Available online at www.isr-publications.com/jnsa
J. Nonlinear Sci. Appl., 10 (2017), 2115–2132
Research Article

ISSN : 2008-1898

Journal of Nonlinear Sciences and Applications

Journal Homepage: www.tjnsa.com - www.isr-publications.com/jnsa

A cooperative coevolution PSO technique for complex bilevel programming problems and application to watershed water trading decision making problems

Tao Zhang[a,*], Zhong Chen[a], Jiawei Chen[b,*]

[a]School of Information and Mathematics, Yangtze University, Jingzhou 434023, China.
[b]School of Mathematics and Statistics, Southwest University, Chongqing 400715, China.

Communicated by Y. J. Cho

Abstract

The complex bilevel programming problem (CBLP) in this paper mainly refers to the optimistic BLP in which the high-dimensional decision variables at both levels. A cooperative coevolutionary particle swarm optimization (CCPSO) is proposed for solving the (CBLP), in which the evolutionary paradigm can efficiently prevent the premature convergence of the swarm. Furthermore, the stagnation detection strategy in our algorithm can further accelerate the convergence speed. Finally, we use the test problems from the reference and practical example about watershed water trading decision-making problem to measure and evaluate the proposed algorithm. The presented results indicate that the proposed algorithm can effectively solve the complex bilevel programming problems.

Keywords: Complex bilevel programming, cooperative coevolutionary particle swarm optimization, watershed water trading decision making problems, elite strategy.
2010 MSC: 65K10, 90C25.

1. Introduction and preliminaries

The bilevel programming problem (BLP) is a nested optimizations problem with two levels in a hierarchy: the upper and lower level decision-makers. The upper level maker makes his decision firstly, followed by the lower level decision maker. The objective function and constraint of the upper level problem not only rely on their own decision variables but also depend on the optimal solution of the lower level problem. The lower level has to optimize its own objective function under the given parameters from the upper level and the upper lever selects the parameters which feedback from the lower level to optimize the whole problem. 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 past decades, some surveys and bibliographic reviews were given by several authors [11, 12, 15, 41]. Reference books on bilevel programming and related issues have emerged [5, 10, 14, 34, 39].

*Corresponding authors
Email addresses: zt_math981@126.com (Tao Zhang), j.w.chen713@163.com (Jiawei Chen)

doi:10.22436/jnsa.010.04.65

Received 2016-12-31

The bilevel programming problem is a nonconvex problem, which is extremely difficult to solve. As we know, BLP is a NP-Hard problem [4, 7, 24]. Vicente et al. [42] also showed that even the search for the local optima to the bilevel linear programming is NP-Hard. Even so, many researchers are devoted to develop the algorithms for solving BLP and propose many efficient algorithms. To date a few algorithms exist to solve BLP, and they can be classified into four types: Karus-Kuhn-Tucker approach (KKT) [2, 3, 16, 17], Branch-and-bound method [6], penalty function approach [1, 23, 31, 38] and descent approach [18, 37]. The properties such as differentiation and continuity are necessary when proposing the traditional algorithms. Unfortunately, the bilevel programming problem is nonconvex. Many researchers tend to propose the heuristic algorithms for solving BLP because of their key characteristics of minimal problem restrictions such as differentiation. Mathieu et al. [33] firstly developed a genetic algorithm (GA) for 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 [8, 22, 44, 45]. Because of the prominent advantage that neural computing can converge to the equilibrium point (optimal solution) rapidly, the neural network approach was used to solve bilevel programming problem in [29, 32, 49]. Tabu search [21, 35, 46], simulated annealing [36], ant colony optimization [9] and λ-cut and goal-programming-based algorithm [20] are also typical intelligent algorithms for solving bilevel programming problem.

Particle swarm optimization (PSO) is a relatively novel heuristic algorithm inspired by the choreography of a bird flock, which has been found to be quite successful in a wide variety of optimization tasks [26]. Due to its high speed of convergence and relative simplicity, the PSO algorithm has been employed for solving BLP problems. For example, Li et al. [30] proposed a hierarchical PSO for solving BLP problem. Kuo and Huang [28] applied the PSO algorithm for solving bilevel linear programming problem. Jiang et al. [25] presented the PSO based on CHKS smoothing function for solving nonlinear bilevel programming problem. Gao et al. [19] presented a method to solve bilevel pricing problems in supply chains using PSO. Zhang et al. [50] presented a new strategic bidding optimization technique which applies bilevel programming and swarm intelligence. In addition, the hybrid algorithms based on PSO are also proposed to solve the bilevel programming problems [27, 43, 48]. However, it is worth noting that the mentioned above only for the relatively simple BLP and the complex bilevel programming problem (CBLP) has seldom been studied using PSO so far. The main reason lies in the complexity of the CBLP, namely that the high-dimensional decision variables at both levels. The difficulty for solving CBLP is that the increasing dimensional lead to the solution space of CBLP increased dramatically, thus the global convergence of the algorithm for CBLP is hard to be guaranteed and the convergence speed of the algorithm becomes slow.

Though Sinha et al. [40] proposed a nested bilevel evolutionary algorithm for CBLP recently, there still exist two problems: (a) for 10-dimensional test problems, the method in [40] can solve the first five test problems successfully in all the runs, but it can hardly solve the remaining test problems; (b) for two sets of test problems, the number of function evaluations of both levels is higher. From the problem (a), it can be seen that the global convergence of the algorithm is hard to be guaranteed with the increase of the dimension. From the problem (b), it is obvious that the convergence speed of the algorithm needs to be improved. In this paper, a coadapted coevolutionary particle swarm optimization (CCPSO) is proposed for solving the CBLP, in which the appropriate number of species can cover multiple niches and the species interact with one another within a shared domain model and have a cooperative relationship, thus the global convergence of the proposed algorithm for CBLP can be greatly improved. Furthermore, the stagnation detection strategy in our algorithm will enable the evolutionary pressure to increase the overall fitness of the ecosystem, which can further accelerate the convergence speed. In order to test the effectiveness and practicability of the proposed algorithm, we introduced the algorithm to settle watershed water trading decision-making problem. For the watershed water trading decision-making model based on bilevel programming in [47], we reoptimize the optimal allocations parameters of water resources by the proposed algorithm and the solutions found by the proposed method are better than those given in

the references.

The rest of this paper is organized as follows. Section 2 introduces the definitions and properties of complex bilevel programming problems. Section 3 proposes a cooperative coevolution particle swarm optimization algorithm for such problems. Some illustrative examples are provided in Section 4. The CCPSO algorithm is applied for solving a practical problem in Section 5, while the conclusion is reached in Section 6.

