A Hybrid Genetic Algorithm for Solving a Class of Nonlinear Bilevel Programming Problems

Hecheng Li[1] and Yuping Wang[2]

[1] Department of Mathematics Science,
Xidian University, Xi'an 710071, China
lihecheng@qhnu.edu.cn
[2] School of Computer Science and Technology,
Xidian University, Xi'an 710071, China
ywang@xidian.edu.cn

Abstract. In this paper, a special nonlinear bilevel programming problem (BLPP), in which the follower's problem is a convex quadratic programming in y, is transformed into an equivalent single-level programming problem by using Karush-Kuhn-Tucker(K-K-T) condition. To solve the equivalent problem effectively, firstly, a genetic algorithm is incorporated with *Lemke* algorithm. For x fixed, the optimal solution y of the follower's problem can be obtained by *Lemke* algorithm, then (x, y) is a feasible or approximately feasible solution of the transformed problem and considered as a point in the population; secondly, based on the best individuals in the population, a special crossover operator is designed to generate high quality individuals; finally, a new hybrid genetic algorithm is proposed for solving this class of bilevel programming problems. The simulation on 20 benchmark problems demonstrates the effectiveness of the proposed algorithm.

1 Introduction

The bilevel programming problem(BLPP) can be defined as

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

and $x \in X \subseteq R^n$, $y \in Y \subseteq R^m$. Here $x(y)$ is called the leader's (follower's) variable. In the same way, F(f) is called the leader's (follower's) objective function. The bilevel programming problem is a mathematical model of the leader-follower game, it can be viewed as a static version of the noncooperative, two-person game introduced by Von Stackelberg [1] in the context of unbalanced economic markets. As an optimization problem with a hierarchical structure, BLPP has a wide variety of applications, such as network design, transport system planning, and

T.-D. Wang et al. (Eds.): SEAL 2006, LNCS 4247, pp. 408–415, 2006.

management and economics [2,3,4]. However, owing to the complex structure, the vast majority of researches on BLPP are concentrated on the linear version of the problem [5,6,7,8], and a few works on the nonlinear BLPP. Moreover, most of existing algorithms for nonlinear BLPP are usually based on the assumption that all of the functions are convex and twice differentiable [9,10,11]. In this paper, we extend the linear case to a special class of nonlinear BLPP, where the follower's problem is a convex quadratic programming in y and the leader's functions may be nonconvex and nondifferentiable. In order to solve the transformed problem effectively by using Genetic Algorithm(GA), firstly, for x fixed, we apply *Lemke* algorithm to obtain an optimal solution y of the follower's problem, and (x, y) is regarded as an individual in the population, which guarantees that we can get an approximately feasible point that at least satisfies the follower's problem; secondly, for the leader's constraint $G(x, y) \leq 0$, we design a fitness function in a magnified feasible region. In the new feasible region, the fitness function makes any feasible solutions better than all infeasible solutions; finally, a new hybrid genetic algorithm with specifically designed crossover operator is proposed to solve the class of bilevel programming problems.

2 Bilevel Programming Problem

We consider the following nonlinear bilevel programming problem:

$$\begin{cases} \min\limits_{x \in X} F(x, y) \\ s.t.\ G(x, y) \leq 0 \\ \min\limits_{y \in Y} f(x, y) = 1/2 \times y^T Q(x) y + c(x)^T y \\ s.t.\ A(x)y + b(x) \leq 0,\ y \geq 0 \end{cases} \tag{2}$$

where $F : R^n \times R^m \to R$, $G : R^n \times R^m \to R^p$, for each $x \in X$, $Q(x) \in R^{m \times m}$ is symmetric and positive definite, $A(x) \in R^{q \times m}$, $b(x) \in R^q$ and $c(x) \in R^m$. For the purpose of convenience, denote $Q(x)$, $c(x)$, $A(x)$ and $b(x)$ by Q, c, A, and b, respectively. The sets of X and Y may represent upper and lower bounds on elements of the vectors x and y. Now we introduce some related definitions [14].

