2010 International Conference on Computational Intelligence and Security

A Hybrid Approach for Solving Nonlinear Bilevel Programming Problems Using Genetic Algorithms

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

Yuping Wang
School of Computer Science and Technology, Xidian University, Xi'an 710071, China

***Abstract*—The paper focuses on a special nonlinear bilevel programming problem(BLPP), and its characteristic is that the follower's programming is convex and quadratic, whereas there are no any additional requirements for the leader's functions. In order to solve the complex problem efficiently, it is first converted into an equivalent single-level programming by using Karush-Kuhn-Tucher (K-K-T) conditions, and then a hybrid genetic algorithm(HGA), combined with an enumeration technique of the bases, is proposed to solve the equivalent problem. At first, a mixed encoding scheme is given, involving the leader's variables and the bases of the follower's linear complementarity system; In addition, we present a fitness function which consists of the leader's objective and a penalty term, and by which the feasible and infeasible individuals can be identified. In order to illustrate the efficiency of HGA, 10 test problems selected from literature are solved, and the computational results show that the proposed algorithm is efficient and robust.**

***Keywords*-nonlinear bilevel programming; genetic algorithm; optimal solutions; linear complementarity system;**

I. INTRODUCTION

Bilevel programming problem (BLPP) involves two optimization problems where the constraint region of the first level problem(leader's problem) is implicitly determined by the second level problem(follower's problem), that is, BLPP has a nested structure. In the real world, this kind of problems are proposed often for dealing with hierarchical decision making processes. 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 the variables of problem (1) are divided into two classes, namely the leader's variables $x \in R^n$ and the follower's variables $y \in R^m$. 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 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. In this model, the process of optimization is made by a hierarchical order. At first, The leader chooses an $x \in X \subseteq R^n$ in an attempt to optimize his/her objective function $F(x, y)$; For the x chosen , the follower optimizes his/her objective with parameter x. In this way, the leader can obtain the minimum $F(x, y)$ by selecting some x.

BLPP arises widely in lots of fields such as economy and management[1], [2], and a variety of exact algorithmic approaches have been developed for solving BLPPs[2-8]. However, due to the nonconvexity and hierarchical structure of BLPPs, the most of existing algorithms are very time consuming and can't deal with the large-scale problems in a reasonable computational time. In order to overcome these shortcomings, the paper is devoted to designing an algorithm which can reduce the computational complexity caused by the follower's solution and can also deal with BLPPs with nondifferentiable and nonconvex leader's functions.

In this paper we consider a special nonlinear BLPP, in which the follower's programming is convex and quadratic with respect to y, whereas the leader's functions may be nonconvex and nondifferentiable, that is, the leader's problem is a general mathematical programming. In order to solve the nonlinear bilevel programming problem efficiently, based on Karush-Kuhn-Tucher (K-K-T) conditions, we first convert the problem to an equivalent single level programming, in which the follower's problem is replaced by the linear complementarity system. Then, we propose a novel genetic algorithm(GA) to solve the equivalent problem. At first, each chromosome is composed of two parts, one is the leader's variable values using real-encoding, whereas the other is an $m + q$ string of integers whose components are the indices of basic variables in the linear complementarity system. In addition, a new fitness function is given, in which the leader's constraints are incorporated into penalty term, and whether an individual is feasible can be identified by the

978-0-7695-4297-3/10 $26.00 © 2010 IEEE
DOI 10.1109/CIS.2010.29

fitness; Finally, based on the novel encoding scheme and the fitness, a hybrid GA is proposed, and the efficiency of the algorithm is illustrated by solving 10 test problems.

This paper is organized as follows. The model of the bilevel programming problem are given, and some preliminaries are made in Section II. In Section III some specific techniques are presented, and a hybrid genetic algorithm(HGA) is developed based on these techniques. Section IV proceeds by making experiment and comparison to illustrate the efficiency of HGA. We finally conclude our paper in Section V.

