2011 Seventh International Conference on Computational Intelligence and Security

A Genetic Algorithm Based on Duality for Linear Bilevel Programming Problems

Hecheng Li
Department of Mathematics,
Key Laboratory of Tibetan Information Processing of Ministry of Education,
Qinghai Normal University, Xining 810008, China
Email: lihecheng@qhnu.edu.cn

***Abstract*—Linear bilevel programming problem, as a NP-hard problem, is the linear version of bilevel programming, in this paper we design an efficient algorithm for solving this kind of problems by combining genetic algorithm with enumeration procedure of extreme points. First, based on the duality principle, the follower problem is replaced by the prime-dual conditions, and the original problem is transformed into an equivalent single-level programming in which all functions are linear except for one constraint. Then, the bases of the duality problem are considered as individuals. For each selected individual (base), the nonlinear constraint can be simplified to linear one, and some constraints can also be removed from the transformed single-level problem. Hence, the single-level problem is converted into a linear programming. At last, the linear programming is solved and the objective value is taken as the fitness of the individual. The distinguished feature of the algorithm is that the bases of follower's duality problem are searched instead of taking all feasible points into account, which makes the search space smaller. In order to illustrate the efficiency of the algorithm, 4 problems selected from literature are solved, and the results show that the proposed algorithm is efficient and robust.**

***Keywords*-linear bilevel programming; genetic algorithm; optimal solutions; duality;**

I. INTRODUCTION

Bilevel programming problem (BLPP) involves two optimization problems, the leader problem and the follower problem. The constraint region of the leader problem is implicitly determined by the follower problem. The leader first select a strategy x to optimize his/her objective, then the follower observes the leader's selection and find a strategy y to optimize his/her own objective. When such pair (x, y) satisfies the leader's constraints, it is called feasible in a bilevel decision-making procedure. The general bilevel programming problem can be formulated as follows

$$\begin{cases} \min\limits_{x\in X} F(x,y) \\ s.t.\ G(x,y) \le 0 \\ \min\limits_{y\in Y} f(x,y) \\ s.t.\ g(x,y) \le 0 \end{cases} \tag{1}$$

where $x \in R^n, y \in R^m$. In this problem,

$$\begin{cases} \min\limits_{x\in X} F(x,y) \\ s.t.\ G(x,y) \le 0. \end{cases} \tag{2}$$

and

$$\begin{cases} \min\limits_{y\in Y} f(x,y) \\ s.t.\ g(x,y) \le 0 \end{cases} \tag{3}$$

are called the leader and the follower problem, respectively. The variables of problem (1) are divided into the leader's variables $x = (x_1, \cdots, x_n)^T$ and the follower's variables $y = (y_1, \cdots, y_m)^T$. Similarly, $F(f) : R^n \times R^m \to R$ is called the leader's (follower's) 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 called the leader's and the follower's constraints, respectively. The sets X and Y place additional constraints on the variables, such as upper and lower bounds or integrality requirements etc. When the functions in (1) are linear, the problem is known as linear BLPP.

BLPP is used extensively to a variety of areas such as economy, engineering and management[1, 2], and some efficient algorithms are proposed[3–8]. The linear bilevel programming problem is a special BLPP, in which an optimality result is reported that 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 *k*-th best method and branch-and-bound algorithm[1, 9–11]. In recent years, genetic algorithm based on the bases of all linear constraints[12], multiobjective method[13] and neural network approach[14] are developed. The most of these methods, in fact, are based on two starting points, K-K-T conditions and bases of all constraints.

In this paper we begin with the bases of the follower's dual problem, and design an efficient genetic algorithm. First, the follower problem is replaced by the prime-dual conditions, and in this process one nonlinear term arises. After doing so, the original problem is transformed into an equivalent single-level programming in which all functions are linear except for one constraint. Then, in order to remove the nonlinear term, we encode individuals by taking the bases of the duality problem. For each individual, the

978-0-7695-4584-4/11 $26.00 © 2011 IEEE
DOI 10.1109/CIS.2011.11

nonlinear constraint can be simplified to linear one, and some constraints can also be removed from the original single-level problem. Hence, the single-level problem is converted into a simplified linear programming(LP). At last, the linear programming is solved and the objective value is taken as the fitness of the individual. The distinguished feature of the algorithm is that the bases of follower's dual problem are searched instead of taking all feasible points into account, which makes the search space smaller.