1) Search space: $\Omega = \{(x, y) | x \in X, y \in Y\}$.

2) Constraint region: $S = \{(x, y) \in \Omega | G(x, y) \leq 0, A(x)y + b(x) \leq 0,\ y \geq 0\}$.

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

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

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

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

In the remainder, we always assume that for $x \in X$ fixed, $S(x)$ is nonempty as well as S. Since, for x fixed, all functions in the follower's problem are differentiable

and convex in y, K-K-T stationary-point problem of follower's programming can be written as following

$$\begin{cases} Qy + A^T\lambda_1 + \lambda_2 + c = 0 \\ Ay + b + u = 0 \\ \lambda_1^T u = 0 \\ \lambda_2^T y = 0 \\ \lambda_i, y, u \geq 0, i = 1, 2. \end{cases} \tag{3}$$

where $\lambda_1 = (\lambda_{11}, \lambda_{12}, \ldots, \lambda_{1q})^T$, $\lambda_2 = (\lambda_{21}, \lambda_{22}, \ldots, \lambda_{2m})^T$ are Lagrangian multipliers, and $u \in R^q$ is an slack vector. Replace the follower's programming in (2) by (3), then (2) is transformed into (4) as follows

$$\begin{cases} \min\limits_{x \in X} F(x, y) \\ s.t.\ G(x, y) \leq 0 \\ E(x, y, \lambda_1, \lambda_2, u) = 0 \\ y, \lambda_1, \lambda_2, u \geq 0 \end{cases} \tag{4}$$

where $E(x, y, \lambda_1, \lambda_2, u) = 0$ stands for all equations in (3). In order to solve the transformed programming (4), at first, we rewrite (3) as the following linear complementary problem

$$\begin{cases} w + Mz = \hat{b} & (c1) \\ w, z \geq 0 & (c2) \\ w^T z = 0 & (c3) \end{cases} \tag{5}$$

where

$$w = \begin{pmatrix} \lambda_2 \\ u \end{pmatrix}, \quad z = \begin{pmatrix} y \\ \lambda_1 \end{pmatrix}, \quad M = \begin{pmatrix} Q & A^T \\ A & 0 \end{pmatrix}, \quad \hat{b} = \begin{pmatrix} -c \\ -b \end{pmatrix}$$

If (w, z) satisfying (c3) is a basic feasible solution of (c1-c2) in (5), (w, z) is called a complementary basic feasible solution (CBFS) of (5)[13]. For x fixed, if one wants to get the optimal solution y of the follower's problem, what he needs to do is to solve problem (5) via *Lemke* algorithm for a CBFS (w, z) [13]. Take y from z, then (x, y) satisfies all the follower's constraints and for x fixed, $f(x, y)$ gets its optimal value at y. Such points (x, y) form the initial population of GA. In the process of evolving, genetic offspring are generated in two steps. At first, the variable x is acted by crossover or mutation operator to get $\bar{x}$; For $\bar{x}$ fixed, to solve problem (5) for $\bar{y}$. Such $(\bar{x}, \bar{y})$ is regarded as the offspring and satisfies the follower's optimization problem. When the populations are generated with the method, all that we need to do is to solve the following single programming problem

$$\begin{cases} \min\limits_{x \in X} F(x, y) \\ s.t.\ G(x, y) \leq 0 \end{cases} \tag{6}$$

3 New GA

3.1 Fitness Function

Let $\epsilon = (\epsilon_1, \ldots, \epsilon_p)^T$,where ϵ_i are small positive numbers and tend to zero with the increasing of the generations, $\bar{S} = \{(x, y) \in \Omega | G(x, y) \leq \epsilon\}$, $v(x, y) = \max(\max(G(x, y) - \epsilon), 0)$, and K be an upper-bound of $F(x, y)$ on the set $\{(x, y)|v(x, y) = 0\}$. The fitness function $\bar{F}(x, y)$ is defined as following

$$\bar{F}(x, y) = \begin{cases} F(x, y), & v(x, y) = 0; \\ K + v(x, y), & v(x, y) \neq 0. \end{cases}$$

The fitness function implies that any point in $\bar{S}$ always is better than all points out of $\bar{S}$, and between two points out of $\bar{S}$, the point with the smaller v is better than the other.

