PERGAMON

Computers & Operations Research 29 (2002) 1913–1925

computers & operations research

www.elsevier.com/locate/dsw

Linear bilevel programming solution by genetic algorithm

S.R. Hejazi[a], A. Memariani[a,*], G. Jahanshahloo[b], M.M. Sepehri[a]

[a] *Department of Industrial Engineering, Tarbiat Modarres University, P.O. Box 14155 4838, Tehran, Iran*
[b] *Department of Mathematics, Teacher Training University, Tehran, Iran*

Received 1 September 1999; received in revised form 1 July 2000

Abstract

Bilevel programming, a tool for modeling decentralized decisions, consists of the objective of the leader at its first level and that of the follower at the second level. Bilevel programming has been proved to be NP-hard problem. Numerous algorithms have been developed so far for solving bilevel programming problem. In this paper, an attempt has been made to develop an efficient approach based on genetic algorithm. The efficiency of the algorithm is ascertained by comparing the results with Gendreau et al. (J. Global Optimization 8 (1996) 217–233) method.

Scope and purpose

Multilevel programming techniques are developed to solve decentralized planning problems with multiple decision makers in a hierarchical organization. The bilevel programming (BLP) problem is a special case of multilevel programming problems with a two level structure. This problem is an important case in nonconvex optimization, and a leader–follower game in which play is sequential and cooperation is not permitted.

In this paper, we propose a method based on genetic algorithm approach for solving a BLP problem. For solving constrained optimization problem with genetic algorithm, the difficulty is that most of the chromosomes may be infeasible. In the existing methods in the literature, less attention has been provided for this difficulty. By providing some theorems, this matter has been tackled and hence makes the algorithm more efficient than other techniques.

Keywords: Optimization; Bilevel programming; Genetic algorithm

* Corresponding author. Tel.: +98-21-8007905; fax: +98-21-8006544.
E-mail address: memar@modares.ac.ir (A. Memariani).

0305-0548/02/$ - see front matter
PII: S0305-0548(01)00066-1

1. Introduction

Most of the mathematical programming models deal with a single decision maker and a single objective function and are used for centralized planning systems. The bilevel programming (BLP) on the other hand is developed for decentralized planning systems in which the first level is termed as the leader and the second level pertains to the objective of the follower. In the BLP problem, each decision maker tries to optimize its own objective function without considering the objective of the other party, but the decision of each party affects the objective value of the other party as well as the decision space. The linear BLP problem is an optimization model formulated as follows:

$$\begin{aligned} \text{Max}_x \quad & F(x, y) = c_1 x + d_1 y \\ \text{Max}_y \quad & f(x, y) = c_2 x + d_2 y \\ \text{s.t.} \quad & A_1 x + A_2 y \leqslant b, \\ & x, y \geqslant 0, \end{aligned} \tag{1}$$

where $F(x, y)$ is the objective function of the leader and $f(x, y)$ is the objective function of the follower. Also, x is an $n_1 \times 1$ vector (the variables under the control of first level) and y is an $n_2 \times 1$ vector (the variables under the control of second level). A_1 is an $(m \times n_1)$ matrix and A_2 is an $(m \times n_2)$ matrix.

The leader imposes his decision on and gets feedback from the follower. The existing methods for solving this problem can be divided into the following categories:

(a) Methods based on vertex enumeration.
(b) Methods based on Kuhn–Tucker conditions.
(c) Fuzzy approach.
(d) Methods based on meta heuristics.

(a) *Methods based on vertex enumeration*: In these methods the basic idea is that vertex points of accessible region for BLP are vertex point for feasible space of the problem, and the optimal solution is one of these vertices.

Falk [1] developed a method on this approach based on Max–Min problem which is a special case of BLP. This method searches the vertex points of the first level problem using the branch and bound approach. Bialas and Karwan [2] presented Kth-best method in which the search process starts from the unaccessible region and terminates as it searches the first vertex point in the accessible region as optimal solution. Bialas and Karwan [3], also proposed a method which starts the search from the vertex point in accessible region, but it may stop in local optima.