This paper is organized as follows. Some notations and transformation are presented in Section II, and a genetic algorithm is given based on the duality principle in Section III. Experimental results and comparison are presented in Section IV. We finally conclude our paper in Section V.

II. Preliminaries

We consider the following linear bilevel programming:

$$\begin{cases} \min\limits_{x\geq 0} c_1x + d_1y, \\ s.t.\ A_1x + B_1y \leq b_1, \\ \min\limits_{y\geq 0} c_2x + d_2y, \\ s.t.\ A_2x + B_2y \leq b_2. \end{cases} \tag{4}$$

where $x \in R^n$, $y \in R^m$, $A_1 \in R^{p\times n}$, $A_2 \in R^{q\times n}$, $B_1 \in R^{p\times m}$, and $B_2 \in R^{q\times m}$.

Now we introduce some related definitions [1].

1) Constraint region: $S = \{(x, y) \mid A_ix + B_iy \leq b_i, i = 1, 2; x, y \geq 0\}$.

2) For x fixed, the feasible region of follower's problem: $S(x) = \{y \mid A_2x + B_2y \leq b_2, y \geq 0\}$.

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

4) The follower's rational reaction set for each $x \in S(X)$: $M(x) = \{y \in R^m \mid y \in \operatorname{argmin}\{c_2x + d_2v, v \in S(x)\}\}$.

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

In terms of aforementioned definitions, problem (4) also can be written as:

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

In order to ensure that problem (4) is well posed and solved, in the remainder, we always assume that S is nonempty. Since for x fixed, the term c_2x is constant in the follower problem, hence, it can be removed when the follower problem is solved. As a result, the follower programming can be replaced by

$$\begin{cases} \min\limits_{y\geq 0} d_2y, \\ s.t.\ A_2x + B_2y \leq b_2. \end{cases} \tag{5}$$

Obviously, problem (5) has the same optimal solution as the follower problem of (4) for each fixed x, it follows that the term c_2x can be ignored.

(5) is a linear programming for each fixed x, then its dual problem can be written as

$$\begin{cases} \max\limits_{u\geq 0} (A_2x - b_2)^T u, \\ s.t.\ -B_2^T u \leq d_2^T. \end{cases} \tag{6}$$

According to the duality principle, For each fixed x, if (5) has an optimal solution y, then there exists an u such that (y, u) solves

$$\begin{cases} d_2y - (A_2x - b_2)^T u = 0, \\ -B_2^T u \leq d_2, \\ A_2x + B_2y \leq b_2, \\ u, y \geq 0. \end{cases} \tag{7}$$

It follows that (4) can be converted to the following single-level programming

$$\begin{cases} \min\limits_{x\geq 0} c_1x + d_1y, \\ s.t.\ A_1x + B_1y \leq b_1, \\ d_2y - (A_2x - b_2)^T u = 0, \\ -B_2^T u \leq d_2^T, \\ A_2x + B_2y \leq b_2, \\ u, y \geq 0. \end{cases} \tag{8}$$

All functions in (8) are linear except for the second constraint which is an equality.

For problem (8), it is difficult for us to solve directly the problem with the nonlinear term. In fact, u is taken from the dual problem (6), and for each x it should be a feasible solution of (6). When the optimality is considered, u is also a basic feasible solution.

