998.
[6] S. Dempe, Foundation of Bilevel Programming, Kluwer Academic Publishers, London, 2002.
[7] R. Jeroslow, The polynomial hierarchy and a simple model for competitive analysis, Mathematical Programming 32 (1985) 146–164.
[8] O. Ben-Ayed, O. Blair, Computational difficulty of bilevel linear programming, Operations Research 38 (1990) 556–560.
[9] J.F. Bard, Some properties of the bilevel linear programming, Journal of Optimization Theory and Applications 68 (1991) 371–378.
[10] L. Vicente, G. Savard, J. Judice, Decent approaches for quadratic bilevel programming, Journal of Optimization Theory and Applications 81 (1994) 379–399.
[11] X. Deng, Complexity issues in bilevel linear programming, in: A. Migdalas, P.M. Pardalos, P. Varbrand (Eds.), Multilevel Optimization: Algorithms and Applications, Kluwer Academic Publishers, Dordrecht, 1998, pp. 149–164.
[12] R. Mathieu, L. Pittard, G. Anandalingam, Genetic algorithm based approach to bi-level linear programming, RAIRO-Operations Research 28 (1) (1994) 1–21.
[13] S.R. Hejazi, A. Memariani, G. Jahanshanloo, M.M. Sepehri, Linear bilevel programming solution by genetic algorithm, Computers & Operations Research 29 (2001) 1913–1925.
[14] Y.P. Wang, Y.C. Jiao, H. Li, An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handing scheme, IEEE Transactions on Systems Man and Cybernetics: Part C 35 (2) (2005) 221–232.

[15] G.M. Wang, X.J. Wang, Z. Wan, et al., An adaptive genetic algorithm for solving bilevel linear programming problem, Applied Mathematics and Mechanics 28 (12) (2007) 1605–1612.

[16] H.I. Calvete, C. Gate, P.M. Mateo, A new approach for solving linear bilevel problems using genetic algorithms, European Journal of Operational Research 188 (2008) 14–28.

[17] K. Deb, A. Sinha, An evolutionary approach for bilevel multi-objective problems, Cutting-Edge Research Topics on Multiple Criteria Decision Making, Part 1 35 (2009) 17–24.

[18] K. Deb, A. Sinha, Solving bilevel multi-objective optimization problems using evolutionary algorithms, Evolutionary Multi-Criterion Optimization 5467 (2009) 110–124.

[19] M.Q. Li, D. Lin, S.Y. Wang, Solving a type of biobjective bilevel programming problem using NSGA-II, Computers & Mathematics with Applications 59 (2) (2010) 706–715.

[20] H.S. Shih, U.P. Wen, E.S. Lee, et al., A neural network approach to multi-objective and multilevel programming problems, Computers and Mathematics with Applications 48 (2004) 95–108.

[21] K.M. Lan, U.P. Wen, H.S. Shih, et al., A hybrid neural network approach to bilevel programming problems, Applied Mathematics Letters 20 (8) (2007) 880–884.

[22] Y.B. Lv, T.S. Hu, G.M. Wang, Z. Wan, A neural network approach for solving nonlinear bilevel programming problem, Computers and Mathematics with Applications 55 (12) (2008) 2823–2829.

[23] S.B. Yaakob, J. Watada, Double-layered hybrid neural network approach for solving mixed integer quadratic bilevel problems, Integrated Uncertainty Management and Applications 68 (2010) 221–230.

[24] U.P. Wen, A.D. Huang, A simple tabu search method to solve the mixed-integer linear bilevel programming problem, European Journal of Operational Research 88 (3) (1996) 563–571.

[25] M. Gendreau, P. Marcotte, G. Savard, A hybrid Tabu-ascent algorithm for the linear bilevel programming problem, Journal of Global Optimization 8 (3) (1996) 217–233.

[26] J. Rajesh, K. Gupta, H.S. Kusumakar, et al., A tabu search based approach for solving a class of bilevel programming problems in chemical engineering, Journal of Heuristics 9 (4) (2003) 307–319.

[27] H. Küçükaydın, N. Aras, İ.K. Altınel, A hybrid tabu search heuristic for a bilevel competitive facility location model, Hybrid Metaheuristics 6373 (2010) 31–45.

[28] J.d. Oña, P. Gómez, E. Mérida-Casermeiro, Bilevel fuzzy optimization to preprocess traffic data to satisfy the law of flow conservation, Transportation Research Part C: Emerging Technologies 19 (1) (2011) 29–39.

[29] M. Sakawa, T. Matsui, Stackelberg solutions for random fuzzy two-level linear programming through possibility-based probability model, Expert Systems with Applications (2012), http://dx.doi.org/10.1016/j.eswa.2012.03.001.

[30] M. Sakawa, H. Katagiri, T. Matsui, Stackelberg solutions for fuzzy random two-level linear programming through probability maximization with possibility, Fuzzy Sets and Systems 188 (1) (2012) 45–57.

