Interactive Fuzzy Programming for Two-level Nonlinear Integer Programming Problems through Genetic Algorithms

Md. Abul Kalam Azad\textsuperscript{a,}\*, Masatoshi Sakawa\textsuperscript{b}, Kosuke Kato\textsuperscript{c} and Hideki Katagiri\textsuperscript{d}

\textsuperscript{a} Doctoral Student, Department of Artificial Complex Systems Engineering, Graduate School of Engineering, Hiroshima University, Japan
\textsuperscript{b} Professor, Department of Artificial Complex Systems Engineering, Graduate School of Engineering, Hiroshima University, Japan
\textsuperscript{c} Associate Professor, Department of Artificial Complex Systems Engineering, Graduate School of Engineering, Hiroshima University, Japan
\textsuperscript{d} Research Associate, Department of Artificial Complex Systems Engineering, Graduate School of Engineering, Hiroshima University, Japan

Accepted in July 2004
Available online

Abstract

In the present paper, we focus on two-level nonlinear integer programming problems (TLNLIPPs) in which there exists a decision maker (DM) with integer decision variables at the upper level and another DM with integer decision variables at the lower level. Various approaches for two-level programming problems could exist according to situations which the DMs are placed in. Assuming the cooperative relationship between the DM at the upper level and one at the lower level, in this paper, an interactive fuzzy programming technique through genetic algorithms for TLNLIPPs is proposed to obtain a satisfactory solution for the DMs. Furthermore, the feasibility of the proposed method is shown by applying it to an illustrative numerical example.

Keywords: Interactive fuzzy programming; Two-level programming; Nonlinear integer programming; Genetic algorithms

1. Introduction

In this paper, we consider two-level nonlinear integer programming problems in which there exists a decision maker (DM) with integer decision variables at the upper level and another DM with integer decision variables at the lower level. Various approaches for two-level programming problems could exist according to situations which the DMs are placed in. Under the assumption that these DMs do not have motivation to cooperate mutually, a Stackelberg solution (Shimizu, Ishizuka and Bard, 1997) 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 division of a company, it seems natural that these DMs cooperate with each other. Recently, Lai (1996) and Shih, Lai and Lee (1996) proposed a solution concept for two-level or multi-level linear programming problems such that decisions of DMs in all levels are sequential and all of the DMs essentially cooperate with each other. In their method, each of the DMs identifies a membership function of a fuzzy goal for its objective function. Furthermore, the DM at the upper level specifies those of fuzzy goals for decision variables. The DM at the lower level solves a fuzzy programming problem with constraints on fuzzy goals of the DM at the upper level.

Unfortunately, however, a final solution obtained by their method often becomes unnatural or undesirable because of inconsistency between the fuzzy goals of the objective function and the decision variables. To overcome this problem, eliminating the fuzzy goals for the decision variables, Sakawa et al. have proposed interactive fuzzy programming for multi-level linear programming problems (1998) and multi-level linear programming problems with fuzzy parameters (2000).

In general, a variety of actual decision making situations are formulated as large-scale mathematical programming problems involving integer decision variables, nonlinear objective functions and nonlinear constraint functions. Since a general solution method does not exist for such nonlinear integer programming problems like the branch and bound method for linear ones, a solution method peculiar to each problem has been proposed. As a general-purpose solution method for nonlinear integer programming problems, we propose the usage of genetic algorithms with double strings using continuous relaxation based on reference solution updating (GADSCRRSU), which is a direct extension of genetic algorithms with double strings based on reference solution updating (GADSRSU) for linear 0-1 programming problems (Sakawa and Kato, 2003). Under these circumstances, in this

* Corresponding Author: Doctoral Student, Department of Artificial Complex Systems Engineering, Graduate School of Engineering, Hiroshima University, 1-4-1 Kagamiyama, Higashi Hiroshima 739-8527, Japan, Tel. +81-82-424-7694, E-mail: azad@msl.sys.hiroshima-u.ac.jp
paper, interactive fuzzy programming through a proposed GADSCRRSU to derive a satisfactory solution for the DMs is presented for two-level nonlinear integer programming problems (TNLNLIPPs) as a first step to multi-level ones. Furthermore, the feasibility of the proposed method is shown through illustrative numerical example.

