Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 471952, 14 pages
doi:10.1155/2012/471952

Research Article

An Evolutionary Algorithm for Solving Bilevel Programming Problems Using Duality Conditions

Hecheng Li[1,2] and Lei Fang[3]

[1] *Department of Mathematics, Key Laboratory of Tibetan Information Processing of Ministry of Education, Qinghai Normal University, Xining 810008, China*
[2] *Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China*
[3] *Business School, Nankai University, Tianjin 300071, China*

Correspondence should be addressed to Lei Fang, fanglei@nankai.edu.cn

Received 31 March 2012; Accepted 12 September 2012

Academic Editor: Yuping Wang

Bilevel programming is characterized by two optimization problems located at different levels, in which the constraint region of the upper level problem is implicitly determined by the lower level problem. This paper is focused on a class of bilevel programming with a linear lower level problem and presents a new algorithm for solving this kind of problems by combining an evolutionary algorithm with the duality principle. First, by using the prime-dual conditions of the lower level problem, the original problem is transformed into a single-level nonlinear programming problem. In addition, for the dual problem of the lower level, the feasible bases are taken as individuals in population. For each individual, the values of dual variables can be obtained by taking the dual problem into account, thus simplifying the single-level problem. Finally, the simplified problem is solved, and the objective value is taken as the fitness of the individual. Besides, when nonconvex functions are involved in the upper level, a coevolutionary scheme is incorporated to obtain global optima. In the computational experiment, 10 problems, smaller or larger-scale, are solved, and the results show that the proposed algorithm is efficient and robust.

1. Introduction

In a hierarchical system involving two levels of decision making with different objectives, the decision makers are divided into two categories: the upper level (leader) and the lower level (follower). The upper level controls a subset of decision variables, while the lower level controls remaining decision variables. To each decision made by the upper level, the lower level responds by a decision optimizing its objective function over a constraint set which depends upon the decision of the upper level. Bilevel programming (BLPP), as a hierarchical optimization problem, has been proposed for dealing with this kind of decision processes,

which is characterized by the existence of two optimization problems, the upper level and the lower level problems. The distinguished feature of this kind of problems is that the constraint region of the upper level problem is implicitly determined by the lower level problem, and any feasible solution must satisfy the optimality of the lower level problem as well as all constraints. The general bilevel programming problem can be formulated as follows:

$$\begin{aligned} &\min_{x\in X} \quad F(x,y), \\ &\text{s.t.} \quad G(x,y)\le 0, \\ &\min_{y\in Y} \quad f(x,y), \\ &\text{s.t.} \quad g(x,y)\le 0, \end{aligned} \tag{1.1}$$

where $x \in R^n$ and $y \in R^m$ are called the upper level and the lower level variables, respectively. In a similar way, $F(f) : R^n \times R^m \to R$ is known as the upper (lower) level objective function, whereas the vector-valued functions $G : R^n \times R^m \to R^p$ and $g : R^n \times R^m \to R^q$ are usually referred to as the upper level and the lower level constraints, respectively. In addition, the sets X and Y place additional constraints on the variables, such as upper and lower bounds or integrality requirements. In problem (1.1),

$$\begin{aligned} &\min_{x\in X} \quad F(x,y), \\ &\text{s.t.} \quad G(x,y)\le 0, \\ &\min_{y\in Y} \quad f(x,y), \\ &\text{s.t.} \quad g(x,y)\le 0 \end{aligned} \tag{1.2}$$

are called the upper level and the lower level problems, respectively.

In fact, it is very difficult to solve BLPPs. At first, BLPPs are strongly NP-hard [1]. In addition, the lower level variables $(y_1, \ldots, y_m)$ always are determined by the upper level variables $x = (x_1, \ldots, x_n)$. Under some relevant assumptions, the relationship between x and y can be denoted by $y = y(x)$. Generally speaking, the solution function $y(x)$ is nonconvex and even nondifferentiable, which makes BLPP harder to handle.

BLPP is applied extensively to a variety of real-world areas such as economy, engineering, and management [1, 2], and some efficient algorithms and theoretical results are developed for BLPPS, linear or nonlinear [3–13]. For the linear bilevel programming problem, the optimal solutions can be achieved at some extreme point of the constraint region. Based on the characteristics, some exact algorithmic approaches are given, such as kth best method and branch-and-bound algorithm [1, 14–16]. For nonlinear bilevel programming, there exist few literature [2, 8], and the most procedures can only guarantee the local optima are obtained. In recent years, evolutionary algorithms (EAs) with global convergence are often adopted to solve BLPPs. Kuo developed a particle swarm optimization (PSO) algorithm for linear BLPPs [17], in which only the upper space is searched by the designed PSO algorithm. Wang et al. discussed nonlinear BLPPs with nonconvex objective functions and proposed a new evolutionary algorithm [18]. Deb presented a hybrid evolutionary-cum-local-search-based

algorithm for general nonlinear BLPPs [19]. In Deb's method, both the upper and the lower level problems are solved by EAs. These proposed EAs share a common hierarchical structure, that is to say, EAs are used to explore the upper level spaces, whereas for each fixed value of the upper level variable, one has to solve the lower level problem. It means that these algorithms are time consuming when the upper level space become larger or the lower level problem is larger scale. In order to overcome the shortcomings, some optimality results are applied in the design of algorithmic approaches. Based on the extreme points of constraint region, Calvete designed a genetic algorithm for linear BLPPs [20], but it is hard to extend the methods to nonlinear cases except for some special cases involving concave functions. Besides, the K-K-T conditions are also applied to convert BLPPs into single-level problems [8, 21].