Grid search algorithm (GSA) was developed by Bard [4] and bicriteria programming (BCP) developed by Unlu [5] utilizes the concept of bicriteria optimization, for solving the BLP problem. Through counter example, Candler [6], Clark and Westerberg [7], Wen and Hsu [8], Ben-Ayed and Blair [9], and Marcotte and Savard [10] showed that pareto optimality in bicriteria and optimality in BLP are two different concepts.

Hansen et al. [11] developed a branch and bound-type algorithm that word not based on the Kuhn–Tacker conditions, but instead tried to determine which of the constraints were binding and which of the follower's variables were equal to zero at the optimal solution.

They showed that, this algorithm was efficient against branch and bound algorithm developed by Bard and Moor [12].

(b) *Methods based on Kuhn–Tucker conditions*: In these methods, the second level problem is replaced by the Kuhn–Tucker optimality conditions, yielding a one-level optimization problem with complementarity constraints. Hence, the problem transfers into a single objective with complementary constraints.

Fortuny-Amat and McCarl [13], for solving this kind of problems, converted the complementary constraints to linear one by adding some binary variables. Bialas and Karwan [2], developed parametric complementary pivot (PCP) algorithm by transferring the problem into a single objective in which the objective function of the first level is augmented into the set of constraints as follows:

$$c_1x + d_1y \geqslant \alpha$$

and thereby the problem becomes linear complementary problem (LCP). If this problem is feasible, by increasing α, a better solution can be obtained. If at any iteration the LCP becomes infeasible, then the last feasible solution is considered to be the solution of the problem according to this method. Judice and Faustino [14] developed a sequential LCP algorithm for solving the BLP problem by using a hybrid enumerative method. They showed that, this algorithm performed quite well for the solution of small and medium scale, but it cannot solve some problems. Bard and Moore [12], presented the branch and bound method for solving this problem. White and Anandalingam [15], Anandalingam and White [16], solved the problem by introducing a penalty function to satisfy the complementary constraints.

(c) *Fuzzy approach*: Shih et al. [17] suggested a fuzzy approach by defining the membership function for the objective functions of two levels and the variables related to top level decision maker, but they ignored the noncooperative principle of the decision makers. Sakava and Nishizaki [18] have defined membership function only for the objective functions of two levels. Also they have extended the variation domain of objective functions.

(d) *Methods based on meta heuristics*: In this category, Mathieu et al. [19] developed a genetic algorithm based bilevel programming algorithm (GABBA). In this method, the leader's decision vector is generated and the follower's reaction is obtained from the solution of a linear programming problem. It is different from the usual genetic algorithms because they only use mutations. Also, in this method the search space is expanded considerably. The reason is that, each feasible chromosome represents an accessible solution, but not necessarily an extreme point.

Sahin and Cirit [20] presented an algorithm based on simulated annealing. The main idea is to fuzzify the problem in order to expand the search area and adopt the search randomly using Markov chains. They solved a few small size problems which seem to be inefficient with respect to time. For example, in the first problem with three variables and five constraints, the algorithm took 44 s to solve. This algorithm was capable of solving any BLP problem i.e., linear, nonlinear or integer.

The other method in this category, is a hybrid tabu-ascent algorithm (HTA) which is developed by Gendreau et al. [21]. The basic idea in this method is to use the concept of penalty function for finding the initial solution as well as for improving the existing solution. Also, it applied a Tabu-ascent algorithm for moving from a point of accessible region to another point of this set.

In the present paper, we propose a method based on the genetic algorithm in which each feasible chromosome represents a vertex point of accessible region and thereby reduce the search space significantly. In the following pages, in Section 2, we first provide basic concepts of GA for solving BLP problem. The GA algorithm for solving BLP problem has been presented in Section 3. Computational experiences have been discussed in Section 4. Finally, we concluded the paper in Section 5.