In proposed genetic algorithm, we begin with (6), and encode each potential base as an individual. For each individual, we solve the equations $-B_2^T u + \bar{u} = d_2^T$ to obtain u, where $u, \bar{u} \geq 0$ should be satisfied. After doing so, there is no nonlinear term in (8) and the constraint $-B_2^T u \leq d_2$ can also be removed. That is to say, if $u = u_0$ is obtained, (8) can be written as

$$\begin{cases} \min\limits_{x\geq 0} c_1x + d_1y, \\ s.t.\ A_1x + B_1y \leq b_1, \\ d_2y - (A_2x - b_2)^T u_0 = 0, \\ A_2x + B_2y \leq b_2, \\ y \geq 0, \end{cases} \tag{9}$$

which is a linear programming.

Thus, for any $u = u_0$, it is easy for us to solve the linear programming (9) and obtain, if any, an optimal solution. It indicates if one can search all bases of problem (6), then he can obtain the optimal solutions of BLPP(4) by taking the minimum one among all objective values of (9) at deferent values of u.

III. Solution Method

In this section, we present a GA for solving (4). At first, we encode each individual by using the potential bases of (6). Then a new fitness function is given which involves the problem (9). Next we describe each step of the algorithm in more detail.

A. Chromosome Encoding

We denote a chromosome by a potential base of (6). First, (6) is standardized as

$$\begin{cases} \max\limits_{u\geq 0}(A_2x-b_2)^Tu, \\ s.t.\ -B_2^Tu+\bar{u}=d_2^T. \end{cases} \tag{10}$$

Let $W=(-B_2^T,I)$ which is an $m\times(m+q)$ matrix. Then m columns in W are chosen at random, and with any loss of generality, denoted by $(i_1,\cdots,i_m)$, which stands for an individual. Further, we denote these columns by a matrix B. If B is a feasible base, then the individual $(i_1,\cdots,i_m)$ is called a feasible individual; otherwise it is an infeasible individual.

B. Fitness Evaluation

For each individual $l=(i_1,\cdots,i_m)$, if it is feasible, the fitness $R(l)$ of l can be obtained by the following steps. First, the basic components of u are taken as $B^{-1}d_2^T$, whereas other components are 0. Then, for the known u, we solve the LP problem (9). 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 are positive number and large enough. For the purpose of convenience, for any infeasible individual, we also take M as the fitness value.

C. 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 $L=l\cup l'$. We choose randomly m different integers from L, these integers are taken as an offspring l_c of crossover; the other offspring l'_c can be obtained by following the same procedure.

Let $l=(i_1,\ldots,i_m)$ be selected parent for mutation and $\vartheta=\{1,2,\cdots,m+q\}$. We give the mutation offspring as follows: Choose randomly a i_k in l, and replace it by i'_k selected from $\vartheta\setminus l$.

D. A Genetic Algorithm Based on Duality (GABD)

In this subsection, we present a GA based on the encoding scheme, fitness function and genetic operators described above.

Step 1 (*Initialization*) Randomly generate N initial points $l^i, i=1,2,\cdots,N$. All of the points l^i form the initial population $pop(0)$ with population size N. Let $k=0$.

Step 2 (*Fitness*) Evaluate the fitness value $R(l)$ of each point in $pop(k)$.

Step 3 (*Crossover*) Let the crossover probability be p_c. For crossover parents $l,l'\in pop(k)$, we first take a $r_1\in[0,1]$ at random, if $r_1\leq p_c$, 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.

Sept 4 (*Mutation*) Let the mutation probability be p_m. For each $\bar{l}\in pop(k)$, we first take a $r_2\in[0,1]$ at random, if $r_2\leq p_m$, then execute the mutation on $\bar{l}$ to get its offspring $\bar{l}_o$. Let $O2$ stand for the set of all these offspring.

Step 5 (*Selection*) Let $O=O1\cup O2$. Evaluate the fitness values of all points in O. Select the best N_1 points from the set $pop(k)\cup O$ and randomly select $N-N_1$ points from the remaining points of the set. These selected points form the next population $pop(k+1)$.

Step 6 If the termination condition is satisfied, then stop; otherwise, let $k=k+1$, go to *Step* 3.

