Pergamon

Computers Math. Applic. Vol. 36, No. 7, pp. 79–89, 1998
© 1998 Elsevier Science Ltd. All rights reserved
Printed in Great Britain
0898-1221/98 $19.00 + 0.00

PII: S0898-1221(98)00174-6

Stackelberg-Nash Equilibrium for Multilevel Programming with Multiple Followers Using Genetic Algorithms

BAODING LIU
Department of Applied Mathematics
Tsinghua University, Beijing 100084, China
liu@midwest.com.cn

(Received and accepted February 1998)

Abstract—Multilevel programming offers a means of studying decentralized noncooperative decision systems. Unfortunately, multilevel programming is lacking efficient algorithms due to its computational difficulties such as nonconvexity and NP-hardness. This paper will design a genetic algorithm for solving Stackelberg-Nash equilibrium of nonlinear multilevel programming with multiple followers in which there might be information exchange among the followers. As a byproduct, we obtain a means for solving classical minimax problems. Finally, some numerical examples are provided to illustrate the effectiveness of the proposed genetic algorithm. © 1998 Elsevier Science Ltd. All rights reserved.

Keywords—Mathematical programming, Multilevel programming, Genetic algorithm.

1. INTRODUCTION

Now we consider a decentralized noncooperative decision system in which one leader and several followers of equal status are involved. We assume that the leader and followers may have their own decision variables and objective functions, and the leader can only influence (rather than dictate) the reactions of followers through his own decision variables, while the followers have full authority to decide how to optimize their objective functions in view of the decisions of the leader and other followers. A powerful tool dealing with decentralized decision systems is the so-called multilevel programming.

The formulations of multilevel programming may vary considerably from one paper to another. Wen and Hsu [1] and Ben-Ayed [2] reviewed the models and algorithms as well as applications of linear bilevel programming. Cassidy *et al.* [3] presented a bilevel programming model for a central government distributing resources to its subdivisions. Bracken and McGill [4] formulated bilevel models for strategic-force planning and general-purpose-force planning. Anandalingam and Apprey [5] discussed an application to a water conflict problem between India and Bangladesh. Fortuny-Amat and McCarl [6] applied bilevel programming for a fertilizer dealer to decide the base price of fertilizer in order to maximize his profit. Liu and Esogbue [7] constructed a bilevel fuzzy programming for fuzzy criterion clustering.

A lot of numerical algorithms to multilevel programming have been developed by several authors. Candler and Townsley [8] presented an implicit enumeration scheme. Bialas and Karwan [9] designed the k^{th} best algorithm and a parametric complementary pivot algorithm.

Bard [10,11] gave various necessary and sufficient conditions of optimal solution and proposed a one-dimensional grid search algorithm. A branch and bound algorithm was presented by Bard and Moore [12] based on Kuhn-Tucker conditions. Savard and Gauvin [13] gave the steepest descent direction for nonlinear bilevel programming problems. And Liu and Esogbue [7] provided a genetic algorithm for a special nonlinear bilevel programming.

Ben-Ayed and Blair [14] showed that multilevel programming is an NP-hard problem via the well-known Knapsack Problem. Thus, in order to obtain the global optimal solution of general multilevel programming models, we should design some heuristic processes or innovative computations. So this paper will design a genetic algorithm for solving Stackelberg-Nash equilibrium of general multilevel programming with multiple followers in which there might be information exchange among the followers. It is known that classical minimax problems are a special kind of bilevel programming, thus as a byproduct, we obtain a means for searching for minimax solutions. Finally, some numerical examples are provided to illustrate the effectiveness of the proposed genetic algorithm.

2. MULTILEVEL PROGRAMMING

As a special case of multilevel programming, bilevel programming has drawn most of attention paid to this field. Now we assume that in a decentralized two-level decision system there are one leader and m followers. Let $\mathbf{x}$ and $\mathbf{y}_i$ be the control vector of the leader and the i^{th} followers, $i = 1, 2, \dots, m$, respectively. We also assume that the objective functions (without loss of generality, all are to be maximized) of the leader and i^{th} followers are $F(\mathbf{x}, \mathbf{y}_1, \dots, \mathbf{y}_m)$ and $f_i(\mathbf{x}, \mathbf{y}_1, \dots, \mathbf{y}_m)$, $i = 1, 2, \dots, m$, respectively.

In addition, let S be the feasible set of control vector $\mathbf{x}$ of the leader, defined by

$$S = \{\mathbf{x} \mid G(\mathbf{x}) \le 0\}, \tag{1}$$

where G is a vector-valued function of decision vector $\mathbf{x}$ and 0 is a vector with zero components. Then for each decision $\mathbf{x}$ chosen by the leader, the feasible set Y of control array $(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m)$ of followers should be dependent on $\mathbf{x}$, and generally represented by

$$Y(\mathbf{x}) = \{(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m) \mid g(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m) \le 0\}, \tag{2}$$

where g is a vector-valued function not only of $\mathbf{x}$ but also of $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m$.

