1998 Second International Conference on Knowledge-Based Intelligent Electronic Systems, 21-23 April 1998, Adelaide, Australia. Editors, L.C. Jain and R.K.Jain

Decentralized two-level 0-1 programming through genetic algorithms with double strings

Keiichi NIWA, †**Ichiro NISHIZAKI** and **Masatoshi SAKAWA**
Department of Industrial and Systems Engineering, Hiroshima University, JAPAN

Abstract **— In this paper, we consider two-level programming problems in which there are one decision maker (the leader) at the upper level and two or more decision makers (the followers) at the lower level and decision variables of the leader and the followers are 0-1 variables. We assume that there is coordination among the followers while between the leader and the group of all the followers, there is no motivation to cooperate each other, and fuzzy goals for objective functions of the leader and followers are introduced in order to take fuzziness of their judgments into consideration. The leader maximizes the degree of satisfaction (the value of the membership function) and the followers choose in concert so as to maximize a minimum among their degrees of satisfaction. A computational method, which is based on the genetic algorithms, for obtaining a solution to the above mentioned problem is developed. To demonstrate the feasibility and efficiency of the proposed algorithm, numerical experiments are carried out.**

KeyWords **— Decentralized two-level 0-1 problems, fuzzy goals, genetic algorithms.**

1. Introduction

In this paper we consider situations where there are two or more decision makers at the same or different level of a hierarchy structure, with independent and sometimes conflicting objectives, and each decision makers can directly control certain variables. Such situations are referred to as multi-level problems. As an example of a two-level problem without cooperation between two decision makers, we cite the Stackelberg duopoly: Firm 1 and Firm 2 supply homogeneous goods to a market. Suppose Firm 1 dominates Firm 2 in the market, and consequently Firm 1 first determines a level of supply and then Firm 2 decides its level of supply after it realizes Firm 1's level of supply. This problem is often formulated as

$$\max_{q_1 \geq 0} (a - b(q_1 + q_2))q_1 - c_1 q_1$$
where q_2 solves
$$\max_{q_2 \geq 0} (a - b(q_1 + q_2))q_2 - c_2 q_2,$$

where q_i denotes Firm i's level of supply, c_i denotes a unit cost of the good of Firm i, and a and b are constants. Objective functions of Firms 1 and 2 are profits and $a - b(q_1 + q_2)$ denotes a price in the market.

Each decision maker (firm) does not have a motivation to cooperate each other in this situation. Studies on such a situation have been seen in the literature on game theory. This problem is modeled as the Stackelberg game, in which there are two decision makers, and one decision maker determines his/her strategy and thereafter the other decision maker decides his/her strategy [12]. Each decision maker completely knows objective functions and constraints of an opponent and himself/herself, and the decision maker at the upper level (leader) first specifies his/her strategy and then the decision maker at the lower level (follower) specifies his/her strategy so as to optimize his/her objectives with full knowledge of decision of the leader. According to the rule, the leader also specifies his/her strategy so as to optimize his/her own objective. Then a solution defined as the above mentioned procedure is called the Stackelberg strategy (solution). The Stackelberg strategy has been employed as a solution concept when decision problems are modeled as two-level programming problems. Even if objective functions of both the leader and the follower are linear and are minimized subject to common linear constraint functions, it is known that this problem is a non-convex programming problem with special structure.

Computational methods for obtaining the Stackelberg solution to two-level linear programming problems are classified roughly into three categories: the vertex enumeration approach taking advantage of a property that an extreme point of a set of best responses of the follower is also an extreme point of a set of the common constraints, the Kuhn-Tucker approach in which the leader's problem with constraints involved optimality conditions of the follower's problem is solved, and the penalty function approach which adds a penalty term to the leader's objective function so as to satisfy optimality of the follower's problem.

Regarding studies on two-level programming problems with discrete variables, Bard and Moore develop algorithms based on the branch-and-bound technique for obtaining the Stackelberg solutions both to two-level 0-1 programming problems [5] and to two-level mixed integer programming prob-

† Corresponding author
1-4-1 Kagamiyama, Higashi-Hiroshima 739 JAPAN
Tel:+81-824-24-7695, Fax:+81-824-22-7195
E-mail: nisizaki@msl.sys.hiroshima-u.ac.jp

0-7803-4316-6/98/$10.00 ©1998 IEEE

lems [4]. Wen and Yang [15] propose algorithms for obtaining the Stackelberg solution or its approximate solution to a two-level programming problem in which decision variables of the leader are 0-1 variables while those of the follower are continuous variables. One of their algorithms is also based on the branch-and-bound technique, and it picks a 0-1 variable of the leader to make a node and creates two branches. Vicente, Savard and Judice [14] establish equivalences between two-level discrete linear programming problems and particular two-level linear programming problems by using penalty function methods.