In this paper, we discuss a class of BLPPs with linear lower level problem, and based on the optimality results of linear programming, develop an efficient evolutionary algorithm which begins with the bases of the follower's dual problem. First, the lower level problem is replaced by the prime-dual conditions. After doing so, the original problem is transformed into an equivalent single-level nonlinear programming. Then, in order to reduce the number of variables and improve the performance of the algorithm, we encode individuals by taking the bases of the duality problem. For each individual given, the dual variables can be solved, and some constraints can also be removed from the original single-level problem. Finally, the simplified nonlinear programming is solved, and the objective value is taken as the fitness of the individual. The proposed algorithm keeps the search space finite and can be used to deal with some nonlinear BLPPs.

This paper is organized as follows. Some notations and transformation are presented in Section 2, an evolutionary algorithm is given based on the duality principle in Section 3, and a revised algorithm is proposed for handling nonconvex cases in Section 4. Experimental results and comparison are presented in Section 5. We finally conclude our paper in Section 6.

2. Preliminaries

In this paper, the following nonlinear bilevel programming is discussed:

$$\begin{aligned} &\min_{x \in X} \quad F(x, y), \\ &\text{s.t.} \quad G(x, y) \leq 0, \\ &\min_{y \geq 0} \quad cx + dy, \\ &\text{s.t.} \quad Ax + By \leq b, \end{aligned} \tag{2.1}$$

where $A \in R^{q \times n}$, $B \in R^{q \times m}$, $c^T \in R^n$, $d^T \in R^m$, and X is a box set.

Now we introduce some related definitions [1].

(1) Constraint region: $S = \{(x, y) \mid G(x, y) \leq 0, Ax + By \leq b, x \in X, y \geq 0\}$.

(2) For x fixed, the feasible region of lower level problem: $S(x) = \{y \mid Ax + By \leq b, y \geq 0\}$.

(3) Projection of S onto the upper level decision space: $S(X) = \{x \in R^n \mid \exists y, (x, y) \in S\}$.

(4) The lower level rational reaction set for each $x \in S(X)$: $M(x) = \{y \in R^m \mid y \in \operatorname{argmin}\{cx + dv, v \in S(x)\}\}$.

(5) Inducible region: $IR = \{(x, y) \in S \mid y \in M(x)\}$.

Hence, problem (2.1) can also be written as

$$\min\{F(x, y) \mid (x, y) \in IR\}. \tag{2.2}$$

We always assume that S is nonempty and compact to ensure that problem (2.1) can be well posed.

Note that for x fixed in the lower level problem, the term c_2x is constant, it can be removed when the lower level problem is solved. As a result, the lower level problem can be replaced by

$$\begin{aligned} &\min_{y \geq 0} \quad dy, \\ &\text{s.t.} \quad Ax + By \leq b. \end{aligned} \tag{2.3}$$

Equation (2.3) is a linear programming parameterized by x, then its dual problem can be written as

$$\begin{aligned} &\max_{u \geq 0} \quad (Ax - b)^T u, \\ &\text{s.t.} \quad -B^T u \leq d^T. \end{aligned} \tag{2.4}$$

According to the duality principle, For each fixed x, if there exists (y, u) satisfying

$$\begin{aligned} &dy - (Ax - b)^T u = 0, \\ &-B^T u \leq d, \\ &Ax + By \leq b, \\ &u, y \geq 0. \end{aligned} \tag{2.5}$$

Then, y is an optimal solution to (2.3). It follows that (2.1) can be converted to the following single-level problem:

$$\begin{aligned} &\min_{x \in X} \quad F(x, y), \\ &\text{s.t.} \quad G(x, y) \leq 0, \\ &\qquad dy - (Ax - b)^T u = 0, \\ &\qquad -B^T u \leq d^T, \\ &\qquad Ax + By \leq b, \\ &\qquad u, y \geq 0. \end{aligned} \tag{2.6}$$

Obviously, (2.6) is a nonlinear programming problem.

It is computationally expensive for one to directly solve the problem (2.6). In fact, u is taken from the dual problem (2.4), for each x, it should be a feasible solution of (2.4). Hence, if u can be obtained in advance, then (2.6) can be simplified.