Assume that the leader first chooses his control vector $\mathbf{x} \in S$, and the followers determine their control array $(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m) \in Y(\mathbf{x})$ after that. Then a general type of bilevel programming should be formulated as follows,

$$\begin{array}{l}
\max F(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m) \\
\text{subject to:} \\
\quad G(\mathbf{x}) \le 0 \\
\quad \text{where each } \mathbf{y}_i (i = 1, 2, \dots, m) \text{ solves} \\
\qquad\qquad \max\limits_{\mathbf{y}_i} f_i(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m) \\
\qquad\qquad \text{subject to:} \\
\qquad\qquad\quad g(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m) \le 0.
\end{array} \tag{3}$$

This bilevel programming is a multiperson noncooperative game with leader-follower strategy. When $m = 1$ and $f_1(\mathbf{x}, \mathbf{y}_1) = -F(\mathbf{x}, \mathbf{y}_1)$, i.e., the objective function of follower is in direct opposition to the objective function of leader, the bilevel programming is a classical minimax problem. For each follower i, $(i = 1, 2, \dots, m)$, if $\mathbf{x}, \mathbf{y}_1, \dots, \mathbf{y}_{i-1}, \mathbf{y}_{i+1}, \dots, \mathbf{y}_m$ are revealed by the leader and other followers, then the reaction $\mathbf{y}_i^*$ of the i^{th} follower must be the optimal solution of the optimization problem

$$\begin{array}{l}\max\limits_{\mathbf{y}_i} f_i(\mathbf{x},\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_m)\\ \text{subject to:}\\ \qquad g(\mathbf{x},\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_m)\le 0,\end{array} \tag{4}$$

whose set of optimal solutions may consist of multiple points, and such a set is also called the *rational reaction set* of the i^{th} follower.

However, all the followers are of equal status, and they must reveal their strategies simultaneously. So, for all followers, a popular solution concept is the so-called *Nash equilibrium* defined as the array $(\mathbf{y}_1^*,\mathbf{y}_2^*,\ldots,\mathbf{y}_m^*)\in Y(\mathbf{x})$ with respect to $\mathbf{x}$, i.e.,

$$f_i(\mathbf{x},\mathbf{y}_1^*,\ldots,\mathbf{y}_{i-1}^*,\mathbf{y}_i,\mathbf{y}_{i+1}^*,\ldots,\mathbf{y}_m^*)\le f_i(\mathbf{x},\mathbf{y}_1^*,\ldots,\mathbf{y}_{i-1}^*,\mathbf{y}_i^*,\mathbf{y}_{i+1}^*,\ldots,\mathbf{y}_m^*), \tag{5}$$

for any $(\mathbf{y}_1^*,\ldots,\mathbf{y}_{i-1}^*,\mathbf{y}_i,\mathbf{y}_{i+1}^*,\ldots,\mathbf{y}_m^*)\in Y(\mathbf{x})$ and $i=1,2,\ldots,m$. If there is a unique Nash equilibrium, perhaps all the followers might make such an equilibrium because any follower cannot improve his own objective by altering his strategy unilaterally. It must be also noted that, in general, Nash equilibrium is neither unique nor Pareto optimal, and any one follower can deviate from the Nash equilibrium and move to a better solution.

Stackelberg-Nash equilibrium of the bilevel programming has been discussed by Cruz [15], Sherali *et al.* [16] and Anandalingam and Apprey [5]. Let $(\mathbf{x}^*,\mathbf{y}_1^*,\mathbf{y}_2^*,\ldots,\mathbf{y}_m^*)$ be an array with $\mathbf{x}^*\in S$ and $(\mathbf{y}_1^*,\mathbf{y}_2^*,\ldots,\mathbf{y}_m^*)$ be a Nash equilibrium of followers with respect to $\mathbf{x}^*$. We call the array $(\mathbf{x}^*,\mathbf{y}_1^*,\mathbf{y}_2^*,\ldots,\mathbf{y}_m^*)$ a *Stackelberg-Nash equilibrium* to the bilevel programming (3) if and only if,

$$F(\overline{\mathbf{x}},\overline{\mathbf{y}}_1,\overline{\mathbf{y}}_2,\ldots,\overline{\mathbf{y}}_m)\le F(\mathbf{x}^*,\mathbf{y}_1^*,\mathbf{y}_2^*,\ldots,\mathbf{y}_m^*) \tag{6}$$

for any $\overline{\mathbf{x}}\in S$ and the Nash equilibrium $(\overline{\mathbf{y}}_1,\overline{\mathbf{y}}_2,\ldots,\overline{\mathbf{y}}_m)$ with respect to $\overline{\mathbf{x}}$.

Unfortunately, not every multilevel programming has a solution even though it has a nonempty compact feasible set. For example, when some control vector $\mathbf{x}^*$ is given by the leader, the rational reaction set of some follower might consist of multiple points, and the followers would be indifferent among these points. However, the leader might not experience the same indifference with respect to his objective function. Thus, the objective function of leader is not well defined for this case. Some illustrative examples were given in Bard and Falk [17] and Ben-Ayed [2].

In this paper, in order to avoid the ambiguity arising in the reactions of followers, we only discuss the case in which the rational reaction set consists of a single point, or the leader is also indifferent among the multiple points in the rational reaction set.