[31] H.S. Kemal, A.R. Ciric, Dual temperature simulated annealing approach for solving bilevel programming problems, Computers & Chemical Engineering 23 (1) (1998) 11–25.

[32] G.M. Wang, X.J. Wang, Z. Wan, A fuzzy interactive decision making algorithm for bilevel multi-followers programming with partial shared variables among followers, Expert Systems with Applications 36 (7) (2009) 10471–10474.

[33] G.Q. Zhang, J. Lu, Fuzzy bilevel programming with multiple objectives and cooperative multiple followers, Journal of Global Optimization 47 (3) (2010) 3–419.

[34] Y. Zheng, Z. Wan, G.M. Wang, A fuzzy interactive method for a class of bilevel multiobjective programming problem, Expert Systems with Applications 38 (8) (2011) 10384–10388.

[35] M. Sakawa, H. Katagiri, T. Matsui, Interactive fuzzy random two-level linear programming through fractile criterion optimization, Mathematical and Computer Modelling 54 (11–12) (2011) 3153–3163.

[36] Z. Zheng, J. Lu, G.Q. Zhang, et al., Rule sets based bilevel decision model and algorithm, Expert Systems with Applications 36 (1) (2009) 18–26.

[37] H.I. Calvete, C. Gale, M.J. Oliveros, Bilevel model for production-distribution planning solved by using ant colony optimization, Computers & Operations Research 38 (1) (2011) 320–327.

[38] S.R. Arora, R. Gupta, Interactive fuzzy goal programming approach for bilevel programming problem, European Journal of Operational Research 194 (2) (2009) 368–376.

[39] S. Dempe, Comment to "Interactive fuzzy goal programming approach for bilevel programming problem" by S.R. Arora and R. Gupta, European Journal of Operational Research 212 (2) (2011) 429–431.

[40] J. Kennedy, R.C. Eberhart, Swarm Intelligence, Morgan Kaufmann Publishers, 2001.

[41] X. Li, P. Tian, X. Min, A hierarchical particle swarm optimization for solving bilevel programming problems, in: Proceedings of the 8th International Conference on Artificial Intelligence and Soft Computing (ICAISC), Poland, Lecture Notes in Computer Science, 2006, pp. 1169–1178.

[42] R.J. Kuo, C.C. Huang, Application of particle swarm optimization algorithm for solving bi-level linear programming problem, Computers and Mathematics with Applications 58 (2009) 678–685.

[43] B. Alatas, E. Akin, A.B. Ozer, Chaos embedded particle swarm optimization algorithms, Chaos, Solitons and Fractals 40 (2009) 1715–1734.

[44] K. Chandrasekaran, S.P. Simon, Multi-objective scheduling problem: hybrid approach using fuzzy assisted cuckoo search algorithm, Swarm and Evolutionary Computation 5 (2012) 1–16.

[45] S. Das, U. Halder, D. Maity, Chaotic dynamic characteristics of social foraging swarms – an analysis, IEEE Transactions on Systems, Man and Cybernetics (SMC) Part B 42 (2012) 1288–1293.

[46] A.H. Gandomi, X.S. Yang, S. Talatahari, et al., Firefly algorithm with chaos, Communications in Nonlinear Science and Numerical Simulation 18 (2013) 89–98.

[47] S. Talatahari, B.F. Azar, R. Sheikholeslami, et al., Imperialist competitive algorithm combined with chaos for global optimization, Communications in Nonlinear Science and Numerical Simulation 17 (2012) 1312–1319.

[48] Y. Wang, J. Zhou, Y. Lu, et al., Chaotic self-adaptive particle swarm optimization algorithm for dynamic economic dispatch problem with valve-point effects, Expert Systems with Applications 38 (2011) 14231–14237.

[49] S.B. Yaakob, J. Watada, A hybrid intelligent algorithm for solving the bilevel programming models, Knowledge-Based and Intelligent Information and Engineering Systems 6277 (2010) 485–494.

[50] R.J. Kuo, Y.S. Han, A hybrid of genetic algorithm and particle swarm optimization for solving bi-level linear programming problem—a case study on supply chain model, Applied Mathematical Modelling 35 (2011) 3905–3917.

[51] J.T. Wong, C.T. Su, C.H. Wang, Stochastic dynamic lot-sizing problem using bi-level programming base on artificial intelligence techniques, Applied Mathematical Modelling 36 (5) (2012) 2003–2016.

[53] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization: an overview, Swarm Intelligence 1 (2007) 33–57.

[54] J.F. Schutte, A.A. Groenwold, A study of global optimization using particle swarms, Journal of Global Optimization 31 (2005) 93–108.

[55] V. Zayats, Chaos searching algorithm for second order oscillatory system, in: Proceedings of the International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science, 2002, pp. 97–98.

[56] K. Shimizu, E. Aiyoshi, A new computational method for Syackelberg and min–max problems by use of a penalty method, IEEE Transactions on Automatic Control AC-26 (2) (1981) 460–466.