Interactive fuzzy programming based on a probability maximization model using genetic algorithms for two-level integer programming problems involving random variable coefficients

Kosuke Kato
Graduate School of Engineering,
Hiroshima University
1-4-1, Kagamiyama, Higashi-hiroshima,
739-8527 Japan
Email: kato@msl.sys.hiroshima-u.ac.jp

Masatoshi Sakawa
Graduate School of Engineering,
Hiroshima University
1-4-1, Kagamiyama, Higashi-hiroshima,
739-8527 Japan
Email: sakawa@msl.sys.hiroshima-u.ac.jp

Abstract—In this paper, we focus on two-level integer programming problems with random variable coefficients in objective functions and/or constraints. Using chance constrained programming approaches in stochastic programming, the stochastic two-level integer programming problems are transformed into deterministic two-level integer programming problems. After introducing fuzzy goals for objective functions, we consider the application of the interactive fuzzy programming technique to derive a satisfactory solution for decision makers. Since several integer programming problems have to be solved in the interactive fuzzy programming technique, we incorporate a genetic algorithm designed for integer programming problems into it. An illustrative numerical example is provided to demonstrate the feasibility of the proposed method.

I. INTRODUCTION

In this paper, we consider decision making situations in hierarchical systems, which are formulated as two-level integer programming problems. In these problems, there exist a decision maker with integer decision variables at the upper level and another decision maker with integer decision variables at the lower level.

For two-level programming problems, a number of approaches are proposed according to the relationship between these decision makers. Under the assumption that they do not have motivation to cooperate mutually, the Stackelberg solution [18] is adopted as a reasonable solution for the situation. On the other hand, in the case of a project selection problem in an administrative office and an autonomous divisions of a company, it seems natural that the decision makers cooperate with each other. For such cooperative situations, solutions that both decision makers can be satisfied with seems reasonable. In order to obtain such the satisfactory solution for the decision makers, Sakawa et al. have proposed interactive fuzzy programming techniques for two-level or multi-level linear programming problems [15], [16].

In actual decision making situations, we must often make a decision on the basis of vague information or uncertain data. For such decision making problems involving uncertainty, there exist two typical approaches: probability-theoretic approach and fuzzy-theoretic one. Stochastic programming, as an optimization method based on probability theory, have been developed in various ways [7], including two stage problem considered by G.B. Dantzig [3], chance constrained programming proposed by A. Charnes and W.W. Cooper [2]. For multiobjective stochastic linear programming problems, I.M. Stancu-Minasian [7] discussed the minimum risk approach, while J.P. Leclercq [5] and J. Teghem Jr. et al. [20] proposed interactive decision making methods.

On the other hand, fuzzy mathematical programming representing the vagueness in decision making situations by fuzzy concepts have been studied by many researchers [8], [9]. Fuzzy multiobjective linear programming, first proposed by H.-J. Zimmermann [21], have been developed rapidly developed by numerous researchers, and an increasing number of successful applications has been appearing [17], [6], [19].

As a hybrid of the stochastic approach and the fuzzy one, Sakawa et al. [11], [12] presented an interactive fuzzy satisficing method to derive a satisficing solution for the decision maker after reformulating a multiobjective stochastic linear programming problem using several models for chance constrained programming. Furthermore, they also proposed decision making methods based on interactive fuzzy programming for two-level stochastic linear programming problems. However, there has never been reported the application of these methods to two-level stochastic programming problems with discrete decision variables.

Under these circumstances, in this paper, we deal with two-level integer programming problems with random variable coefficients in objective functions and/or constraints. First, the two-level stochastic integer programming problem is transformed into deterministic ones based on a probability maximization model. Then, we attempt to obtain a satisfactory solution for decision makers through interactive fuzzy
programming [15], [16]. In this derivation of a satisfactory solution through interactive fuzzy programming, several deterministic linear and nonlinear integer programming problems have to be solved. Since it is generally difficult to strictly solve linear and nonlinear integer programming problems with considerably many decision variables and/or constraints, we use a genetic algorithm designed for integer programming problems.

II. TWO-LEVEL STOCHASTIC INTEGER PROGRAMMING

PROBLEMS

Two-level stochastic integer programming problems involving random variable coefficients are formulated as follows.

\[
\begin{align*}
& \text{minimize}_{\text{DM1 (upper level)}} \quad z_1(x_1, x_2, \omega) = c_{11}(\omega) x_1 + c_{12}(\omega) x_2 \\
& \text{minimize}_{\text{DM2 (lower level)}} \quad z_2(x_1, x_2, \omega) = c_{21}(\omega) x_1 + c_{22}(\omega) x_2 \\
& \text{subject to} \quad A_1 x_1 + A_2 x_2 \leq b(\omega) \\
& \quad x_{1,j} \in \{0, 1, \ldots, \nu_{1,j}\}, \quad j_1 = 1, \ldots, n_1 \\
& \quad x_{2,j_2} \in \{0, 1, \ldots, \nu_{2,j_2}\}, \quad j_2 = 1, \ldots, n_2
\end{align*}
\]