3.2 Crossover Operator

In the process of evolving, the best point found always is recorded as (x_{best}, y_{best}). Let (x, y) be a crossover parent. We, at first, get an $\bar{x}$ by the formula $\bar{x} = x + diag(\tau) \times r \times (x_{best} - x)$, where $\tau \in R^n$ is a positive vector determined by the upper and lower bounds of variable x, and $r \in [0, 1]$ is a random number. For $\bar{x}$ fixed, we solve the problem (5) for $\bar{y}$. The resulting $(\bar{x}, \bar{y})$ is the crossover offspring of (x, y).

3.3 Mutation Operator

Let (x, y) be a mutation parent. We, at first, get an $\hat{x}$ by Gaussian mutation $\hat{x} = x + \triangle(0, \sigma^2)$, where each component of $\triangle(0, \sigma^2)$ obeys Gaussian distribution $N(0, \sigma_i^2)$, $i = 1, 2, \ldots, n$. For $\hat{x}$ fixed, we solve the problem (5) for $\hat{y}$. The resulting $(\hat{x}, \hat{y})$ is the mutation offspring of (x, y).

3.4 Proposed Algorithm

Algorithm 1
Step 1 (Initialization). Randomly generate a set *pop* of N_p points in X. For each $x \in$ *pop* fixed, we solve the problem (5) for y. If $y \notin Y$, randomly generate another x to replace the original one, the process doesn't stop until $y \in Y$ is satisfied. All such points (x, y) form the initial population *pop*(0) with N_p individuals. Let N stand for the set of all the best points found by *Algorithm* 1 so far and $k = 0$.
Step 2. Evaluate the fitness value $\bar{F}(x, y)$ of each point in *pop*(k), and let $N = \{p_{i_1}, \cdots, p_{i.}\}$.
Step 3 (*Crossover*). Randomly select a parent p from *pop*(k) according to the crossover probability p_c. We can obtain a crossover offspring $\bar{p}$ via the crossover operator. Let $O1$ stand for the set of all these offspring.

Table 1. Comparison of the best results found by *Algorithm*1 and by the related algorithms in references for Problems 1-20

		$F(x^*, y^*)$		$f(x^*, y^*)$	
No.	CPU(s)	*Algorithm*1	Ref.	*Algorithm*1	Ref.
$F1$([15]E4.1)	10.0115	0.5015	0.5	-15.7475	-14.4879
$F2$ ([15]E4.2)	8.1894	0.5015	0.5	-4.5000	-4.5
$F3$ ([15]E4.3)	12.0063	1.8605	1.859	-10.9321	-10.9310
$F4$ ([15]E4.4)	10.6969	0.8485	0.919	-22.9505	-19.4686
$F5$ ([15]E4.5)	9.3870	0.8975	0.897	-14.9249	-14.9311
$F6$ ([15]E4.6)	10.1625	1.5629	1.562	-11.6826	-11.6808
$F7$ ([15]E3.1)	10.1292	-8.9172	-8.92	-6.1179	-6.054
$F8$ ([15]E3.2)	9.2068	-7.5784	-7.56	-0.5723	-0.5799
$F9$ ([15]E3.3)	11.4052	-11.9985	-12	-438.4165	-112.71
$F10$ ([15]E3.4)	10.3932	-3.6	-3.6	-2	-2
$F11$ ([15]E3.5)	9.5588	-3.92	-3.15	-2	-16.29
$F12$ ([16]E5.2)	10.625	225	225	100	100
$F13$ ([17]E3.2)	11.2531	0	5	200	0
$F14$ ([9]E3.2)	10.2531	-12.6787	-12.6787	-1.016	-1.016
$F15$ ([18]E4.2)	23.2859	-6600	-6599.99	$f_1 = 23.6358$ $f_2 = 30.5833$	$f_1 = 23.47$ $f_2 = 30.83$
$F16$ ([12]E6.2)	6.0922	81.3279	82.44	-0.3359	0.271
$F17$ ([12]E6.3)	6.4875	100	100.58	0	0.001
$F18$ ([12]E6.4)	13.6151	-1.2099	3.57	7.6172	2.4
$F19$	23.0260	0	NA	13.3283	NA
$F20$	11.7354	0	NA	200	NA

