Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Genetic algorithm based on simplex method for solving linear-quadratic bilevel programming problem[☆]

Guangmin Wang[a,*], Zhongping Wan[b], Xianjia Wang[c], Yibing Lv[d]

[a] School of Economics and Management, China University of Geosciences, Wuhan 430074, PR China
[b] School of Mathematics and Statistics, Wuhan University, Wuhan 430072, PR China
[c] Institute of System Engineering, Wuhan University, Wuhan 430072, PR China
[d] School of Information and Mathematics, Yangtze University, Jingzhou, 434102, PR China

ARTICLE INFO

Article history:
Received 24 April 2008
Accepted 27 May 2008

Keywords:
The linear-quadratic bilevel programming problem
Genetic algorithm
Simplex method
Fitness value

ABSTRACT

The bilevel programming problems are useful tools for solving the hierarchy decision problems. In this paper, a genetic algorithm based on the simplex method is constructed to solve the linear-quadratic bilevel programming problem (LQBP). By use of Kuhn–Tucker conditions of the lower level programming, the LQBP is transformed into a single level programming which can be simplified to a linear programming by the chromosome according to the rule. Thus, in our proposed genetic algorithm, only the linear programming is solved by the simplex method to obtain the feasibility and fitness value of the chromosome. Finally, the feasibility of the proposed approach is demonstrated by the example.

1. Introduction

The bilevel programming problems are nested optimization problems with two levels in a hierarchy, the upper level and lower level decision-makers who have their own objective functions and constraints. The decision maker at the lower level (the follower) has to optimize its own objective function under the given parameters from the decision maker at the upper level (the leader), who, in return, selects the parameters so as to optimize its own objective function, with complete information on the possible reactions of the follower.

The bilevel programming problem(BLP) is defined as [1]:

$$\begin{aligned} &\min_x F(x,y) && (1)\\ &\text{subject to}\quad G(x,y)\le 0\\ &\qquad\qquad\quad \min_y f(x,y)\\ &\qquad\qquad\quad \text{subject to } g(x,y)\le 0 \end{aligned}$$

where $F, f : R^{n_1}\times R^{n_2} \longrightarrow R$ are called the objective functions of the leader and the follower, respectively. $G : R^{n_1}\times R^{n_2} \longrightarrow R^p$ and $g : R^{n_1}\times R^{n_2} \longrightarrow R^q$ are the constraints of the leader and the follower, respectively. $x\in R^{n_1}$, $y\in R^{n_2}$ are the decision variables of the leader and the follower, respectively.

☆ Supported partly by the National Natural Science Foundation of China(Nos. 70771080, 60574071) and the Research Foundation for Outstanding Young Teachers, China University of Geosciences (Wuhan) (Nos. CUGQNW0801).

* Corresponding author.
E-mail address: wgm97@163.com (G. Wang).

Although the designation bilevel and multilevel programming was firstly used by Candler and Norton [2], Bracken and McGill [3] gave the original formulation for bilevel programming in 1973. However, till 1980s, these problems started receiving the attention motivated by the game theory [4]. And many authors studied bilevel programming intensively and contributed themselves into those fields. Some surveyed the bilevel programming by presenting both theoretical results as well as solution approaches and a large number of applications [5–11], while some researched those problems in monographes [1,12–14]. Various approaches developed for the bilevel programming problems can be classified into the following categories [14]: extreme point algorithms mainly for the linear bilevel programming, branch-and-bound approach, complementary pivot approach, descent approach, penalty function approach and intelligent computation.

However, the bilevel programming is neither continuous anywhere nor convex even if the objective functions of the upper level and lower level and the constraints are all linear because the objective function of the upper level, generally speaking, is neither linear nor differentiable, because it is decided by the solution function of the lower level problem. Bard proved that the bilevel linear programming is a NP-Hard problem [15] and even it is a NP-Hard problem to search for the locally optimal solution of the bilevel linear programming [16]. So, it is greatly difficult to solve the bilevel programming for its non-convexity and non-continuity, especially the non-linear bilevel programming problem.

This paper considers the linear-quadratic bilevel programming problem(LQBP) where the lower level objective function is a convex quadratic and all remaining functions are linear. The remaining of the paper is organized as follows: The model and definitions of linear-quadratic bilevel programming problem is presenter in Section 2; Section 3 describes the genetic algorithm approach for solving LQBP; Some examples are illustrated to demonstrate the feasibility and efficiency in Section 4; Finally, the paper is concluded in Section 5.