2. Problem Formulation

Two-level nonlinear integer programming problems are generally formulated as follows:

\[
\begin{align*}
\text{minimize} & \quad f_1(x_1, x_2) \\
\text{subject to} & \quad g_i(x_1, x_2) \leq 0, \quad i = 1, \ldots, m \\
& \quad x_j \in \{0, \ldots, v_j\}, \quad l = 1, 2, \quad j = 1, \ldots, n_j,
\end{align*}
\]

where \(x_1\) is an \(n_1\) dimensional integer decision variable column vector for the decision maker at the upper level (DM1), \(x_2\) is an \(n_2\) dimensional integer decision variable column vector for the decision maker at the lower level (DM2), and objective functions \(f_l(x_1, x_2), l = 1, 2\), constraint functions \(g_i(x_1, x_2), i = 1, \ldots, m\), may be linear or nonlinear. For notational convenience, let \(x = (x_1^T, x_2^T)^T\), and denote the feasible region of problem in Equation 1 by \(X\).

For example, consider a project selection problem in an administrative office at the upper level and several autonomous divisions of a company. In this case, the situation that all of the DMs can cooperate with each other seems natural rather than one that all the DMs do not have motivation to cooperate mutually.

Under the hypothesis of cooperation among all DMs, Sakawa et al. (1998) proposed the interactive fuzzy programming for multi-level linear programming problems in order to derive satisfying solutions for the DMs through interactions with the DM at the upper level by introducing fuzzy goals to consider the imprecise nature of DMs’ judgements for objective functions.

In this paper, focusing on the case of cooperative relation between DM1 at the upper level and DM2 at the lower level as in (Sakawa, Nishizaki and Uemura, 1998), we present the interactive fuzzy programming technique in order to derive a satisfactory solution for the DMs to the problem in Equation 1 through genetic algorithms.

3. Interactive Fuzzy Programming

The interactive fuzzy programming proposed by Sakawa et al. (1998) is summarized as follows.

3.1 Interactive Fuzzy Programming

Considering the ambiguity or fuzziness of the decision makers’ judgements on each of objective functions \(f_l(x_1, x_2), l = 1, 2\), in Equation 1, it seems natural to introduce such fuzzy goals for objective functions as “\(f_l(x_1, x_2)\) should be subjectively less than or equal to a certain value”. Let \(X\) denotes the feasible region of problem in Equation 1, the individual minimum

\[
f_l^{\min} = \min_{(x_1, x_2) \in X} f_l(x_1, x_2), \quad l = 1, 2
\]

and the individual maximum

\[
f_l^{\max} = \max_{(x_1, x_2) \in X} f_l(x_1, x_2), \quad l = 1, 2
\]

of each objective function are referred to when the DMs elicit membership functions prescribing the fuzzy goals for the objective functions \(f_l(x_1, x_2), l = 1, 2\). Since these problems are single-objective nonlinear integer programming problems, genetic algorithms with double strings using continuous relaxation based on reference solution updating (GADSCRSRU) is applicable, which is an extension of genetic algorithms with double strings based on reference solution updating (GADRSRU) for linear 0-1 programming problems (Sakawa and Kato, 2003).

The DMs determine the membership functions \(\mu_l(f_l(x_1, x_2)), l = 1, 2\), which are strictly monotone decreasing for \(f_l(x_1, x_2)\), consulting the variation ratio of degree of satisfaction in the interval between the individual minimum of problem in Equation 2 and the individual maximum of problem in Equation 3. The domain of the membership functions is in the interval \([f_l^{\min}, f_l^{\max}]\), \(l = 1, 2\), and the DM specifies the value \(f_l^0\) of the objective function for which the degree of satisfaction is 0 and the value \(f_l^1\) of the objective function for which the degree of satisfaction is 1. For the value undesired (larger) than \(f_l^0\), it is defined that \(\mu_l(f_l(x_1, x_2)) = 0\), and for the value desired (smaller) than \(f_l^1\), it is defined that \(\mu_l(f_l(x_1, x_2)) = 1\). Here a linear membership function in Figure 1 is considered, which characterizes the fuzzy goal of the DM at each level. The corresponding linear membership function \(\mu_l(f_l(x_1, x_2))\) is defined as:

\[
\mu_l(f_l(x_1, x_2)) = \begin{cases} 
1 & f_l(x_1, x_2) < f_l^0 \\
\frac{f_l^1 - f_l^0}{f_l^1 - f_l^0} & f_l^0 \leq f_l(x_1, x_2) < f_l^1 \\
0 & f_l(x_1, x_2) \geq f_l^1,
\end{cases}
\]

where \(f_l^0\) and \(f_l^1\) denote the value of the objective function \(f_l(x_1, x_2)\) such that the degree of membership function is 0 and 1, respectively, and it is assumed that the DMs subjectively assess \(f_l^0\) and \(f_l^1\).
Zimmermann (1978) proposed a method for determining the parameters \( f^0_l \) and \( f^1_l \) of the linear membership function in the following way. That is, using the individual minimum

\[
f^0_l = f(x^0_l, x^0_l) = \min_{(x_l, x_2) \in X} f_l(x_l, x_2), \quad l = 1, 2
\]

together with

\[
f^1_l = f(x^1_l, x^0_l), \quad l = 1, 2, \quad k = \begin{cases} 1 & \text{if } l = 2 \\ 2 & \text{if } l = 1, \end{cases}
\]

the DMs determine the linear membership functions as Equation 4 by choosing \( f^1_l = f^0_l, \quad f^0_l = f^1_l, \quad l = 1, 2 \).

Having elicited the membership functions \( \mu_1(f_l(x^1_l, x_2)) \) and \( \mu_2(f_l(x^1_l, x_2)) \) from both the DMs for the objective functions \( f_1(x^1_l, x_2) \) and \( f_2(x^1_l, x_2) \), the original two-level nonlinear integer programming problem in Equation 1 can be interpreted as the membership functions maximization problem defined by:

\[
\begin{align*}
\text{maximize (upper level)} & \quad \mu_1(f_1(x_l, x_2)) \\
\text{maximize (lower level)} & \quad \mu_2(f_2(x_l, x_2)) \\
\text{subject to} & \quad x_l \in \mathcal{X}.
\end{align*}
\]

Since the problem in Equation 7 is a membership functions maximization problem, in general a complete optimal solution that simultaneously maximizes both the DMs’ degree of satisfaction of their objective functions does not always exist when the objective functions conflict with each other. Thus, solution concept, called M-Pareto optimality (Sakawa, 2002) which is defined in terms of membership function is introduced in the problem in Equation 7.

For deriving an overall satisfactory solution to the formulated problem in Equation 7, first the maximizing decision of the fuzzy decision proposed by Bellman and Zadeh (1970) is found. Namely, the following problem is solved for obtaining a solution which maximizes the smaller degree of satisfaction between the two DMs:

\[
\begin{align*}
\text{maximize} & \quad \min \{ \mu_1(f_l(x_l, x_2)) \} \\
\text{subject to} & \quad g_l(x_l, x_2) \leq 0, \quad i = 1, \ldots, m \\
& \quad x_l \in \{0, \ldots, v_l^i\}, \quad l = 1, 2, \quad j = 1, \ldots, n_i.
\end{align*}
\]

The problem in Equation 8 can also be solved by GADSCRRSU.

By introducing the auxiliary variable \( \lambda \), this problem can be transformed into the following equivalent maximization problem:

\[
\begin{align*}
\text{maximize} & \quad \lambda \\
\text{subject to} & \quad \mu_1(f_l(x_l, x_2)) \geq \lambda, \quad l = 1, 2 \\
& \quad g_l(x_l, x_2) \leq 0, \quad i = 1, \ldots, m \\
& \quad x_l \in \{0, \ldots, v_l^i\}, \quad l = 1, 2, \quad j = 1, \ldots, n_i.
\end{align*}
\]

Assuming that a hypothetical decision maker DM2 at the lower level is on an equal position with DM1 at the upper level, the augmented maximin problem in Equation 10 instead of the maximin problem in Equation 8 is solved.