As for researches on two-level programming problems using genetic algorithms, Anandalingam et al. [2] present a genetic algorithm, which employs a representation of individuals by not 0-1 bit strings but a string of base-10 digits, for obtaining the Stackelberg solution to two-level linear programming problems.

In this paper, we consider situations where there are one decision maker (leader) at the upper level and two or more decision makers (followers) at the lower level, which are referred to as decentralized two-level problems. Especially, as an example of the problems, consider a problem relevant to Stackelberg duopoly. Suppose that there are $n+1$ firms supply homogeneous goods to a market, and a firm dominates the others, remaining n firms in the market, and consequently the dominating firm first determines a level of supply and then the remaining n firms decide their levels of supply simultaneously after they realize the dominating firm's level of supply.

Simaan and Cruz [13] and Anandalingam [1] modeled such problems as follows. The leader first specifies his/her strategy and then the followers specifies their strategies so as to equilibrate their objectives for a given strategy of the leader. In the study of Shimize and Aiyoshi [11], the leader optimize his/her objective function under the conditions that decisions of the followers become Pareto optimal.

We intend to devote this paper to examining situations when there is coordination among the followers while between the leader and the followers, there is not a motivation to cooperate each other, and decision variables of the leader and the followers are 0-1 variables. We assume that fuzzy goals for objective functions of the leader and followers are introduced, the leader maximizes the degree of satisfaction (the value of the membership function) and the followers choose in concert so as to maximize a minimum among their degrees of satisfaction. A computational method, which is based on the genetic algorithms, for obtaining a solution to the above mentioned problem is developed. To demonstrate the feasibility and efficiency of the proposed algorithm, numerical experiments are carried out.

2. Decentralized two-level 0-1 programming problems

Consider the following situation; the leader is a decision maker at the upper level and is denoted by DM0; the followers are p decision makers at the lower level and are denoted by DM1, ..., DMp; DM0 first determines his/her decision $\boldsymbol{x} \in \{0,1\}^{n_0}$ and thereafter DM1, ..., DMp specify their decisions $\boldsymbol{y}_j \in \{0,1\}^{n_j}$, $j = 1,\ldots,p$ simultaneously so as to optimize their objectives with full knowledge of decision $\boldsymbol{x}$ of DM0. According to the rule, it is supposed that DM0 determines his/her decision $\boldsymbol{x}$ so as to optimize his/her own objective.

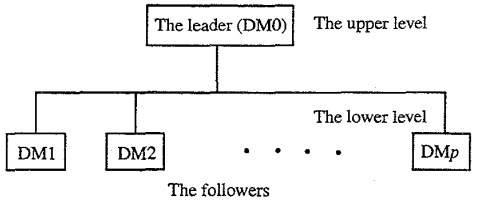

Figure 1. Decentralized two-level structure

A objective function of DM0, which is assumed to be minimized, is represented as

$$z_0 = \boldsymbol{c}_0\boldsymbol{x} + \boldsymbol{d}_{01}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{0p}\boldsymbol{y}_p, \qquad (1)$$

where $\boldsymbol{c}_0$ is n_0-dimensional row constant vector, and $\boldsymbol{d}_{0j}$, $j = 1,\ldots,p$ are n_j-dimensional row constant vectors $j = 1,\ldots,p$, respectively. Objective functions of DMi, $i = 1,\ldots,p$, which are also assumed to be minimized, are represented as

$$\left.\begin{aligned} z_1 &= \boldsymbol{c}_1\boldsymbol{x} + \boldsymbol{d}_{11}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{1p}\boldsymbol{y}_p, \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ z_p &= \boldsymbol{c}_p\boldsymbol{x} + \boldsymbol{d}_{p1}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{pp}\boldsymbol{y}_p, \end{aligned}\right\} \qquad (2)$$

respectively, where $\boldsymbol{c}_i$, $i = 1,\ldots,p$ are n_i-dimensional row constant vectors, and $\boldsymbol{d}_{ij}$, $i = 1,\ldots,p$, $j = 1,\ldots,p$ are n_j-dimensional row constant vectors, respectively.