2. Genetic algorithm for solving the BLP problem

In this section, basic concepts and necessary theorems for the development of the algorithm are discussed. First of all Kuhn–Tucker conditions for the second level problem are derived and then problem (1) is transferred into a single level problem of the form

$$\begin{aligned} \text{Max} \quad & c_1x + d_1y \\ \text{s.t.} \quad & A_1x + A_2y + u = b, \\ & zA_2 - v = d_2, \\ & uz = 0, \quad vy = 0, \\ & x, y, z, u, v \geqslant 0. \end{aligned} \tag{2}$$

We now propose a genetic algorithm to solve problem (2). It is assumed that the problem consists of m constraints and the decision maker of the second level has n_2 variables under its control.

In the proposed algorithm, each chromosome is represented by a string consisting of $m + n_2$ binary components. The first m components are associated with the vector z and the remaining n_2 components are associated with the vector v. For example, in a BLP problem if $m = 5$ and $n_2 = 6$, then the following string is a typical chromosome for this algorithm:

$$\overbrace{01011}^{m=5}\underbrace{010011}_{n_2=6}.$$

This chromosome, according to the following rule, transforms problem (2) into problem (3).

If the value of the ith component corresponding to z_i is equal to zero, then the variable z_i is also equal to zero. In addition, its complementary variable u_i is greater than or equal to zero, otherwise z_i is greater than or equal to zero and its complementary variable u_i is equal to zero. On the other hand, if the jth component corresponding to the variable v_j is zero, then the value of variable v_j is equal to zero and its complementary variable y_j 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.

This rule is applied with each chromosome for the simplification of problem (2). The simplified problem is as follows:

$$\begin{aligned} \text{Max} \quad & c_1x + d_1'y' \\ \text{s.t.} \quad & A_1x + A_2'y' + u' = b, \\ & z'A_2'' - v' = d_2, \\ & x, y', z', u', v' \geqslant 0. \end{aligned} \tag{3}$$

In problem (3), y', u', z' and v' are the variables of y, u, z and v that are greater than or equal to zero. Also, d'_1 is the component of d_1 associated with y'. The columns of the matrices A'_2 and A''_2 are the columns and rows of A_2 which are associated with the variables y' and z', respectively.

For example, if "01010" is a chromosome of the proposed GA then problem (A) is transferred into problem (B) by this chromosome, as follows:

$$
\begin{aligned}
\text{Max}\quad & c_1x + d_{11}y_1 + d_{12}y_2 \\
\text{s.t.}\quad & a_1x + a_{11}y_1 + a_{12}y_2 + u_1 = b_1, \\
& a_2x + a_{21}y_1 + a_{22}y_2 + u_2 = b_2, \\
& a_3x + a_{31}y_1 + a_{32}y_2 + u_3 = b_3, \\
& a_{11}z_1 + a_{21}z_2 + a_{31}z_3 - v_1 = d_{21}, \\
& a_{12}z_1 + a_{22}z_2 + a_{32}z_3 - v_2 = d_{22}, \\
& u_1z_1 = 0, \quad u_2z_2 = 0, \quad u_3z_3 = 0, \\
& y_1v_1 = 0, \quad y_2v_2 = 0, \\
& x, y, u, v, z \geqslant 0.
\end{aligned}
\tag{A}
$$

$$
\begin{aligned}
\text{Max}\quad & c_1x + d_{12}y_2 \\
\text{s.t.}\quad & a_1x + a_{12}y_2 + u_1 = b_1, \\
& a_2x + a_{22}y_2 = b_2, \\
& a_3x + a_{23}y_2 + u_3 = b_3, \\
& a_{21}z_2 - v_1 = d_{21}, \\
& a_{22}z_2 = d_{22}, \\
& x, y_2, u_1, u_3, z_2, v_1 \geqslant 0.
\end{aligned}
\tag{B}
$$