where \(x_1\) is the \(n_1\) dimensional integer decision variable column vector for the upper level decision maker DM1, \(x_2\) is the \(n_2\) dimensional integer decision variable column vector for the lower level decision maker DM2, \(A_1\) is the \(m \times n_1\) coefficient matrix, \(A_2\) is the \(m \times n_2\) coefficient matrix. It should be noted that \(c_{ij}(\omega), \ i = 1, 2, \ j = 1, 2\) are Gaussian random variable row vectors with finite mean \(c_{ij}\) and finite covariance matrix \(V_{pp} = \left(\begin{smallmatrix} h_p h_q \end{smallmatrix}\right) = \text{Cov}[c_{ij}(\omega), c_{ij}(\omega)]\), \(p = 1, 2, q = 1, 2, h_p = 1, \ldots, n_p, h_q = 1, \ldots, n_q\), and \(b(\omega), \ i = 1, \ldots, m\) are random variables with finite mean \(b_i\) which are independent of each other, and the distribution function of each of them is also assumed to be continuous and increasing. Furthermore, \(n_{j_i}, i = 1, 2, j_i = 1, \ldots, n_i\) are positive integer values.

Since (1) contains random variable coefficients, definitions and solution methods for ordinary deterministic mathematical programming problems cannot be directly applied. Consequently, we deal with the constraints in (1) as chance constrained conditions [2] which mean that the constraints need to be satisfied not always but with a certain probability (satisficing level) and over. Namely, replacing the constraints in (1) by chance constrained conditions with satisfying levels \(\beta_i, i = 1, \ldots, m\), (1) can be converted as:

\[
\begin{align*}
& \text{minimize}_{\text{DM1 (upper level)}} \quad z_1(x_1, x_2, \omega) = c_{11}(\omega) x_1 + c_{12}(\omega) x_2 \\
& \text{minimize}_{\text{DM2 (lower level)}} \quad z_2(x_1, x_2, \omega) = c_{21}(\omega) x_1 + c_{22}(\omega) x_2 \\
& \text{subject to} \quad \Pr\{a_{ij} x_1 + a_{i2} x_2 \leq b_i(\omega)\} \geq \beta_i \\
& \quad x_{1,j} \in \{0, 1, \ldots, \nu_{1,j}\}, \quad j_1 = 1, \ldots, n_1 \\
& \quad x_{2,j_2} \in \{0, 1, \ldots, \nu_{2,j_2}\}, \quad j_2 = 1, \ldots, n_2
\end{align*}
\]

where \(a_{ij}\) is the \(i\)th row vector of \(A\) and \(b_i(\omega)\) is the \(i\)th element of \(b(\omega)\).

Because distribution functions \(F_i(r) = \Pr\{b_i(\omega) \leq r\}\) of random variables \(b_i(\omega), \ i = 1, \ldots, m\) are continuous and increasing, the \(i\)th constraint in (2) can be rewritten as:

\[
\Pr\{a_{ij} x_1 + a_{i2} x_2 \leq b_i(\omega)\} \geq \beta_i \iff a_{ij} x_1 + a_{i2} x_2 \leq \hat{b}_i
\]

where \(\hat{b}_i = F_i^{-1}(1 - \beta_i)\).

Thereby, (2) can be transformed into the following equivalent problem:

\[
\begin{align*}
& \text{minimize}_{\text{DM1 (upper level)}} \quad z_1(x_1, x_2, \omega) = c_{11}(\omega) x_1 + c_{12}(\omega) x_2 \\
& \text{minimize}_{\text{DM2 (lower level)}} \quad z_2(x_1, x_2, \omega) = c_{21}(\omega) x_1 + c_{22}(\omega) x_2 \\
& \text{subject to} \quad A_1 x_1 + A_2 x_2 \leq \hat{b} \\
& \quad \hat{b}_j = (\hat{b}_1, \ldots, \hat{b}_m)^T. \text{ In the following, for notational convenience, the feasible region of (3) is denoted by } X.
\end{align*}
\]

For the two-level chance constrained programming problem (3), several models such as an expectation optimization model, a variance minimization model, a probability maximization model, a fractile criterion optimization model, have been proposed depending on the concern of the decision maker.

In this paper, we study the probability maximization model, which aims to maximize the probability that each of objective functions is less than or equal to a certain permissible level.

III. PROBABILITY MAXIMIZATION MODEL

In (3), substituting the maximization of the probability that each of the objective functions \(z_i(x_1, x_2, \omega)\) is less than or equal to a certain permissible level \(f_i\) for the minimization of the objective functions \(z_i(x_1, x_2, \omega) = c_{1i}(\omega) x_1 + c_{2i}(\omega) x_2, \ l = 1, 2\), the problem can be converted as follows.

\[
\begin{align*}
& \text{maximize}_{\text{DM1 (upper level)}} \quad p_1(x_1, x_2) = \Pr\{z_1(x_1, x_2, \omega) \leq f_1\} \\
& \text{maximize}_{\text{DM2 (lower level)}} \quad p_2(x_1, x_2) = \Pr\{z_2(x_1, x_2, \omega) \leq f_2\} \\
& \text{subject to} \quad A_1 x_1 + A_2 x_2 \leq \hat{b} \\
& \quad \hat{b}_j = (\hat{b}_1, \ldots, \hat{b}_m)^T, \ j = 1, \ldots, n_1 \\
& \quad \hat{b}_j = (\hat{b}_1, \ldots, \hat{b}_m)^T, \ j = 1, \ldots, n_2
\end{align*}
\]