All of the decision variables $\boldsymbol{x}$ and $\boldsymbol{y}_1,\ldots,\boldsymbol{y}_p$ are constrained the following conditions in common.

$$\left.\begin{aligned} &A\boldsymbol{x} + B_1\boldsymbol{y}_1 + \cdots + B_p\boldsymbol{y}_p \geq \boldsymbol{e}, \\ &\boldsymbol{x} \in \{0,1\}^{n_0}, \\ &\boldsymbol{y}_j \in \{0,1\}^{n_j},\ j = 1,\ldots,p, \end{aligned}\right\} \qquad (3)$$

where A is an $m \times n_0$ coefficient matrix, B_j, $j = 1,\ldots,p$ are $m \times n_j$ coefficient matrices and $\boldsymbol{e}$ is an m-dimensional column constant vector. Let S denote a feasible solution area expressed by (3).

It is natural that decision makers have fuzzy goals for their objective functions when they take fuzziness of human judgments into consideration. For each of the objective functions z_i, $i = 0, 1, \ldots, p$ represented as (1) and (2), we assume that the decision makers have fuzzy goals such as "the objective function z_i should be substantially less than

or equal to some value p_i."

For DMi, $i = 0, 1, \ldots, p$, the individual minimum

$$z_i^{\min} = \min_{\boldsymbol{x},\boldsymbol{y}\in S} \boldsymbol{c}_i\boldsymbol{x} + \boldsymbol{d}_{i1}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{ip}\boldsymbol{y}_p, \quad (4)$$

and the individual maximum

$$z_i^{\max} = \max_{\boldsymbol{x},\boldsymbol{y}\in S} \boldsymbol{c}_i\boldsymbol{x} + \boldsymbol{d}_{i1}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{ip}\boldsymbol{y}_p, \quad (5)$$

of the objective function are referred to when DMi elicits a membership function prescribing the fuzzy goal for his/her objective function z_i, where $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p)$. DM$i$ determines the membership function $\mu_i(z_i)$, which is strictly monotone decreasing for z_i, consulting the variation ratio of degree of satisfaction in the interval between the individual minimum (4) and the individual maximum (5). The domain of the membership function is the interval $[z_i^{\min}, z_i^{\max}]$, $i = 0, 1, \ldots, p$, and DMi specifies the values z_i^0 of the objective functions for which the degree of satisfaction is 0 and the value z_i^1 of the objective function for which the degree of satisfaction is 1. For the value undesired (larger) than z_i^0, it is defined that $\mu_i(z_i) = 0$, and for the value desired (smaller) than z_i^1, it is defined that $\mu_i(z_i) = 1$.

For the sake of simplicity, in this paper, we adopt a linear membership function, which characterizes the fuzzy goal of DMi. The corresponding linear membership function $\mu_i(z_i)$ is defined as:

$$\mu_i(z_i) = \begin{cases} 0, & \text{if } z_i > z_i^0 \\ \dfrac{z_i - z_i^0}{z_i^1 - z_i^0}, & \text{if } z_i^1 < z_i \le z_i^0 \\ 1, & \text{if } z_i \le z_i^1, \end{cases} \quad (6)$$

where z_i^0 and z_i^1 denote the values of the objective function $z_i(\boldsymbol{x})$ such that the degree of membership function is 0 and 1, respectively, and it is assumed that the DMi subjectively assess z_i^0 and z_i^1.

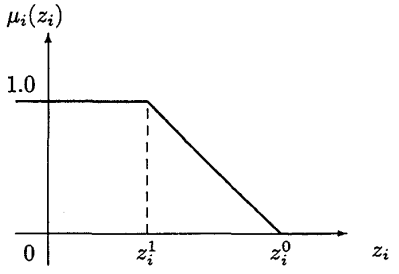

Figure 2. Linear membership function

For the followers, suppose that applying the way suggested by Zimmermann [16], DMi specifies z_i^0 and z_i^1 for $i = 1, \ldots, p$ in the following. That is, using the individual minimum $z_i^{\min}$ together with

$$z_i^{\mathrm{m}} = \max\{\boldsymbol{c}_i\boldsymbol{x} + \boldsymbol{d}_{i1}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{ip}\boldsymbol{y}_p \mid (\boldsymbol{x},\boldsymbol{y}) \in \{(\boldsymbol{x}^{0\mathrm{opt}}, \boldsymbol{y}^{0\mathrm{opt}}), \ldots, (\boldsymbol{x}^{p\mathrm{opt}}, \boldsymbol{y}^{p\mathrm{opt}})\}\}, \quad (7)$$

where $(\boldsymbol{x}^{j\mathrm{opt}}, \boldsymbol{y}^{j\mathrm{opt}})$ denote an optimal solution to the individual minimum (4) for DMj, DMi determines his/her linear membership function as in (6) by choosing $z_i^1 = z_i^{\min}$ and $z_i^0 = z_i^{\mathrm{m}}$. For the leader, suppose that DM0 determines his/her linear membership function by choosing $z_i^1 = z_i^{\min}$ and $z_i^0 = z_i^{\max}$.