2. Problem formulation

Let $x \in R^{n_1}, y \in R^{n_2}$, $F, f : R^{n_1} \times R^{n_2} \to R$, $G : R^{n_1} \times R^{n_2} \to R^p$, $g : R^{n_1} \times R^{n_2} \to R^q$. The optimistic formulation of BLP can be written as follows:

$$\begin{aligned} &\min_{x,y} \quad F(x,y), \\ &\text{s.t.} \quad G(x,y) \leqslant 0, \\ &\text{where } y \text{ solves the following problem:} \\ &\min_{y} \quad f(x,y), \\ &\text{s.t.} \quad g(x,y) \leqslant 0, \end{aligned} \tag{2.1}$$

where $F(x,y)$ and $f(x,y)$ are the upper level and the lower level objective functions, respectively. $G(x,y)$ and $g(x,y)$ are the upper level and the lower level constraints, respectively. $x \in R^{n_1}$ and $y \in R^{n_2}$ are the decision variables under the control of the upper and lower level problems, respectively. The basic notions of BLP are recalled as follows:

(a) Constraint region of the BLP:

$$S = \{(x,y) | G(x,y) \leqslant 0, g(x,y) \leqslant 0\}.$$

(b) Feasible set for the lower level problem for each fixed x:

$$S(x) = \{y | g(x,y) \leqslant 0\}.$$

(c) Projection of onto the upper level maker's decision space:

$$S(X) = \{x | \exists y, G(x,y) \leqslant 0, g(x,y) \leqslant 0\}.$$

(d) The lower level maker's rational reaction set for each fixed with $x \in S(X)$:

$$P(x) = \{y | y \in \operatorname{argmin}[f(x,y) : y \in S(x)]\}.$$

(e) Inducible region:

$$IR = \{(x,y) | (x,y) \in S, y \in P(x)\}.$$

Definition 2.1. A point (x,y) is feasible if $(x,y) \in IR$.

Definition 2.2. A feasible point (x^*,y^*) is an optimal solution if $(x^*,y^*) \in IR$ and $F(x^*,y^*) \leqslant F(\bar{x},\bar{y})$, $\forall (\bar{x},\bar{y}) \in IR$.

For problem (2.1), it is noted that a solution (x^*,y^*) is feasible for the upper level problem if and only if y^* is an optimal solution for the lower level problem with $x = x^*$. In practice, we often make the approximate optimal solutions of the lower level problem as the optimal response feedback to the upper level problem, and this point of view is accepted usually. Based on this fact, the CCPSO algorithm may have a great potential for solving BLP. In the following, an algorithm based on the CCPSO is presented for solving problem (2.1).

3. The algorithm

3.1. The CCPSO algorithm

We now describe a generalized architecture for evolving interacting coadapted subcomponents. The architecture, which we call cooperative coevolution, models an ecosystem consisting of two or more species. As in nature, the species are genetically isolated—meaning that individuals only mate with other members of their species. Mating restrictions are enforced simply by evolving the species in separate populations. The species interact with one another within a shared domain model and have a cooperative relationship.

The basic coevolutionary model is shown in Fig. 1. Although this particular illustration shows three species, the actual number in the ecosystem may be more or less. Each species is evolved in its own population and adapts to the environment through the repeated application of the PSO. The figure shows the fitness evaluation phase of the PSO from the perspective of each of the three species. Although most of our implementations of this model have utilized a sequential pattern of evaluation, where the complete population of each species is evaluated in turn, the species could also be evaluated in parallel. To evaluate individuals from one species, collaborations are formed with representatives from each of the other species. There are many possible methods for choosing representatives with which to collaborate. In this paper, it is appropriate to simply let the current best individual from each species be the representative.

Now we discuss the uses of cooperative coevolution particle swarm optimization (CCPSO) to minimize a function $f(x, y)$ of n independent variables. The problem was decomposed into n species and each assigned to one of the independent variables. Each species consisted of a population of alternative values for its assigned variable. To evaluate an individual from one of the species, we first selected the current best individual from every one of the other species and combined them, along with the individual being evaluated, into a vector of variable values. This vector was then applied to the target function. An individual was rewarded based on how well it minimized the function within the context of the variable values selected from the other species. The details of the proposed algorithm are given as follows:

Algorithm 3.1.

Step 1. Initialization scheme. Initialized the species randomly and each species represents a variable of a complete solution. Then, go to Step 2 (a).

Step 2. Generate the complete solution.

- (a) Select the individual from each of the other species randomly and combine them, along with the individual being evaluated, into a vector. Then, go to Step 3.
- (b) Select the current best individual from every one of the other species and combine them, along with the individual being evaluated, into a vector. Then, go to Step 3.

Step 3. Credit evaluation. The credit assignment at the species level is defined in terms of the fitness value of the complete solutions in which the species members participate.

Step 4. Species coevolutionary. Each of the species is coevolved in a round-robin fashion using a standard PSO.

Step 5. Evolutionary stagnation detection. If the evolutionary stagnation condition of a species is false, then, go to Step 7. Otherwise, go to Step 6.

Step 6. Reinitialization the stagnation species. Keep the member with the best credit assignment, the remaining members are reinitialized randomly and the credit evaluation of these members is computed as Step 3.

Step 7. Termination check. If termination condition is false, go to Step 2 (b). Otherwise, output the optimal solution.

Evolutionary stagnation condition

It is well-known that PSO has the advantage of good convergence performance and the disadvantage of easily trapping in local optimal. If a species is unproductive, determined by the contribution its

individuals make to the collaborations they participate in, the species will be destroyed. Stagnation can be detected by monitoring the quality of the collaborations through the application of the inequality:

$$f(t) - f(t - L) < G,$$

where $f(t)$ is the fitness of the best collaboration at the generation t, G is a constant specifying the fitness gain considered to be a significant improvement, and L is a constant specifying the length of an evolutionary window in which significant improvement must be made. In this paper, the G is set as 10^{-3}.

Constraint handling

In these approaches, the pair-wise comparison used in tournament selection is exploited to make sure that:

(i) when two feasible solutions are compared, the one with better objective function value is chosen;
(ii) when one feasible and one infeasible solutions are compared, the feasible solution is chosen;
(iii) when two infeasible solutions are compared, the one with smaller constraint violation is chosen.

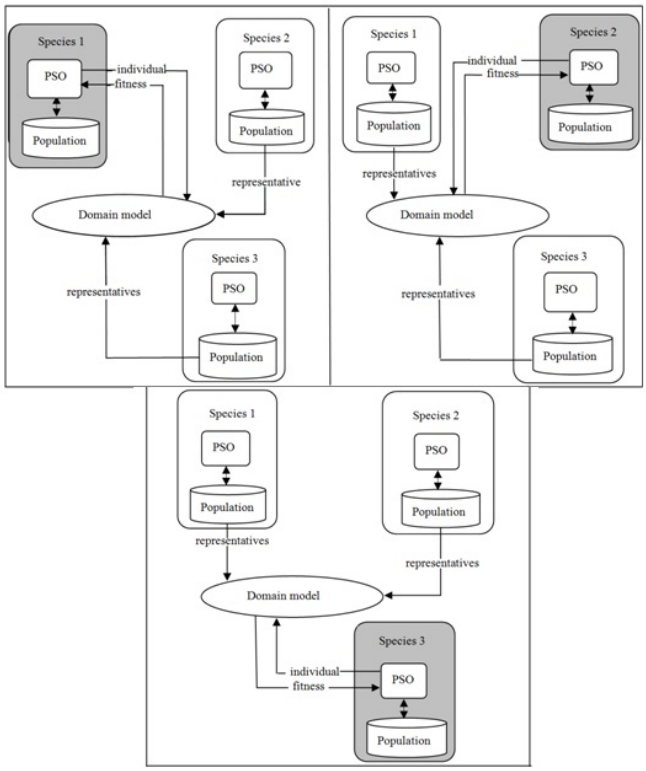

Figure 1: Cooperative coevolutionary model of three species.