The algorithmic profile can be described as follows: We begin with (2.4), and encode each potential basis as an individual. For each individual, we solve the constraints $-B_2^T u + \overline{u} = d_2^T$ to obtain a basic feasible solution u. Also, the constraint $-B^T u \leq d$ can be removed since the inequality always holds. That is to say, if $u = u_0$ is obtained, (2.6) can be written as

$$\begin{aligned} \min_{x \in X} \quad & F(x, y), \\ \text{s.t.} \quad & G(x, y) \leq 0, \\ & dy - (Ax - b)^T u_0 = 0, \\ & Ax + By \leq b, \\ & y \geq 0, \end{aligned} \tag{2.7}$$

which is a simplified version of (2.6).

For each u_0 obtained from (2.4), we solve the nonlinear programming (2.7) and if any, obtain an optimal solution. It indicates if one can search all bases of problem (2.4), then he can obtain the optimal solutions of BLPP (2.1) by taking the minimum one among all objective values of (2.7). In spite of the fact that when the dual variable vector u is obtained, y can also be got according to the duality theorems, we do not intend to adopt the procedure to obtain y. The main reason is that when the follower has more than one optimal solution, the procedure cannot guarantee that the obtained y is the best one to F.

3. Solution Method

In spite of the fact that (2.6) has been simplified as (2.7), it is still hard to solve when the upper level functions are nonconvex and even nondifferentiable. In this section, we first present an EA for the case in which the upper level problem can be solved by deterministic methods, such as convex programs. In the next section, the method will be extended to more general cases.

3.1. Chromosome Encoding

The feasible bases of (2.4) are encoded as chromosomes. First, (2.4) is standardized as

$$\begin{aligned} \max_{u \geq 0} \quad & (Ax - b)^T u, \\ \text{s.t.} \quad & -B^T u + \overline{u} = d^T. \end{aligned} \tag{3.1}$$

Let $W = (-B^T, I)$ which is an $m \times (m + q)$ matrix, and m columns in W, $(i_1, \ldots, i_m)$, are chosen at random. If the matrix $\overline{B}$ consisting of these columns is a feasible basis, then it is referred to as an individual (chromosome) and denoted by $(i_1, \ldots, i_m)$.

3.2. Initial Population

The initial population consists of N individuals associated with the feasible bases of (3.1). In order to obtain the N individuals, we first take an x, then solve problem (3.1) using the simplex method. Since different feasible bases can be found at each iteration of vertices, these bases can be put into the initial population. If the number of these feasible bases is less than N, we take a substitutive x and repeat the procedure of solving (3.1) until N individuals are found.

3.3. Fitness Evaluation

For each individual $l = (i_1, \ldots, i_m)$, the fitness $R(l)$ of l can be obtained by the following steps. First, the basic components of u are taken as $\overline{B}^{-1} d_2^T$, whereas other components are 0. Then, for the known u, we solve problem (2.7). If there exists an optimal solution, then the objective value is taken as the fitness $R(l)$ of l; otherwise, $R(l) = M$, where M is a positive number large enough.

3.4. Crossover and Mutation Operators

Let $l = (i_1, \ldots, i_m)$ and $l' = (i'_1, \ldots, i'_m)$ be selected parents for crossover. We give the crossover offspring of l and l' as follows. Let $L_1 = l' \setminus l$ and $|L_1| = s$ standing for the size of L_1. If $s > 0$, then s_1 ($s_1 \leq s$) elements are chosen at random in L_1. These selected elements are taken as entering variables for the present basis associated with l and the leaving variables in l can be determined by using the minimum ratio principle. As a result, an offspring l_c is generated. If $s = 0$, let $l' = (m + q + 1) - l'$ and then execute the procedure mentioned above. Let $L_2 = l \setminus l'$, then the other offspring l'_c can be obtained.

Let $l = (i_1, \ldots, i_m)$ be a selected parent for mutation. We give the mutation offspring as follows. First, one element $\bar{l}_i$ is randomly chosen from the set $\bar{l} = \{1, 2, \ldots, m + p\} - l$, then $\bar{l}_i$ is taken as an entering variable, and the leaving variable from l can be determined according to the minimum ratio principle. The new basis obtained by the pivot operation is referred as the offspring of mutation.

3.5. An Evolutionary Algorithm Based on Duality Principle (EADP)

In this subsection, we present an evolutionary algorithm based on the encoding scheme, fitness function, and evolutionary operators described above. In each generation, an archive $\hat{S}$ of s_0 (larger than the size of population) individuals is maintained to avoid evaluating repeatedly the fitness of the same individuals generated in genetic operations.

Step 1 (initialization). Generate N initial points l^i, $i = 1, 2, \ldots, N$ by taking different x. All of the points l^i form an initial population pop(0) with population size N. Let $v = 0$.

Step 2 (fitness). Evaluate the fitness value $R(l)$ of each point in pop(v). Set $\widehat{S} = [\text{pop}(v), R(l)]$.

Step 3 (crossover). Let the crossover probability be p_c. For crossover parents l, $l' \in \text{pop}(v)$, we first take a $r \in [0, 1]$ at random, if $r \leq p_c$, then execute the crossover on l, l' to get its offspring l_o and l'_o. Let $O1$ stand for the set of all these offspring.