We assume that the followers (DM1, ..., DMp) choose in concert so as to maximize a minimum among degrees of their membership functions. Then we can model the situation as the following decentralized two-level 0-1 programming problem:

$$\left.\begin{array}{l} \max\limits_{\boldsymbol{x}} \mu_0(\boldsymbol{c}_0\boldsymbol{x} + \boldsymbol{d}_{01}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{0p}\boldsymbol{y}_p) \\ \text{where } \boldsymbol{y} \text{ solves} \\ \max\limits_{\boldsymbol{y}} \min\limits_{i=1,\ldots,p} \mu_i(\boldsymbol{c}_i\boldsymbol{x} + \boldsymbol{d}_{i1}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{ip}\boldsymbol{y}_p) \\ \text{s. t. } A\boldsymbol{x} + B_1\boldsymbol{y}_1 + \cdots + B_p\boldsymbol{y}_p \ge \boldsymbol{e}, \\ \quad \boldsymbol{x} \in \{0,1\}^{n_0}, \\ \quad \boldsymbol{y}_j \in \{0,1\}^{n_j},\ j = 1, \ldots, p. \end{array}\right\} \quad (8)$$

In this formulation, the followers (DM1, ..., DMp) maximize their aggregated degree of satisfaction $\min_{i=1,\ldots,p} \mu_i(\boldsymbol{c}_i\boldsymbol{x} + \boldsymbol{d}_{i1}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{ip}\boldsymbol{y}_p)$. This aggregated degree of satisfaction is nothing else but the fuzzy decision proposed by Bellman and Zadeh [6], which is often employed as a solution concept in fuzzy environments.

3. Computational method using genetic algorithms with double strings

We assume that all the coefficients (entries of A, B_j, $j = 1, \ldots, q$ and $\boldsymbol{e}$) of the constraints (3) are positive in order to effectively solve the problem through genetic algorithms with double strings proposed by Sakawa et. al. [10].

To solve the problem (8), we intend to phase usage of genetic algorithms (GAs) with double strings. That is, the decision variables $\boldsymbol{x}$ of the leader (DM0) are reproduced according to the GAs and, for given $\boldsymbol{x}$, the decision variables $\boldsymbol{y}$ of the followers (DMi, $i = 1, \ldots, p$) are also reproduced. We propose an algorithm summarized as follows:

An algorithm for obtaining solutions to decentralized two-level 0-1 programming problems

Step 1 Generate N_l initial individuals for the decision variable $\boldsymbol{x}$ of the leader at random.

Step 2 For each individual of $\boldsymbol{x}$, the following procedure is repeated.

Step 2-1 Generate N_f initial individuals for the decision variable $\boldsymbol{y}$ of the followers at random.

Step 2-2 Evaluate each individual for the decision variable $\boldsymbol{y}$ with the given $\boldsymbol{x}$ on the basis

of phenotype ($(n_0+n_1+\cdots+n_p)$-dimensional vector) decoding from genotype (string).

Step 2-3 Go to Step 3 after the procedure of the GA with respect to the followers is repeated M_f times, which is the number of generations specified in advance.

Step 2-3 Apply a reproduction operator, and thereafter apply a crossover operator and a mutation operator to $\boldsymbol{y}$ according to the probabilities of crossover and mutation. Return to Step 2-2.

Step 3 Evaluate each individual for the decision variable $\boldsymbol{x}$ with an optimal individual $\boldsymbol{y}(\boldsymbol{x})$ on the basis of phenotype decoding from genotype.

Step 4 Stop the algorithm after the procedure of the GA with respect to the leader is repeated M_l times, which is the number of generations specified in advance. Regard an individual with the maximal fitness as an optimal individual $(\boldsymbol{x}, \boldsymbol{y})$.

Step 5 Apply a reproduction operator, and thereafter apply a crossover operator and a mutation operator to $\boldsymbol{x}$ according to the probabilities of crossover and mutation. Return to Step 2.

We will describe the above procedure in detail.