It should be noted that GABD is different from other existing algorithms. First, since GABD searches only the bases of the follower's dual problem instead of taking all constraints into account, it makes the searching space smaller. In addition, there is a local search procedure in GABD. When u is fixed, (9) is solved, which is a local search process for x and can improve the efficiency of the proposed algorithm. At last, it is necessary for the most of the existing algorithmic procedures to assume that the follower's solution is unique, since the assumption can make the problem become easier. In the proposed GABD, y is not calculated directly and chosen for a fixed x, as a result, the uniqueness restriction of the follower's solutions can be removed.

Since the search space of GABD is finite, the populations $pop(t)$ are monotonic and any two individuals are accessible by the mutation operation, the proposed algorithm is convergent with probability 1[15]. We have

Theorem 1. *GABD converges to the optimal solutions with probability 1.*

IV. Simulation Results

We select 4 test problems from the literature, these problems, as examples, are frequently solved to illustrate the performance of the proposed algorithms.

The values of the parameters are given in Table 1.

For examples 1-4, the algorithm stops when the maximum generation of 20 is achieved. We execute GABD in 20

Table II
COMPARISON OF THE RESULTS FOUND BY GABD AND BY THE RELATED ALGORITHMS IN REFERENCES.

Problems	GABD							Ref.-
	$F(x^*, y^*)$	F_{mean}	F_{median}	$F(\bar{x}, \bar{y})$	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	0.7	$(1, 2, 0)$	6
Example 3	-8.7778	-8.7778	-8.7778	-8.7778	0	0.5	$(2, 0, 0.77778)$	-8.2230
Example 4	-85.0909	-85.0909	-85.0909	-85.0909	0	0.5	$(17.4545, 10.9091)$	-85.0909

Table I
THE VALUES OF THE PARAMETERS TAKEN IN GABD.

N	5
p_c	0.8
p_m	0.1
N_1	2
M	10000

independent runs on each problem on a PC(Intel Pentium IV-2.66GHz), and record the following data:
(1) Best solution (x^*, y^*).
(2) Leader's objective value $F(x^*, y^*)$ at the best solution.
(3) The leader's objective function value $F(\bar{x}, \bar{y})$ at the worst solution $(\bar{x}, \bar{y})$.
(4) Mean value (F_{mean}), median(F_{median}), and standard deviation(Std) of the leader's objective values.
(5) Mean CPU time in 20 runs.

Example 1[7]

$$\begin{cases} \min\limits_{x\geq 0} -8x_1 - 4x_2 + 4y_1 - 40y_2 - 4y_3 \\ \min\limits_{y\geq 0} x_1 + 2x_2 + y_1 + y_2 + 2y_3 \\ s.t.\ -y_1 + y_2 + y_3 \leq 1,\ 2x_1 - y_1 + 2y_2 - 0.5y_3 \leq 1, \\ 2x_2 + 2y_1 - y_2 - 0.5y_3 \leq 1. \end{cases}$$

Example 2[13]

$$\begin{cases} \min\limits_{x\geq 0} -2x_1 + 4x_2 + 3y \\ x_1 - x_2 \leq -1, \\ \min\limits_{y\geq 0} -y \\ s.t.\ x_1 + x_2 + y \leq 4,\ 2x_1 + 2x_2 + y \leq 6. \end{cases}$$

Example 3[14]

$$\begin{cases} \min\limits_{x\geq 0} -4x - y_1 - y_2 \\ \min\limits_{y\geq 0} -x - 3y_1 \\ s.t.\ x + y_1 + y_2 \leq \dfrac{25}{9},\ x + y_1 \leq 2, \\ y_1 + y_2 \leq \dfrac{8}{9}. \end{cases}$$

Example 4[6]

$$\begin{cases} \min\limits_{x\geq 0} 2x - 11y \\ \min\limits_{y\geq 0} x + 3y \\ s.t.\ x - 2y \leq 4,\ 2x - y \leq 24, 3x + 4y \leq 96, \\ x + 7y \leq 126, -4x + 5y \leq 65, -x - 4y \leq -8. \end{cases}$$

All results are presented in Table 2. Table 2 provides the comparison of the results found by GABD in 20 runs and by the compared algorithms for the four examples, and the best solutions found by GABD in 20 runs are also presented in the table. Where Ref. stands for the related algorithms in references in Table 2.