II. Bilevel Programming Problem with Transformation

The nonlinear BLPP considered in this paper can be formulated as follows:

$$\begin{cases} \min F(x,y) \\ s.t.\ G(x,y) \le 0 \\ \min f(x,y) = \dfrac{1}{2}y^T Q(x) y + c(x)^T y \\ s.t.\ A(x)y \ge b(x),\ y \ge 0 \end{cases} \tag{2}$$

where for each x given by the leader, $Q(x) \in R^{m\times m}$ is symmetric and positive semi-definite, $A(x) \in R^{q\times m}$, $b(x) \in R^q$ and $c(x) \in R^m$, and for convenience, are denoted by Q, A, b, and c, respectively.

Now we introduce some related notions [2].

1) Search space: $\Omega = \{(x,y) \mid x \in X, y \in Y\}$, where X and Y are box sets determined by the upper and lower bounds of x and y.

2) Constraint region: $S = \{(x,y) \in \Omega \mid G(x,y) \le 0, A(x)y + b(x) \ge 0, y \ge 0\}$.

3) For x fixed, the feasible region of follower's problem: $S(x) = \{y \in Y \mid A(x)y + b(x) \ge 0, y \ge 0\}$.

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

5) The follower's rational reaction set for each $x \in S(X)$: $M(x) = \{y \in Y \mid y \in \operatorname{argmin}\{f(x,v), v \in S(x)\}\}$.

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

In terms of aforementioned definitions, problem (2) also can be written as:

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

In order to ensure that problem (2) is well posed and solved, in the remainder, we always assume that S is nonempty and $M(x)$ is a single-point set for each x. Since for x fixed, all functions in the follower's problem are differentiable and convex in y, Karush-Kuhn-Tucker stationary-point problem of the follower's programming can be written as

$$\begin{cases} v - Qy + A^T u = c \\ y_0 - Ay = -b \\ v^T y = 0 \\ y_0^T u = 0 \\ u, v, y, y_0 \ge 0. \end{cases} \tag{3}$$

where u, v are Lagrangian multipliers, and $y_0 \in R^q$ is an slack vector. (3) is also called linear complementarity system of the follower's programming. Further, let

$$w = \begin{pmatrix} v \\ y_0 \end{pmatrix}, \quad z = \begin{pmatrix} y \\ u \end{pmatrix}$$

$$M = \begin{pmatrix} Q & -A^T \\ A & 0 \end{pmatrix}, \quad q = \begin{pmatrix} p \\ -b \end{pmatrix}$$

then, (3) can be re-written as follows

$$\begin{cases} w - Mz = q & (c1) \\ w, z \ge 0 & (c2) \\ w^T z = 0 & (c3) \end{cases} \tag{4}$$

Replace the follower's programming in (2) by (3) or (4), then we can transform (2) into a single level programming as follows

$$\begin{cases} \min\limits_{x,w,z} F(x,y) \\ s.t.\ G(x,y) \le 0 \\ E(w,z) = 0 \\ w, z \ge 0 \end{cases} \tag{5}$$

where $E(w,z) = 0$ represents all equations in (4).

For problem (4), if (w,z) satisfying (c3) is a basic feasible solution of (c1-c2), (w,z) is called the complementary basic feasible solution (CBFS) of (4)[9]. For x fixed, if one wants to obtain the optimal solution y of the follower's problem, all that needs to do is to solve problem (4) for a CBFS. In fact, it is computation expensive to solve (4) directly. Considering the characteristics of solutions to (4), there exist at least one feasible base B for each fixed x, which consists of some columns of the coefficient matrix $(I, -M)$, and for each variable pair (w_i, z_i), $i = 1, 2, \ldots, m+q$, only one variable can be selected as basic variable. Hence, when a genetic algorithm is used to solve problem (5), we only search the leader's values and the columns of $(I, -M)$.

III. Solution Method

In this section, We present a new hybrid GA for solving (5). At first, we encode each individual by using the leader's variable values and the bases of $(c1-c2)$ by taking $(c3)$ into account. Then a new fitness function is given, which consists of leader's objective and a penalty term. Next we describe each step of the algorithm in more detail.