\[
\begin{align*}
\text{maximize} & \quad \min \left\{ \mu_1(f_l(x_l, x_2)) + \rho \sum_{k=1}^{2} \mu_k(f_k(x_l, x_2)) \right\} \\
\text{subject to} & \quad \mu_1(f_l(x_l, x_2)) \geq \lambda, \quad l = 1, 2 \\
& \quad g_l(x_l, x_2) \leq 0, \quad i = 1, \ldots, m \\
& \quad x_l \in \{0, \ldots, v_l^i\}, \quad l = 1, 2, \quad j = 1, \ldots, n_i.
\end{align*}
\]

The term augmented is adopted here because the term \( \rho \sum_{k=1}^{2} \mu_k(f_k(x_l, x_2)) \) is added to the standard maximin problem in Equation 8, where \( \rho \) is a sufficiently small positive number. This problem can also be solved by GADSCRRSU. Let \( x^* \) denotes an optimal solution to the problem in Equation 10. Then, we define the satisfactory degree of both the DMs under the constraints as

\[
\lambda = \min \{ \mu_1(f_1(x^*)), \mu_2(f_2(x^*)) \}.
\]

If DM1 is satisfied with the optimal solution \( x^* \), it follows that the optimal solution \( x^* \) becomes a satisfactory solution; however, DM1 is not always satisfied with the solution \( x^* \). It is quite natural to assume that DM1 would like to subjectively specify a minimal satisfactory level \( \delta \in [0, 1] \) for his membership function \( \mu_1(f_1(x, x_2)) \), taking into account a ratio of the satisfactory degree of DM2 to that of DM1 \( \Delta = \mu_2(f_2(x, x_2))/\mu_1(f_1(x, x_2)) \), the following problem in Equation 12 is solved.

\[
\begin{align*}
\text{maximize} & \quad \mu_2(f_2(x_l, x_2)) \\
\text{subject to} & \quad \mu_1(f_1(x_l, x_2)) \geq \delta \\
& \quad x_l \in \mathcal{X}
\end{align*}
\]

where DM2’s membership function is maximized under the condition that DM1’s membership function \( \mu_1(f_1(x_l, x_2)) \) is larger than or equal to the minimal satisfactory level \( \delta \) specified by DM1. It should be noted that GADSCRRSU is applicable to the problem in Equation 12.

At an iteration \( k \), let \( \mu_1(f^k_1), \mu_2(f^k_2), \lambda^k \) and \( \Delta^k = \mu_2(f^k_2)/\mu_1(f^k_1) \) respectively denote DM1’s and DM2’s satisfactory degrees, a satisfactory degree of both the DMs and the ratio of satisfactory degrees between the DMs, and let the corresponding optimal solution be \( x^k \). The interactive process terminates if the following two conditions are satisfied and DM1 concludes the solution as an overall satisfactory solution.
Termination conditions of the interactive process

**Condition 1**: DM1’s satisfactory degree is larger than or equal to the minimal satisfactory level \( \hat{\delta} \) specified by DM1, i.e., \( \mu_i(f_i^k) \geq \hat{\delta} \).

**Condition 2**: The ratio of \( \Delta^k \) of satisfactory degrees lies in the interval between the lower and the upper bounds specified by DM1, i.e., \( \Delta^k \in [\Delta_{\min}, \Delta_{\max}] \).

Condition 1 is DM1’s required condition for solutions, and condition 2 is provided in order to keep overall satisfactory balance between both levels. Unless these two conditions are satisfied simultaneously, DM1 needs to update the minimal satisfactory level \( \hat{\delta} \).

**Procedure for updating minimal satisfactory level \( \hat{\delta} \)**

**Case 1**: If condition 1 is not satisfied, then DM1 decreases the minimal satisfactory level \( \hat{\delta} \).

**Case 2**: If the ratio \( \Delta^k \) exceeds its upper bound, then DM1 increases the minimal satisfactory level \( \hat{\delta} \). Conversely, if \( \Delta^k \) is below its lower bound, then DM1 decreases the minimal satisfactory level \( \hat{\delta} \).

**Case 3**: Although conditions 1 and 2 are satisfied, if DM1 is not satisfied with the obtained solution and judges that it is desirable to increase the satisfactory degree of DM1 at the expense of the satisfactory degree of DM2, then DM1 increases the minimal satisfactory level \( \hat{\delta} \). Conversely, if DM1 judges that it is desirable to increase the satisfactory degree of DM2 at the expense of the satisfactory degree of DM1, then DM1 decreases the minimal satisfactory level \( \hat{\delta} \).