Solution of problem (3), if it exists, is an accessible solution for BLP and the optimal value of the objective function is the fitness value for this chromosome. The difficulty is to solve problem (3) for obtaining accessible and unaccessible chromosomes. To overcome this difficulty, problem (3) can be decomposed into two separate problems as follows:

$$
\begin{aligned}
& z'A''_2 - v' = d_2, \\
& z', v' \geqslant 0.
\end{aligned}
\tag{4}
$$

$$
\begin{aligned}
\text{Max}\quad & c_1x + d'_1y' \\
\text{s.t.}\quad & A_1x + A'_2y' + u' = b, \\
& x, y', u' \geqslant 0.
\end{aligned}
\tag{5}
$$

This decomposition is allowed, because problems (4) and (5) do not have common variables. Problem (4) is solved first. If this problem is infeasible, then the chromosome is unaccessible,

otherwise problem (5) is solved. If this problem is infeasible, then the chromosome is unaccessible, otherwise, the chromosome is accessible and the optimal value of the objective function of (5) is the fitness value of the chromosome.

The following theorem helps us to find a way to search and evaluate the vertex points of accessible region of BLP.

Theorem 1. *If the optimal solution of the simplified problem by each chromosome exists, then this solution is a vertex point of accessible region of the BLP problem.*

Proof. Suppose that the optimal solution of (3) is $(x_0, y_0', u_0', z_0', v_0')$. This solution satisfies the Kuhn–Tucker conditions for the second level problem. Then, this solution is accessible for BLP. The solution (x_0, y_0', u_0') is an optimal solution of problem (5) and hence is a basic feasible solution to this problem. On the other hand, the columns of the matrices A_1, A_2 and coefficient of u_0' associated with the nonzero variables are linearly independent. Also, (x_0, y_0, u_0) is a feasible solution for the set of $S = \{A_1x + A_2y + u = b;\ x, y, u \geqslant 0\}$. The corresponding components of y_0 and y_0' are equal and the remaining components of y_0 are equal to zero. The same argument is true for u_0 and u_0'. The columns of A_2' are the subset of the columns of A_2, therefore the columns of A_1, A_2 and a coefficient of u_0 associated with nonzero components of (x_0, y_0, u_0) are linearly independent. As a result, this solution is a basic feasible solution for S (i.e., it corresponds with a vertex point of S). But, the set of accessible solutions of BLP is a subset of S. Consequently, this solution is the vertex point of accessible region of BLP. □

Another difficulty is with the complexity of the algorithm, which increases the CPU time. This is because very often unaccessible solutions are repeatedly generated. To distinguish this point, at least problem (4) has to be solved. Theorems 2 and 3 are proved for decreasing this complexity. By applying Theorems 2 and 3, there will be no need for solving problems (4) and (5).

Theorem 2. *For an unaccessible chromosome p, if problem* (4) *corresponding to it is infeasible, and the zero components of p corresponding to any chromosome* (*say* p_1) *is zero, then* p_1 *is also unaccessible.*

Proof. Let problem (4) associated with chromosome p be

$$z'A_2'' - v' = d_2,$$
$$z', v' \geqslant 0. \tag{6}$$

For simplicity, we can write (6) as

$$Aw = d_2,$$
$$w \geqslant 0. \tag{7}$$

Let problem (4) associated with chromosome p_1 be

$$A'w = d_2,$$
$$w \geqslant 0. \tag{8}$$

where the nonzero columns of A' are a subset of nonzero columns of A. We know that problem (7) is infeasible, so problem (8) is also infeasible. Therefore, p_1 is unaccessible. □

Theorem 3. *For unaccessible chromosome p, if problem* (5) *corresponding to it is infeasible, and for unity components of p, the corresponding one in any chromosome* (*say* p_1) *is* 1, *then* p_1 *is also unaccessible.*

Proof. Let the feasible space of problem (5) associated with chromosome p be

$$A_1x + A_2'y' + u' = b,$$
$$x, y', u' \geqslant 0. \tag{9}$$

For simplicity, we can write (9) as

$$Aw = b,$$
$$w \geqslant 0. \tag{10}$$

Let the feasible space of problem (5) associated with chromosome p_1 be

$$A'w = b,$$
$$w \geqslant 0, \tag{11}$$

where the nonzero columns of A' are a subset of nonzero columns of A. We know that problem (10) is infeasible, so that problem (11) is also infeasible. Therefore p_1 is unaccessible. □