The objective functions \(\Pr\{z_l(x_1, x_2, \omega) \leq f_l\}, l = 1, 2\) are rewritten as:

\[
\begin{align*}
& \Pr\{z_l(x_1, x_2, \omega) \leq f_l\} \\
= & \Pr\{c_{1l}(\omega) x_1 + c_{2l}(\omega) x_2 \leq f_l\} \\
= & \Pr\left\{\frac{\sqrt{\left|x_1^T f_1 - c_{1l} x_1 - c_{2l} x_2\right|^2}}{\sqrt{\left|x_1^T f_1 - c_{1l} x_1 - c_{2l} x_2\right|^2}} \leq \frac{f_l - (c_{1l} x_1 + c_{2l} x_2)}{\sqrt{\left|x_1^T f_1 - c_{1l} x_1 - c_{2l} x_2\right|^2}}\right\} \\
= & \Phi_l\left(\frac{f_l - (c_{1l} x_1 + c_{2l} x_2)}{\sqrt{\left|x_1^T f_1 - c_{1l} x_1 - c_{2l} x_2\right|^2}}\right)
\end{align*}
\]

where \(\Phi_l(\cdot)\) is the distribution function of a standard Gaussian random variable.
Then, (4) can be transformed into the following equivalent problem:

\[
\begin{align*}
\text{maximize} & \quad \Phi_1 \left( \frac{f_1 - (\bar{c}_{11}x_1 + \bar{c}_{12}x_2)}{\sqrt{[x_1^1, x_1^2]_V [x_1^2, x_1^1]^T}} \right) \\
\text{maximize} & \quad \Phi_2 \left( \frac{f_2 - (\bar{c}_{21}x_1 + \bar{c}_{22}x_2)}{\sqrt{[x_2^1, x_2^2]_V [x_2^2, x_2^1]^T}} \right) \\
\text{subject to} & \quad A_1x_1 + A_2x_2 \leq b \\
& \quad x_{1j_1} \in \{0, 1, \ldots, n_{1j_1}\}, j_1 = 1, \ldots, n_1 \\
& \quad x_{2j_2} \in \{0, 1, \ldots, n_{2j_2}\}, j_2 = 1, \ldots, n_2 \\
\end{align*}
\]

(5)

IV. INTERACTIVE FUZZY PROGRAMMING

In general, it seems natural that there exists the ambiguity or fuzziness in the evaluation of each objective function by the decision maker. In order to consider the imprecise nature of the decision maker’s judgement for each objective function in (5), we introduce the fuzzy goals such as “\(p_l(x_1, x_2)\) should be substantially greater than or equal to a certain value”. Then, (5) can be rewritten as:

\[
\begin{align*}
\text{maximize} & \quad \mu_1(p_1(x_1, x_2)) \\
\text{maximize} & \quad \mu_2(p_2(x_1, x_2)) \\
\text{subject to} & \quad A_1x_1 + A_2x_2 \leq b \\
& \quad x_{1j_1} \in \{0, 1, \ldots, n_{1j_1}\}, j_1 = 1, \ldots, n_1 \\
& \quad x_{2j_2} \in \{0, 1, \ldots, n_{2j_2}\}, j_2 = 1, \ldots, n_2 \\
\end{align*}
\]

(6)

where \(\mu_1(\cdot)\) is a membership function to quantify a fuzzy goal for the \(l\)th objective function in (5), as shown in Fig. 1.

We attempt to derive a satisfactory solution by the interactive fuzzy programming technique using fuzzy goals to consider the ambiguity of the decision makers’ judgement and a ratio of the satisfactory degree for DM2 \(\mu_2\) to that for DM1 \(\mu_1\).

Computational procedure of interactive fuzzy programming

Step 1: Determine the satisfying levels \(\beta_i, i = 1, \ldots, m\) for constraints in (2).

Step 2: Solve problems (7), (8) to obtain the individual minimum \(\bar{z}_{l_{\text{min}}}\) and maximum \(\bar{z}_{l_{\text{max}}}\) of \(E\{z_l(x_1, x_2, \omega)\} = \bar{c}_{1l}x_1 + \bar{c}_{2l}x_2, l = 1, 2\) under the chance constrained conditions with satisfying levels \(\beta_i, i = 1, \ldots, m\).

\[
\begin{align*}
\text{minimize} & \quad \bar{z}_{l_{\text{min}}}(x_1, x_2), \quad l = 1, 2 \quad (7) \\
\text{maximize} & \quad \bar{z}_{l_{\text{max}}}(x_1, x_2), \quad l = 1, 2 \quad (8)
\end{align*}
\]

Since these problems are deterministic integer programming problems, in order to find (approximate) optimal solutions to them, we apply Genetic Algorithm with Double Strings using Continuous Relaxation based on Reference Solution Updating (GAD-SCRRSU) designed for integer programming problems [10]. Then, specify permissible levels \(f_l, l = 1, 2\) for the objective functions in consideration of \(\bar{z}_{l_{\text{min}}}\) and \(\bar{z}_{l_{\text{max}}}\).