A. Chromosome Encoding

We denote a chromosome by two parts: one is the values of x, the other is the basic variable indices of $(c1 - c2)$, which are $m+q$ integers. These indices are taken as follows: At first, $k(0 \leq k \leq m+q)$ columns are chosen randomly from I, let the k column indices be $(i_1, i_2, \ldots, i_k)$. Then the remaining $m+q-k$ columns are chosen at random from all columns of $-M$ except for the i_1th-i_kth columns, and these column indices are denoted by $(i_{k+1}, i_{k+2}, \ldots, i_{m+q})$. According to the scheme, a chromosome can be represented as $(x_1, \ldots, x_n, i_1, \ldots, i_{m+q})$. We denote by B the matrix consisting of columns taken from I and $-M$, if B is singular or $B^{-1}q$ is infeasible, the chromosome is called an infeasible individual, if not, it will be a quasi-feasible individual. Further, If a quasi-feasible individual l satisfies the leader's constraints, l is called a feasible individual.

B. Fitness Evaluation

For each individual $l = (x_1, \cdots, x_n, i_1, i_2, \ldots, i_{m+q})$, if it is a quasi-feasible individual, the fitness $R(l)$ of l is given as follows

$$R(l) = F(x, y) + M_1 \max\{0, G(x, y)\} \tag{6}$$

otherwise, $R(l) = M_2$; where $M_i, i = 1, 2$, are positive real number and large enough.

Note that in (6) y can be obtained from z, and the nonzero components of z can be got from $B^{-1}q$, it means that $R(l)$ can be calculated for each quasi-feasible individual.

Obviously, the feasible individuals and infeasible individuals can be identified by the fitness function.

C. Crossover Operator and Mutation Operator

From the chromosome encoding scheme, one can see that the column selection from $-M$ depends on the column indices taken from I, and so we only consider the x and the column indices from I in the crossover and mutation processes. Let D_I, D_{-M} denote the column indices of $I, -M$, respectively.

Let

$$l = (x_1, \ldots, x_n, i_1, \ldots, i_{m+q})$$
$$l' = (x'_1, \ldots, x'_n, i'_1, \ldots, i'_{m+q})$$

be selected parents for crossover, $\{i_1, \ldots, i_{k_1}\} \subseteq D_I$, $\{i'_1, \ldots, i'_{k_2}\} \subseteq D_I$, and $V = \{i_1, \ldots, i_{k_1}\} \bigcup \{i'_1, \ldots, i'_{k_2}\}$. We give the crossover offspring of l and l' as follows:

$$l_o = (\alpha x + (1-\alpha)x', J_1, J_2), l'_o = (\alpha x' + (1-\alpha)x, J'_1, J'_2)$$

where $\alpha \in [0, 1]$ is random; $x = (x_1, \ldots, x_n)$, $x' = (x'_1, \ldots, x'_n)$; J_1, J'_1 are subsets of V taken randomly, whereas J_2, J'_2 can be got according to J_1, J'_1, respectively.

Let $\bar{l} = (x_1, \ldots, x_n, i_1, \ldots, i_{m+q})$ be a selected parent for mutation. We denote the offspring of mutation by $l_m = (x_o, K_1, K_2)$. Where $x_o = (x_1, \cdots, x_n) + \Delta, \Delta \sim N(0, 1)$; K_1, as a subset of D_I, is taken randomly, and based on the K_1, K_2 can be obtained accordingly.

D. Local Search Scheme

To improve the quality of infeasible individuals, we use a local search scheme when some infeasible chromosomes are found. Let $l = (x, J_1, J_2)$ be an infeasible individual, where $J_1 \subseteq D_I, J_2 \subseteq D_{-M}$. For the fixed x, we change J_1 by adding a column index $i(\notin J_1)$ or deleting an element from J_1, after doing so, J_2 is changed accordingly to satisfy the complementary conditions. The process is not stopped until l become quasi-feasible or a maximum run number of K is reached.

E. Hybrid Genetic Algorithm(HGA)

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

Step 1 (*Initialization*) Initial population $pop(0)$ with population size N is generated randomly. 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. Crossover parents $l, l' \in pop(k)$ are chosen according to p_c, and then the crossover is executed on l and l' to get offspring l_o, l'_o. Let $O1$ stand for the set of all these offspring.

Sept 4 (*Mutation*) For each $\bar{l} \in pop(k)$, we execute the mutation on the individual according to mutation probability p_m, and its offspring is denoted by 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. The local search is executed for at most t offspring, and these individuals are re-evaluated. 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

IV. Simulation Results

In order to illustrate the efficiency of HGA, in this section we select 10 test problems from the references[6], [7], [8]. These problems, as computational examples, are frequently solved to illustrate the performance of the algorithms in literature. In all these problems, the follower's problems are convex quadric in y.