For applying the above theorems, two groups of chromosomes are preserved. The first group (say G_1) consists of some chromosomes such that their corresponding problem (4) is infeasible and also they have more components of 1. The second group (say G_2) consists of some chromosomes such that their corresponding problem (5) is infeasible and that they have more components of zeros. For each generated chromosome:

(a) If for zero components in one of the G_1 chromosomes corresponding components in the generated chromosome are zero, then this chromosome is infeasible.
(b) If for unity components in one of the G_2 chromosomes corresponding components in the generated chromosome are 1, then this chromosome is infeasible.

Now, we are concerned with the chromosomes, which are repeatedly generated. To avoid unnecessary computation, the accessible chromosomes are preserved in a list and the generated chromosomes are checked with this list. If the generated chromosome already exists in the list, then problems (4) and (5) are not required to be solved.

3. The steps of proposed GA

The algorithm consists of the following steps:

Step 1: *Generating the initial population*. The initial population consists of a set of accessible chromosomes. For generating these chromosomes, the following problem is solved:

$$\begin{aligned} \text{Max} \quad & rx + d_2 y \\ \text{s.t.} \quad & A_1 x + A_2 y \leqslant b, \\ & x, y \geqslant 0, \end{aligned} \tag{12}$$

where r is a random vector. By changing the components of r, the optimal solution also changes, but the optimality conditions hold for $d_2 y$. These solutions are vertex points of the accessible region. These solutions are converted into chromosomes and the value of the objective functions of the first level is used for fitness value of each chromosome. After generating such sufficient chromosomes, go to step 2.

Step 2: *Crossover*. In this step, first, a random number $P_c \in [0, 1]$ is generated. This number is the percentage of the population on which the crossover is performed. Then, two chromosomes are selected randomly from the population as parents. Children are generated using the following procedure:

Random integer c is generated in the interval $[1, l-1]$, where l is the number of components of a chromosome ($l = m + n_2$). The cth first 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 $(c+i)$th component of the first child is replaced by the $(l-i+1)$th component of the second parent (for $i = 1, 2, \ldots, l-c$).
(ii) The $(c+i)$th component of the second child is replaced by the $(l-i+1)$th component of the first parent (for $i = 1, 2, \ldots, l-c$).

For example, by applying the proposed operator for the following parents, and asuming $c = 5$, we obtaind the following children:

Parents	Children
10110 1100	10110 0100
11010 0010	11010 0011

Note that the proposed operator generates chromosomes with more variety in comparison with the standard operators CX and PMX, because this operator can generate different children from similar parents, where as but CX and PMX cannot.

If each new chromosome is not recognized as unaccessible by applying Theorems 2 and 3 and it has not been generated already, as accessible, then it is evaluated with problems (4) and (5) for feasibility and fitness value. The crossover operation continues for P_c percent of the population.

Step 3: *Mutation*. In this step, first, a random number $P_m \in [0, 1]$ is generated. This number is the percentage of population on which the mutation is performed. Then one chromosome is selected randomly from the population. An integer random number u is generated in the interval

$[1, l]$, where l is the length of the chromosome $(m + n_2)$. For generating the new chromosome, the uth component is changed to 0, if it was initially 1 and to 1 if it was initially 0.

If by applying Theorems 2 and 3, the new chromosome is not recognized as unaccessible and it has not been generated already, as accessible, then it is evaluated with problems (4) and (5) for feasibility and fitness value. In case the changed component was initially 0 then problem (4) would be feasible and only problem (5) needs to be considered. The mutation operation is performed for P_m percent of the population.

Step 4: *Selection*. If the number of generated races are sufficient then we proceed to the next step, otherwise we arrange the chromosomes of the present population and the chromosomes that are generated in this iteration in the descending order of fitness value.

A population corresponding to the size of the original population is selected from the top of the list. This is considered as the new population.

Step 5: *Termination*. The best generated solution which has been recorded in all iterations in the earliest time is reported as the solution for BLP problem by GA algorithm.