Step 3: For the purpose of obtaining the individual minimum \(p_{l_{\text{min}}}\) and maximum \(p_{l_{\text{max}}}\) of \(p_l(x_1, x_2)\), \(l = 1, 2\) in (5), solve the following problems.

\[
\begin{align*}
\text{minimize} & \quad f_l - (\bar{c}_{1l}x_1 + \bar{c}_{2l}x_2) \quad \text{maximize} \\
& \quad \sqrt{[x_1^1, x_1^2]_V [x_1^2, x_1^1]^T} \\
& \quad \sqrt{[x_2^1, x_2^2]_V [x_2^2, x_2^1]^T}
\end{align*}
\]

(9)

(10)

We also apply GADSCRRSU [10] to solve these problems. Then, specify membership functions for objective functions in (5), \(\mu_l(p_l(x_1, x_2)), l = 1, 2\), and set the upper bound \(\Delta_{\text{max}}\) and the lower bound \(\Delta_{\text{min}}\) of a ratio of the satisfactory degree for DM2 to that for DM1. \(\Delta = \mu_2(p_2(x_1, x_2)) / \mu_1(p_1(x_1, x_2))\).

Step 4: Based on the maximizing decision of Bellman and Zadeh [1], solve the following maximin problem through GADSCRRSU [10].

\[
\text{maximize} \min_{l=1,2} \{\mu_l(p_l(x_1, x_2))\} \\
\text{subject to} \quad x \in X
\]

(11)

If DM1 is satisfied with the optimal solution to (11), terminate the interaction procedure. Otherwise, taking into account a ratio of satisfactory degrees \(\Delta\), DM1 subjectively specifies the minimal satisfactory level \(\delta\) for \(\mu_1(p_1(x_1, x_2))\).

Step 5: Solve the following problem for \(\delta\) using GADSCRRSU [10].

\[
\begin{align*}
\text{maximize} & \quad \mu_2(p_2(x_1, x_2)) \\
\text{subject to} & \quad \mu_1(p_1(x_1, x_2)) \geq \delta \\
& \quad x \in X
\end{align*}
\]

(12)

Then, calculate the value of \(\Delta\) corresponding to the optimal solution \((x_1^*, x_2^*)\) to (12).

Step 6: If DM1 is satisfied with \(\mu_1(p_1(x_1^*, x_2^*))\), \(l = 1, 2\) and \(\Delta \in [\Delta_{\text{min}}, \Delta_{\text{max}}]\), stop. Otherwise, ask DM1 to update the minimal satisfactory level \(\delta\). To be more specific, if \(\Delta \leq \Delta_{\text{min}}, \text{i.e., } \mu_1(p_1(x_1^*, x_2^*))\) is much greater than \(\mu_2(p_2(x_1^*, x_2^*))\), DM1 should decrease the value of \(\delta\). If \(\Delta \geq \Delta_{\text{max}}, \text{i.e., } \mu_1(p_1(x_1^*, x_2^*))\) is much less than \(\mu_2(p_2(x_1^*, x_2^*))\), DM1 should increase the value of \(\delta\). Go to step 5.
V. GENETIC ALGORITHM WITH DOUBLE STRINGS USING CONTINUOUS RELAXATION BASED ON REFERENCE SOLUTION UPDATING (GADSCRRSU)

In this section, we explain GADSCRRSU [10] proposed as a general solution method for linear integer programming problems defined as (13).

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad A x \leq b \\
& \quad x_j \in \{0, 1, \ldots, v_j\}, \quad j = 1, \ldots, n
\end{align*}
\]

(13)

where \( A = [p_1, \ldots, p_m] \) is an \( m \times n \) coefficient matrix, \( b = (b_1, \ldots, b_m)^T \) is an \( m \) dimensional column vector and \( c = (c_1, \ldots, c_n) \) is an \( n \) dimensional row vector.

A. Individual Representation

In GADSCRRSU, double strings shown in Fig. 2 is used as the individual representation.

\[
\begin{array}{cccc}
\text{Individual} \mathbf{S} & \begin{array}{cccc}
s(1) & s(2) & \cdots & s(n) \\
g_1(1) & g_2(1) & \cdots & g_n(1) \\
g_1(2) & g_2(2) & \cdots & g_n(2) \\
\end{array}
\end{array}
\]

Fig. 2. Double string

In the figure, each of \( s(j), \ j = 1, \ldots, n \) is the index of an element in a solution vector and each of \( g_i( j) \in \{0, 1, \ldots, v_i( j)\}, \ j = 1, \ldots, n \) is the value of the element, respectively. For example, the first column of the double string in Fig. 2 means that the candidate of the value of \( x_{s(1)} \) is \( g_1(1) \).

B. Decoding Algorithm

Since a solution corresponding to a double string, i.e., \( \mathbf{x} = (g_1, g_2, \ldots, g_n)^T \) does not always satisfy the constraint \( A x \leq b \) in (13), the decoding procedure is needed to repair infeasible solutions. In [10], a decoding algorithm of double strings for linear integer programming problems is constructed as follows. In the algorithm, a feasible solution \( \mathbf{x}^* \), called a reference solution, is used as the origin of decoding.

Decoding algorithm using a reference solution

In this algorithm, it is assumed that a feasible solution \( \mathbf{x}^* \) to (13) is obtained in advance. Let \( n \) and \( N \) be the number of variables and the number of individuals in the population, respectively. Also, \( b^+ \) means a column vector of positive right-hand side constants, and the corresponding coefficient matrix is denoted by \( A^+ = [p_1^+, \ldots, p_n^+] \).

Step 1: Let \( j := 1 \) and \( \text{psum} := 0 \).

Step 2: If \( g_{s(j)} = 0 \), set \( q_{s(j)} := 0 \) and \( j := j + 1 \), and go to step 4. If \( g_{s(j)} \neq 0 \), go to step 3.