3.2 Algorithm of the Interactive Fuzzy Programming through GADSCRRSU

We are now ready to present an interactive algorithm for deriving an overall satisfactory solution to two-level nonlinear integer programming problem in Equation 1 through genetic algorithms with double strings using continuous relaxation based on reference solution updating (GADSCRRSU), which is summarized in the following.

**Step 1**: Solve the problems in Equation 5 through GADSCRRSU for individual minimum and by using Equation 6 calculate individual maximum of each objective function of both the DMs and ask the DMs to identify their membership functions \( \mu_i(f^k_i) \) of the fuzzy goals for their own objective functions.

**Step 2**: Set iteration \( k = 1 \) and solve the problem in Equation 10 through GADSCRRSU, in which a smaller degree between the satisfactory degrees of DM1 and DM2 is maximized. If DM1 is satisfied with the obtained optimal solution, the solution becomes a satisfactory solution and stop. Otherwise, ask DM1 to specify the minimal satisfactory level \( \hat{\delta} \) together with the lower and the upper bounds \([\Delta_{\min}, \Delta_{\max}]\) of the ratio of satisfactory degrees \( \Delta \).

**Step 3**: Solve the problem in Equation 12 through GADSCRRSU, in which the satisfactory degree of DM2 is maximized under the condition that the satisfactory degree of DM1 is larger than or equal to the minimal satisfactory level \( \hat{\delta} \) specified by DM1, and then propose an optimal solution \( \chi^k \) to the problem in Equation 12 to DM1 together with \( \mu_i(f^k_i), \mu_2(f^k_i), \) and \( \Delta^k \).

**Step 4**: If the solution proposed to DM1 satisfies the termination conditions and DM1 concludes the solution as a satisfactory solution, the algorithm stops.

**Step 5**: Ask DM1 to update the minimal satisfactory level \( \hat{\delta} \) in accordance with the procedure of updating minimal satisfactory level.

**Step 6**: Solve the problem in Equation 12 again and propose the obtained optimal solution to DM1 together with the related information. Return to step 4.

4. Genetic Algorithms with Double Strings using Continuous Relaxation based on Reference Solution Updating (GADSCRRSU)

In this section, we mention GADSCRRSU proposed as a general solution method for nonlinear integer programming problems defined as Equation 13.

**minimize** \( f(x) \)

**subject to** \( g_i(x) \leq 0, \quad i = 1, \ldots, m \)

\( x_j \in \{0, 1, \ldots, v_j\}, \quad j = 1, \ldots, n \) \hspace{1cm} (13)

In the problem in Equation 13, \( x \) is an \( n \) dimensional integer decision variable vector, \( f(x), g_i(x), i = 1, \ldots, m \), are nonlinear functions and \( v_j, j = 1, \ldots, n \), is the upper bound of each decision variable.

4.1 Individual Representation

The individual representation (Sakawa and Kato, 2003) by double strings shown in Figure 2 is adopted in GADSCRRSU.

In the figure, each of \( s(j), j = 1, \ldots, n \), is the index of an element in a solution vector and each of \( y_{s(j)} \in \{0, 1, \ldots, v_j\}, \quad j = 1, \ldots, n \), is the integer value of the element, respectively.

<table>
<thead>
<tr>
<th>Indices</th>
<th>( s(1) )</th>
<th>( s(2) )</th>
<th>\cdots</th>
<th>( s(j) )</th>
<th>\cdots</th>
<th>( s(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
<td>( y_{s(1)} )</td>
<td>( y_{s(2)} )</td>
<td>\cdots</td>
<td>( y_{s(j)} )</td>
<td>\cdots</td>
<td>( y_{s(n)} )</td>
</tr>
</tbody>
</table>

**Figure 2. Double Strings Representation**
4.2 Decoding Algorithm

Let \( N \) be the total number of population (\( pop\_size \)). The individuals \( x \) with the dimension \( n \) are generated randomly. Unfortunately, however, the direct mapping of the individuals can generate infeasible solutions. To eliminate such solutions, as in (Sakawa and Kato, 2003), a decoding algorithm of double strings for nonlinear integer programming problem using a reference solution \( x^0 \), which is a feasible solution and used as the origin of decoding, is constructed as follows.