Sept 4 (Mutation). Randomly select parents from $pop(k)$ according to the mutation probability p_m. For each selected parent p, we can get its offspring $\hat{p}$ via the mutation operator. Let $O2$ stand for the set of all these offspring.
Step 5 (*Selection*). Let $O = O1 \cup O2$ and 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 $N_p - n_1$ points from the remaining points of the set. All these selected points form the next population $pop(k+1)$.
Step 6. If the termination condition is satisfied, then stop; otherwise, renew N, let $k = k + 1$ and $\epsilon = \theta\epsilon$, $\theta \in [0, 1]$, go to *Step* 3.

4 Simulation Results

In this section, 20 benchmark problems are used for simulation, in which the first 18 problems are selected from the related references. In order to illustrate the efficiency of the proposed algorithm for solving complex nonlinear BLPPs with nondifferentiable , even nonconvex, leader's objective function, we construct 2 new benchmark problems, $F19$ and $F20$ as follows

Table 2. Comparison of the best solutions found by *Algorithm*1 in 30 runs and by the related algorithms in references for problems 1-20

No.	(x^*, y^*)	
	*Algorithm*1	Ref.
$F1$([15]E4.1)	$(2.25.5, 2.9985, 2.9985)$	$(2.07, 3, 3)$
$F2$ ([15]E4.2)	$(0, 2.9985, 2.9985)$	$(0, 3, 3)$
$F3$ ([15]E4.3)	$(3.4564, 1.7071, 2.5685)$	$(3.456, 1.707, 2.569)$
$F4$ ([15]E4.4)	$(2.8563, 3.8807, 3.0402)$	$(2.498, 3.632, 2.8)$
$F5$ ([15]E4.5)	$(3.9977, 1.6651, 3.8865)$	$(3.999, 1.665, 3.887)$
$F6$ ([15]E4.6)	$(1.9095, 2.9786, 2.2321)$	$(1.90, 2.979, 2.232)$
$F7$ ([15]E3.1)	$(1.2046, 3.0972, 2.5968, 1.7922)$	$(0.97, 3.14, 2.6, 1.8)$
$F8$ ([15]E3.2)	$(0.2785, 0.4759, 2.3439, 1.0327)$	$(0.28, 0.48, 2.34, 1.03)$
$F9$ ([15]E3.3)	$(46.7128, 124.9863, 2.9985, 2.9985)$	$(20.26, 42.81, 3, 3,)$
$F10$ ([15]E3.4)	$(2, 0.0296, 2, 0)$	$(2, 0.06, 2, 0)$
$F11$ ([15]E3.5)	$(-0.4050, 0.7975, 2, 0)$	$(2.42, -3.65, 0, 1.58)$
$F12$ ([16]E5.2)	$(20, 5, 10, 5)$	$(20, 5, 10, 5)$
$F13$ ([17]E3.2)	$(0, 0, -10, -10)$	$(25, 30, 5, 10)$
$F14$ ([9]E3.2)	$(0, 2, 1.8750, 0.9063)$	$(0, 2, 1.8750, 0.9063)$
$F15$ ([18]E4.2)	$(7.034, 3.122, 11.938, 17.906, 0.25, 9.906, 29.844, 0)$	$(7.05, 4.13, 11.93, 17.89, 0.26, 9.92, 29.82, 0)$
$F16$ ([12]E6.2)	$(10.0164, 0.8197)$	$(10.04, 0.1429)$
$F17$ ([12]E6.3)	$(10.0000, 10.0000)$	$(10.03, 9.969)$
$F18$ ([12]E6.4)	$(1.8889, 0.8889, 0)$	NA
$F19$	$(13.6508, 8.5111, 10.0000, 8.5111)$	NA
$F20$	$(0, 0, -10, -10)$	NA