4. Computational experiences

In order to test the efficiency of the proposed GA method against HTA method and estimate some parameters of GA method, a series of different sizes of problems which were randomly generated was solved. The components of A_1 and A_2 are generated randomly in the interval $[-35, 20]$. The coefficient of the first and second level variables associated with the first level objective function are randomly generated in the intervals $[1, 30]$ and $[-30, -1]$, respectively. Moreover, the coefficient the the second level variables associated to the second level objective function is randomly generated in the interval $[1, 50]$.

All computations in Sections 4.1 and 4.2 were performed on a pentium II 400 MHz using the Turbo C++ *v*.3.1 compiler. LINDO software was used as a subroutine for solving LP problems.

In Section 4.3, computations were performed on a pentium II 400 MHz using the GAMS 2.25 compiler, and MINOS 5 was used for solving LP problems.

4.1. Suitable population size

For obtaining suitable population size, four groups of different size problems were solved. The population sizes were considered as $3, 4, \ldots, 10, 15$ chromosomes. Ten unaccessible chromosomes for each problem were preserved for applying Theorems 2 and 3. Termination time was fixed for 200 s. The average CPU arrival time for the solutions is shown in Table 1.

In Table 1, it is observed that for problems 1–4, the best population sizes are 5, 7, 8 and 8, respectively. The best population size increases with the size of the problem, where as the rate is low.

4.2. Appropriate number of unaccessible chromosomes for applying Theorems 2 and 3

Ten problems solved with different sizes and population sizes of 3 and 5 in order to generate 50 races while retaining different number of unaccessible chromosomes for applying Theorems 2 and 3. Results are presented in Table 2.

Table 1
Relationship between population size and CPU time (s)

Prob. no.	Prob. size m–$n1$–$n2$	Population sizes								
		3	4	5	6	7	8	9	10	15
1	10–8–5	2.26	2.64	1.93	2.60	3.30	4.20	4.06	4.90	3.69
2	10–5–10	4.35	5.20	4.43	4.38	4.25	5.55	5.90	6.48	5.73
3	15–10–10	13.51	28.50	16.96	12.43	14.96	11.82	14.33	18.24	19.60
4	20–5–10	29.14	33.62	32.40	24.76	24.68	20.01	28.53	30.70	43.16

Table 2
Relationship between CPU time and the number of the preserved unaccessible chromosomes

Prob. no.	Prob. size m–$n1$–$n2$	No. of unacce. chro. to be preserved									
		P[a]–N[b]	1	5	10	15	20	25	30	35	40	45
1	6–2–4	3–50	16	7	2	2	2	2	2	2	2	2
2	7–3–5	3–50	18	11	6	5	5	5	5	5	5	5
3	8–4–5	3–50	26	17	6	3	3	3	3	3	3	3
4	8–3–6	3–50	23	16	11	8	9	8	7	7	7	7
5	8–3–7	3–50	21	15	12	10	9	9	9	9	9	9
6	9–4–7	3–50	23	17	12	9	8	8	8	8	8	8
7	10–4–7	5–50	41	31	25	21	19	16	16	16	15	15
8	10–3–8	5–50	43	34	26	19	16	14	11	10	10	10
9	12–3–7	5–50	48	40	35	31	29	26	23	23	21	21
10	10–5–10	5–50	45	37	32	28	26	23	22	22	19	19

[a] P is population size.
[b] N is the number of races to be generated.

In this table, CPU time of the algorithm decreases with an increase in retaining unaccessible chromosomes, upto a certain amount and beyond that level the CPU time remains stationary. This amount is less for small problems and is more for larger problems.

If the memory of the computer is not a constraint then retaining more unaccessible chromosomes will not affect the efficiency of the algorithm.

4.3. Comparing the proposed GA with HTA method

HTA method is presented by Gendreau et al. [21]. The basic idea in this method is to use the concept of penalty function for finding both the initial solution as well as the improvement of the existing solution.