It can be seen from Table 2 that for example 3, the all results found by GABD in 20 runs are better than the result given by the compared algorithm in the reference, which indicates the algorithm in the related conference can't find out the globally optimal solutions of the problem. For other problems , the best results found by GABD are as good as those by the compared algorithms.

In all 20 runs, GABD found the best results of all problems, and the standard deviations are 0, which means that GABD is stable and robust.

From the CPU time, one can see that GABD needs less time for obtaining the best results, it means that the algorithm is efficient.

V. CONCLUSION

For linear bilevel programming problems, the paper develops a genetic algorithm based on the duality principle. In the proposed algorithm the bases of dual problem of the follower are used to encode individuals, which leads to that the searching space become finite. Also, it should be noted that the proposed algorithm can be extended to the nonlinear cases in which the leader problem can be solved by the existing approaches, such as convex programming.

ACKNOWLEDGMENT

This research work was supported by the National Natural Science Foundations of China (No.61065009, No. 60873099) and the Natural Science Foundation of Qinghai Province(No. 2011-z-756).

REFERENCES

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

[2] B. Colson, P. Marcotte, and G. Savard, "Bilevel programming: A survey," A Quarterly Journal of Operations Research (4OR), vol.3, pp. 87-107, 2005

[3] L. D. Muu and N. V. Quy, "A global optimization method for solving convex quadratic bilevel programming problems," Journal of Global Optimization, vol. 26, pp. 199-219, 2003

[4] J. B. E. Etoa, "Solving convex quadratic bilevel programming problems using an enumeration sequential quadratic programming," J. Glob. Optim. vol. 47, pp. 615-637, 2010

[5] 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, pp. 211-227, 2005

[6] Kuen-Ming Lan, Ue-Pyng Wen, and Hsu-Shih Shih, *et al*, "A hybrid neural network approach to bilevel programming problems," Applied Mathematics Letters,vol. 20, pp. 880–884, 2007

[7] Yuping Wang, Yong-Chang Jiao, and Hong Li, "An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint - handling scheme," IEEE Trans. on Systems, Man, and Cybernetics-Part C, vol. 35, no. 2, pp. 221-232, 2005

[8] B. D. Liu, "Stackelberg-Nash equilibrium for mutilevel programming with multiple followers using genetic algorithms," Comput. Math. Appl., vol. 36, pp.79-89, 1998

[9] Ue-Pyng Wen and Shuh-Tzy Hsu, "Linear bi-Level programming problems–A review," The Journal of the Operational Research Society, vol. 42, pp. 125–133, 1991

[10] 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, pp. 247–254, 2006

[11] Chenggen Shi, Jie Lu, Guangquan Zhang, and Hong Zhou, "An extended branch and bound algorithm for linear bilevel programming," Applied Mathematics and Computation, vol.180, pp.529–537, 2006

[12] H. I. Calvete, C. Gale, and P. M. Mateo, "A new approach for solving linear bilevel problems using genetic algorithms," European Journal of Operational Research, vol. 188, pp. 14–28, 2008

[13] J. Glackin, J.G. Ecker, and M. Kupferschmid, "Solving bilevel linear programs using multiple objective linear programming," J. Optim. Theory Appl., vol.140, pp. 197–212, 2009

[14] Tiesong Hu, Bing Huang, and Xiang Zhang, "A neural network approach for solving linear bilevel programming problem," Proceedings of the sixth ISNN 2009, AISC 56, pp. 649–658, 2009.

[15] Yuping Wang, *Theory and Methods of Evolutionary Computation*. Beijing: Science Publishers, 2011 (in Chinese)