Step 4 (mutation). Let the mutation probability be p_m. For each $\bar{l} \in \text{pop}(v)$, we take a $r \in [0,1]$ at random, if $r \leq p_m$, then execute the mutation on $\bar{l}$ to get its offspring l_o. Let $O2$ stand for the set of all these offspring.

Step 5 (selection). Let $O = O1 \cup O2$. For any point in O, if the individual belongs to $\widehat{S}$, then the fitness value is there, otherwise, one has to evaluate the fitness value of the point. Select the best N_1 points from the set $\text{pop}(v) \cup O$ and randomly select $N - N_1$ points from the remaining points of the set. These selected points form the next population $\text{pop}(v+1)$.

Step 6 (updating). Merge all points in O into $\widehat{S}$. If the number of elements in $\widehat{S}$ is larger than s_0, then randomly delete some point such that S has only s_0 individuals.

Step 7. If the termination condition is met, then stop; otherwise, let $v = v + 1$, go to Step 3.

EADP differs from other existing algorithms in three ways. First, since EADP only searches the feasible bases of the lower level dual problem instead of taking all feasible points into account, it makes the searching space smaller. In addition, there is a local search procedure in EADP. Sine for u fixed, to solve (2.7) is a local search process for (x, y), which can improve the performance of the proposed algorithm. Finally, it is necessary for the most of the existing algorithmic procedures to assume that the lower level solution is unique, since the assumption can make the problem become easier. In the proposed EADP, y is not directly calculated and chosen for any fixed x, as a result, the uniqueness restriction can be removed.

4. A Revised Version for Nonconvex Cases

The solutions of a BLPP must be feasible, that is, for any x fixed, y must be optimal to the follower problem. The proposed algorithm can guarantee all points (x, y) are in IR if (2.7) can be solved well. In fact, if (2.7) is a convex programming, there are dozens of algorithms for this kind of problems. However, when the problem is nonconvex, it is very hard for us to obtain a globally optimal solution since the most of existing classical algorithms cannot guarantee to obtain globally optimal solutions. In order to overcome the shortcomings, we use a coevolutionary algorithm to search globally optimal solutions. Set $U = \{u_1, u_2, \ldots, u_k\}$, and for each $u_i \in U$, (x_i, y_i) stands for the solution obtained by solving (2.7), which may be locally optimal. We generate an initial population $p(0)$ as follows: first, (x_i, y_i), $i = 1, 2, \ldots, k$, are put into $p(0)$, and then for each (x_i, y_i), randomly generate $r - 1$ points according to Gaussian distribution $N((x_i, y_i), \sigma^2)$, and these points are also put into the set. As a result, the total of kr points are included in $p(0)$. The fitness is given with a multicriteria type; that is, there are k evaluation functions: $f_i = F(x, y) + M \max\{0, G(x, y), Ax + By - b, | dy - (Ax - b)^T u_i |\}$, $i = 1, \ldots, k$. To evaluate individual is to compute f_i, $i = 1, \ldots, k$, at each point. Next, we present a co-evolutionary algorithmic profile (Co-EA) as follows.

(S1) Generate an initial population $p(0)$. Let $g = 0$.

(S2) Evaluate all points in $p(g)$.

(S3) Execute the arithmetical crossover and Gaussian mutation to generate offspring set O.

(S4) Evaluate all offspring in O, and according to each f_i, $i = 1, \ldots, k$, select the best r points in $p(g) \bigcup O$, total kr points are taken as the next generation of population $p(g+1)$.

(S5) If the termination criterion is satisfied, then output the best individual associated with each f_i, otherwise, $g = g + 1$, turn to (S3).

The purpose of Co-EA is to obtain globally optimal solutions to (2.7). Next, we present the revised version of EADP by imbedding Co-EA.

Step 1 (initialization). Generate N initial points l^i, $i = 1, 2, \ldots, N$ by taking different x. All of the points l^i form the initial population pop(0) with population size N. Let $v = 0$, $\widehat{S} = \phi$.

Step 2 (fitness). Evaluate the fitness value $R(l)$ of each point in pop(v). Let $\widehat{S} = [\text{pop}(v), R]$.

Step 3 (crossover). Let the crossover probability be p_c. For crossover parents $l, l' \in \text{pop}(v)$, we first take a $r \in [0, 1]$ at random, if $r \leq p_c$, then execute the crossover on l, l' to get its offspring l_o, l'_o. Let $O1$ stand for the set of all these offspring.

Step 4 (mutation). Let the mutation probability be p_m. For each $\bar{l} \in \text{pop}(v)$, we first take a $r \in [0, 1]$ at random, if $r \leq p_m$, then execute the mutation on $\bar{l}$ to get its offspring l_o. Let $O2$ stand for the set of all these offspring.

Step 5 (offspring evaluating). Let $O = O1 \cup O2$. For any point in O, if the individual is in S, the fitness value is there, otherwise, one has to evaluate the fitness value of the point.