The parameters are chosen as follows: the population size $N = 30$, $p_c = 0.8$, $p_m = 0.1$, $N_1 = 20, t = K = 5$, $M_1 = M_2 = 10000$. For $T01$-$T10$, The algorithm stops when the maximum generations of 100 is reached. We execute HGA in 20 independent runs on each problem on a PC(Intel Pentium IV-2.66GHz), and record the following data:

(1) Best solution (x^*, y^*), and $F(x^*, y^*)$.

(2) $F(\bar{x}, \bar{y})$ at the worst solution $(\bar{x}, \bar{y})$.

(3) Mean value (F_{mean}) of the leader's objective values in 20 runs, median(F_{median}) and standard deviation(std).

Table I
COMPARISON OF THE RESULTS FOUND BY HGA AND THE RELATED ALGORITHMS IN LITERATURE.

	HGA						REF-
$No.$	$F(x^*, y^*)$	F_{mean}	F_{median}	$F(\bar{x}, \bar{y})$	std	(x^*, y^*)	$F(x^*, y^*)$
$T01$[7]	-8.9172	-8.9172	-8.9172	-8.9172	$1.7e-6$	$(1.031, 3.097, 2.597, 1.793)$	-8.92
$T02$[7]	-7.58	-7.58	-7.58	-7.58	$6.3e-4$	$(0.28, 0.47, 2.3, 1.03)$	-7.58
$T03$[7]	-11.999	-11.999	-11.999	-11.999	0	$(51.4, 64.3, 2.99, 2.99)$	-11.999
$T04$[7]	-3.6	-3.5996	-3.5996	-3.599	$3.1e-4$	$(2.0, -0.004, 2.0, 0)$	-3.6
$T05$[7]	-3.92	-3.919	-3.92	-3.917	$8.8e-4$	$(-0.396, 0.8, 2, 0)$	-3.92
$T06$[8]	-1.6509	-1.6509	-1.6509	-1.6509	0	$(1.2, 0.6, 0.3)$	-1.4074
$T07$[8]	231.25	231.31	231.30	231.35	0.04	$(15, 7.5, 10, 7.5)$	231.25
$T08$[8]	0.6389	0.639	0.639	0.639	$3.5e-4$	$(0.61, 0.39, 0, 0)$	0.6389
$T09$[6]	17	17	17	17	0	$(1, 0)$	17
$T10$[6]	28.25	28.25	28.25	28.25	$7.5e-4$	$(0, 0)$	31.25

All results are shown in Table 1. Table 1 provides the comparison of the results found by HGA and the compared algorithms for $T01$-$T10$, and the best solutions found by HGA in 20 runs are also presented in the table. Where REF stands for the related algorithms in references.

We can see from Table 1 that for problems $T06$ and $T10$, all results found by HGA, including the worst one, are better than the best results given by the references, which indicates the compared algorithms get stuck at local optima. For other problems except for $T01$, the best results found by HGA are almost as good as those provided in literature. For $T01$, although the results found by HGA are a little bit worse than that by the compared algorithm, the relative error between the results is less than $3.2e^{-4}$.

It should be noted that in all 20 runs, HGA found uniformly the best results of problems $T03, T06$ and $T09$. For other problems, we can see that the worst results are very close to the best results and the standard deviations are also very small, which means that HGA is stable and robust.

V. CONCLUSIONS

For nonlinear bilevel programming problems in which the follower's problem is a convex quadric programming with respect to y, the paper develops a genetic algorithm based on a base enumeration method. The technique avoids solving the follower's programming directly, and so the amount of computation can be reduced.

ACKNOWLEDGMENT

This research work was supported by the National Natural Science Foundation of China (No. 60873099, 61065009), and the Innovation Research Foundation of Qinghai Normal University.

REFERENCES

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

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

[3] 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.

[4] 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.

[5] 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.

[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, pp. 211-227, 2005.

[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] 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.

[9] M. S. Bazaraa and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*. Wiley, Chichester, 1979.