Decoding algorithm using a reference solution

In the algorithm, it is assumed that a feasible solution \( x^0 \) is obtained in advance. Let \( n \) and \( N \) be the number of variables and the number of individuals in the population, respectively.

**Step 1:** If the index of an individual to be decoded is in \([1, \ldots, [N/2]]\), go to step 2. Otherwise, go to step 8.

**Step 2:** Let \( j := 1, x := [0, \ldots, 0], l := 1 \).

**Step 3:** Let \( x_{a(0)} := y_{a(0)} \).

**Step 4:** If \( g_{i}(x) \leq 0, i = 1, \ldots, m, \) let \( l := j, j := j+1, \) and go to step 5. Otherwise, let \( j := j+1, \) and go to step 5.

**Step 5:** If \( j \leq n \), go to step 3. Otherwise, go to step 6.

**Step 6:** If \( l > 0 \), go to step 7. Otherwise, go to step 8.

**Step 7:** By substituting \( x_{a(l)} := y_{a(l)} \), \( 1 \leq j \leq l \) and \( x_{a(0)} := 0 \), \( l < j \leq n \), we obtain a feasible solution \( x \) corresponding to the individual \( s \) and stop.

**Step 8:** Let \( j := 1, x := x^0 \).

**Step 9:** Let \( x_{a(0)} := y_{a(0)} \). If \( y_{a(j)} = x_{a(j)}^0 \), let \( j := j+1 \), and go to step 11. If \( y_{a(j)} \neq x_{a(j)}^0 \), go to step 10.

**Step 10:** If \( g_{i}(x) \leq 0, i = 1, \ldots, m, \) let \( j := j+1, \) and go to step 11. Otherwise, let \( x_{a(j)} := x_{a(j)}^0, j := j+1, \) and go to step 11.

**Step 11:** If \( j \leq n \), go to step 9. Otherwise, we obtain a feasible solution \( x \) corresponding to the individual \( s \) and stop.

This decoding algorithm enables us to decode each of the individuals represented by the double strings to the corresponding feasible solution. However, the diversity of the solution \( x \) greatly depends on the reference solution, because solutions obtained by the decoding algorithm using reference solution tend to concentrate around the reference solution. To overcome such situations, the reference solution updating procedure (Sakawa and Kato, 2003) is adopted here.

4.3 Fitness

Nature obeys the principle of Darwinian “survival of the fittest”; the individuals with high fitness values will, on average, reproduce more often than those low fitness values.

For obtaining satisfactory solution for both the DMs to two-level nonlinear integer programming problem in Equation 1 through GADSCRRSU, the objective function value is used as the fitness value \( f \) of an individual \( s \). When the variance of fitness in a population is small, it is often observed that the ordinary roulette wheel selection does not work well because there is little difference between the probability of a good individual surviving and that of a bad one surviving (Sakawa and Kato, 2003). In order to overcome this problem, the linear scaling (Sakawa and Kato, 2003) is adopted here. The fitness \( f(s) \) of the DM at the upper level and the fitness \( f_d(s) \) of the DM at the lower level are obtained by using the following linear scaling

\[
f(s) = a_i f(s) + b_i
\]

where \( f(s), l = 1, 2, \) is the fitness value of the DM at each level with respect to each decoded individual \( s \).

4.4 Genetic Operators

For obtaining satisfactory solution for both the DMs to two-level nonlinear integer programming problem in Equation 1 through GADSCRRSU, four genetic operators such as reproduction, partially matched crossover (PMX), bit reverse mutation and inversion (Sakawa and Kato, 2003) are adopted here.

4.5 Usage of Continuous Relaxation

In order to find an approximate optimal solution with high accuracy in reasonable time, we need some schemes such as the restriction of the search space to a promising region, the generation of individuals near the optimal solution and so forth. From the point of view, the information about an optimal solution to the corresponding continuous relaxation problem

\[
\text{minimize } f(x) \\
\text{subject to } g_i(x) \leq 0, i = 1, \ldots, m \\
0 \leq x_j \leq v_j, j = 1, \ldots, n
\]

is used in the generation of the initial population and the bit reverse mutation. When this problem is convex, we can obtain a global optimal solution by some convex programming technique, e.g., the sequential quadratic programming. Otherwise, i.e., when it is non-convex, because it is difficult to find a global optimal solution, we search an approximate optimal solution by some approximate solution method such as genetic algorithms or simulated annealing. Here GENOCOP V is used to find the solution of corresponding continuous relaxation problem in Equation 15.