3. COMPUTING NASH EQUILIBRIUM

Now we define symbols

$$\mathbf{y}_{-i}=(\mathbf{y}_1,\ldots,\mathbf{y}_{i-1},\mathbf{y}_{i+1},\ldots,\mathbf{y}_m),\qquad i=1,2,\ldots,m. \tag{7}$$

For any decision $\mathbf{x}$ revealed by the leader, if the i^{th} follower knows the strategies $\mathbf{y}_{-i}$ of other followers, then the optimal reaction of the i^{th} follower is represented by a mapping $\mathbf{y}_i=r_i(\mathbf{y}_{-i})$ which should solve the subproblem

$$\begin{array}{l}\max\limits_{\mathbf{y}_i} f_i(\mathbf{x},\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_m)\\ \text{subject to:}\\ \qquad g(\mathbf{x},\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_m)\le 0.\end{array} \tag{8}$$

The Nash equilibrium of the m followers will be the solution $(\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_m)$ of the system of equations

$$\mathbf{y}_i=r_i(\mathbf{y}_{-i}),\qquad i=1,2,\ldots,m. \tag{9}$$

In other words, we should find a fixed point of the vector-valued function (r_1, r_2, j, r_m). In order to solve the system of equations (9), we should design some efficient algorithms. The argument breaks down into three cases.

CASE I. If we have explicit expressions of all functions r_i, $i = 1, 2, \dots, m$, then we might get an analytic solution to system (9). Unfortunately, it is almost impossible to do this in practice.

CASE II. In most cases, no analytic solution of (9) can be obtained, so the system (9) might be solved by some iterative method that generates a sequence of points $\mathbf{y}^k = (\mathbf{y}_1^k, \mathbf{y}_2^k, \dots, \mathbf{y}_m^k)$, $k = 0, 1, 2, \dots$ via the iteration formula

$$\mathbf{y}_i^{k+1} = r_i(\mathbf{y}_{-i}^k), \qquad i = 1, 2, \dots, m, \tag{10}$$

where $\mathbf{y}_{-i}^k = (\mathbf{y}_1^k, \dots, \mathbf{y}_{i-1}^k, \mathbf{y}_{i+1}^k, \dots, \mathbf{y}_m^k)$. However, generally speaking, it is not easy to verify the conditions on the convergence of iterative method for practical problems. When we solve some given problem on a computer, we might employ the iterative method. If we indeed find a solution, then the problem is solved. Otherwise, we have to try other methods.

CASE III. If the iterative method fails to find a fixed point, we may employ a genetic algorithm to solve the following minimization problem,

$$\begin{aligned} &\min R(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m) = \sum_{i=1}^{m} \|\mathbf{y}_i - r_i(\mathbf{y}_{-i})\| \\ &\text{subject to:} \\ &\qquad g(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m) \le 0. \end{aligned} \tag{11}$$

If an array $(\mathbf{y}_1^*, \mathbf{y}_2^*, \dots, \mathbf{y}_m^*)$ satisfies that $R(\mathbf{y}_1^*, \mathbf{y}_2^*, \dots, \mathbf{y}_m^*) = 0$, then $\mathbf{y}_i^* = r_i(\mathbf{y}_{-i}^*)$, $i = 1, 2, \dots, m$ and $(\mathbf{y}_1^*, \mathbf{y}_2^*, \dots, \mathbf{y}_m^*)$ must be a solution of (9). In other words, if the minimizing solution $(\mathbf{y}_1^*, \mathbf{y}_2^*, \dots, \mathbf{y}_m^*)$ of the minimization problem (11) is such that the objective value $R(\mathbf{y}_1^*, \mathbf{y}_2^*, \dots, \mathbf{y}_m^*)$ is zero, then $(\mathbf{y}_1^*, \mathbf{y}_2^*, \dots, \mathbf{y}_m^*)$ is a solution of (9). Otherwise, the system of equations (9) might be considered inconsistent, i.e., there is no Nash equilibrium of followers in the given bilevel programming. Although this method can deal with general problems, it is a slow way to find a Nash equilibrium.

4. A GENETIC ALGORITHM FOR STACKELBERG-NASH EQUILIBRIUM

Genetic algorithms have demonstrated considerable success in providing good solutions to many complex optimization problems and received more and more attentions. They have been well documented by numerous pieces of literature, such as [18–20], and applied to a wide variety of optimization problems.

In this section, we will design a genetic algorithm for solving the Stackelberg-Nash equilibrium of general bilevel programming models with multiple followers in which there might be information exchange among the followers. As opposed to the other literature dealing with multilevel programming, we will not assume any linearity, convexity, continuity, and differentiability in the bilevel programming models. In view of the complexity of structure of the bilevel programming models, the proposed genetic algorithm might involve some iterative process or evolution subprocess for solving Nash equilibrium of followers for each control vector revealed by the leader. In fact, in order to enhance the ability of genetic algorithms, any algorithm essential for subproblems can be inserted into the evolution process. For example, stochastic simulations were involved in a genetic algorithm for solving expected value models [21], chance constrained programming [22], and dependent-chance programming [23–25]; and fuzzy simulations were also involved in a genetic algorithm for solving fuzzy programming [26].