Step 6 (selection). Select the best N_1 points from the set pop(v) $\cup$ O and randomly select $N - N_1$ points from the remaining points of the set. These selected points form the next population pop($v + 1$).

Step 7 (coevolution). If $v > 0$ and v mod $t = 0$, then select the best one and other $k - 1$ individuals from pop($v + 1$), execute Co-EA, and update the fitness values of these individuals.

Step 8 (updating). If element number in $\widehat{S}$ is less than s_0, then some points in O, especially, modified points by Co-EA, are put into $\widehat{S}$ until there are s_0 elements.

Step 9. If the termination condition is met, then stop; otherwise, let $v = v + 1$, go to Step 3.

5. Simulation Results

In this section, we first select 7 test problems from the literature as follows, which are denoted by Example 1–Example 7 (Group I). These smaller-scale problems, as examples, are frequently solved to illustrate the performance of the algorithms in the literature.

Example 1 (see [8]). Consider

$$\begin{aligned}
&\min_{x\geq 0} \quad -8x_1 - 4x_2 + 4y_1 - 40y_2 - 4y_3,\\
&\min_{y\geq 0} \quad x_1 + 2x_2 + y_1 + y_2 + 2y_3,\\
&\text{s.t.} \quad -y_1 + y_2 + y_3 \leq 1,\ 2x_1 - y_1 + 2y_2 - 0.5y_3 \leq 1,\\
&\qquad 2x_2 + 2y_1 - y_2 - 0.5y_3 \leq 1.
\end{aligned} \tag{5.1}$$

Example 2 (see [11]). Consider

$$\begin{aligned}
&\min_{x\geq 0} \quad -2x_1 + 4x_2 + 3y,\\
&\qquad\quad x_1 - x_2 \leq -1,\\
&\min_{y\geq 0} \quad -y,\\
&\text{s.t.} \quad x_1 + x_2 + y \leq 4,\ 2x_1 + 2x_2 + y \leq 6.
\end{aligned} \tag{5.2}$$

Example 3 (see [21]). Consider

$$\begin{aligned}
&\min_{x\geq 0} \quad -4x - y_1 - y_2,\\
&\min_{y\geq 0} \quad -x - 3y_1,\\
&\text{s.t.} \quad x + y_1 + y_2 \leq \frac{25}{9},\ x + y_1 \leq 2,\ y_1 + y_2 \leq \frac{8}{9}.
\end{aligned} \tag{5.3}$$

Example 4 (see [7]). Consider

$$\begin{aligned}
&\min_{x\geq 0} \quad 2x - 11y,\\
&\min_{y\geq 0} \quad x + 3y,\\
&\text{s.t.} \quad x - 2y \leq 4,\ 2x - y \leq 24,\ 3x + 4y \leq 96,\\
&\qquad x + 7y \leq 126,\ -4x + 5y \leq 65,\ -x - 4y \leq -8.
\end{aligned} \tag{5.4}$$

Example 5 (see [8]). Consider

$$\begin{aligned}
&\max_{1\geq x\geq 0} \quad 100x + 1000y,\\
&\max_{y} \quad y_1 + y_2,\\
&\text{s.t.} \quad x + y_1 - y_2 \leq 1,\ y_1 + y_2 \leq 1.
\end{aligned} \tag{5.5}$$

Example 6 (see [1]). Consider

$$\begin{aligned}
&\min_{50\geq x\geq 0} \quad (y_1 - x_1 + 20)^2 + (y_2 - x_2 + 20)^2,\\
&\min_{20\geq y\geq -10} \quad 2x_1 + 2x_2 - 3y_1 - 3y_2 - 60,\\
&\text{s.t.} \quad x_1 + x_2 + y_1 - 2y_2 \leq 40,\ 2y_1 - x_1 + 10 \leq 0,\ 2y_2 - x_2 + 10 \leq 0.
\end{aligned} \tag{5.6}$$

Example 7 (see [12]). Consider

$$\begin{aligned}
&\min_{x\geq 0} \quad (1 + x_1 - x_2 + 2y_2)(8 - x_1 - 2y_1 + y_2 + 5y_3),\\
&\min_{y\geq 0} \quad 2y_1 - y_2 + y_3,\\
&\text{s.t.} \quad -y_1 + y_2 + y_3 \leq 1,\ 2x_1 - y_1 + 2y_2 - 0.5y_3 \leq 1,\ 2x_2 + 2y_1 - y_2 - 0.5y_3 \leq 1.
\end{aligned} \tag{5.7}$$

In order to illustrate the performance of EADP, we construct 3 larger-scale linear BLPPs (Group II, Example 8–Example 10) with the following type:

$$\begin{aligned}
&\min_{x\geq 0} \quad c_1 x + d_1 y,\\
&\min_{y\geq 0} \quad c_2 x + d_2 y,\\
&\text{s.t.} \quad A_2 x + B_2 y \leq b_2.
\end{aligned} \tag{5.8}$$

All coefficients are taken as follows. The coefficients of the upper level objective are randomly generated from the uniform distribution on $(-10, 10)$, whereas those of the lower level objective are randomly chosen in $(0, 10)$. A_2 and B_2 are generated from a uniform distribution on $(-50, 50)$, and the right-hand side of each constraint is the sum of absolute values of the left-hand side coefficients. In order to assure the constraint region of the generated BLPP is bounded, we select a constraint randomly, and the left-hand side coefficients are taken as the absolute values of the corresponding coefficients. The scales of the constructed problems are given in Table 1.