HTA is composed of three main building blocks: the start-up phase desgined to produce a good initial solution, the local ascent phase which is the reverse implement of the startup phase, and the Tabu phase for moving away from the current, locally optimal solution and improving it if possible.

In the initialization phase, the second level problem is replaced by the Kuhn–Tucker optimality conditions, yielding a one-level optimization problem with complementarity constraints. Then, as in Anandalingam and White [16], the complementarity constraints are penalized to obtain the linear constrained program as follows:

$$\begin{aligned} \text{Max}_{x,y,z} \quad & c_1x + d_1y - M[(zA_2 - d_2)y + z(b - A_1x - A_2y)] \\ \text{s.t.} \quad & A_1x + A_2y \leqslant b, \\ & zA_2 \geqslant d_2, \\ & x, y, z \geqslant 0. \end{aligned} \tag{13}$$

Whenever M is sufficiently large, the penalty is similar in the sense that problem (13) and the original BLP (1) admit the same solution sets.

If an optimal dual vector z was known a priori, then (13) would reduce to a standard linear program. In this method, z is estimated in a simple way by solving the dual vector of the lower level problem for a given x. The penalized problem is then solved with respect to the x and y-variables. If the resulting vector y is not in the rational reaction set of x, the penalty parameter M is increased and the procedure is repeated. Otherwise (x, y) is the initial solution for the algorithm.

The Tabu phase is to determine from a point (x^0, y^0) another point (x^+, y^+) in the accessible region such that, once the Tabu phase is successfuly completed, the local ascent phase is used for improving (x^+, y^+) solution. If the Tabu phase fails to discover a solution, then the overal procedure halts.

In this paper, the proposed algorithm is compared with HTA method. The comparison is based on computational efficiency and quality of the solution. The computational efficiency is measured in terms of CPU time in seconds and the quality of the solution is measured by

$$\Delta_{\text{GA}} = \frac{F^* - F_1}{F^*} \quad \text{and} \quad \Delta_{\text{HTA}} = \frac{F^* - F_2}{F^*}$$

where F^* is the upper bound of the leader's function, F_1 and F_2 are the values obtained by GA and HTA algorithm for a leader's function, respectively. Also, Δ_{GA} and Δ_{HTA} are the relational deviations from the optimal solution.

Table 3
Results obtained by the proposed GA and HTA

Prob. no.	Prob. size m–$n1$–$n2$	Proposed GA		HTA	
		E(CPU) (s)	Δ_{GA}	E(CPU) (s)	Δ_{HTA}
1	50–35–40	119.40	0.0	122.25	0.257
2	100–75–75	368.20	0.0	436.40	0.024
3	150–125–75	751.00	0.307	703.0	0.008
4	150–250–150	694.25	0.439	880.25	0.371
5	200–150–100	1171.75	0.987	1750.25	0.537

For comparing the two algorithms, five groups of problems with different sizes are solved by both the methods and results, which are presented in Table 3. For each group, 5 problems have been solved.

It is observed that Δ_{GA} is equal to zero for the first two groups and it is greater than Δ_{HTA} for other groups, but the difference is less. The proposed GA has solved four groups of problems faster than the HTA method on the average, while only HTA solved the third group faster than the proposed GA method from computational efficiency point of view.

5. Conclusion

Meta heuristic methods are presented for reducing computational complexity. These algorithms seem to be efficient to solve BLP, which is an NP-hard problem.

The main purpose of this paper is to present a method based on genetic algorithm approach. To this end, Kuhn–Tucker conditions for the second level problem are derived and then the BLP problem is transferred into a single level problem with complementary constraints. Then, the genetic algorithm is employed to solve the transferred problem. For simplification of the computations, some theorems have been proved and an operator for crossover has been suggested.

Empirical results showed that the proposed GA against HTA is efficient from computational point of view and the quality of solutions are almost the same for both the methods.

Acknowledgements

The authors would like to thank the anonymous referees whose comments and suggestions improved the algorithm and the quality of presentation.