4.1. Representation Structure

We can use a binary vector as a chromosome to represent real value of decision variable, where the length of the vector depends on the required precision. The necessity for binary coding has received considerable criticism. An alternative approach to represent a solution is the floating point implementation in which each chromosome vector is coded as a vector of floating numbers, of the same length as the solution vector. Here, we use a vector $V = (x_1, x_2, \dots, x_n)$ as a chromosome to represent the control vector of leader, where n is the dimension.

4.2. Initialization Process

At first, we should eliminate all equality constraints by replacing some variables by the representation of the remaining variables. Then we define an integer pop_size as the number of chromosomes and initialize pop_size chromosomes randomly on the feasible set S. Usually, it is not easy for complex optimization problems to produce feasible chromosome explicitly. So we employ one of the following two ways as the initialization process, depending on what kind of information the decision maker can give. First case is that the decision maker can determine an interior point, denoted by V_0, in the constraint set S. This is very possible for a real decision problem. We also need to define a large positive number M which ensures that all the genetic operators are probabilistically complete for the feasible solutions. This number M is used not only for initialization process but also for mutation operation. The pop_size chromosomes will be produced as follows. We randomly select a direction $\mathbf{d}$ in $\Re^n$ and define a chromosome V as $V_0 + M \cdot \mathbf{d}$ if it is in the feasible set S, otherwise, we set M as a random number between 0 and M until $V_0 + M \cdot \mathbf{d}$ is feasible. We mention that a feasible solution for the inequality constraints can be found in finite times by taking random number since V_0 is an interior point. Repeat this process pop_size times and produce pop_size initial feasible solutions $V_1, V_2, \dots, V_{\text{pop\_size}}$. If the decision maker fails to give such an interior point, but can predetermine a region which contains the optimal solution. Usually, this region will be designed to have nice sharp, for example, an n-dimensional hypercube, because the computer can easily sample points from a hypercube. We generate a random point from the hypercube and check the the feasibility of this point. If it is feasible, then it will be accepted as a chromosome. If not, then we regenerate a point from the hypercube randomly until a feasible one is obtained. Repeat the above process pop_size times, we can make pop_size initial feasible chromosomes $V_1, V_2, \dots, V_{\text{pop\_size}}$.

4.3. Evaluation Function

Evaluation function, denoted by eval(V), is to assign a probability of reproduction to each chromosome V so that its likelihood of being selected is proportional to its fitness relative to the other chromosomes in the population, that is, the chromosomes with higher fitness will have more chance to produce offspring by using *roulette wheel selection.* Let $V_1, V_2, \dots, V_{\text{pop\_size}}$ be the pop_size chromosomes at the current generation. The pop_size chromosomes $V_1, V_2, \dots, V_{\text{pop\_size}}$ can be rearranged from good to bad, i.e., the better the chromosome is, the smaller ordinal number it has. This arrangement is usually based on the values of objective function of the leader. Since the objective value of the leader is also dependent on the Nash equilibrium, so at present we must compute the Nash equilibrium for each given chromosome (control vector of leader) by an iterative method or a genetic algorithm. Now let a parameter $a \in (0, 1)$ in the genetic system be given, then we can define the so-called *rank-based evaluation function* as follows,

$$\text{eval}(V_i) = a(1-a)^{i-1}, \qquad i = 1, 2, \dots, \text{pop\_size}. \tag{12}$$

We mention that $i = 1$ means the best individual, i =pop_size the worst individual.

4.4. Selection Process

The selection process is based on spinning the roulette wheel pop_size times, each time we select a single chromosome for a new population in the following way.

Step 1. Calculate the cumulative probability q_i for each chromosome V_i,

$$q_0 = 0 \quad q_i = \sum_{j=1}^{i} \text{eval}\,(V_j), \qquad i = 1, 2, \dots, \text{pop\_size}. \tag{13}$$

Step 2. Generate a random real number r in $(0, q_{\text{pop\_size}}]$.
Step 3. Select the i^{th} chromosome V_i $(1 \le i \le \text{pop\_size})$ such that $q_{i-1} < r \le q_i$.
Step 4. Repeat the 2nd and 3rd steps pop_size times and obtain pop_size copies of chromosomes.

4.5. Crossover Operation

We define a parameter P_c of a genetic system as the probability of crossover. In order to determine the parents for crossover operation, let us do the following process repeatedly from $i = 1$ to pop_size: generating a random real number r from the interval $[0, 1]$, the chromosome V_i is selected as a parent if $r < P_c$. We denote the selected parents as $V_1', V_2', V_3', \dots$ and divide them to the following pairs: $(V_1', V_2'), (V_3', V_4'), (V_5', V_6'), \dots$. Let us illustrate the crossover operator on each pair by (V_1', V_2'). At first, we generate a random number c from the open interval $(0, 1)$, then the crossover operator on V_1' and V_2' will produce two children X and Y as follows:

$$X = c \cdot V_1' + (1 - c) \cdot V_2' \quad \text{and} \quad Y = (1 - c) \cdot V_1' + c \cdot V_2'. \tag{14}$$

If the feasible set is convex, this arithmetical crossover operation ensures that both children are feasible if both parents are. However, in many cases, the feasible set is not necessarily convex or hard to verify the convexity. So we must check the feasibility of each child. If both children are feasible, then we replace the parents by them. If not, we keep the feasible one if exists, and then redo the crossover operator by regenerating the random number c until two feasible children are obtained or a given number of cycles is finished. In this case, we only replace the parents by the feasible children.