The proposed algorithm is used to solve the problems in Group I and Group II. The values of the parameters in EADP are given in Table 2, where Max-gen stands for the maximum number of generations that the algorithm runs. Notice that in the proposed algorithmic approach, the total of feasible points in search space is not greater than C_{m+q}^{q}. Hence, the value can be taken as a reference point when N, s_0 and Max-gen need to be determined. For a specific problem, if one wants to determine Max-gen in advance, then he can tentatively take an experiment value of this parameter on other problems with same lower level scale. For parameter M, one can first take a positive integer by considering the maximum value of F on S and then examine the constraint violations of obtained points. If the constraint violations can not be permitted, then the integer is multiplied by 10. This process is not stopped until the constraint violations are permitted.

For Example 1–Example 10, the algorithm stops when the maximum number of generations is achieved. In the executed algorithm, (2.7) is solved by different methods.

Table 1: The scales of the constructed problems.

Factors	Example 8	Example 9	Example 10
n	100	100	100
m	60	80	100
q	40	60	80

Table 2: The values of the parameters taken in EADP.

Parameters	Group I	Group II
N	5	50
p_c	0.8	0.8
p_m	0.1	0.1
N_1	2	10
M	10000	—
s_0	20	200
σ	1	1
Max-gen	20	5000

The simplex method is adopted in Example 1–Example 5 and Group II, the active-set method is applied in Example 6 in which (2.7) is a convex quadratic programming. For Example 7, the revised EADP is executed since the objective is nonconvex. k is taken as 5, $r = 4$, $t = 5$, and the maximum generation number of Co-EA is 100.

For Group I, we execute EADP in 20 independent runs on each problem on a PC (Intel Pentium IV-2.66 GHz) and record the following data.

(1) Best solution (x^*, y^*).

(2) Upper level objective value $F(x^*, y^*)$ at the best solution.

(3) Upper level objective value $F(\overline{x}, \overline{y})$ at the worst solution $(\overline{x}, \overline{y})$.

(4) Mean value (F_{mean}), median (F_{median}), and standard deviation (STD) of the upper level objective values.

(5) Mean CPU time in 20 runs.

All results for Group I are presented in Table 3. Table 3 provides the comparison of the results found by EADP in 20 runs and by the compared algorithms for these examples, and the best solutions found by EADP in 20 runs are also presented in Table 3, where Ref. stands for the related algorithms in references in Table 3.

It can be seen from Table 3 that for Example 3 and Example 7, all results found by EADP in 20 runs are better than the results given by the compared algorithms in the references, which indicates the algorithms in the related references can not find out the globally optimal solutions of these two problems. Especially, for Example 7, the compared algorithm found a local solution $(0, 0.75, 0, 0.5, 0)$ with the objective value of 10.625. For other problems, the best results found by EADP are as good as those by the compared algorithms. In all 20 runs, EADP found the best results of all problems, and the standard deviations are 0, which means that EADP is stable.

Table 3: Comparison of the results found by EADP and other algorithms in reference.

Prob.	EADP							Ref.-
	$F(x^*, y^*)$	F_{mean}	F_{median}	$F(\overline{x}, \overline{y})$	STD	CPU(s)	(x^*, y^*)	$F(x^*, y^*)$
Example 1	−29.2	−29.2	−29.2	−29.2	0	0.6	(0, 0.9, 0, 0.6, 0.4)	−29.2
Example 2	6	6	6	6	0	0.5	(1, 2, 0)	6
Example 3	−8.7778	−8.7778	−8.7778	−8.7778	0	0.6	(2, 0, 0.7778)	−8.2230
Example 4	−85.0909	−85.0909	−85.0909	−85.0909	0	0.5	(17.4545, 10.9091)	−85.0909
Example 5	1000	1000	1000	1000	0	0.4	(0, 1, 0)	1000
Example 6	0	0	0	0	0	0.4	(25, 30, 5, 10)	0
Example 7	7.5	7.5	7.5	7.5	0	6.2	(0.5, 0.5, 0, 0, 0)	10.625

Table 4: The computational results given by EADP on Group II.

Examples	CPU	TB	STD	F^*
Example 8	$6.7e + 002$	9	9.7	$-4.6663e + 003$
Example 9	$1.4e + 003$	9	7.5	$-3.8536e + 003$
Example 10	$2.7e + 003$	7	$1.8e + 001$	$-4.5548e + 003$

From the CPU time, one can see that the CPU time that EADP needs is short for obtaining these results, it means that the algorithm is efficient.