3.2. The CCPSO algorithm for CBLP

The process of the proposed algorithm for solving the CBLP is an interactive coevolutionary process. We first initialize population, and then the CBLP is transformed to solve single level optimization problems in the upper level and the lower level interactively by the CCPSO. For each iteration, an approximate optimal solution for problem 1 is obtained and this interactive procedure is repeated until the accurate optimal solutions of the original problem are found. The details of the proposed algorithm are given as follows:

Algorithm 3.2.

Step 1. Initialization scheme.

Step 1.1. Initialize a random population (N_u) of the upper level variables X_u. For each upper level member $X_u^i(i = 1, 2, \cdots, N_u)$, initialize a random population (N_l) of the lower level variables Y_l.

Step 1.2. For the fixed $X_u^i(i = 1, 2, \cdots, N_u)$, perform a lower level optimization procedure to determine the corresponding optimal lower level variables using Algorithm 3.1 and the optimal solution of the lower level $Y^{optimal}$ is obtained .

Step 2. Combine the upper level variables and the corresponding optimal lower level variables to generate the complete upper level solution $Z_u^i = (X_u^i, Y^{optimal})(i = 1, 2, \cdots, N_u)$. Evaluate the fitness value of the complete upper level solutions based on the upper level function and constraints.

Step 3. Check the stop condition. If the stop condition is satisfied, the optimal solution is output; otherwise go to Step 5.

Step 4. Update the upper level decision variables using the simulated binary crossover operator (SBX) and the polynomial variation method (PM).

Step 5. Reinitialize the lower level members. For the updated $\overline{X_u^i}(i = 1, 2, \cdots, N_u)$, solve $\min \left|\overline{X_u^i} - X_u^j\right|$, where $X_u^j(j = 1, 2, \cdots, N_u)$ is the upper decision variables before updating. If X_u^p meets the condition and the $Y_p^{optimal}$ is the optimal lower lever variable corresponding to X_u^p, the $Y_p^{optimal}$ is reserved for next generation. The other $(N_l - 1)$ level particles are produced randomly in the feasible region. According to the above method, the n_s sub-swarms are produced.

Step 6. Combine the updated upper level variables and the corresponding lower level variables to generate the complete lower level solution $Z_l^i = ((\overline{X_u^i}, \overline{Y_i^1}), (\overline{X_u^i}, \overline{Y_i^2}), \cdots, (\overline{X_i}, \overline{Y_i^{optimal}}) \cdots, (\overline{X_i}, \overline{Y_i^{N_l}}))$. Then, evaluate the fitness value of the complete lower level solutions based on the lower level function and constraints.

Step 7. Fix the upper level variables of the complete lower level solution, perform a lower level optimization procedure to produce the corresponding optimal lower level variables using Algorithm 3.1 and the optimal lower level variable $Y^{optimal}(i = 1, 2, \cdots, N_u)$ is obtained. Then, go to Step 2.

Termination condition

At Step 1.2 and Step 7, Algorithm 3.1 uses a variance based termination criteria. When the value of α_j, described in the following equation becomes less than α_{stop}, the optimization task terminates. In the following, we state the termination criteria at the lower level, which can be similarly extended to the upper level. Let the variance of the lower level population members at generation j for each lower level variable j be v_i^j. If the number of lower level variables is m_l, then α_j is computed as:

$$\alpha_j = \sum_{i=1}^{m_l} \frac{v_i^j}{v_0^j}, \tag{3.1}$$

where v_0^i denotes the variance for the variable i in the initial lower level population, the value of α_j usually lies between 0 and 1 in (3.1). In this paper, the value of α_{stop} is set as 10^{-5} for the lower level and the value of α_{stop} is set as 10^{-4} for the upper level.

4. Numerical experiments

In this section, the parameters are set as follows: the PSO parameters are set as follows: $r_1, r_1 \in$ random$(0, 1)$ the inertia weight $\omega = 0.7298$ and acceleration coefficients with $r_1 = r_1 = 1.49618$. For the CCPSO, the population sizes of each species were set as 10 at the upper level and lower level. All results

presented in this paper have been obtained on a personal computer (CPU:AMD Phenon(tm)X6 1055T 2.80GHz; RAM:3.25GB) using a $C^{\#}$ implementation of the proposed algorithm.

4.1. Problem statement and its properties

In this section, we test our algorithm using two sets of twelve problems [40] which are scalable in terms of number of variables. The first set is the 5-dimensional instance of these problems and the second set is the 10-dimensional instance of these problems. Tables 1, 2, and 3 give the twelve problems and describe the properties of these problems.

4.2. Experimental comparison

The results obtained by the CCPSO are compared with those obtained by the nested bilevel evolutionary algorithm in [40]. We performed 11 runs for 5-dimension and 10-dimension problems. For 5-dimensional problems, the problems 1-5 and the problems 7-12, we choose $p = 2, q = 3$ and $r = 1$, and the problem 6 we choose $p = 2, q = 0, r = 1$ and $s = 2$. For 10-dimensional problems, the problems 1-5 and the problems 7-12 we choose $p = 5, q = 5$ and $r = 2$, and the problem 6 we choose $p = 5, q = 3, r = 2$ and $s = 2$.

The results for 5-dimensional test problems are reported in Tables 4 and 5. Table 4 provides the comparisons of function evaluations at upper and lower levels. The comparisons of the accuracy of both levels and the number of lower level calls for 11 runs, as well as the average lower level function evaluations required per lower level call are reported in Table 5. The similar result comparisons for 10-dimensional test problems are reported in Tables 6 and 7.

The fifth column and sixth column in Table 4 provides the median function evaluations required at the lower and upper levels respectively. The numbers in the brackets provide a ratio of the function evaluations required using the method in [40] against the function evaluations required using CCPSO. Both the approaches are able to successfully handle all the test problems with five dimensions, however, as can be observed from Table 4, the method in [40] requires 2.7 to 5.01 times function evaluations at the upper level, and requires 5.21 to 9.10 times function evaluations at the lower level as compared to CCPSO. From Table 5, we can obtain the following comments about our algorithm and the algorithm in [40]: for some problems, such as problems 1, 2, 4, 6, 7, 9, 10, 11 and 12, the upper level's accuracy obtained by our algorithm is not worse than those obtained by the algorithm in [40], while the lower level's accuracy obtained by our algorithm is better than those obtained by the algorithm in [40]. For the problems 3, 5, and 8, the upper level's accuracy by our algorithm is worse than those by the algorithm in [40], while the lower level's accuracy by our algorithm is not worse than those by the algorithm in [40]. On the other hand, these results also provide the median number of lower level calls, and the average number of lower level function evaluations required per lower level call. The method in [40] requires nearly three times lower level calls, and requires nearly two times function evaluations at the lower level for per lower level call compared to CCPSO. From the result comparisons, it suggests that the cooperative coevolutionary strategy adopted in the CCPSO can improve the global search ability of the algorithm. In addition, the evolutionary stagnation detection technology adopted in both levels greatly enhances the speed of convergence of the algorithm, thus it saves the number of function evaluations at both levels.