4.6. Mutation Operation

We define a parameter P_m of a genetic system as the probability of mutation. Similar to the process of selecting parents for crossover operation, we repeat the following steps from $i = 1$ to pop_size: generating a random real number r from the interval $[0, 1]$, the chromosome V_i is selected as a parent for mutation if $r < P_m$. For each selected parent, denoted by $V = (x_1, x_2, \dots, x_n)$, we mutate it by the following way. We choose a mutation direction $\mathbf{d}$ in $\Re^n$ randomly, if $V + M \cdot \mathbf{d}$ is not feasible for the constraint set S, then we set M as a random number between 0 and M until it is feasible. If the above process cannot find a feasible solution in a predetermined number of iterations, then set $M = 0$. We replace the parent V by its child $X = V + M \cdot \mathbf{d}$.

4.7. The Procedure

Following selection, crossover and mutation, the new population is ready for its next evaluation. The genetic algorithm will terminate after a given number of cyclic repetitions of the above steps. We can summarize the genetic algorithm for solving the Stackelberg-Nash equilibrium of general bilevel programming models as follows.

Step 0. Input parameters pop_size, P_c and P_m.
Step 1. Initialize pop_size chromosomes (control vectors of leader).

Step 2. Update the chromosomes by crossover and mutation operations.
Step 3. Determine the Nash equilibrium for each chromosome (control vector of leader) by iterative methods or genetic algorithms.
Step 4. Calculate the objective values of the leader for all chromosomes according to the Nash equilibrium reactions of followers.
Step 5. Compute the fitness of each chromosome by rank-based evaluation function based on the objective values.
Step 6. Select the chromosomes by spinning the roulette wheel.
Step 7. Repeat the 2nd to 6th steps a given number of cycles.
Step 8. Report the best chromosome as the optimal solution.

5. NUMERICAL EXAMPLES

The computer code for the genetic algorithm to general multilevel programming models with multiple followers has been written in C language. In order to illustrate the effectiveness of genetic algorithm, here we give some numerical examples performed on a personal computer with the following parameters: the population size is 30, the probability of crossover P_c is 0.3, the probability of mutation P_m is 0.2, the parameter a in the rank-based evaluation function is 0.05. We also mention that the genetic algorithm is very robust in setting of these parameters.

EXAMPLE 1. Assume that in a bilevel programming model there exist one leader and one follower. Suppose that the leader has a control vector $\mathbf{x} = (x_1, x_2)$ and the follower has a control vector $\mathbf{y} = (y_1, y_2)$. The bilevel programming is formulated as follows,

$$\begin{aligned}
&\max F(\mathbf{x},\mathbf{y}) = \frac{(x_1+y_1)(x_2+y_2)}{1+x_1y_1+x_2y_2}\\
&\text{subject to:}\\
&\qquad x_1 \ge 0,\ x_2 \ge 0,\ x_1^2 + x_2^2 \le 100,\\
&\qquad\qquad \max f(\mathbf{x},\mathbf{y}) = -F(\mathbf{x},\mathbf{y})\\
&\qquad\qquad \text{subject to:}\\
&\qquad\qquad 0 \le y_1 \le x_1,\ 0 \le y_2 \le x_2,
\end{aligned}$$

which is equivalent to the following minimax problem,

$$\begin{aligned}
&\max_{x_1,x_2}\min_{y_1,y_2} \frac{(x_1+y_1)(x_2+y_2)}{1+x_1y_1+x_2y_2}\\
&\text{subject to:}\\
&\qquad x_1 \ge 0,\ x_2 \ge 0,\ x_1^2 + x_2^2 \le 100\\
&\qquad 0 \le y_1 \le x_1,\ 0 \le y_2 \le x_2.
\end{aligned}$$

This is one of the simplest examples of bilevel programming. A run of genetic algorithm with 300 generations shows that the Stackelberg-Nash equilibrium (here leader-follower strategy) is

$$\mathbf{x}^* = (7.0854,\ 7.0291), \qquad \mathbf{y}^* = (7.0854,\ 0.0000)$$

with $F(\mathbf{x}^*, \mathbf{y}^*) = 1.9760$. In view the complexity of the objective function of follower, we have to employ genetic algorithm to search for the reaction of the follower. This makes the evolution process very slow. For this example, the total CPU time is about 28 minutes. This result also implies that the proposed genetic algorithm is applicable to solve classical minimax problems.