For Group II, we execute EADP in 10 independent runs on each problem. Since there is no computational result given in the literature, we give a criterion for measuring the simulation results obtained by EADP. For the best one in 10 independent runs, we select the best individual and randomly generate other $N - 1$ individuals to form a population. The algorithm begins with the population and is executed successively 5000 generations again. The obtained solutions are taken as the "best" results.

All obtained results in Group II are presented in Table 4. Where CPU stands for the mean CPU time in 10 runs, TB represents the times that the best result appears in 10 runs; STD denotes the standard deviation of the objectives in 10 runs, and F^* is the best objective value. From Group I to Group II, one can see that the increments of CPU are larger, which implies that the computational complex of BLPP will sharply increase as the dimension become larger. Despite the fact that STD is larger due to larger objective values, TB shows that EADP is reliable in solving larger-scale BLPPs since the minimum rate of success is 70%.

In order to present the convergence of EADP on the larger-scale problems, we select one in 10 runs for each problem and plot the curves of objective values, and the generations, refer to Figure 1(a) for Example 8, (b) for Example 9, and (c) for Example 10. From Figure 1, we can see the proposed algorithm converges in 3000 generations.

6. Conclusion

For a class of nonlinear bilevel programming problems, the paper develops an evolutionary algorithm based on the duality principle. In the proposed algorithm, the bases of the dual problem of the lower level are used to encode individuals, which leads to that the searching space becomes finite. For nonconvex cases, we design a Co-EA technique to obtain globally

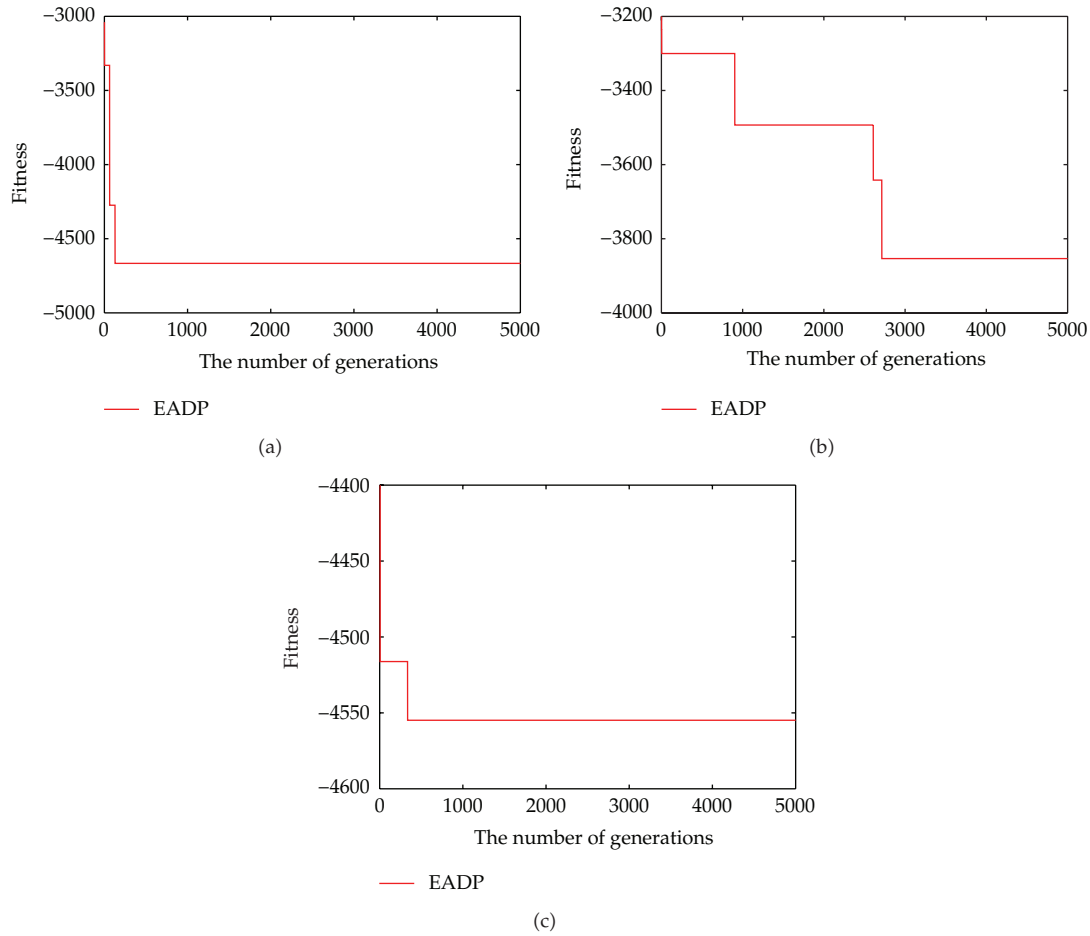

Figure 1: The convergency of EADP on the larger-scale linear BLPPs.

optimal solutions. The experiment results show the proposed algorithm is efficient and effective. In the future work, based on optimality results, we intend to research nonlinear BLPPs with more general lower level problems.