2. The concepts and properties of the LQBP

When the objective function of the lower level is convex quadratic and all remaining functions are linear, the linear-quadratic bilevel programming problem can be formulated as follows:

$$\begin{aligned}
(LQBP)\quad & \min_x F(x,y) = a^{\mathrm{T}}x + b^{\mathrm{T}}y \\
& \text{where } y \text{ solves} \\
& \min_y f(x,y) = c^{\mathrm{T}}x + d^{\mathrm{T}}y + (x^{\mathrm{T}}, y^{\mathrm{T}})Q(x^{\mathrm{T}}, y^{\mathrm{T}})^{\mathrm{T}} \\
& s.t.\ Ax + By \le r \\
& \qquad x, y \ge 0
\end{aligned} \tag{2}$$

where $F(x,y), f(x,y)$ are the objective functions of the leader and the follower, respectively. $a, c \in R^{n_1}$, $b, d \in R^{n_2}$, $A \in R^{m\times n_1}, B \in R^{m\times n_2}, r \in R^m.Q \in R^{(n_1+n_2)\times(n_1+n_2)}$ is a symmetric positive semi-definite matrix. $x \in R^{n_1}, y \in R^{n_2}$ are the decision variables under the control of the leader and the follower, respectively.

Definition 2.1. The constraint region of LQBP:

$$S = \{(x,y) | Ax + By \le r, x, y \ge 0\}.$$

Definition 2.2. The projection of S onto the leader's decision space:

$$S(X) = \{x \ge 0 | \text{there exists a } y \text{ , such that } (x,y) \in S\}.$$

In order to ensure that the problem (2) is well posed we make assumption that S is non-empty and bounded.

In the problem (2), let

$$Q = \begin{bmatrix} Q_2 & Q_1^{\mathrm{T}} \\ Q_1 & Q_0 \end{bmatrix}$$

where $Q_0 \in R^{n_2\times n_2}, Q_1 \in R^{n_2\times n_1}, Q_2 \in R^{n_1\times n_1}$. Then $f(x,y)$ is transformed into

$$f(x,y) = c^{\mathrm{T}}x + x^{\mathrm{T}}Q_2x + (d + 2Q_1x)^{\mathrm{T}}y + y^{\mathrm{T}}Q_0y.$$

Note that $c^{\mathrm{T}}x + x^{\mathrm{T}}Q_2x$ is constant for each fixed $x \in S(X)$, we can assume $c = 0, Q_2 = 0$ to ignore those terms without loss of generality when solving the lower level programming. Thus the optimal solution to the follower problem can be obtained by solving the following problem:

$$\begin{aligned}
& \min_y f(x,y) = (d + 2Q_1x)^{\mathrm{T}}y + y^{\mathrm{T}}Q_0y \\
& s.t.\ By \le r - Ax \\
& \qquad y \ge 0.
\end{aligned} \tag{3}$$

Because Q_0 is a symmetric positive semi-definite matrix from the supposition that Q is a symmetric positive semi-definite matrix, there exists a unique and global solution, denoted by $y(x)$, to the problem (3) for each fixed $x \in S(X)$ [17].

Definition 2.3. The inducible region of LQBP:

$$IR = \{(x, y) | (x, y) \in S,\ y = y(x)\}.$$

Definition 2.4. A point (x, y) is called to be feasible if $(x, y) \in IR$.

Definition 2.5. A point (x^*, y^*) is called to be optimal if $F(x^*, y^*) \le F(x, y)$ for any $(x, y) \in IR$.

3. The development of the problem

Here, by applying Kuhn–Tucker conditions for the lower level problem, we have the following theorem:

Theorem 3.1. *Let* $(\bar{x}, \bar{y}) \in S$, *then a necessary and sufficient condition that* $(\bar{x}, \bar{y}) \in IR$ *is that there exist a* $w \ge 0$ $(w \in R^m)$, $u \ge 0$ $(u \in R^m)$, $v \ge 0$ $(v \in R^{n_2})$, *such that*

$$\begin{aligned} &A\bar{x} + B\bar{y} + w = r \\ &2Q_1\bar{x} + 2Q_0\bar{y} - B^{\mathrm{T}}u + v = -d \\ &u^{\mathrm{T}}w = 0 \\ &\bar{y}^{\mathrm{T}}v = 0 \\ &\bar{x}, \bar{y}, w, u, v \ge 0. \end{aligned} \tag{4}$$