EXAMPLE 2. Now we consider a decentralized decision system in which there is one leader and two followers [27,28]. Assume that the control vector of the leader is $\mathbf{x} = (x_1, x_2, x_3, x_4)$, and the control vectors of two followers are $\mathbf{y}_1 = (y_{11}, y_{12})$ and $\mathbf{y}_2 = (y_{21}, y_{22})$. The bilevel programming is formulated as follows,

$$\begin{aligned}
&\max F(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2) = (200 - y_{11} - y_{21})(y_{11} + y_{21}) + (160 - y_{12} - y_{22})(y_{12} + y_{22})\\
&\text{subject to:}\\
&\quad x_1 + x_2 + x_3 + x_4 \le 40\\
&\quad 0 \le x_1 \le 10,\ 0 \le x_2 \le 5,\ 0 \le x_3 \le 15,\ 0 \le x_4 \le 20\\
&\qquad \min f_1(\mathbf{y}_1) = (y_{11} - 4)^2 + (y_{12} - 13)^2\\
&\qquad \text{subject to:}\\
&\qquad\quad 0.4y_{11} + 0.7y_{12} \le x_1\\
&\qquad\quad 0.6y_{11} + 0.3y_{12} \le x_2\\
&\qquad\quad 0 \le y_{11}, y_{12} \le 20,\\
&\qquad \min f_2(\mathbf{y}_2) = (y_{21} - 35)^2 + (y_{22} - 2)^2\\
&\qquad \text{subject to:}\\
&\qquad\quad 0.4y_{21} + 0.7y_{22} \le x_3\\
&\qquad\quad 0.6y_{21} + 0.3y_{22} \le x_4\\
&\qquad\quad 0 \le y_{21}, y_{22} \le 40.
\end{aligned}$$

This is a simple example because there is no information exchange between the two followers. Thus any optimal solutions $\mathbf{y}_1^*$ and $\mathbf{y}_2^*$ to the two subproblems can form a Nash equilibrium $(\mathbf{y}_1^*, \mathbf{y}_2^*)$. The known optimal solution given by Bard [28] is $\mathbf{x}^* = (7.91, 4.37, 11.09, 16.63)$, $\mathbf{y}_1^* = (2.29, 10)$, $\mathbf{y}_2^* = (27.21, 0.00)$ with objective value $F(\mathbf{x}^*, \mathbf{y}_1^*, \mathbf{y}_2^*) = 6600$. For each control vector $\mathbf{x}$ revealed by the leader, there is a unique Nash equilibrium $(\mathbf{y}_1^*, \mathbf{y}_2^*)$ as the reactions of followers. However, the optimal control vector $\mathbf{x}^*$ of the leader is not unique. A run of genetic algorithm with 600 generations shows that a Stackelberg-Nash equilibrium $(\mathbf{x}^*, \mathbf{y}_1^*, \mathbf{y}_2^*)$ is

$$\mathbf{x}^* = (7.05, 3.13, 11.93, 17.89), \qquad \mathbf{y}_1^* = (0.26, 9.92), \qquad \mathbf{y}_2^* = (29.82, 0.00)$$

with objective value $F(\mathbf{x}^*, \mathbf{y}_1^*, \mathbf{y}_2^*) = 6599.99$, while the objective values of the followers are $f_1(\mathbf{y}_1^*) = 23.47$ and $f_2(\mathbf{y}_2^*) = 30.83$. Hence, the genetic algorithm is considered successful for this bilevel programming model. The total CPU time spent for this example is 65 seconds.

EXAMPLE 3. Now we consider a bilevel programming with three followers in which the leader has a control vector $\mathbf{x} = (x_1, x_2, x_3)$ and the three followers have control vectors $\mathbf{y}_i = (y_{i1}, y_{i2})$, $i = 1, 2, 3$,

$$\begin{aligned}
&\max F(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2) = y_{11}y_{12}\sin x_1 + y_{21}y_{22}\sin x_2 + y_{31}y_{32}\sin x_3\\
&\text{subject to:}\\
&\quad x_1 + x_2 + x_3 \le 10,\ x_1 \ge 0,\ x_2 \ge 0,\ x_3 \ge 0\\
&\qquad \max f_1(\mathbf{y}_1) = y_{11}\sin y_{12} + y_{12}\sin y_{11}\\
&\qquad \text{subject to:}\\
&\qquad\quad y_{11} + y_{12} \le x_1,\ y_{11} \ge 0,\ y_{12} \ge 0,\\
&\qquad \max f_2(\mathbf{y}_2) = y_{21}\sin y_{22} + y_{22}\sin y_{21}\\
&\qquad \text{subject to:}\\
&\qquad\quad y_{21} + y_{22} \le x_2,\ y_{21} \ge 0,\ y_{22} \ge 0,
\end{aligned}$$

$$\max f_3(\mathbf{y}_3) = y_{31}\sin y_{32} + y_{32}\sin y_{31}$$
subject to:
$$y_{31} + y_{32} \le x_3,\ y_{31} \ge 0,\ y_{32} \ge 0.$$

For this bilevel programming model, the traditional methods cannot work because of the complexity and multimodality of objective functions of each player. For each optimization problem of leader and followers, we have to employ genetic algorithms to search for the best solutions. A run of genetic algorithm with 300 generations shows that the Stackelberg-Nash equilibrium is

$$\mathbf{x}^* = (1.946, 8.054, 0.000), \quad \mathbf{y}_1^* = (0.973, 0.973), \quad \mathbf{y}_2^* = (1.315, 6.793), \quad \mathbf{y}_3^* = (0.000, 0.000)$$

with optimal objective values

$$F(\mathbf{x}^*, \mathbf{y}_1^*, \mathbf{y}_2^*, \mathbf{y}_3^*) = 9.566, \quad f_1(\mathbf{y}_1^*) = 1.609, \quad f_2(\mathbf{y}_2^*) = 7.099, \quad f_3(\mathbf{y}_3^*) = 0.000.$$

The total CPU time spent for this example is about 36 minutes.