F19).The problem is the same as $F12$ except for

$$F(x, y) = |\sin((x_1 - 30)^2 + (x_2 - 20)^2 - 20y_1 + 20y_2 - 225)|$$

F20). The problem is the same as $F13$ except for

$$F(x, y) = |\sin(2x_1 + 2x_2 - 3y_1 - 3y_2 - 60)|$$

In all problems, the follower's problems are quadratic programming problems in y for x fixed, while the leader's functions may be nonlinear and nondifferentiable.

The parameters are chosen as follows: the population size $N_p = 30$, $p_c = 0.8$, $p_m = 0.3$, $n_1 = 10$, the initial $\epsilon = (1, \cdots, 1) \in R^p$, $\theta = 0.7$ for $k \leq k_{max}/2$, while $\theta = 0$ for $k > k_{max}/2$, where k represents the generation number, while k_{max} the maximum generation number. For $F1 - F20$, The algorithm stops after 50 generations. We execute *Algorithm* 1 in 30 independent runs on each problem on a computer(Intel Pentium IV-2.66GHz), and record the following data: (1) best solution (x^*, y^*), and the leader's(follower's) objective values $F(x^*, y^*)(f(x^*, y^*))$ at the best solution; (2) worst solution $(\bar{x}, \bar{y})$ and the leader's (follower's) objective function value $F(\bar{x}, \bar{y})$ $(f(\bar{x}, \bar{y}))$ at the point $(\bar{x}, \bar{y})$; (3) mean value of CPU time(denoted by CPU in short).

Table 3. The worst results found by *Algorithm*1

No.	$F(\bar{x},\bar{y})$	$f(\bar{x},\bar{y})$	$(\bar{x},\bar{y})$
$F6$ ([15]E4.6)	1.5633	-11.6859	$(1.9101, 2.9785, 2.2319)$
$F8$ ([15]E3.2)	-7.5782	-0.5726	$(0.2827, 0.4668, 2.3432, 1.0307)$
$F15$ ([18]E4.2)	-6599.6	$f_1 = 25.1041$ $f_2 = 28.4778$	$(6.9191, 3.0261, 12.0210, 18.0337,$ $0.1417, 9.8035, 30.0525, 0)$

All results are presented in Tables 1-3, in which Table 1 provides the comparison of the best results found by *Algorithm*1 in 30 runs and by the related algorithms in references for problems 1-20 , as well as mean CPU time needed by *Algorithm*1 in a run. Table 2 shows the best solutions found by *Algorithm*1 in 30 runs and by the related algorithms in references for problems 1-20. For those functions for which *Algorithm*1 finds different results in 30 runs, Table 3 gives the worst solutions $(\bar{x},\bar{y})$ and the related objective function values. The first volume of Tables 1-3 consists of two parts, Fi and $([l]Ej.k)$. Fi stand for test function numbers $(i = 1, 2, \cdots, 20)$, and $([l]Ej.k)$ the kth example in Section j of reference $[l]$. NA means that the result is not available for the algorithm and Ref. stands for the related algorithms in references in Tables 1-3.

It can be seen from Table 1 that for problems $F4, F11, F13, F16, F17$ and $F18$, the best results found by *Algorithm* 1 are better than those by the compared algorithms in the references, which indicates these algorithms can't find the optimal solutions of these problems; For the constructed $F19$ and $F20$, *Algorithm*1 can find the best results; For other problems, the best results found by *Algorithm*1 are almost as good as those by the compared algorithms.