Obviously, the problem (1) is transformed into a single level problem of the following form by replacing the lower level programming with its Kuhn–Tucker conditions:

$$\begin{aligned} \max_{x,y,w,u,v}\ & F(x, y) = a^{\mathrm{T}}x + b^{\mathrm{T}}y \\ s.t.\ & Ax + By + w = r \\ & 2Q_1x + 2Q_0y - B^{\mathrm{T}}u + v = -d \\ & u^{\mathrm{T}}w = 0 \\ & v^{\mathrm{T}}y = 0 \\ & x, y, w, u, v \ge 0 \end{aligned} \tag{5}$$

where $w \in R^m$ is a slack variable and $u \in R^m$, $v \in R^{n_1}$ are Kuhn–Tucker multipliers associated with constraints of the lower level programming. So we have the following theorem [18]:

Theorem 3.2. *A necessary and sufficient condition that* (x^*, y^*) *solves the problem* (1) *is that there exist* $w^* \in R^m$, $u^* \in R^m$, $v^* \in R^{n_1} \ge 0$, *such that* $(x^*, y^*, w^*, u^*, v^*)$ *is the optimal solution of the problem* (5).

Based on this reformulation, Bard and Moore [19] have proposed a branch and bound algorithm to investigate LQBP. Later, Júdice and Faustino [20] developed a sequential LCP (SLCP) algorithm for solving LQBP. Furthermore, they illustrated computational experience with the SLCP algorithm and the branch and bound algorithm in Ref. [19] for small and medium scale LQBPs to show that the former is consistently more efficient than the later and the gap increases with the dimension of LQBP.

While concerning that the genetic algorithm (GA) is a numerical algorithm compatible for the optimization problem since it has no special requirement for the differentiability of the function and it has been applied to a wide variety of problem domains including engineering, sciences and commerce for its simplicity, minimal problem restrictions, global perspective and implicit parallelism. Mathieu et al. [21] firstly developed a genetic algorithm for solving bi-level programming problem, in which chromosomes are $n_1 + n_2$ strings of base-10 digits that represent feasible solutions, not necessarily extreme points. Hejazi et al. [22] proposed a method based on genetic algorithm approach for solving a bilevel linear programming problem by tackling the difficulty that most of the chromosomes may be infeasible in solving constrained optimization problem with genetic algorithm for each chromosome represents a bilevel feasible extreme point. Recently, Calvete et al. [23] developed a new approach for solving linear bilevel problems using genetic algorithms, which combined classical extreme point enumeration techniques with genetic search methods by associating chromosomes with extreme points of the polyhedral constraint region.

Hence, in this paper we also proposed the genetic algorithm to solve LQBP by use of the favorable characteristic of the genetic algorithm, which is search and optimization procedures motivated by natural principles and selection [24]. It is obvious that the complementary slack conditions are the difficulties for solving the problem (5). So, we get rid of the complementary slack conditions to simplify the problem (5) by use of the rule in Ref. [22].

Each chromosome is described by using a string consisting of $m + n_2$ binary components in the proposed algorithm for the problem, in which there are m constraints and the follower has n_2 decision variables under his/her control. The first m components are associated with the vector u and the remaining n_2 components are associated with the vector v. For example, in a linear-quadratic bilevel programming problem if $m = 6$, $n_2 = 5$, then the following string is a typical chromosome for this algorithm:

$$\overbrace{010110}^{m=6}\underbrace{10011}_{n_2=5}.$$

This chromosome, according to the following rule [22], transforms the problem (5) into the problem (6) below. If the value of the *ith* component of the chromosome corresponding to u_i, which is the *ith* component of u, is equal to zero, then the variable u_i is also equal to zero; In addition, its complementary variable w_i, which is the *ith* component of w, is greater than or equal to zero, otherwise u_i is greater than or equal to zero and its complementary variable w_i is equal to zero, where $i = 1, \ldots, m$. On the other hand, if the $(m+j)th$ component of the chromosome corresponding to the variable v_j, which is the *jth* component of v, is zero, then the value of variable v_j is equal to zero and its complementary variable y_j, which is the *jth* component of y, is greater than or equal to zero, otherwise v_j is greater than or equal to zero and its complementary variable y_j is equal to zero, where $j = 1, \ldots, n_2$. This rule is applied with each chromosome for simplification of problem (5). The simplified problem is as follows:

$$\begin{aligned} &\max_{x,y',w',u',v'} F(x,y') = a^{\mathrm{T}}x + b'^{\mathrm{T}}y' \\ &s.t.\ Ax + B'y' + w' = r \\ &\qquad 2Q_1x + 2Q_0{}'y' - B''^{\mathrm{T}}u' + v' = -d \\ &\qquad x, y', w', u', v' \geq 0 \end{aligned} \tag{6}$$

where components of y', w', u' and v' are ones of y, u, v and w which are greater than or equal to zero. Also, the components of b' are those of b associated with y'. The columns of the matrices B' and B'' are the columns and rows of B, which are associated with the variables y' and u', respectively. The columns of the matrix $Q_0{}'$ are the columns of Q_0, which are associated with the variable y'. Then the problem (6) has no complementary slack conditions and is a linear programming problem, so it can be solved by using simplex method and much easier to solve than the problem (5).

To search and evaluate the feasible solution of LQBP we give the following theorem:

Theorem 3.3. *If there exists an optimal solution* $(x^*, y'^*, w'^*, u'^*, v'^*)$ *to the simplified problem* (6)*, then* (x, y, w, u, v) *corresponding to this optimal solution is a feasible solution to the problem* (5)*, and* (x, y) *is the feasible solution to the original problem* (1)*.*

Proof. $(x^*, y'^*, w'^*, u'^*, v'^*)$ is the optimal solution to the problem (6), then

$$\begin{aligned} &Ax^* + B'y'^* + w'^* = r \\ &2Q_1x^* + 2Q_0{}'y'^* - B''^{\mathrm{T}}u'^* + v'^* = -d. \end{aligned} \tag{7}$$

According to the above rule, it is obvious that (x, y, w, u, v) corresponding to the optimal solution $(x^*, y'^*, w'^*, u'^*, v'^*)$ to the problem (6) satisfies the constraints of the problem (5), that is, (x, y, w, u, v) is a feasible solution to the problem (5). By Theorem 3.1, (x, y) is a feasible solution to the original problem (1). □

Definition 3.1. A chromosome is called to be feasible if the problem simplified by the chromosome has an optimal solution.

To avoid unnecessary computation resulted from the repeat solving the problem (6), we keep the feasible chromosomes and infeasible chromosomes, respectively, so the computation time can be saved if the chromosome exists for we need not solve the problem (6).

4. Steps of the proposed algorithm

In this section, the steps of the proposed algorithm are described in details.

Encoding of chromosomes is the first question to ask when starting to solve a problem with GA. The most used ways of encoding are float encoding and binary encoding. And the binary encoding is used in our paper. The proposed algorithm consists of the following steps:

Step 1: Initialing. Set the parameters the population size M, probability of crossover and mutation P_c and P_m and the maximal generation of terminating the algorithm T, then set the counter of generation $t = 0$;

Step 2: Generating the initial population $P(0)$. The initial population $P(0)$ consists of a set of feasible chromosomes. For generating these chromosomes, the following problem is solved:

$$\begin{aligned} &\max_{y\geq 0} f(x,y) = (d + 2Q_1x)^{\mathrm{T}}y + y^{\mathrm{T}}Q_0y \\ &s.t.\ By \leq r - Ax \end{aligned} \tag{8}$$

where $x \in S(X)$ is randomly selected. The optimal solution to the problem (8) changes as x, but the optimality conditions also hold for y. Then, the feasible solutions are converted into chromosomes and the objective function value of the upper level is used for fitness value of each chromosome. After generating sufficient such chromosomes, goto next step;

Step 3: Keeping the current best chromosome. If the the counter of current generation t equals to 0, then keep the best chromosome of the initial population as the current best chromosome; Otherwise, compare the best chromosome of the current population with the current best chromosome, if the former is better than the latter, then keep the former as the current best chromosome; Otherwise, the current best chromosome does not change.