Step 3: \( \text{psum} + p_{s(j)}^+ g_{s(j)} \leq b^+ \), set \( q_{s(j)} := g_{s(j)} \), \( \text{psum} := \text{psum} + p_{s(j)}^+ g_{s(j)} \) and \( j := j + 1 \), and go to step 4. Otherwise, set \( q_{s(j)} := 0 \) and \( j := j + 1 \), and go to step 4.

Step 4: If \( j > n \), go to step 5. If \( j \leq n \), go to step 2.

Step 5: Let \( j := 1, l := 0 \) and \( \text{sum} := 0 \).

Step 6: If \( g_{s(j)} = 0 \), set \( j := j + 1 \) and go to step 8. If \( g_{s(j)} \neq 0 \), set \( \text{sum} := \text{sum} + p_{s(j)}^+ g_{s(j)} \) and go to step 7.

Step 7: If \( \text{sum} \leq b \), set \( l := j, \ j := j + 1 \), and go to step 8. Otherwise, set \( j := j + 1 \) and go to step 8.

Step 8: If \( j > n \), go to step 9. If \( j \leq n \), go to step 6.

Step 9: If \( l > 0 \), go to step 10. If not, go to step 11.

Step 10: For \( x_{s(j)} \) satisfying \( 1 \leq j \leq l \), let \( x_{s(j)} := g_{s(j)} \). For \( x_{s(j)} \) satisfying \( l + 1 \leq j \leq n \), let \( x_{s(j)} := 0 \), and stop.

Step 11: Let \( \text{sum} := \sum_{k=1}^{n} p_{s(k)} \cdot x_{s(k)} \) and \( j := 1 \).

Step 12: If \( g_{s(j)} = x_{s(j)}^* \), let \( x_{s(j)} := g_{s(j)} \) and \( j := j + 1 \), and go to step 16. Otherwise, go to step 13.

Step 13: If \( \text{sum} - p_{s(j)} \cdot x_{s(j)}^* + p_{s(j)} \cdot t_{s(j)} \leq b \), set \( \text{sum} := \text{sum} - p_{s(j)} \cdot x_{s(j)}^* + p_{s(j)} \cdot t_{s(j)} \) and \( x_{s(j)} := t_{s(j)} \), and go to step 16. Otherwise, go to step 14.

Step 14: Let \( t_{s(j)} := 0.5 \cdot (x_{s(j)}^* + g_{s(j)}) \) and go to step 15.

Step 15: If \( \text{sum} - p_{s(j)} \cdot x_{s(j)}^* + p_{s(j)} \cdot t_{s(j)} \leq b \), set \( \text{sum} := \text{sum} - p_{s(j)} \cdot x_{s(j)}^* + p_{s(j)} \cdot t_{s(j)} \) and \( x_{s(j)} := t_{s(j)} \), and go to step 16. Otherwise, set \( x_{s(j)} := x_{s(j)}^* \) and go to step 16.

Step 16: If \( j > n \), stop. Otherwise, return to step 12.

Because solutions obtained the decoding algorithm using a reference solution tend to concentrate around the reference solution, the reference solution updating procedure is adopted.

C. Reference solution updating

The diversity of solutions \( \mathbf{x} \) greatly depends on the reference solution used in the above decoding algorithm. In order to widen the search region, we propose the following reference solution updating procedure such that the current reference solution is updating by another feasible solution if the diversity of solutions seems to be lost. To do so, for every generation, check the dependence on the reference solution through the calculation of the mean of the Hamming distance between all solutions corresponding to individuals and the reference solution, and when the dependence on the reference solution is strong, replace the reference solution by the solution corresponding to an individual having maximum Hamming distance.

Let \( N, \mathbf{x}^*, \eta (\eta < 1.0) \) and \( \mathbf{x}^r \) respectively denote the number of individuals, the reference solution, a parameter for reference solution updating and a feasible solution decoded by the \( r \) th individual, then the reference solution updating procedure can be described as follows.

Reference solution updating procedure

Step 1: Set \( r := 1, r_{\text{max}} := 1, d_{\text{max}} := 0 \) and \( d_{\text{sum}} := 0 \).

Step 2: Calculate \( d_r = \sum_{j=1}^{n} |x_{s(j)}^r - x_{s(j)}^*| \) and let \( d_{\text{sum}} := d_{\text{sum}} + d_r \). If \( d_r > d_{\text{max}} \) and \( \mathbf{c}_{\mathbf{x}^r} < \mathbf{c}_{\mathbf{x}^*} \), let \( d_{\text{max}} := d_r, r_{\text{max}} := r \) and \( r := r + 1 \), and go to step 3. Otherwise, let \( r := r + 1 \) and go to step 3.

Step 3: If \( r > n \), go to step 4. Otherwise, return to step 2.

Step 4: If \( d_{\text{sum}} / (N \cdot \sum_{j=1}^{n} v_j) < \eta \), then update the reference solution as \( \mathbf{x}^* := \mathbf{x}_{\text{max}}^r \), and stop. Otherwise,
proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (MCDM 2007)

Table 1

<table>
<thead>
<tr>
<th>$x^*_i = \hat{x}_j$</th>
<th>$x^*_i \neq \hat{x}_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>606</td>
</tr>
<tr>
<td>$\neq 0$</td>
<td>279</td>
</tr>
</tbody>
</table>

stop without updating the reference solution.