It can be seen from Tables 6 and 7 that the number of function evaluations increase significantly when the size of the test problems is increased to 10. The method in [40] is able to successfully solve the first five test problems in all the runs. From Table 6, it follows that the method in [40] requires 2.84 to 3.84 times function evaluations at the upper level, and requires 5.79 to 11.38 times function evaluations at the lower level for the first eight test problems as compared to CCPSO. From Table 7, it can be observed that the method in [40] requires 2.79 to 3.82 times lower level calls, and requires 2.16 to 3.05 times function evaluations at the lower level for per lower level call as compared to CCPSO. From the result comparisons, it can be seen that the global convergence of the proposed algorithm is greatly improved. Furthermore, the results demonstrate that the available computational resources quickly become insufficient to solve the problems with an increase in the number of dimensions.

Table 1: Test problems and their properties.

No.	Problem	Properties
1	$\min\limits_{x,y} F(x,y) = \sum\limits_{i=1}^{p} x_i^2 + \sum\limits_{i=r+1}^{q} y_i^2 + \sum\limits_{i=1}^{r} (x_i - \tan y_i)^2$	convex upper problem
	$\min\limits_{y} f(x,y) = \sum\limits_{i=r+1}^{p} x_i^2 + \sum\limits_{i=r+1}^{q} y_i^2 + \sum\limits_{i=1}^{r} (x_i - \tan y_i)^2$	convex level problem
	$x_i \in [-5,10], \quad i = 1,2,\cdots,p;$	the two levels cooperate with each other
	$y_i \in [-\pi/2, \pi/2], \quad i = 1,2,\cdots,r;$	without constrains at two levels
	$y_i \in [-5,10], \quad i = r+1, r+2,\cdots,q.$	
2	$\min\limits_{x,y} F(x,y) = \sum\limits_{i=1}^{p} x_i^2 - \sum\limits_{i=r+1}^{q} y_i^2 - \sum\limits_{i=1}^{r} (x_i - \log y_i)^2$	convex upper problem
	$\min\limits_{y} f(x,y) = \sum\limits_{i=r+1}^{p} x_i^2 + \sum\limits_{i=r+1}^{q} y_i^2 + \sum\limits_{i=1}^{r} (x_i - \log y_i)^2$	convex level problem
	$x_i \in [-5,1], \quad i = 1,2,\cdots,r;$	the two levels conflict with each other
	$x_i \in [-5,10], \quad i = r+1,2,\cdots,p;$	without constrains at two levels
	$y_i \in [0,e], \quad i = 1,2,\cdots,r;$	
	$y_i \in [-5,10], \quad i = r+1,2,\cdots,q.$	
3	$\min\limits_{x,y} F(x,y) = \sum\limits_{i=r+1}^{p} x_i^2 - \sum\limits_{i=r+1}^{q} y_i^2 + \sum\limits_{i=1}^{r} (x_i^2 - \tan y_i)^2$	convex upper problem
	$\min\limits_{y} f(x,y) = \sum\limits_{i=r+1}^{p} x_i^2 + \sum\limits_{i=r+1}^{q} (y_i^2 - \cos 2\pi y_i)$	convex level problem
	$+ \sum\limits_{i=1}^{r} (x_i^2 - \tan y_i)^2$	the two levels cooperate with each other
	$x_i \in [-5,10], \quad i = 1,2,\cdots,p;$	without constrains at two levels
	$y_i \in [-\pi/2, \pi/2], \quad i = 1,2,\cdots,r;$	
	$y_i \in [-5,10], \quad i = r+1,2,\cdots,q.$	
4	$\min\limits_{x,y} F(x,y) = \sum\limits_{i=r+1}^{p} x_i^2 - \sum\limits_{i=r+1}^{q} y_i^2 - \sum\limits_{i=1}^{r} (\lvert x_i\rvert - \log(1+y_i))^2$	convex upper problem
	$\min\limits_{y} f(x,y) = \sum\limits_{i=r+1}^{p} x_i^2 + \sum\limits_{i=r+1}^{q} (y_i^2 - \cos 2\pi y_i)$	multi-modality at the lower level
	$+ \sum\limits_{i=1}^{r} (\lvert x_i\rvert - \log(1+y_i))^2 + q$	the two levels conflict with each other
	$x_i \in [-5,10], \quad i = 1,2,\cdots,r;$	without constrains at two levels
	$x_i \in [-5,10], \quad i = r+1,2,\cdots,p;$	
	$y_i \in [0,e], \quad i = 1,2,\cdots,r;$	
	$y_i \in [-5,10], \quad i = r+1,2,\cdots,q.$	
5	$\min\limits_{x,y} F(x,y) = \sum\limits_{i=1}^{p} x_i^2 - \sum\limits_{i=r+1}^{q} ((y_{i+1} - y_i^2) + (y_i - 1)^2)$	convex upper problem
	$- \sum\limits_{i=1}^{r} (\lvert x_i\rvert - y_i^2)^2$	multi-modality at the lower level
	$\min\limits_{y} f(x,y) = \sum\limits_{i=1}^{p} x_i^2 + \sum\limits_{i=r+1}^{q} ((y_{i+1} - y_i^2) + (y_i - 1)^2)$	the two levels conflict with each other
	$+ \sum\limits_{i=1}^{r} (\lvert x_i\rvert - y_i^2)^2$	without constrains at two levels
	$x_i \in [-5,10], \quad i = 1,2,\cdots,p; \quad y_i \in [-5,10], \quad i = 1,2,\cdots,q.$	

Table 2: Test problems and their properties.

No.	Problem	Properties
6	$\min\limits_{x,y} F(x,y) = \sum\limits_{i=r+1}^{p} x_i^2 - \sum\limits_{i=r+1}^{q} y_i^2 + \sum\limits_{i=q+1}^{q+s} y_i^2 - \sum\limits_{i=1}^{r}(x_i - y_i)^2$	many global solutions at the lower level
	$\min\limits_{y} f(x,y) = \sum\limits_{i=r+1}^{p} x_i^2 + \sum\limits_{i=r+1}^{q} y_i^2 + \sum\limits_{i=q+1}^{q+s}(y_{i+1} - y_i)^2$	the two levels conflict with each other
	$+ \sum\limits_{i=1}^{r}(x_i - y_i)^2$	without constrains at two levels
	$x_i \in [-5,10], \quad i = 1,2,\cdots,p;$	
	$y_i \in [-\pi/2, \pi/2], \quad i = 1,2,\cdots,q+s.$	
7	$\min\limits_{x,y} F(x,y) = 1 + \frac{1}{400}\sum\limits_{i=r+1}^{p} x_i^2 - \prod\limits_{i=r+1}^{p}\cos\frac{x_i}{\sqrt{i}} - \sum\limits_{i=r+1}^{q} y_i^2$	multi-modality at the upper level
	$+ \sum\limits_{i=1}^{r} x_i^2 - \sum\limits_{i=1}^{r}(x_i - \log y_i)^2$	the two levels conflict with each other
	$\min\limits_{y} f(x,y) = \sum\limits_{i=r+1}^{p} x_i^3 + \sum\limits_{i=r+1}^{q} y_i^2 + \sum\limits_{i=1}^{r}(x_i - \log y_i)^2$	without constrains at two levels
	$x_i \in [-5,1], \quad i = 1,2,\cdots,r;$	
	$x_i \in [-5,10], \quad i = r+1,2,\cdots,p;$	
	$y_i \in [0,e], \quad i = 1,2,\cdots,r;$	
	$y_i \in [-5,10], \quad i = r+1,2,\cdots,q.$	
8	$\min\limits_{x,y} F(x,y) = 20 + e - 20\exp(-0.2\sqrt{\frac{1}{p-r}\sum\limits_{i=r+1}^{p} x_i^2})$	multi-modality at the upper level
	$- \exp(\frac{1}{p-r}\sum\limits_{i=r+1}^{p}\cos 2\pi x_i)$	multi-modality at the lower level
	$- \sum\limits_{i=r+1}^{q}((y_{i+1} - y_i^2) + (y_i - 1)^2)$	the two levels conflict with each other
	$+ \sum\limits_{i=1}^{r} x_i^2 - \sum\limits_{i=1}^{r}(x_i - y_i^3)^2$	without constrains at two levels
	$\min\limits_{y} f(x,y) = \sum\limits_{i=r+1}^{p}\lvert x_i\rvert + \sum\limits_{i=r+1}^{q}((y_{i+1} - y_i^2) + (y_i - 1)^2)$	
	$+ \sum\limits_{i=1}^{r}(x_i - y_i^3)^2$	
	$x_i \in [-5,10], \quad i = 1,2,\cdots,p;$	
	$y_i \in [-5,10], \quad i = 1,2,\cdots,r;$	
9	$\min\limits_{x,y} F(x,y) = \sum\limits_{i=r+1}^{p} x_i^2 - \sum\limits_{i=r+1}^{q} y_i^2 - \sum\limits_{i=1}^{p}(x_i - \log(y_i - 1))^2$	the two levels conflict with each other
	$\min\limits_{y} f(x,y) = \sum\limits_{i=1}^{p} x_i^2 + \sum\limits_{i=r+1}^{q} y_i^2 + \sum\limits_{i=1}^{p}(x_i - \log(y_i - 1))^2$	without constrains at two levels
	$x_i \in [-5,1], \quad i = 1,2,\cdots,r;$	
	$x_i \in [-5,10], \quad i = r+1,2,\cdots,p;$	
	$y_i \in [-1,-1+e], \quad i = 1,2,\cdots,r;$	
	$y_i \in [-5,10], \quad i = r+1,2,\cdots,q.$	