We notice that the optimal solution of the follower is not unique. For example, an optimal solution of the second follower is $\mathbf{y}_2^* = (1.315, 6.739)$, by the symmetry of the control variables y_{21} and y_{22}, the vector $\mathbf{y}_2^{*\prime} = (6.739, 1.315)$ is also an optimal solution. Fortunately, the leader is indifferent between the two solutions. From the symmetry of the followers, we know that the Stackelberg-Nash equilibrium is not unique too. For example, we have also the following Stackelberg-Nash equilibrium

$$\mathbf{x}^* = (8.054, 1.946, 0.000), \quad \mathbf{y}_1^* = (1.315, 6.793), \quad \mathbf{y}_2^* = (0.973, 0.973), \quad \mathbf{y}_3^* = (0.000, 0.000).$$

If the objective function of the leader is replaced by

$$F(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3) = y_{11}y_{12}\sin x_1 + 2y_{21}y_{22}\sin x_2 + 3y_{31}y_{32}\sin x_3.$$

A run of genetic algorithm with 300 generations shows that the Stackelberg-Nash equilibrium is

$$\mathbf{x}^* = (0.000, 1.936, 8.064), \quad \mathbf{y}_1^* = (0.000, 0.000), \quad \mathbf{y}_2^* = (0.968, 0.968), \quad \mathbf{y}_3^* = (1.317, 6.747)$$

with optimal objective values

$$F(\mathbf{x}^*, \mathbf{y}_1^*, \mathbf{y}_2^*, \mathbf{y}_3^*) = 27.822, \quad f_1(\mathbf{y}_1^*) = 0.000, \quad f_2(\mathbf{y}_2^*) = 1.595, \quad f_3(\mathbf{y}_3^*) = 7.120.$$

Although the Nash equilibrium is not unique, the optimal control vector $\mathbf{x}^*$ of the leader is unique.

EXAMPLE 4. Let us now consider a bilevel programming with three followers in which there is information exchange among the three followers. Assume that the leader has a control vector $\mathbf{x} = (x_1, x_2)$ and the three followers have control vectors $\mathbf{y}_i = (y_{i1}, y_{i2})$, $i = 1, 2, 3$, respectively. The bilevel programming is formulated as follows,

$$\min F(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3) = \frac{3(y_{11}+y_{12})^2 + 5(y_{21}+y_{22})^2 + 10(y_{31}+y_{32})^2}{2x_1^2 + x_2^2 + 3x_1x_2}$$
subject to:
$$x_1 + 2x_2 \le 10,\ x_1 > 0,\ x_2 > 0$$

$$\min f_1(\mathbf{y}_1) = y_{11}^2 + y_{12}^2$$
subject to:
$$y_{11} + y_{21} + y_{31} \ge x_1$$

$$y_{12} + y_{22} + y_{32} \geq x_2$$
$$y_{11} \geq 1,\ y_{12} \geq 2,$$

$$\min f_2(\mathbf{y}_2) = y_{21} + y_{22} + \frac{y_{11}}{y_{21}} + \frac{y_{12}}{y_{22}}$$
subject to:
$$y_{21} > 0,\ y_{22} > 0,$$

$$\min f_3(\mathbf{y}_3) = \frac{(y_{31} - y_{21})^2}{y_{31}} + \frac{(y_{32} - y_{22})^2}{y_{32}}$$
subject to:
$$2y_{31} + 3y_{32} = 5$$
$$y_{31} > 0,\ y_{32} > 0.$$

For the bilevel programming model, each subproblem of followers is a parametric optimization which can be solved by any existing means of mathematical programming when the parameters are given. However, in order to obtain the Nash equilibrium of followers, we might employ the iterative method. Fortunately, for this problem, all iterative processes converge to Nash equilibrium.

A run of genetic algorithm with 300 generations shows that the Stackelberg-Nash equilibrium of the bilevel programming is

$$\mathbf{x}^* = (5.768,\ 2.116), \qquad \mathbf{y}_1^* = (2.885,\ 2.000), \qquad \mathbf{y}_2^* = (1.699,\ 1.414), \qquad \mathbf{y}_3^* = (1.183, 0.878)$$

with optimal value

$$F(\mathbf{x}^*, \mathbf{y}_1^*, \mathbf{y}_2^*, \mathbf{y}_3^*) = 1.510, \qquad f_1(\mathbf{y}_1^*) = 12.323, \qquad f_2(\mathbf{y}_2^*) = 6.225, \qquad f_3(\mathbf{y}_3^*) = 0.835.$$

The total CPU time spent for the evolution process is about nine minutes.