It should be observed here that when the constraints of the problem are strict, there exist a possibility that all of the individuals are decoded in the neighborhood of the reference solution. To avoid a such possibility, in addition to the reference solution updating procedure, after every $P$ generations, the reference solution is replaced by another feasible solution.

D. Usage of Continuous Relaxation

It is expected that an optimal solution to the continuous relaxation problem becomes a good approximate optimal solution of the original integer programming problem. With this observation in mind, after generating 20 single-objective solution of the original integer programming problem. With relaxation problem becomes a good approximate optimal so-

such variables by uniform integer random numbers in $[-999, 0]$ and $[0, 999]$, respectively, while $b_i, i = 1, \ldots, m$ are defined as

$$b_i = \gamma \sum_{j=1}^{n} a_{ij}, \quad i = 1, \ldots, m$$

where a positive constant $\gamma$ is a parameter to control the degree of strictness of the constraints, determined by a uniform real random number ranging from 5 to 10. In addition, upper bounds $\nu_j$ of $x_j, j = 1, \ldots, n$ are set at 20 for all $j$.

Table 1 shows relations between the optimal solution to integer knapsack problems $x^*_i$ and that to corresponding continuous relaxation problems $\hat{x}_j$, while Fig. 3 shows the frequency distribution of differences between the values of an optimal solution $x_i$ of integer programming problems and an optimal solution $\hat{x}_i$ of linear programming relaxation problems.

As a result, it is recognized that each variable $x_i$ takes exactly or approximately the same value that $\hat{x}_i$ does, especially, such variables $x_i$ as $\hat{x}_i = 0$ are very likely to be equal to 0.

Based on the fact, the information about the optimal solution to the continuous relaxation problem is used in the generation of the initial population and the mutation [10].

E. Reproduction

As a reproduction operator, elitist expected value selection, which is the combination of expected value selection and elitist preserving selection, is adopted. In [14], elitist expected value selection is defined as a combination of elitism and expected value selection as mentioned below.

F. Crossover

If a single-point crossover or multi-point crossover is directly applied to individuals of double string type, the $k$th element of an offspring may take the same number that the $k$th element takes. Similar violation occurs in solving traveling salesman problems or scheduling problems through genetic algorithms as well. In order to avoid this violation, a crossover method called partially matched crossover (PMX) was proposed [4] and was modified so as to be suitable for double strings [14].

PMX for double string

Step 0: Select two individuals $X, Y$ from the population as parent individuals and prepare copies $X'$ and $Y'$ of $X$ and $Y$, respectively.

Step 1: Choose two crossover points at random on these strings, say, $h$ and $k$ ($h < k$).

Step 2: (a) Set $j = h$.

(b) Find $j'$ such that $s_{X'}(j') = s_Y(j)$. Then, interchange $(s_{X'}(j')$, $g_{s_{X'}(j')})^T$ with $(s_{X'}(j'), g_{s_{X'}(j')})^T$ and set $j = j + 1$.

(c) If $j > k$, stop. Otherwise, return to (b).
Step 3: Replace the part from \( h \) to \( k \) of \( X' \) with that of \( Y \) and let \( X'' \) be the offspring of \( X \).

An illustrative example of crossover is shown in Fig. 4.

<table>
<thead>
<tr>
<th>( h )</th>
<th>( k )</th>
<th>( X' = 5 7 1 3 4 6 2 )</th>
<th>( y = 3 1 6 5 7 2 4 )</th>
<th>( 0 5 4 2 0 9 3 )</th>
</tr>
</thead>
</table>

\[ X = \begin{bmatrix} 5 & 7 & 1 & 3 & 4 & 6 & 2 \\ 6 & 3 & 0 & 4 & 8 & 2 & 3 \end{bmatrix} \quad Y = \begin{bmatrix} 3 & 1 & 6 & 5 & 7 & 2 & 4 \\ 0 & 5 & 4 & 2 & 0 & 9 & 3 \end{bmatrix} \]

\[ X' = \begin{bmatrix} 5 & 7 & 1 & 3 & 4 & 6 & 2 \\ 6 & 3 & 0 & 4 & 8 & 2 & 3 \end{bmatrix} \quad Y' = \begin{bmatrix} 3 & 1 & 6 & 5 & 7 & 2 & 4 \\ 0 & 5 & 4 & 2 & 0 & 9 & 3 \end{bmatrix} \]

Fig. 4. Partially matched crossover (PMX) for double strings.

G. Mutation

It is considered that mutation plays the role of local random search in genetic algorithms. In this paper, two mutation operators (bit-reverse type and inversion) are used. A direct extension of mutation for 0-1 programming problems is to change the value of \( CV \) at random in \( CJ/BC \).

<table>
<thead>
<tr>
<th>( h )</th>
<th>( k )</th>
<th>( X = \begin{bmatrix} 4 &amp; 1 &amp; 5 &amp; 7 &amp; 6 &amp; 2 &amp; 3 \ 2 &amp; 5 &amp; 0 &amp; 9 &amp; 3 &amp; 2 &amp; 1 \end{bmatrix} \quad X' = \begin{bmatrix} 4 &amp; 1 &amp; 5 &amp; 7 &amp; 6 &amp; 2 &amp; 3 \ 2 &amp; 5 &amp; 0 &amp; 9 &amp; 3 &amp; 2 &amp; 1 \end{bmatrix} )</th>
</tr>
</thead>
</table>

\[ X = \begin{bmatrix} 4 & 1 & 5 & 7 & 6 & 2 & 3 \\ 2 & 5 & 0 & 9 & 3 & 2 & 1 \end{bmatrix} \quad X' = \begin{bmatrix} 4 & 1 & 5 & 7 & 6 & 2 & 3 \\ 2 & 5 & 0 & 9 & 3 & 2 & 1 \end{bmatrix} \]

and go to step 4.