References

[1] Falk JE. A linear max–min problem. Mathematical Programming 1973;5:169–88.
[2] Bialas WF, Karwan MH. On two-level optimization. IEEE Transactions on Automatic Control 1982;AC-25(1):211–4.
[3] Bialas WF, Karwan MH. Two-level linear programming. Management Science 1983;30:1004–20.
[4] Bard JF. An efficient point algorithm for a linear two-stage optimization programming. Operations Research 1983;38:556–60.
[5] Unlu G. A linear bilevel programming algorithm based on bicriteria programming. Computers & Operations Research 1987;14:173–9.
[6] Candler W. A linear bilevel programming algorithm: a comment. Computers & Operations Research 1988;15(3):297–8.
[7] Clark PA, Westerberg AW. A note on the optimality conditions for the bilevel programming problem. Naval Research Logistic Quarterly 1988;35:413–8.
[8] Wen UP, Hsu ST. A note on a linear bilevel programming algorithm based on bicriteria programming. Computers & Operations Research 1989;16(1):79–83.
[9] Ben-Ayed O, Blair CE. Computational difficulties of bilevel linear programming. Operations Research 1990;38:556–60.

[10] Marcotte P, Savard G. A note on the Pareto optimality of solutions to the linear bilevel programming problem. Computers & Operations Research 1991;18(4):355–9.
[11] Hansen P, Jaumard B, Savard G. New branch and bound rules for linear bilevel programming. SIAM Journal on Scientific Statistical and Computing 1992;13(5):1194–247.
[12] Bard JF, Moore JT. A branch and bound algorithm for the bilevel programming problem. SIAM Journal on Scientific and Statistical Computing 1990;11(2):281–92.
[13] Fortuny-Amat J, McCarl B. A representation and economic interpretation of a two-level programming problem. Journal of Operational Research Society 1981;32:783–92.
[14] Judice JJ, Faustino AM. A sequential LCP method for bilevel linear programming. Annals of Operations Research 1992;34:89–106.
[15] White DJ, Anandalingam G. A penalty function approach for solving bi-level linear programs. Journal of Global Optimization 1993;3:397–419.
[16] Anandalingam G, White DJ. A solution for the linear static Stackelberg problem using penalty functions. IEEE Transactions on Automatic Control 1990;35:1170–3.
[17] Shih HS, Lai YJ, Lee ES. Fuzzy approach for multi-level programming problems. Computers & Operations Research 1983;23(1):773–91.
[18] Sakava M, Nishizaki I, Uemura Y. Interactive fuzzy programming for multilevel linear programming problem. Computers & Mathematics with Applications 1997;36(2):71–86.
[19] Mathieu R, Pittard L, Anandalingam G. Genetic algorithm based approach to bilevel linear programming. Recherch Ope'rationnelle/Operations Research 1994;28(1):1–21.
[20] Sahin KH, Cirit AR. A dual temperature simulated annealing approach for solving bilevel programming problems. Computers and Chemical Engineering 1998;23:11–25.
[21] Gendreau M, Marcotte P, Savard G. A hybrid Tabu-Ascent algorithm for the linear bilevel programming problem. Journal of Global Optimization 1996;8:217–33.

S.R. Hejazi is a Ph.D. candidate in Industrial Engineering at Tarbiat Modarres University, Tehran. He is working on the solution of Bilevel programming problem with meta heuristics.

A. Memariani is an Associate Professor and Head of Industrial Engineering Department at Tarbiat Modarres University. He obtained his Ph.D. in Mathematics from Banaras Hindu University, India. His research interests include Multiple Criteria Decision Making and Fuzzy Systems.

G. Jahanshahloo is a Professor of Mathematics at Teacher Training University, Tehran. He has supervised many M.S. and Ph.D. theses on Data Envelopment Analysis and Optimization.

M.M. Sepehri is an Assistant Professor of Industrial Engineering at Department of I.E., School of Engineering, Tarbiat Modarres University, Tehran, Iran. His area of interests are scheduling, network flows, routing and meta heuristic. He obtained his M.S. and Ph.D. in management Science from the University of Tennessee, Knoxville, USA.