Table 3: Test problems and its properties.

No.	Problem	Properties
10	$\min\limits_{x,y} F(x,y) = \sum\limits_{i=1}^{p}(x_i-2)^2 + \sum\limits_{i=r+1}^{q} y_i^2 - \sum\limits_{i=1}^{r}(x_i-\tan y_i)^2$	the two levels conflict with each other
	s.t. $G_1(x,y) = \sum\limits_{i=r+1,i\neq j}^{p} x_i^3 + \sum\limits_{i=1}^{r} x_i^3 - x_j \leqslant 0 \quad j = r+1,2,\cdots,p$	constraints at both levels
	$G_2(x,y) = \sum\limits_{i=1,i\neq j}^{r} x_i^3 + \sum\limits_{i=r+1}^{p} x_i^3 - x_j \leqslant 0 \quad j = 1,2,\cdots,r$	
	$\min\limits_{y} f(x,y) = \sum\limits_{i=r+1}^{p} x_i^2 + \sum\limits_{i=r+1}^{q}(y_i-2)^2 + \sum\limits_{i=1}^{r}(x_i-\tan y_i)^2$	
	s.t. $g_1(x,y) = \sum\limits_{i=r+1,i\neq j}^{q} y_j^3 - y_j \leqslant 0 \quad j = r+1,2,\cdots,q$	
	$x_i \in [-5,10], \quad i = 1,2,\cdots,p;$	
	$y_i \in [-\pi/2,\pi/2], \quad i = 1,2,\cdots,r;$	
	$y_i \in [-5,10], \quad i = r+1,r+2,\cdots,q.$	
11	$\min\limits_{x,y} F(x,y) = \sum\limits_{i=1}^{p} x_i^2 - \sum\limits_{i=r+1}^{q} y_i^2 - \sum\limits_{i=1}^{r}(x_i-\log y_i)^2$	many global solutions at the lower level
	s.t. $G_1(x,y) = \frac{1}{\sqrt{r}} + \log y_j - x_j \leqslant 0 \quad j = r+1,2,\cdots,p$	the two levels conflict with each other
	$\min\limits_{y} f(x,y) = \sum\limits_{i=r+1}^{p} x_i^2 + \sum\limits_{i=r+1}^{q} y_i^2 + \sum\limits_{i=1}^{r}(x_i-\log y_i)^2$	constraints at both levels
	s.t. $g_1(x,y) = 1 - \sum\limits_{i=1}^{r}(x_i-\log y_i)^2 \leqslant 0 \quad j = r+1,2,\cdots,p$	
	$x_i \in [-5,10], \quad i = 1,2,\cdots,p;$	
	$y_i \in [1/e,e], \quad i = 1,2,\cdots,r;$	
	$y_i \in [-5,10], \quad i = r+1,r+2,\cdots,q.$	
12	$\min\limits_{x,y} F(x,y) = \sum\limits_{i=r+1}^{p}(x_i-2)^2 + \sum\limits_{i=r+1}^{q} y_i^2 + \sum\limits_{i=1}^{r}\tan\lvert y_i\rvert$	many global solutions at the lower level
	$- \sum\limits_{i=1}^{r}(x_i-y_i)^2$	the two levels conflict with each other
	s.t. $G_1(x,y) = \tan y_i - x_i \leqslant 0 \quad i = 1,2,\cdots,r$	constraints at both levels
	$G_2(x,y) = \sum\limits_{i=r+1,i\neq j}^{p} x_i^3 + \sum\limits_{i=1}^{r} x_i^3 - x_j \leqslant 0 \quad j = r+1,r+2,\cdots,p$	
	$G_3(x,y) = \sum\limits_{i=1,i\neq j}^{r} x_i^3 + \sum\limits_{i=r+1}^{q} x_i^3 - x_j \leqslant 0 \quad j = 1,2,\cdots,r$	
	$\min\limits_{y} f(x,y) = \sum\limits_{i=r+1}^{p} x_i^2 + \sum\limits_{i=r+1}^{q}(y_i-2)^2 + \sum\limits_{i=1}^{r}(x_i-\tan y_i)^2$	
	s.t. $g_1(x,y) = 1 - \sum\limits_{i=1}^{r}(x_i-\tan y_i)^2 \leqslant 0$	
	$g_2(x,y) = \sum\limits_{i=r+1,i\neq j}^{p}(y_j)^3 - y_j \leqslant 0 \quad j = r+1,2,\cdots,q$	
	$x_i \in [-14.1,14.1], \quad i = 1,2,\cdots,r;$	
	$x_i \in [-5,10], \quad i = r+1,r+2,\cdots,p;$	
	$y_i \in [-1.5,1.5], \quad i = 1,2,\cdots,r;$	
	$y_i \in [-5,10], \quad i = r+1,r+2,\cdots,q;$	

Table 4: Comparison of the function evaluations (FE) for the upper level (UL) and the lower level (LL) from 11 runs for 5-dimensional test problems.