Step 4: If \( j < n \), set \( j := j + 1 \) and return to step 2. Otherwise, go to step 5.

Step 5: If \( r < N \), set \( r := r + 1 \) and return to step 1. Otherwise, stop.

H. Computational procedures of GADSCRRSU

Step 0: Determine values of the parameters used in the genetic algorithm: the population size \( N \), the generation gap \( G \), the probability of crossover \( p_c \), the probability of mutation \( p_m \), the probability of inversion \( p_i \), the minimal search generation \( I_{min} \), the maximal search generation \( I_{max} \), the scaling constant \( c_{max} \), the convergence criterion \( \varepsilon \), the degree of use of information about solutions to linear programming relaxation problems \( R \), and set the generation counter \( t \) at 0.

Step 1: Generate the initial population consisting of \( N \) individuals based on the information of the optimal solution to the continuous relaxation problem.

Step 2: Decode each individual (genotype) in the current population and calculate its fitness based on the corresponding solution (phenotype).

Step 3: If the termination condition is fulfilled, stop. Otherwise, let \( t := t + 1 \) and go to step 4.

Step 4: Apply reproduction operator using elitist expected value selection after linear scaling.

Step 5: Apply crossover operator, called PMX (Partially Matched Crossover) for double string.

Step 6: Apply mutation based on the information of a solution to the continuous relaxation problem.

Step 7: Apply inversion operator. Go to step 2.

VI. NUMERICAL EXAMPLE

To demonstrate the feasibility of the proposed method, consider the following two-level integer programming problem...
Involving random variable coefficients.

\[
\begin{align*}
\text{minimize} & \quad z_1(x_1, x_2, \omega) = c_{11}(\omega) x_1 + c_{12}(\omega) x_2 \\
\text{subject to} & \quad a_{11} x_1 + a_{12} x_2 \leq b_1(\omega) \\
& \quad a_{21} x_1 + a_{22} x_2 \leq b_2(\omega) \\
& \quad \vdots \\
& \quad a_{101} x_1 + a_{102} x_2 \leq b_{10}(\omega) \\
& \quad x_{1j}, \quad x_{2j} \in \{0, 1, \ldots, 30\}, \quad j = 1, \ldots, 15
\end{align*}
\]

where \( x_1 = (x_1, \ldots, x_{15})^T, \) \( x_2 = (x_{16}, \ldots, x_{30})^T, \) each element of random variable vectors \( c_{i}(\omega) = (c_{i1}(\omega), c_{i2}(\omega)), \) \( i = 1, 2 \) is a Gaussian random variable whose mean is given as Table II, and random variables \( b_i(\omega), i = 1, \ldots, 10 \) are also Gaussian random variables, \( N(6753, 20^2), N(-3588, 50^2), \) \( N(5534, 25^2), \) \( N(10225, 80^2), \) \( N(-2837, 15^2), \) \( N(6122, 30^2), N(3169, 70^2), N(5811, 100^2), N(7565, 120^2), N(-1433, 30^2), \) respectively. All of \( p_{ij}, s \) are equal to 30.

Table III shows the result of the application of the proposed interactive fuzzy programming based on the probability maximization model.

Parameters of GADSCRSSU [10] is set as: population size \( N = 100, \) crossover rate \( p_c = 0.9, \) generation gap \( G = 0.9, \) mutation rate \( p_m = 0.05, \) inversion rate \( p_i = 0.05, \) parameter for reproduction \( \lambda = 0.9, \) minimal search generation number \( I_{\min} = 500, \) maximal search generation number \( I_{\max} = 1000, \) scaling constant \( \varepsilon_{\text{mult}} = 1.6, \) parameter for reference solution updating \( \eta = 0.2, \) penalty constant \( \theta = 5. \)

First, following step 1 in the interactive fuzzy programming, DM1 specifies satisficing levels \( \beta_i, i = 1, \ldots, 10 \) as:

\[
(\beta_1, \ldots, \beta_{10})^T = (0.95, 0.80, 0.85, 0.90, 0.90, 0.85, 0.85, 0.95, 0.80, 0.95).
\]

Then, the corresponding \( \hat{b} \) are calculated as:

\[
\hat{b} = (6720.10, -3630.08, 5508.09, 10122.48, -2856.22, 6000.91, 3096.45, 5646.51, 7464.01, -1482.35)^T.
\]

Next, following step 2, minimal values \( z_{1,\min} \) and maximal values \( z_{1,\max} \) of objective functions \( E[z_1(x_1, x_2)] \) under the chance constrained conditions corresponding to given satisficing levels are calculated using GADSCRSSU as

\[
\begin{align*}
z_{1,\min} &= -2206, \quad z_{2,\min} = -1278 \\
z_{1,\max} &= 1670, \quad z_{2,\max} = 3323
\end{align*}
\]

Based on these values, the permissible levels \( f_l, l = 1, 2 \) for objective functions are specified as \( f_1 = -200, f_2 = 1000. \)