4.6 Computational Procedures of GADSCRRSU

Now the genetic algorithms with double strings using continuous relaxation based on reference solution updating
(GADSCRRSU) for solving nonlinear integer programming problems in Equation 13 are summarized in the following.

**Step 0:** Determine values of the parameters used in GADSCRRSU: the population size \( N \), the minimal search generation \( I_{\text{min}} \), the maximal search generation \( I_{\text{max}} \), the convergence criterion \( \varepsilon \), the degree of use of information about solutions to nonlinear programming relaxation problems \( R \), the parameter for feasible solution \( \theta \), the scaling constant \( c_{\text{mult}} \), the parameter for reference solution updating \( \eta \), the generation gap \( G \), the probability of crossover \( p_c \), the probability of mutation \( p_m \), the probability of inversion \( p_i \), and set the generation counter \( t \) at 0.

**Step 1:** Generate the initial population consisting of \( N \) individuals based on the information of a solution to the continuous relaxation problem in Equation 15.

**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 the reproduction operator based on the elitist expected value selection, after carrying out linear scaling.

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

**Step 6:** Apply the mutation operator based on the information of an optimal solution to the continuous relaxation problem in Equation 15.

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

5. Numerical Example

For obtaining satisfactory solution for both the DMs, we have considered the following two-level nonlinear integer programming problem to test the proposed algorithm.

\[
\begin{align*}
\text{maximize} & \quad f_1(x_1, x_2) = \prod_{j=1}^n \left[ 1 \left(1 - (1-r_j)^{V_j} \right) \right] \\
\text{minimize} & \quad f_2(x_1, x_2) = \sum_{j=1}^n q_j \left[ x_j + \exp \left( \frac{x_j}{4} \right) \right] \\
\text{subject to} & \quad g_1(x_1, x_2) = \sum_{j=1}^n p_j x_j - P \leq 0 \\
& \quad g_2(x_1, x_2) = \sum_{j=1}^n w_j x_j \exp \left( \frac{x_j}{4} \right) - W \leq 0 \\
& \quad x_j \in \{1, 2, \ldots, 10\}, \quad j = 1, \ldots, n
\end{align*}
\]

The parameter values used in GADSCRRSU for solving two-level nonlinear integer programming problem in Equation 16 were set as follows: \( N = 100, I_{\text{min}} = 100, I_{\text{max}} = 1000, \varepsilon = 0.02, R = 0.9, \theta = 5.0, c_{\text{mult}} = 1.8, \eta = 0.1, G = 0.9, p_c = 0.9, p_m = 0.05, p_i = 0.03, P = 50, \sigma = 2.0, \tau = 3.0, \rho = 0.005, \) and \( \nu = 10 \). Several problems with different numbers of variables were considered to test the proposed algorithm for solving the problem in Equation 16. The data were generated randomly. In the following section the result has been discussed briefly for a problem consists number of variables 10 and the data are shown in Table 1.

5.1 Result and Discussion

At first, by using Zimmermann method, the individual minimum and maximum of each objective function of both the DMs are calculated and are shown in Table 2. After calculating individual minimum and maximum of each objective function of both the DMs, the corresponding membership functions are specified subjectively by the DMs.

Since DM1’s objective function is a maximization type, the linear membership function in Equation 17 in Figure 3 is used to specify the fuzzy goal of the DM1.

\[ \mu_i(f_i(x_1, x_2)) = \begin{cases} 0 & f_i(x_1, x_2) < f_i^0 \\ \frac{f_i(x_1, x_2) - f_i^0}{f_i^0 - f_i^\text{min}_i} & f_i^\text{min}_i \leq f_i(x_1, x_2) < f_i^0 \\ 1 & f_i(x_1, x_2) \geq f_i^0 \end{cases} \]

On the other hand, since DM2’s objective function is a minimization type, the linear membership function in Equation 4 in Figure 1 is used to specify the fuzzy goal of the DM2. The corresponding linear membership functions of both the DMs are shown in Table 3.