No.	Method	the best FE		the median FE		the worst FE	
		LL	UL	LL	UL	LL	UL
1	CCPSO	103464	386	192174(8.46)	712(3.56)	242250	1046
	The method in [40]	859875	1124	1625794	2534	2091127	3512
2	CCPSO	99786	478	138481(10.93)	642(3.61)	203458	1021
	The method in [40]	1123645	1467	1513514	2316	2103519	3592
3	CCPSO	192736	433	250662(5.79)	714(3.26)	215384	1023
	The method in [40]	887673	1143	1452379	2326	2013258	2976
4	CCPSO	69298	243	122884(9.16)	572(2.84)	137011	725
	The method in [40]	583769	746	1125625	1624	1359875	2132
5	CCPSO	583769	519	238302(8.72)	892(3.27)	267553	1248
	The method in [40]	1295156	1684	2078546	2917	2513763	3579
6	CCPSO	138232	537	208760(11.38)	774(3.81)	269872	1172
	The method in [40]	1278636	1578	2375682	2948	3786498(×)	4248(×)
7	CCPSO	233310	494	259457(6.30)	856(2.90)	318746	1126
	The method in [40]	966238	1376	1635278	2476	2435994(×)	3858(×)
8	CCPSO	230724	756	286808(9.75)	1096(3.84)	323172	1546
	The method in [40]	1489672	2168	2796316	4212	5294734(×)	5986(×)
9	CCPSO	977846	1189	1725678	3425	3412248(×)	4628(×)
	The method in [40]	––	––	––	––	––	––
10	CCPSO	977846	1189	1725678	3425	3412248(×)	4628(×)
	The method in [40]	––	––	––	––	––	––
11	CCPSO	977846	1189	1725678	3425	3412248(×)	4628(×)
	The method in [40]	––	––	––	––	––	––
12	CCPSO	977846	1189	1725678	3425	3412248(×)	4628(×)
	The method in [40]	––	––	––	––	––	––

Table 5: Comparison of the accuracy for the upper and lower levels, and the lower level calls from 11 runs for 5-dimensional test problems.

No.	Method	the best FE		the median	the average FE
		UL accuracy	LL accuracy	LL calls	LL calls
1	CCPSO	0.000529	0.000061	239	340.90
	The method in [40]	0.000615	0.000112	763	628.96
2	CCPSO	0.000207	0.000045	225	342.31
	The method in [40]	0.000224	0.000216	638	625.33
3	CCPSO	0.000079	0.000029	216	273.77
	The method in [40]	0.000051	0.000037	686	579.62
4	CCPSO	0.000048	0.000023	217	374.52
	The method in [40]	0.000068	0.000074	623	693.76
5	CCPSO	0.000084	0.000045	333	236.13
	The method in [40]	0.000071	0.000063	1012	483.36
6	CCPSO	0.000139	0.000056	249	245.48
	The method in [40]	0.000217	0.000103	714	616.07
7	CCPSO	0.000089	0.000041	233	310.54
	The method in [40]	0.000365	0.000983	672	572.53
8	CCPSO	0.000893	0.000076	360	211.29
	The method in [40]	0.000572	0.000153	1127	544.39
9	CCPSO	0.000531	0.000062	184	236.01
	The method in [40]	0.000617	0.000154	576	552.06
10	CCPSO	0.009156	0.009876	271	292.55
	The method in [40]	0.087354	0.091426	793	376.75
11	CCPSO	0.003219	0.001534	1211	1299.07
	The method in [40]	0.043512	0.038767	6176	2317.98
12	CCPSO	0.007463	0.000102	264	8835.99
	The method in [40]	0.087312	0.004138	813	18695.66

Table 6: Comparison of the function evaluations (FE) for the upper level (UL) and the lower level (LL) from 11 runs for 10-dimensional test problems.

No.	Method	the best FE		the median FE		the worst FE	
		LL	UL	LL	UL	LL	UL
1	CCPSO	103464	386	192174(8.46)	712(3.56)	242250	1046
	The method in [40]	859875	1124	1625794	2534	2091127	3512
2	CCPSO	99786	478	138481(10.93)	642(3.61)	203458	1021
	The method in [40]	1123645	1467	1513514	2316	2103519	3592
3	CCPSO	192736	433	250662(5.79)	714(3.26)	215384	1023
	The method in [40]	887673	1143	1452379	2326	2013258	2976
4	CCPSO	69298	243	122884(9.16)	572(2.84)	137011	725
	The method in [40]	583769	746	1125625	1624	1359875	2132
5	CCPSO	168304	519	238302(8.72)	892(3.27)	267553	1248
	The method in [40]	1295156	1684	2078546	2917	2513763	3579
6	CCPSO	138232	537	208760(11.38)	774(3.81)	269872	1172
	The method in [40]	1278636	1578	2375682	2948	3786498(×)	4248(×)
7	CCPSO	233310	494	259457(6.30)	856(2.90)	318746	1126
	The method in [40]	966238	1376	1635278	2476	2435994(×)	3858(×)
8	CCPSO	230724	756	286808(9.75)	1096(3.84)	323172	1546
	The method in [40]	1489672	2168	2796316	4212	5294734(×)	5986(×)
9	CCPSO	977846	1189	1725678	3425	3412248(×)	4628(×)
	The method in [40]	−−	−−	−−	−−	−−	−−
10	CCPSO	977846	1189	1725678	3425	3412248(×)	4628(×)
	The method in [40]	−−	−−	−−	−−	−−	−−
11	CCPSO	977846	1189	1725678	3425	3412248(×)	4628(×)
	The method in [40]	−−	−−	−−	−−	−−	−−
12	CCPSO	977846	1189	1725678	3425	3412248(×)	4628(×)
	The method in [40]	−−	−−	−−	−−	−−	−−

Table 7: Comparison of the accuracy for the upper and lower levels, and the lower level calls from 11 runs for 10-dimensional test problems.

No.	Method	The best FE		The median	The average FE
		UL accuracy	LL accuracy	LL calls	LL calls
1	CCPSO	0.008794	0.003125	712	269.91
	The method in [40]	0.005841	0.002174	2534	641.59
2	CCPSO	0.003126	0.001975	642	215.70
	The method in [40]	0.001973	0.001182	2316	653.50
3	CCPSO	0.008914	0.004126	714	351.07
	The method in [40]	0.008717	0.002633	2278	655.76
4	CCPSO	0.005939	0.003912	572	214.83
	The method in [40]	0.007379	0.002708	1598	685.43
5	CCPSO	0.001938	0.000987	892	267.15
	The method in [40]	0.002652	0.003487	2890	716.82
6	CCPSO	0.005413	0.000792	774	269.72
	The method in [40]	0.008463	0.006952	2936	768.34
7	CCPSO	0.006734	0.000317	856	303.10
	The method in [40]	0.009216	0.000527	2394	654.34
8	CCPSO	0.002145	0.000081	1096	261.69
	The method in [40]	0.006935	0.000337	4188	647.12
9	CCPSO	0.061873	0.016795	3425	503.85
	The method in [40]	––	––	––	––
10	CCPSO	0.035127	0.071963	2876	500.03
	The method in [40]	––	––	––	––
11	CCPSO	0.389578	0.216874	2962	1005.51
	The method in [40]	––	––	––	––
12	CCPSO	0.513681	0.009765	3124	708.57
	The method in [40]	––	––	––	––

5. Application of CCPSO algorithm for a practical problem

Taking into account water rights and emission rights to establish a water market system based on watershed management is the most effective economic method to solve water shortage and pollution of watershed. In this water market system, the watershed management agency usually maximizes his profits, whereas each user's goal is to maximize its own profit. The problem involves uncertainty and is bilevel in nature, thus, a watershed water trading decision-making model based on bilevel programming is constructed, in which the upper decision-maker is the watershed management agency as the planning, controlling and coordinating center of watershed and each user is the lower decision-maker. We present a deterministic version of the case study from [47].