6. CONCLUSION

In this paper, we showed that the iterative method and genetic algorithm can be employed to solve the Nash equilibrium of followers in a bilevel programming. We also designed a genetic algorithm for searching for the Stackelberg-Nash equilibrium of general bilevel programming models without linearity, convexity, continuity and differentiability assumptions. As a byproduct we obtain a genetic algorithm for solving classical minimax problems. Finally, we presented some numerical examples to illustrate the effectiveness of genetic algorithm. Although genetic algorithm is a slow and costly way to find optimal solutions and can only deal with small-scale problems, it indeed provides a powerful means of obtaining global optimal solutions of complicated bilevel programming that cannot be solved by existing method else.

Although the proposed genetic algorithm has been devoted to the case of two levels, they are clearly applicable to solve the Stackelberg-Nash equilibrium of general multilevel programming.

REFERENCES

1. U.P. Wen and S.T. Hsu, Linear bilevel programming problems—A review, *Journal of Operational Research Society* **42** (2), 125–133, (1991).
2. O. Ben-Ayed, Bilevel linear programming, *Computers and Operations Research* **20** (5), 485–501, (1993).
3. R.G. Cassidy, M.J.L. Kirby and W.M. Raike, Efficient distribution of resources through three-levels of government, *Management Science* **17**, B462–B472, (1971).
4. J. Bracken and J.T. McGill, Defense applications of mathematical programs with optimization problems in the constraints, *Operations Research* **22**, 1086–1096, (1974).
5. G. Anandalingam and V. Apprey, Multi-level programming and conflict resolution, *European Journal of Operational Research* **51**, 233–247, (1991).

6. J. Fortuny-Amat and B. McCarl, A representation and economic interpretation of a two-level programming problem, *Journal of Operational Research Society* **32**, 783–792, (1981).
7. B. Liu and A.O. Esogbue, Cluster validity for fuzzy criterion clustering, In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Piscataway, NJ, U.S.A., Volume 5, IEEE, pp. 4702–4705, (1995).
8. W. Candler and R. Townsley, A linear two-level programming problem, *Computers and Operations Research* **9**, 59–76, (1982).
9. W.F. Bialas and M.H. Karwan, Two-level linear programming, *Management Science* **30**, 1004–1020, (1984).
10. J.F. Bard, An algorithm for solving the general bilevel programming problem, *Mathematics of Operations Research* **8**, 260–272, (1983).
11. J.F. Bard, Optimality conditions for the bilevel programming problem, *Naval Research Logistics Quarterly* **31**, 13–26, (1984).
12. J.F. Bard and J.T. Moore, A branch and bound algorithm for the bilevel programming problem, *SIAM J. Sci. Statist. Comput.* **11**, 281–292, (1990).
13. G. Savard and J. Gauvin, The steepest descent direction for nonlinear bilevel programming problem, *Operations Research Letters* **15**, 265–272, (1994).
14. O. Ben-Ayed and C.E. Blair, Computational difficulties of bilevel linear programming, *Operations Research* **38**, 556–560, (1990).
15. J.B. Cruz, Leader-follower strategies for multi-level systems, *IEEE Transactions on Automatic Control* **23** (2), 244–255, (1978).
16. H.D. Sherali, A.L. Soyster and F.H. Hurphy, Stackelberg-Nash-Cournot equilibria: Characterizations and computations, *Operations Research* **31** (2), 253–276, (1983).
17. J.F. Bard and J.E. Falk, An explicit solution to the multi-level programming problem, *Computers and Operations Research* **9** (1), 77–100, (1982).
18. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, (1989).
19. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd edition, Springer-Verlag, New York, (1994).
20. D.B. Fogel, An introduction to simulated evolutionary optimization, *IEEE Transactions on Neural Networks* **5**, 3–14, (1994).
21. M. Gen, B. Liu and K. Ida, Evolution program for deterministic and stochastic optimizations, *European Journal of Operational Research* **94** (3), 618–625, (1996).
22. K. Iwamura and B. Liu, A genetic algorithm for chance constrained programming, *Journal of Information & Optimization Sciences* **17** (2), 409–422, (1996).
23. B. Liu, Dependent-chance goal programming and its genetic algorithm based approach, *Mathl. and Comput. Modelling* **24** (7), 43–52, (1996).
24. B. Liu and K. Iwamura, Modelling stochastic decision systems using dependent-chance programming, *European Journal of Operational Research* **101** (1), 193–203, (1997).
25. B. Liu, Dependent-chance programming: A class of stochastic optimization, *Computers Math. Applic.* **34** (12), 89–104, (1997).
26. B. Liu and K. Iwamura, Chance constrained programming with fuzzy parameters, *Fuzzy Sets and Systems* **94** (2), 227–237, (1998).
27. E. Aiyoshi and K. Shimizu, Hierarchical decentralized systems and its new solution by a barrier method, *IEEE Transactions on Systems, Man, and Cybernetics* **11**, 444–449, (1981).
28. J.F. Bard, Convex two-level optimization, *Mathematical Programming* **40**, 15–27, (1988).