3.1. Coding and decoding

For solving 0-1 programming problems through genetic algorithms, an individual is usually represented by a binary 0-1 string [8], [9]. This representation, however, may weaken ability of genetic algorithms since an individual whose phenotype is feasible is scarcely generated under this representation. In this paper, as one possible approach to generate only feasible solutions, a double string as is shown in Figure 3 is adopted for representing an individual, where $s_{i_x(k)} \in \{1,0\}$, $i_x(k) \in \{1,\ldots,n_0\}$, and $i_x(k) \neq i_x(k')$ for $k \neq k'$, and similarly $s_{i_y(k)} \in \{1,0\}$, $i_y(k) \in \{1,\ldots,n_1+\cdots+n_p\}$, and $i_y(k) \neq i_y(k')$ for $k \neq k'$ [10]. Note that $n_1+\cdots+n_p$ entries of the variable vectors $\boldsymbol{y}_j$, $j=1,\ldots,p$ are arranged in the second part of the double string without distinction of vectors $\boldsymbol{y}_j$.

individual for $\boldsymbol{x}$			individual for $\boldsymbol{y}$		
$i_x(1)$	$\cdots$	$i_x(n_0)$	$i_y(1)$	$\cdots$	$i_y(n_1+\cdots+n_p)$
$s_{i_x(1)}$	$\cdots$	$s_{i_x(n_0)}$	$s_{i_y(1)}$	$\cdots$	$s_{i_y(n_1+\cdots+n_p)}$

Figure 3. Double string

In the double string, regarding $i_x(k)$, $i_y(k)$ and $s_{i_x(k)}$, $s_{i_y(k)}$ as the index of an element in a solution vector and the value of the element respectively, a string $\boldsymbol{s}$ can be transformed into a solution $\boldsymbol{x}=(x_1,\ldots,x_{n_0})$ and $\boldsymbol{y}_j=(y_{j1},\ldots,y_{j\,n_j})$, $j=1,\ldots,p$ as:

$$x_{i(k)} = s_{i_x(k)},\ k=1,\ldots,n_0,$$
$$y_{1i(k)} = s_{i_y(k)} \text{ for } 1 \le i_y(k) \le n_1,$$
$$y_{2i(k)} = s_{i_y(k)} \text{ for } n_1+1 \le i_y(k) \le n_2,$$
$$\cdots\cdots\cdots$$
$$y_{pi(k)} = s_{i_y(k)} \text{ for } n_{p-1}+1 \le i_y(k) \le n_p.$$

Unfortunately, however, since this mapping may generate infeasible solutions, we propose the following decoding procedures for eliminating infeasible solutions at the Steps 1 and 3, and Step 2-2 of the proposed algorithm.

For Steps 1 and 3 of the proposed algorithm, it is possible for some individuals $\boldsymbol{x}$ to violate the constraints even if all the entries of $\boldsymbol{y}$ are one. Assuming all the coefficients are nonnegative, we can employ the following decoding procedure which suppresses birth of individuals generating infeasible solutions.

Step 0 Let $\boldsymbol{e}' = \boldsymbol{e} - \sum_{j=1}^{p}\sum_{\ell=1}^{n_j} \boldsymbol{b}^j_{\cdot\ell}$, where $\boldsymbol{b}^j_{\cdot\ell}$ denotes the ℓth column vector of the matrix B_j.

Step 1 Set $k=1$, $\boldsymbol{\Sigma} = \sum_{\ell=1}^{n_0} \boldsymbol{a}_{\cdot\ell}$, where $\boldsymbol{a}_{\cdot\ell}$ denotes the ℓth column vector of the matrix A.

Step 2 If $s_{i_x(k)}=0$, set $k=k+1$ and go to step 3. Otherwise, i.e., if $s_{i_x(k)}=1$, set $k=k+1$ and go to step 4.

Step 3 If $\boldsymbol{\Sigma} - \boldsymbol{a}_{\cdot i_x(k)} \ge \boldsymbol{e}$, set $x_{i_x(k)}=0$, $\boldsymbol{\Sigma} = \boldsymbol{\Sigma} - \boldsymbol{a}_{\cdot i_x(k)}$ and go to step 4. Otherwise, set $x_{i_x(k)}=1$ and go to step 4.

Step 4 If $k > n_0$, stop and regard $\boldsymbol{x}$ as phenotype of the individual represented by the double string. Otherwise, return to Step 2.

For Step 2-2 of the proposed algorithm, we can use the following similar decoding procedure.

Step 0 Let $\boldsymbol{e}' = \boldsymbol{e} - \sum_{\ell=1}^{n_0} \boldsymbol{a}_{\cdot\ell} x_\ell$.

Step 1 Set $k=1$, $\boldsymbol{\Sigma} = \sum_{j=1}^{p}\sum_{\ell=1}^{n_j} \boldsymbol{b}^j_{\cdot\ell}$.

Step 2 If $s_{i(k)}=0$, set $k=k+1$ and go to step 3. Otherwise, i.e., if $s_{i(k)}=1$, set $k=k+1$ and go to step 4.