$$
\begin{aligned}
&\max_{w,t,r_1,g_1,r_2,g_2} V_T = 0.4w + t(q_1+q_2) + f_1 + f_2,\\
&\quad \text{s.t. } r_1 + r_2 + w = 90,\\
&\qquad q_1 + q_2 + w \leqslant 90,\\
&\qquad g_1 + g_2 = 20,\\
&\qquad r_1 \geqslant 38, r_2 \geqslant 42, g_1 \geqslant 7, g_2 \geqslant 8, w \geqslant 6, 0.3 \leqslant t \leqslant 2,\\
&\max_{q_1,l_1} f_1 = 0.7q_1 - q_1 t - 0.3(45-q_1)^2 + (r_1-q_1)[0.9 - 0.01(r_1+r_2-q_1-q_2)]\\
&\qquad\qquad - 0.2(0.2q_1 - l_1)^2 + (g_1-l_1)[0.8 - 0.01(g_1+g_2-l_1-l_2)],\\
&\max_{q_2,l_2} f_2 = 0.8q_2 - q_2 t - 0.2(47-q_2)^2 + (r_2-q_2)[0.9 - 0.01(r_1+r_2-q_1-q_2)]\\
&\qquad\qquad - 0.1(0.3q_2 - l_2)^2 + (g_2-l_2)[0.8 - 0.01(g_1+g_2-l_1-l_2)],\\
&\quad \text{s.t. } l_1 + l_2 \leqslant 20,\\
&\qquad q_1 \geqslant 0, l_1 \geqslant 0,\\
&\qquad q_2 \geqslant 0, l_2 \geqslant 0,
\end{aligned}
$$

where q_1 and q_1 are actual water intake volume of water consumer A and water consumer B, respectively. l_1 and l_1 are waste water discharge volume of two users, respectively. r_1 and r_2 are water rights of two users, respectively. g_1 and g_1 are emission rights of two users, respectively. w is ecological water requirement of watershed. t is water resource fees.

The optimal solution of the problem is obtained by Algorithm 3.2. Note that, the optimal solution of the lower level is obtained by solving multiobjective problem using CCPSO combining the fast non-

Table 8: Results of the Spacing (SP) metrics for the above four examples.

Parameter	CCPSO	The best results in [40]	The best results in [45]
q_1	42.0000	42.0000	41.2016
q_2	42.0000	41.9680	42.5388
l_1	7.0216	6.9984	6.4772
l_2	9.0013	9.1751	9.1611
r_1	40.0000	39.9679	39.4861
r_2	44.0000	44.0000	44.2542
g_1	9.0000	9.0000	9.0015
g_2	11.0000	11.0000	10.9985
ω	6.0000	6.0321	6.02596
t	0.3000	0.3000	0.3226
F	59.0613	58.97837	54.4482
f_1	13.4240	13.40286	10.964
f_2	18.0244	17.97226	17.964

dominated sort technology [13]. Table 8 shows the obtained results compared with the results in the relevant references. From Table 8, we can see that the solutions found by the proposed method are better than those given in the references. Therefore, the proposed method can also be applied to the practical problem. In this problem, the population sizes of each species for both levels were chosen as 10 for the problem and we execute the algorithm in 10 independent runs.

6. Conclusion and future works

In this paper, the CCPSO is proposed to solve the complex bilevel programming problem (CBLP) in which the evolutionary stagnation detection technology is adopted. The algorithm is tested on two sets of twelve problems which are scalable in terms of number of variables. The first set is the 5-dimensional instance of these problems and the second set is the 10-dimensional instance of these problems. The results indicate that the global convergence of the proposed algorithm is greatly improved, as well as the evolutionary stagnation detection technology adopted in both levels greatly enhances the speed of convergence of the algorithm. In the future works, the following problems will be considered: (1) Constraint handling. It is possible that the infeasible solution with good performance near the optimum can induce global searching to the boundary, so we will further discuss how to efficiently use the infeasible solution. This kind of discussion could improve the performance of our CCPSO, particularly when the optimal front lies on the boundaries between the feasible and infeasible regions. (2) Algorithm parallelization. We will design the parallel CCPSO for CBLP so as to improve further the computation efficiency.

Acknowledgment

This work is supported by the National Science Foundation of China (61673006), the Young Project of Hubei Provincial Department of Education (Q20141304), the Dr. Start-up fund by the Yangtze University (2014) and the Basic and Advanced Research Project of Chongqing (cstc2016jcyjA0239, cstc2015jcyjBX0131).

References

[1] E. Aiyoshi, K. Shimuzu, *A solution method for the static constrained Stackelberg problem via penalty method*, IEEE Trans. Automat. Control, **29** (1984), 1112–1114. 1
[2] M. A. Amouzegar, *A global optimization method for nonlinear bilevel programming problems*, IEEE Trans. Systems Man Cybernet., **29** (1999), 771–777. 1
[3] J. F. Bard, *An algorithm for solving the general bilevel programming problem*, Math. Oper. Res., **8** (1983), 260–272. 1
[4] J. F. Bard, *Some properties of the bilevel programming problem*, J. Optim. Theory Appl., **68** (1991), 371–378. 1
[5] J. F. Bard, *Practical bilevel optimization*, Algorithms and applications, Nonconvex Optimization and its Applications, Kluwer Academic Publishers, Dordrecht, (1998). 1
[6] J. F. Bard, J. E. Falk, *An explicit solution to the multilevel programming problem*, Mathematical programming with parameters and multilevel constraints, Comput. Oper. Res., **9** (1982), 77–100. 1
[7] O. Ben-Ayed, C. E. Blair, *Computational difficulties of bilevel linear programming*, Oper. Res., **38** (1990), 556–560. 1
[8] H. I. Calvete, C. Gaté, P. M. Mateo, *A new approach for solving linear bilevel problems using genetic algorithms*, European J. Oper. Res., **188** (2008), 14–28. 1
[9] H. I. Calvete, C. Galé, M. J. Oliveros, *Bilevel model for productiondistribution planning solved by using ant colony optimization*, Comput. Oper. Res., **38** (2011), 320–327. 1
[10] J.-W. Chen, Q. H Ansari, Y.-C. Liou, J.-C. Yao, *A proximal point algorithm based on decomposition method for cone constrained multiobjective optimization problems*, Comput. Optim. Appl., **65** (2016), 289–308. 1
[11] B. Colson, P. Marcotte, G. Savard, *Bilevel programming: a survey*, 4OR, **3** (2005), 87–107. 1
[12] B. Colson, P. Marcotte, G. Savard, *An overview of bilevel optimization*, Ann. Oper. Res., **153** (2007), 235–256. 1
[13] K. Deb, S. Agrawalan, A. Pratap, T. Meyarivan, *A fast and elitist multiobjective genetic algorithm: NSGA-II*, IEEE Trans. Evol. Comput., **6** (2002), 182–197. 5
[14] S. Dempe, *Foundations of bilevel programming*, Nonconvex Optimization and its Applications, Kluwer Academic Publishers, Dordrecht, (2002). 1
[15] S. Dempe, *Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints*, Optimization, **52** (2003), 333–359. 1
[16] T. A. Edmunds, J. F. Bard, *Algorithms for nonlinear bilevel mathematical programs*, IEEE Trans. Systems Man Cybernet., **21** (1991), 83–89. 1