Following step 3, after (9) and (10) are solved by GADSCRSSU, minimal values \( p_{l,\min} \) and maximal values \( p_{l,\max} \) are calculated as

\[
\begin{align*}
p_{1,\min} &= 0.003387, \quad p_{2,\min} = 0.183073 \\
p_{1,\max} &= 0.908240, \quad p_{2,\max} = 0.995727
\end{align*}
\]

In consideration of these values, the membership functions to quantify fuzzy goals for objective functions are subjectively determined. Here, the following linear membership function, as shown in Fig. 6, is adopted.

\[
\mu_l(p_l(x_1, x_2)) = \begin{cases} 
1 & \text{if } p_l(x_1, x_2) \geq p_{l,1} \\
\frac{p_{l,1} - p_{l,0}}{p_{l,1} - p_{l,0}} & p_{l,0} \leq p_l(x_1, x_2) < p_{l,1} \\
0 & \text{if } p_l(x_1, x_2) \leq p_{l,0}
\end{cases}
\]

In this paper, parameters \( p_{l,1}, p_{l,0}, l = 1, 2 \) which characterize membership functions \( \mu_l(\cdot) \) are determined by Zimmermann’s method [21].

\[
\begin{align*}
p_{1,1} &= p_1(x_1, x_2) = 0.908240 \\
p_{1,0} &= p_1(x_1, x_2) = 0.420420 \\
p_{2,1} &= p_2(x_1, x_2) = 0.997077 \\
p_{2,0} &= p_2(x_1, x_2) = 0.507786
\end{align*}
\]

Furthermore, the upper bound and the lower bound of the ratio of satisfactory degrees \( \Delta \) are set as \( \Delta_{\max} = 0.85 \) and \( \Delta_{\min} = 0.75. \)

Following step 4, the maxmin problem, (11), are solved by GADSCRSSU. For the obtained optimal solution \( x_1^*, x_2^* \), corresponding objective function values and membership function values are calculated as \( p_1(x_1^*, x_2^*) = 0.7392, p_2(x_1^*, x_2^*) = 0.8290, \) \( \mu_1(p_1(x_1^*, x_2^*)) = 0.6535, \mu_2(p_2(x_1^*, x_2^*)) = 0.6529. \) Then, the ratio of satisfactory degrees \( \Delta \) is equal to 0.9091. Since DM1 is not satisfied with this solution, DM1 sets the minimal satisfactory level \( \delta \) for \( p_1(x_1, x_2) \) to 0.70.

In step 5, (12) for \( \delta = 0.70 \) is solved by GADSCRSSU. Then, the ratio of satisfactory degrees for the optimal solution is calculated as \( \Delta = 0.9033. \)

Following step 6, DM1 judges if he is satisfied with the solution obtained in step 5. Since the ratio of satisfactory
degrees $\Delta$ is greater than $\Delta_{\text{max}} = 0.85$, DM1 can not be satisfied with it and he updates the minimal satisfactory level $\hat{\delta}$ from 0.70 to 0.80.

Again in step 5, (12) for $\hat{\delta} = 0.80$ is solved by GAD-SCRRSU. Then, the ratio of satisfactory degrees for the optimal solution is calculated as $\Delta = 0.6715$.

In step 6, since the ratio of satisfactory degrees $\Delta$ is less than $\Delta_{\text{min}} = 0.75$, DM1 can not be satisfied with it and he updates the minimal satisfactory level $\hat{\delta}$ from 0.80 to 0.75.

Again in step 5, (12) for $\hat{\delta} = 0.75$ is solved by GAD-SCRRSU. Then, the ratio of satisfactory degrees for the optimal solution is calculated as $\Delta = 0.8131$.

In step 6, since the ratio of satisfactory degrees $\Delta$ exists in the interval $[0.75, 0.85]$ and DM1 is satisfied with it, the satisfactory solution is obtained and the interaction procedure is stopped.

VII. CONCLUSIONS

In this paper, focusing on two-level integer programming problems involving random variable coefficients, we presented interactive fuzzy programming based on a probability maximization model. Since several integer programming problems in the proposed interactive fuzzy programming approach showed its feasibility to have been solved, we adopt the genetic algorithm with double strings using continuous relaxation based on reference solution updating (GADSCRRSU) [10]. Furthermore, we showed its feasibility for a simple numerical example. As future problems, we are going to consider other stochastic programming models such as the expectation optimization model, the variance minimization model and so forth, and two-level integer programming problems involving fuzzy random variable coefficients.

REFERENCES


TABLE III

<table>
<thead>
<tr>
<th>Interaction</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta$</td>
<td>0.70</td>
<td>0.80</td>
<td>0.75</td>
</tr>
<tr>
<td>$\mu_1(x=1, x=2)$</td>
<td>0.5585</td>
<td>0.7001</td>
<td>0.6528</td>
</tr>
<tr>
<td>$\mu_2(x=1, x=2)$</td>
<td>0.6529</td>
<td>0.5310</td>
<td>0.6121</td>
</tr>
<tr>
<td>$p_1(x=1, x=2)$</td>
<td>0.7562</td>
<td>0.6022</td>
<td>0.5810</td>
</tr>
<tr>
<td>$p_2(x=1, x=2)$</td>
<td>0.8200</td>
<td>0.8192</td>
<td>0.7722</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>0.9901</td>
<td>0.9465</td>
<td>0.7120</td>
</tr>
</tbody>
</table>