Step 3 If $\boldsymbol{\Sigma} - \boldsymbol{b}^j_{\cdot i_y(k)} \ge \boldsymbol{e}$ for $n_{j-1}+1 \le i_y(k) \le n_j$, set $y_{j\,i_y(k)}=0$, $\boldsymbol{\Sigma} = \boldsymbol{\Sigma} - \boldsymbol{b}^j_{\cdot i_y(k)}$ and go to step 4. Otherwise, set $y_{i_y(k)}=1$ and go to step 4.

Step 4 If $k > n_1+\cdots+n_p$, stop and regard $\boldsymbol{y}$ as phenotype of the individual represented by the double string. Otherwise, return to Step 2.

Note that a location of a variable in the string does not influence a value specified by the above decoding procedures.

3.2. Evaluation and selection

In the proposed algorithm, individuals for variables $\boldsymbol{y}$ of the followers are evaluated for a given $\boldsymbol{x}$ in Step 2-2, and individuals for variable $\boldsymbol{x}$ of the leader with optimal individuals $\boldsymbol{y}(\boldsymbol{x})$ are also evaluated in Step 3.

A fitness of an individual for variable y of the followers is directly defined, for a given x, as a value of the aggregated fuzzy goal

$$\min_{i=1,\ldots,p} \mu_i(\boldsymbol{c}_i\boldsymbol{x} + \boldsymbol{d}_{i1}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{ip}\boldsymbol{y}_p). \quad (9)$$

For an individual for variable x of the leader, a fitness is also defined as the fuzzy goal of the leader (DM0)

$$\mu_0(\boldsymbol{c}_0\boldsymbol{x} + \boldsymbol{d}_{01}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{0p}\boldsymbol{y}_p). \quad (10)$$

In this paper, we employ the ranking selection proposed by Baker [3]. Ranks of individuals are arranged in decreasing order of the fitness, i.e., an individual with the largest fitness is determined as Rank 1. The probability p_i that the individual s_i of Rank i is reproduced is

$$p_i = \frac{1}{N}\left(\eta^+ - (\eta^+ - \eta^-)\frac{i-1}{N-1}\right), \quad (11)$$

where N is a population size, η^+ is set at 2, and η^- becomes 0 because $\eta^- = 2 - \eta^+$. Moreover the elitist preserving selection is also incorporated, that is, if the fitness of an individual in the past populations is larger than that of every individual in the current population, preserve this individual into the current generation.

3.3. Crossover and mutation

If a single-point or multi-point crossover operator is applied to individuals represented by double strings, an index $i_x(k)$ ($i_y(k)$) in an offspring may take the same number that an index $i_x(k')$, ($i_y(k')$,) $k \neq k'$ takes. Recall that the same violation occurs in solving traveling salesman problems or scheduling problems through genetic algorithms. One possible approach to circumvent such violation, a crossover method called partially matched crossover (PMX) is useful. The PMX was first proposed by Goldberg and Lingle [7] for tackling a blind traveling salesman problem. It enables us to generate desirable offsprings without changing the double string structure unlike the ordinal representation. We employ the following crossover operation which is a revised version of PMX to deal with double strings [10] and is carried out according to the prespecified probability p_c.

Step 1 For two individuals s_1 and s_2 represented by double strings, choose two crossover points.

Step 2 According to the PMX, reorder upper strings of s_1 and s_2 together with the corresponding lower strings which yields s_1' and s_2'.

Step 3 Exchange lower substrings between two crossover points of s_1' and s_2' for obtaining the resulting offsprings s_1'' and s_2'' after the revised PMX for double strings.

It is well recognized that a mutation operator plays a role of local random search in genetic algorithms. In this paper, for the lower string of a double string, mutation of bit-reverse type is adopted. We also introduce another genetic operator, an inversion. The inversion proceeds as follows:

Step 1 For an individual s, choose two inversion points at random, i.e.,

$$s = \begin{pmatrix} i(1)\cdots \mid i(l)\ i(l+1)\cdots i(m) \mid \cdots i(n) \\ s_{i(1)}\cdots \mid s_{i(l)}\ s_{i(l+1)}\ \cdots s_{i(m)} \mid \cdots s_{i(n)} \end{pmatrix}.$$

Step 2 Invert both upper and lower substrings between two inversion points, i.e.,

$$s' = \begin{pmatrix} i(1)\cdots \mid i(m)\ i(m-1)\cdots i(l) \mid \cdots i(n) \\ s_{i(1)}\cdots \mid s_{i(m)}\ s_{i(m-1)}\ \cdots s_{i(l)} \mid \cdots s_{i(n)} \end{pmatrix}.$$

Either of these two genetic operations is carried out according to the prespecified probability p_m and the constant value for splitting the two operations.