[17] J. B. Etoa Etoa, *Solving quadratic convex bilevel programming problems using a smoothing method*, Appl. Math. Comput., **217** (2011), 6680–6690. 1
[18] J. E. Falk, J.-M. Liu, *On bilevel programming*, I, General nonlinear cases, Math. Programming, **70** (1995), 47–72. 1
[19] Y. Gao, G.-Q. Zhang, J. Lu, H.-M. Wee, *Particle swarm optimization for bi-level pricing problems in supply chains*, J. Global Optim., **51** (2011), 245–254. 1
[20] Y. Gao, G.-Q. Zhang, J. Ma, J. Lu, *A λ-cut and goal-programming-based algorithm for fuzzy-linear multiple-objective bilevel optimization*, IEEE Trans. Fuzzy Syst., **18** (2010), 1–13. 1
[21] M. Gendreau, P. Marcotte, G. Savard, *A hybrid tabu-ascent algorithm for the linear bilevel programming problem*, Hierarchical and bilevel programming, J. Global Optim., **8** (1996), 217–233. 1
[22] S. R. Hejazi, A. Memariani, G. Jahanshahloo, M. M. Sepehri, *Linear bilevel programming solution by genetic algorithm*, Comput. Oper. Res., **29** (2001), 1913–1925. 1
[23] Y. Ishizuka, E. Aiyoshi, *Double penalty method for bilevel optimization problems*, Hierarchical optimization, Ann. Oper. Res., **34** (1992), 73–88. 1
[24] R. G. Jeroslow, *The polynomial hierarchy and a simple model for competitive analysis*, Math. Programming, **32** (1985), 146–164. 1
[25] Y. Jiang, X.-Y. Li, C.-C. Hang, X.-N. Wu, *Application of particle swarm optimization based on CHKS smoothing function for solving nonlinear bilevel programming problem*, Appl. Math. Comput., **129** (2013), 4332–4339. 1
[26] J. Kennedy, R. C. Eberhart, Y.-H. Shi, *Swarm intelligence*, Morgan Kaufmann Publishers, San Francisco, CA, (2001). 1
[27] 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*, Appl. Math. Model., **35** (2011), 3905–3917. 1
[28] R. J. Kuo, C. C. Huang, *Application of particle swarm optimization algorithm for solving bi-level linear programming problem*, Comput. Math. Appl., **58** (2009), 678–685. 1
[29] K.-M. Lan, U.-P. Wen, H.-S. Shih, E. S. Lee, *A hybrid neural network approach to bilevel programming problems*, Appl. Math. Lett., **20** (2007), 880–884. 1
[30] X.-Y. Li, P. Tian, X.-P. Min, *A hierarchical particle swarm optimization for solving bilevel programming problems*, Proceedings of the 8th International Conference on Artificial Intelligence and Soft Computing (ICAISC), Poland, Lecture Notes in Computer Science, **2016** (2006), 1169–1178. 1
[31] Y.-B. Lv, T.-S. Hu, G.-M. Wang, Z.-P. Wan, *A penalty function method based on Kuhn-Tucker condition for solving linear bilevel programming*, Appl. Math. Comput., **188** (2007), 808–813. 1
[32] Y.-B. Lv, T.-S. Hu, G.-M. Wang, Z.-P. Wan, *A neural network approach for solving nonlinear bilevel programming problem*, Comput. Math. Appl., **55** (2008), 2823–2829. 1
[33] R. Mathieu, L. Pittard, G. Anandalingam, *Genetic algorithm based approach to bi-level linear programming*, RAIRO Rech. Opér., **28** (1994), 1–21. 1
[34] A. Migdalas (Ed.), P. M. Pardalos (Ed.), P. Värbrand (Ed.), *Multilevel optimization: algorithms and applications*, Nonconvex Optimization and its Applications, Kluwer Academic Publishers, Dordrecht, (1998). 1
[35] J. Rajesh, K. Gupta, H. S. Kusumakar, V. K. Jayaraman, B. D. Kulkarni, *A tabu search based approach for solving a class of bilevel programming problems in chemical engineering*, J. Heuristics, **9** (2003), 307–319. 1
[36] K. H. Sahin, A. R. Ciric, *A dual temperature simulated annealing approach for solving bilevel programming problems*, Comput. Chem. Eng., **23** (1998), 11–25. 1
[37] G. Savard, J. Gauvin, *The steepest descent direction for the nonlinear bilevel programming problem*, Oper. Res. Lett., **15** (1994), 265–272. 1
[38] K. Shimizu, E. Aiyoshi, *A new computational method for Stackelberg and min-max problems by use of a penalty method*, IEEE Trans. Automat. Control, **26** (1981), 460–466. 1
[39] K. Shimizu, Y. Ishizuka, J. F. Bard, *Nondifferentiable and two-level mathematical programming*, Kluwer Academic Publishers, Boston, MA, (1997). 1
[40] A. Sinha, P. Malo, K. Deb, *Test problem construction for single-objective bilevel optimization*, Evol. Comput., **22** (2014), 439–477. 1, 4.1, 4.2, 4.2, 4.2, 4, 5, 6, 7, 8
[41] L. N. Vicente, P. H. Calamai, *Bilevel and multilevel programming: a bibliography review*, J. Global Optim., **5** (1994), 291–306. 1
[42] L. Vicente, G. Savard, J. Júdice, *Descent approaches for quadratic bilevel programming*, J. Optim. Theory Appl., **81** (1994), 379–399. 1
[43] Z.-P. Wan, G.-M. Wang, B. Sun, *A hybrid intelligent algorithm by combining particle swarm optimization with chaos searching technique for solving nonlinear bilevel programming problems*, Swarm Evol. Comput., **8** (2013), 26–32. 1
[44] Y.-P. Wang, Y.-C. Jiao, H. Li, *An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme*, IEEE Trans. Systems Man Cybernet., **35** (2005), 221–232. 1
[45] G.-M. Wang, X.-J. Wang, Z.-P. Wan, S.-H. Jia, *An adaptive genetic algorithm for solving bilevel linear programming problem*, Appl. Math. Mech. (English Ed.), **28** (2007), 1605–1612. 1, 8
[46] U. P. Wen, A. D. Huang, *A simple tabu search method to solve the mixed-integer linear bilevel programming problem*, European J. Oper. Res., **88** (1996), 563–571. 1
[47] C.-Y. Wu, Y.-Z. Zhao, *Watershed water trading decision-making model based on bi-level programming*, Oper. Res. Manag. Sci., **20** (2011), 30–37. 1, 5

[48] S. B. Yaakob, J. Watada, *A hybrid intelligent algorithm for solving the bilevel programming models*, Knowledge-Based Intell. Inf. Eng. Syst., **6277** (2010), 485–494. 1

[49] S. B. Yaakob, J. Watada, *Double-layered hybrid neural network approach for solving mixed integer quadratic bilevel problems*, Integr. Uncertain. Manag. Appl., **68** (2010), 221–230. 1

[50] G.-Q. Zhang, G.-L. Zhang, Y. Gao, J. Lu, *Competitive strategic bidding optimization in electricity markets using bilevel programming and swarm technique*, IEEE Trans. Ind. Electron., **58** (2011), 2138–2146. 1