After specifying the linear membership functions, the two-level nonlinear integer programming problem in Equation 16 was solved interactively. The best results after each iteration are summarized in Table 4. In each iteration consecutive 10 trials were performed. In the numerical experiment after the fourth iteration, termination conditions were satisfied for the problem in Equation 16 with number of variables 10.

At the first iteration, the problem in Equation 16 is solved by using the problem in Equation 10. Here, if the DM1 is satisfied with this solution then the solution becomes satisfactory solution for both the DMs. However, DM1 is not always satisfied with this solution. Then he subjectively specifies his minimal satisfactory level of his membership function \( \delta \in [0, 1] \) and also specifies \( \Delta_{\text{min}} \) and \( \Delta_{\text{max}} \in [0, 1] \). Suppose that DM1 would like to subjectively specify a minimal satisfactory level \( \delta = 0.75 \) and the lower and upper bound of \( \Delta \) i. e., \( \Delta_{\text{min}} = 0.65 \), and \( \Delta_{\text{max}} = 0.80 \). At the second iteration, the problem is solved by using Equation 12 where DM2’s membership function is maximized under the DM1’s given conditions. Here the second termination condition is not satisfied so DM1 needs to update his minimal satisfactory level according the procedure for updating the minimal satisfactory level. At
Table 1. Data for Problem of Equation 16

<table>
<thead>
<tr>
<th>Number of Variables</th>
<th>Co-efficient Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>r = 0.680, 0.749, 0.630, 0.606, 0.770, 0.774, 0.836, 0.644, 0.876, 0.612</td>
</tr>
<tr>
<td></td>
<td>q = 10.000, 7.000, 0.110, 0.130, 13.000, 1.000, 1.200, 0.030, 0.140, 0.900</td>
</tr>
<tr>
<td></td>
<td>p = 1.000, 4.000, 1.000, 1.000, 2.000, 2.000, 5.000, 3.000, 2.000, 4.000</td>
</tr>
<tr>
<td></td>
<td>w = 1.000, 5.000, 0.600, 9.000, 0.300, 0.700, 0.800, 0.300, 1.000, 0.900</td>
</tr>
<tr>
<td></td>
<td>P = 481.000</td>
</tr>
<tr>
<td></td>
<td>W = 277.262</td>
</tr>
</tbody>
</table>

Table 2. Calculated Individual Minimum and Maximum

<table>
<thead>
<tr>
<th>Number of Variables</th>
<th>Objective</th>
<th>( f_1^{\text{max}} )</th>
<th>( f_1^{\text{min}} )</th>
<th>Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>( f_1(x_1, x_2) )</td>
<td>0.0334</td>
<td>0.9468</td>
<td>39.33</td>
</tr>
<tr>
<td></td>
<td>( f_2(x_1, x_2) )</td>
<td>76.5377</td>
<td>235.4866</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3. Linear Membership Function for DM1

Table 3. DMs’ Linear Membership Functions

<table>
<thead>
<tr>
<th>Decision Maker</th>
<th>( f_1^0 )</th>
<th>( f_1^1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>DM1</td>
<td>0.03</td>
<td>0.95</td>
</tr>
<tr>
<td>DM2</td>
<td>250.00</td>
<td>75.00</td>
</tr>
</tbody>
</table>

Figure 4. Total Computational Time

the fourth iteration all the termination conditions are satisfied and hence the interactive process gives a satisfactory solution for both the DMs and terminated.

We considered a test problem in Equation 16 to test the feasibility of the proposed interactive fuzzy programming problem for a two-level nonlinear integer programming problem through genetic algorithms with different numbers of variables to find the computational time. Figure 4 shows the total computational time to solve the problem in Equation 16 with different numbers of variables. From the figure it seems that the computational time increases polynomially as the number of variables increases in a given test problem.

6. Conclusion

In this paper, focusing on two-level nonlinear integer programming problems, an interactive fuzzy programming procedure for them through genetic algorithms with double strings using continuous relaxation based on reference solution updating (GADSCRRSU) is presented. Furthermore, the feasibility of the proposed method is shown by applying it to an illustrative numerical example with different numbers of variables considered. The extension of the proposed method to multi-level nonlinear integer programming problems will be desirable in the future.

References