4. Numerical experiments

To demonstrate the feasibility and efficiency of the proposed algorithm, consider the following decentralized two-level 0-1 programming problem with 40 variables and 5 constraints, in which there are one leader (denoted by DM0) at the upper level and three followers (denoted by DM1, DM2, and DM3) at the lower level.

$$\begin{aligned} &\max_{\boldsymbol{x}} \mu_0(\boldsymbol{c}_0\boldsymbol{x} + \boldsymbol{d}_{01}\boldsymbol{y}_1 + \boldsymbol{d}_{02}\boldsymbol{y}_2 + \boldsymbol{d}_{03}\boldsymbol{y}_3) \\ &\text{where } \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3 \text{ solves} \\ &\max_{\boldsymbol{y}_1,\boldsymbol{y}_2,\boldsymbol{y}_3} \min_{i=1,2,3} \mu_i(\boldsymbol{c}_i\boldsymbol{x} + \boldsymbol{d}_{i1}\boldsymbol{y}_1 + \boldsymbol{d}_{i2}\boldsymbol{y}_2 + \boldsymbol{d}_{i3}\boldsymbol{y}_3) \\ &\quad \text{s. t. } A\boldsymbol{x} + B_1\boldsymbol{y}_1 + B_2\boldsymbol{y}_2 + B_3\boldsymbol{y}_3 \geq \boldsymbol{e}, \\ &\qquad \boldsymbol{x} \in \{0,1\}^{n_0}, \\ &\qquad \boldsymbol{y}_j \in \{0,1\}^{n_j},\ j = 1,2,3, \end{aligned}$$

where $\boldsymbol{x} = (x_1, \ldots, x_{20})^T$, $\boldsymbol{y}_1 = (y_{11}, \ldots, y_{17})^T$, $\boldsymbol{y}_2 = (y_{21}, \ldots, y_{27})^T$, and $\boldsymbol{y}_3 = (y_{31}, \ldots, y_{36})^T$, and ^{T} denotes transposition; each entry of 5×20 coefficient matrices A, 5×7 coefficient matrices B_1 and B_2, and 5×6 coefficient matrices B_3 is a random value in the interval $[10, 99]$; each entry of the right hand side constant column vector $\boldsymbol{e}$ is a sum of entries of the corresponding row vector of A, B_1, B_2 and B_3 multiplied by a random number in the interval $[0.45, 0.55]$. The coefficients are shown in Table 1. Parameters of fuzzy goals for DM0, DM1, DM2 and DM3 are determined as shown in Section 2.

In the proposed algorithm for obtaining an approximate optimal solution to the problem, we set 0.7 to the probability of the crossover, 0.10 to the probability of the mutation, 0.5 to the constant value for splitting into two types of mutation, 100 to the population sizes for the GAs with respect to both the leader and the followers, 100 to the

numbers of the generations for the GAs with respect to both the leader and the followers, and we made this computational trial ten times. The result of the computational experiments are shown in Table 3. As seen in Table 3, we suppose that the proposed algorithm works well because variances of the degree of satisfaction are sufficiently small.

Table 2. Parameters of fuzzy goals

	Leader	Followers		
	DM0	DM1	DM2	DM3
z_j^1	178	457	396	375
z_j^0	1906	653	840	549

Table 3. Results of computational experiments

	degree of satisfaction			
	best	worst	mean	variance
Leader	0.887	0.892	0.888	0.0000
Followers	0.643	0.439	0.561	0.0031

5. Conclusions

In this paper, we have considered decentralized two-level 0-1 problems, where there is coordination among the followers while between the leader and the group of all the followers, there is no motivation to cooperate each other. To take fuzziness of judgments of the leader and the followers into consideration, fuzzy goals for objective functions of them have been introduced. We have considered the situation when the leader maximizes the degree of satisfaction and the followers choose in concert so as to maximize a minimum among their degrees of satisfaction, and have developed a computational method through the genetic algorithms for obtaining a solution to the decentralized two-level 0-1 problem. Numerical experiments have been performed in order to demonstrate the feasibility and efficiency of the proposed algorithm.