Acknowledgments

This research work was supported in part by the National Natural Science Foundations of China (no. 61065009) and the Natural Science Foundation of Qinghai Province (Innovation Research Foundation of Qinghai Normal University) (no. 2011-z-756).

References

[1] J. F. Bard, *Practical Bilevel Optimization*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[2] B. Colson, P. Marcotte, and G. Savard, "Bilevel programming: a survey," *Journal of Operations Research*, vol. 3, no. 2, pp. 87–107, 2005.

[3] M. Zhao and Z. Gao, "A globally convergent algorithm for solving the bilevel linear programming problem," *OR Transacton*, vol. 9, no. 2, pp. 57–62, 2005.
[4] L. D. Muu and N. V. Quy, "A global optimization method for solving convex quadratic bilevel programming problems," *Journal of Global Optimization*, vol. 26, no. 2, pp. 199–219, 2003.
[5] J. B. E. Etoa, "Solving convex quadratic bilevel programming problems using an enumeration sequential quadratic programming algorithm," *Journal of Global Optimization*, vol. 47, no. 4, pp. 615–637, 2010.
[6] B. Colson, P. Marcotte, and G. Savard, "A trust-region method for nonlinear bilevel programming: algorithm and computational experience," *Computational Optimization and Applications*, vol. 30, no. 3, pp. 211–227, 2005.
[7] K.-M. Lan, U.-P. Wen, H.-S. Shih, and E. S. Lee, "A hybrid neural network approach to bilevel programming problems," *Applied Mathematics Letters*, vol. 20, no. 8, pp. 880–884, 2007.
[8] Y. Wang, Y. C. Jiao, and H. Li, "An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme," *IEEE Transactions on Systems, Man and Cybernetics Part C*, vol. 35, no. 2, pp. 221–232, 2005.
[9] B. Liu, "Stackelberg-Nash equilibrium for multilevel programming with multiple followers using genetic algorithms," *Computers & Mathematics with Applications*, vol. 36, no. 7, pp. 79–89, 1998.
[10] S. K. Suneja and B. Kohli, "Optimality and duality results for bilevel programming problem using convexifactors," *Journal of Optimization Theory and Applications*, vol. 150, no. 1, pp. 1–19, 2011.
[11] J. Glackin, J. G. Ecker, and M. Kupferschmid, "Solving bilevel linear programs using multiple objective linear programming," *Journal of Optimization Theory and Applications*, vol. 140, no. 2, pp. 197–212, 2009.
[12] Z. Wan, G. Wang, and Y. Lv, "A dual-relax penalty function approach for solving nonlinear bilevel programming with linear lower level problem," *Acta Mathematica Scientia. Series B*, vol. 31, no. 2, pp. 652–660, 2011.
[13] H. I. Calvete, C. Galé, S. Dempe, and S. Lohse, "Bilevel problems over polyhedra with extreme point optimal solutions," *Journal of Global Optimization*, vol. 53, no. 3, pp. 573–586, 2012.
[14] U. P. Wen and S. T. Hsu, "Linear bi-level programming problems. A review," *Journal of the Operational Research Society*, vol. 42, no. 2, pp. 125–133, 1991.
[15] A. G. Mersha and S. Dempe, "Linear bilevel programming with upper level constraints depending on the lower level solution," *Applied Mathematics and Computation*, vol. 180, no. 1, pp. 247–254, 2006.
[16] C. Shi, J. Lu, G. Zhang, and H. Zhou, "An extended branch and bound algorithm for linear bilevel programming," *Applied Mathematics and Computation*, vol. 180, no. 2, pp. 529–537, 2006.
[17] R. J. Kuo and C. C. Huang, "Application of particle swarm optimization algorithm for solving bi-level linear programming problem," *Computers & Mathematics with Applications*, vol. 58, no. 4, pp. 678–685, 2009.
[18] Y. Wang, H. Li, and C. Dang, "A new evolutionary algorithm for a class of nonlinear bilevel programming problems and its global convergence," *INFORMS Journal on Computing*, vol. 23, no. 4, pp. 618–629, 2011.
[19] K. Deb and S. Sinha, "An efficient and accurate solution methodology for bilevel multi-objective programming problems using a hybrid evolutionary-local-search algorithm," *Journal Evolutionary Computation*, vol. 18, no. 3, pp. 403–449, 2010.
[20] H. I. Calvete, C. Galé, and P. M. Mateo, "A new approach for solving linear bilevel problems using genetic algorithms," *European Journal of Operational Research*, vol. 188, no. 1, pp. 14–28, 2008.
[21] T. Hu, B. Huang, and X. Zhang, "A neural network approach for solving linear bilevel programming problem," in *Proceedings of the 6th ISNN Conference*, vol. 56, pp. 649–658, AISC, 2009.