Step 4: Crossover. Crossover (also called recombination), which is the main method to generate new chromosomes, is to exchange some genes of the two chromosomes selected for recombination to produce offsprings. Before crossover, every two chromosomes need be matched. The matching strategy is the random matching, that is, the M chromosomes are randomly arranged to $\llcorner M/2 \lrcorner$ matching group. For every two chromosomes in each matching group, a integer n is randomly generated in the interval $[1, l-1]$ (where $l = m+n_2$ is the length of the chromosome) to set the nth gene as the crossover point. And then according to the crossover probability P_c, the one-point crossover is done for them at the crossover point as follows:

The first n components of the children are the same components as the respective parents (i.e. The first child from the first parent and the second child from the second parent.). The remaining components are selected according to the following rules:

i. The $(n+i)$th component of the first child is replaced by the $(l-i+1)$th component of the second parent (for $i = 1, 2, \dots, l-n$).

ii. The $(n+i)$th component of the second child is replaced by the $(l-i+1)$th component of the first parent (for $i = 1, 2, \dots, l-n$).

For example, by applying the proposed operator for the following parents, and assuming $n = 5$, we obtained the following children.

Parents		Children	
10110	101100	10110	010100
11010	001010	11010	001101.

Note that the proposed operator generates chromosomes with more variety since this operator can generate different children from the same parents. If the new chromosome is not in the feasible or infeasible lists, it is evaluated with the problem (6) for feasibility and fitness value. If the new chromosome generated is unfeasible then it is added in the infeasible list and eliminated and the algorithm continues.

Step 5: Mutation. In genetic algorithm, mutation is a genetic operator used to maintain genetic diversity from one generation of a population of chromosomes to the next. The purpose of mutation in GA is to allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. For each gene of every chromosome, it is chosen to be the mutation point according to mutation probability P_m and be mutated as follows:

the value of the chosen gene is changed to 0, if it was initially 1 and to 1 if it was initially 0. If the new chromosome is not in the feasible or infeasible lists, it is evaluated with problem (6) for feasibility and fitness value. If the new chromosome generated is unfeasible then it is added in the infeasible list and eliminated.

Step 6: Selection. The chromosomes are arranged in ascending order as their fitness values. And the selected probability is set as the the order, then, the next population corresponding to the size of the original population is selected by fitness-proportionate selection (roulette wheel). The advantage of this selection is only to compare the fitness values of every two chromosomes without consideration the sign of the fitness value of every chromosome and the difference between the fitness values of chromosomes [25].

Step 7: Termination. The algorithm terminates at the maximal iteration number. The best generated solution, which can be obtained by the current best chromosome kept in all iterations in the earliest time, is reported as the solution for the LQBP by proposed GA algorithm.

5. Computational experience

To demonstrate the feasibility and efficiency of the proposed algorithm, the following example is solved by use of the algorithm proposed in this paper.

Example 1. $\max x_1 + x_2 + 3y_1 - y_2$

$$\max 5y_1 + 8y_2 + (x_1, x_2, y_1, y_2) \begin{pmatrix} 1 & 3 & 2 & 0 \\ 3 & 1 & 4 & -2 \\ 2 & 4 & -2 & 1 \\ 0 & -2 & 1 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \end{pmatrix}$$

$$\begin{aligned} s.t.\ & x_1 + x_2 + y_1 + y_2 \le 12 \\ & -x_1 + x_2 \le 2 \\ & 3x_1 - 4y_2 \le 5 \\ & y_1 + y_2 \le 4. \end{aligned}$$

Here, we choose the parameters as follows: $M = 10$, $P_c = 0.7$, $P_m = 0.1$ and $T = 30$. We execute the proposed algorithm in 20 independent runs, the best solution is $(x^*, y^*) = (6.3, 1.7, 4.0, 0.0)$ and the upper level's objective function $F(x^*, y^*) = 20$ as well as the lower level's objective function $f(x^*, y^*) = 230.04$ at the best solution (x^*, y^*), which are all near the exact values $(x, y) = (6.31250, 1.68750, 4.00000, 0.00000)$, $F(x, y) = 20$ and $f(x, y) = 229.60975$ in Ref. [26]. It can be seen that our proposed approach is feasible from the result.