References

1. G. Anandalingam, "A mathematical programming model of decentralized multi-level systems," *Journal of Operational Research Society*, vol. 39, pp. 1021-1033, 1988.
2. G. Anandalingam, R. Mathieu, C.L. Pittard, and N. Sinha, "Artificial intelligence based approaches for solving hierarchical optimization problems," in *Impacts of Recent Computer Advances on Operations Research,* Sharda, Golden, Wasil, Balci and Stewart, Eds. North-Holland, 1989, pp. 289–301.
3. J.E. Baker, "Adaptive selection mothods for genetic algorithms," in *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1985, pp. 14–21.
4. J. Bard and J. Moore, "The mixed integer linear bilevel programming problem," *Operations Research*, vol. 38, pp. 911–921, 1990.
5. J. Bard and J. Moore, "An algorithm for the discrete bilevel programming problem," *Naval Research Logistics*, vol. 39, pp. 419–435, 1992.
6. R.E. Bellman and L.A. Zadeh, "Decision making in a fuzzy environment," *Management Science*, vol. 17, pp. 141-164, 1970.
7. D.E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1985, pp. 154–159.
8. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Massachusetts: Addison Wesley, 1989.
9. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Berlin: Springer-Verlag, 1992, Second, extended edition, 1994, Third, revised and extended edition, 1996.
10. M. Sakawa, K. Kato, H. Sunada and T. Shibano, "Fuzzy programming for multiobjective 0-1 programming problems through revised genetic algorithms," *European Journal of Operational Research*, vol. 97, pp. 149–158, 1997.
11. K. Shimizu and E. Aiyoshi, "Hierarchical multi-objective decision systems for general resource allocation problems," *Journal of Optimization Theory and Applications*, vol. 35, pp. 517-533, 1981.
12. M. Simaan and J.B. Cruz, JR., "On the Stackelberg strategy in nonzero-sum games," *Journal of Optimization Theory and Applications*, vol. 11, pp. 533–555, 1973.
13. M. Simaan and J.B. Cruz, JR., "Stackelberg solution for games with many players," *IEEE Transactions on Automatic Control*, vol. AC-18, pp. 322–324, 1973.
14. L. Vicente, G. Savard and J. Judice, "Discrete linear bilevel programming problem," *Journal of Optimization Theory and Applications*, vol. 89, pp. 597–614, 1996.
15. W.P. Wen and Y.H. Yang, "Algorithms for solving the mixed integer two-level linear programming problem," *Computers and Operations Research*, vol. 17, pp. 133–142, 1990.

Table 1. Coefficients

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
c_{0j}	21	51	90	27	17	33	85	13	36	26	98	16	48	74	61	30	68	20	69	99	
c_{1j}	38	73	10	90	33	59	71	61	27	46	58	75	31	26	70	39	85	61	98	20	
c_{2j}	21	76	41	23	71	88	16	15	64	27	90	52	67	30	64	21	46	12	49	99	
c_{3j}	66	58	93	33	27	38	73	66	49	40	78	18	10	69	89	73	96	51	19	33	
a_{1j}	16	96	38	29	36	44	50	23	75	50	30	51	18	31	78	91	91	74	91	31	
a_{2j}	28	77	80	70	19	24	48	22	36	57	10	30	32	77	52	24	47	97	36	62	
a_{3j}	10	12	11	67	70	89	39	97	80	35	54	35	54	24	65	94	38	82	76	40	
a_{4j}	56	65	31	61	87	30	50	90	21	92	19	87	12	18	40	36	67	73	95	27	
a_{5j}	36	52	83	70	10	74	83	95	15	73	48	67	34	27	35	29	40	37	83	14	
k				1							2							3			
j	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	
d^k_{0j}	17	88	23	58	13	59	26	91	37	26	35	80	96	54	11	20	86	12	61	31	
d^k_{1j}	43	63	84	84	87	87	53	43	37	21	18	73	62	12	29	86	77	48	45	31	
d^k_{2j}	26	32	49	81	72	18	50	87	52	46	50	82	77	43	63	78	54	46	80	46	
d^k_{3j}	36	12	97	54	35	17	22	86	99	41	87	94	28	24	10	92	64	24	57	55	e
b^k_{1j}	65	32	35	70	23	99	41	93	95	57	55	57	50	81	44	59	30	85	52	72	559
b^k_{2j}	77	38	18	59	42	98	29	93	34	22	10	32	72	78	64	34	87	39	20	16	472
b^k_{3j}	30	98	32	53	47	16	65	16	39	72	17	69	96	46	33	66	28	55	56	95	525
b^k_{4j}	46	26	47	27	85	12	36	48	84	99	64	35	75	51	88	17	55	43	84	48	531
b^k_{5j}	20	34	59	90	49	10	42	48	91	13	14	17	66	24	37	62	26	46	93	50	474

16. H.-J. Zimmermann, "Fuzzy programming and linear programming with several objective functions," *Fuzzy Sets and Systems*, vol. 1, pp. 45–55, 1978.