In all 30 runs, *Algorithm*1 finds the best results of all problems except the problems 6, 8, and 15. For the three problems, the proposed algorithm also found the best solutions in 25, 24 and 19 runs, respectively. The worst results are shown in Table 3. From Table 3, we can find the worst results are close to the best results. This means that the proposed *Algorithm* 1 is stable and robust.

5 Conclusion

For the nonlinear bilevel programming problem, whose follower level problem is a convex quadratic programming in y, we transform it into a single-level programming problem by using K-K-T condition. To solve the equivalent problem effectively, we propose a new hybrid genetic algorithm incorporated with *Lemke* algorithm, in which an efficient crossover operator is designed to generate high quality individuals. The simulation on 20 benchmark problems demonstrates the robustness and the effectiveness of the proposed algorithm.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 60374063.

References

1. Stackelberg, H. Von: The Theory of the Market Economy. William Hodge, Lond, UK (1952)
2. Marcotte, P.: Network optimization with continuous parameters. Trans. Sci. 17 (1983) 181–197
3. Suh, S. , Kim, T.: Solving nonlinear bilevel programming models of equilibrium network design problem: A comparative review. Ann. Oper. Res. 34 (1992) 203–218
4. Milier, T. , Friesz, T., Robin, R.: Heuristic algorithms for delivered price spatially competitive net work facility location problems. Ann. Oper. Res. 34 (1992) 177–202
5. Hansen, P., Jaumard, B., Savard, G.: New branch-and-bound rules for linear bilevel programming. SIAM J. Sci. Stat. Comput. 13(5) (1992) 1194–1217
6. Amouzegar, M. A., Moshirvaziri, K.: A penalty method for linear bilevel programming problems. In Multilevel optimization: Algorithms, complexity and Applications. A. Migdalas, M. Pardalos, and P. Varbrand, Eds. Norwell,Kluwer MA (1997) ch.11
7. Bard, J. ,Moore, J.: A branch and bound algorithm for the bilevel programming problem. SIAM J. Sci. Stat. Comput. 11(5) (1990) 281–292
8. Candler, W. , Townsley, R.: A linear bi-level programming problem. Comput. Oper. Res. 9(1) (1982) 59–76
9. Amouzegar, M. A.: A global optimization method for nonlinear bilevel programming problems. IEEE Trans. Syst., Man, Cybern. B, Cybern. 29(6) (1999) 771–777
10. Vicente, L., Savard, G., Judice, J.:Descent approach for quadratic bilevel programming. J. Optim. Theory Appl. 81(2) (1994) 379–399
11. Al-Khayyal, F., Horst, R., Pardalos, P.: Global optimization of concave function subject to quadratic constraints: An application in nonlinear bilevel programming. Ann. Oper. Res. 34 (1992) 125–147
12. Oduguwa, V., Roy, R.: Bi-level optimization using genetic algorithm . In Proc. IEEE Int. Conf. Artificial Intelligence Systems (2002) 123–128
13. Bazaraa, M. S., Shetty, C. M.: Nonlinear Programming: Theory and Algorithms. Wiley, Chichester (1979)
14. Bard, J. F.: Practical Bilevel Optimization. Norwell, Kluwer, MA(1998)
15. Outrata, J. V.: On the numerical solution of a class of Stackelberg problem. Zeitschrift Fur Operational Reseach 34 (1990) 255–278
16. Shimizu, K., Aiyoshi, E.: A new computational method for Syackelberg and min-max problems by use of a penalty method. IEEE Trans. Autom. Control AC-26(2) (1981) 460–466
17. Aiyoshi, E., Shimuzu, K.: A solution method for the static constrained Stackelberg problem via penalty method. IEEE Trans. Autom. Control AC-29(12) (1984) 1111–1114
18. Bard, J. F.: Covex two-level optimization. Math. Programming 40 (1988) 15–27