6. Conclusion and future work

In this paper, a genetic algorithm is given to solve the linear-quadratic bilevel programming problem. The LQBP is transformed into a single level programming by using the Kuhn–Tucker conditions of the lower level programming. And then it is simplified to a linear programming, which can be solved by simplex method, by the chromosome according to the rule. For avoiding unnecessary computation, we also keep the feasible chromosomes and infeasible chromosomes in the algorithm. In all, our algorithm is a novel attempt to solve the LQBP. And in future, more and larger scale examples should be provided to demonstrate efficiency of our algorithm and further research of the parameters' (such as the population size, probabilities of crossover and mutation) impact on our algorithm will also be the future work.

References

[1] J.F. Bard, Practical Bilevel Optimization: Algorithms and Applications, Kluwer Academic Publishers, Dordrecht, 1998.
[2] W. Candler, R. Norton, Multilevel programming. Technical Report 20, World Bank Development Research Center, Washington, DC, 1977.
[3] J. Bracken, J.M. McGill, Mathematical programs with optimization problems in the constraints, Operations Research 21 (1973) 37–44.
[4] H.V. Stackelberg, The Theory of the Market Economy, Oxford University Press, Oxford, UK, 1952.
[5] O. Ben-Ayed, Bilevel linear programming, Computers and Operations Research 20 (1993) 485–501.
[6] B. Colson, P. Marcotte, G. Savard, An overview of bilevel optimization, Annals Of Operations Research 153 (1) (2007) 235–256.
[7] B. Colson, P. Marcotte, G. Savard, Bilevel programming: A survey, A Quarterly Journal of Operations Research 3 (2005) 87–107.
[8] S. Dempe, Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints, Optimization 52 (2003) 333–359.
[9] L. Vicente, P.H. Calamai, Bilevel and multibilevel programming: A bibliography review, Journal of Global Optimization 5 (1994) 291–306.
[10] Guangmin Wang, Zhongping Wan, Xianjia Wang, Bibliography on bilevel programming, Advances in Mathematics 36 (5) (2007) 513–529 (in Chinese).
[11] U.P. Wen, S.T. Hsu, Linear bi-level programming problems—a review, Journal of Operational Research Society 42 (2) (1991) 125–133.
[12] S. Dempe, Foundations of Bilevel Programming, Kluwer Academic Publishers, 2002.
[13] K. Shimizu, Y. Ishizuka, J.F. Bard, Nondifferentiable and Two-level Mathematical Programming, Kluwer Academic Publisher, 1997.
[14] Xianjia Wang, Shangyou Feng, The Optimality Theory of Bilevel System, The Science Press, 1995.
[15] J.F. Bard, Some properties of the bilevel linear programming, Journal of Optimization Theory and Applications 68 (1991) 371–378.
[16] L. Vicente, G. Savard, J. Judice, Descent approaches for quadratic bilevel programming, Journal of Optimization Theory and Applications 81 (1994) 379–399.
[17] T. Edmunds, J.F. Bard, Algorithms for nonlinear bilevel mathematical programming, IEEE Transactions on Systems, Man and Cybernetics 21 (1991) 83–89.
[18] J.F. Bard, Convex two-level optimization, Mathematical Programming 40 (1988) 15–27.
[19] J.F. Bard, J. Moore, A branch and bound algorithm for the bilevel programming problem, SIAM Journal on Scientific Computing 11 (1990) 281–292.
[20] J. Júdice, A. Faustino, The linear-quadratic bilevel programming problem, INFOR 32 (1994) 87–98.
[21] R. Mathieu, L. Pittard, G. Anandalingam, Genetic algorithm based approach to bi-level linear programming, Operations Research 28 (1994) 1–21.
[22] S.R. Hejazi, A. Memariani, G. Jahanshanloo, M.M. Sepehri, Linear bilevel programming solution by genetic algorithm, Computers and Operations Research 29 (2002) 1913–1925.
[23] H.I. Calvete, C. Galé, P.M. Mateo, A new approach for solving linear bilevel programming problems, European Journal of Operational Research 188 (1) (2008) 14–28.
[24] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, Reading,MA, 1989.
[25] T. Back, The interaction of mutation rate, selection and self-adaptation within a genetic algorithm, in: R. Manner, B. Manderick (Eds.), Parallel Problem Solving from Nature, Elsevier, Amsterdam, 1992, pp. 85–94.
[26] H.X. Yin, R.S. Wang, Q. Wang, S.Y. Wang, Linear-quadratic bilevel programming, Systems Engineering- Theory & Practice (4) (1994) 1–8.