Hindawi Publishing Corporation
Advances in Operations Research
Volume 2009, Article ID 372548, 17 pages
doi:10.1155/2009/372548

Research Article

An Interactive Fuzzy Satisficing Method for Multiobjective Nonlinear Integer Programming Problems with Block-Angular Structures through Genetic Algorithms with Decomposition Procedures

Masatoshi Sakawa and Kosuke Kato

Department of Artificial Complex Systems Engineering, Graduate School of Engineering, Hiroshima University, Higashi-Hiroshima 739-8527, Japan

Correspondence should be addressed to Masatoshi Sakawa, sakawa@hiroshima-u.ac.jp

Received 19 August 2008; Revised 27 March 2009; Accepted 29 May 2009

Recommended by Walter J. Gutjahr

We focus on multiobjective nonlinear integer programming problems with block-angular structures which are often seen as a mathematical model of large-scale discrete systems optimization. By considering the vague nature of the decision maker's judgments, fuzzy goals of the decision maker are introduced, and the problem is interpreted as maximizing an overall degree of satisfaction with the multiple fuzzy goals. For deriving a satisficing solution for the decision maker, we develop an interactive fuzzy satisficing method. Realizing the block-angular structures that can be exploited in solving problems, we also propose genetic algorithms with decomposition procedures. Illustrative numerical examples are provided to demonstrate the feasibility and efficiency of the proposed method.

1. Introduction

Genetic algorithms (GAs) [1], initiated by Holland, his colleagues, and his students at the University of Michigan in the 1970s, as stochastic search techniques based on the mechanism of natural selection and natural genetics, have received a great deal of attention regarding their potential as optimization techniques for solving discrete optimization problems or other hard optimization problems. Although genetic algorithms were not much known at the beginning, after the publication of Goldberg's book [2], genetic algorithms have recently attracted considerable attention in a number of fields as a methodology for optimization, adaptation, and learning. As we look at recent applications of genetic algorithms to

optimization problems, especially to various kinds of single-objective discrete optimization problems and/or to other hard optimization problems, we can see continuing advances [3–13].

Sakawa et al. proposed genetic algorithms with double strings (GADS) [14] for obtaining an approximate optimal solution to multiobjective multidimensional 0-1 knapsack problems. They also proposed genetic algorithms with double strings based on reference solution updating (GADSRSU) [15] for multiobjective general 0-1 programming problems involving both positive coefficients and negative ones. Furthermore, they proposed genetic algorithms with double strings using linear programming relaxation (GADSLPR) [16] for multiobjective multidimensional integer knapsack problems and genetic algorithms with double strings using linear programming relaxation based on reference solution updating (GADSLPRRSU) for linear integer programming problems [17]. Observing that some solution methods for specialized types of nonlinear integer programming problems have been proposed [18–23], as an approximate solution method for general nonlinear integer programming problems, Sakawa et al. [24] proposed genetic algorithms with double strings using continuous relaxation based on reference solution updating (GADSCRRSU).

In general, however, actual decision making problems formulated as mathematical programming problems involve very large numbers of variables and constraints. Most of such large-scale problems in the real world often have special structures that can be exploited in solving problems. One familiar special structure is the block-angular structure to the constraints and several kinds of decomposition methods for linear and nonlinear programming problems with block-angular structure have been proposed [25]. Unfortunately, however, for large-scale problems with discrete variables, it seems quite difficult to develop an efficient solution method for obtaining an exact optimal solution. For multidimensional 0-1 knapsack problems with block-angular structures, by utilizing the block-angular structures that can be exploited in solving problems, Sakawa et al. [9, 26] proposed genetic algorithms with decomposition procedures (GADPs). For dealing with multidimensional 0-1 knapsack problems with block angular structures, using triple string representation, Sakawa et al. [9, 26] presented genetic algorithms with decomposition procedures. Furthermore, by incorporating the fuzzy goals of the decision maker, they [9] also proposed an interactive fuzzy satisficing method for multiobjective multidimensional 0-1 knapsack problems with block angular structures.

Under these circumstances, in this paper, as a typical mathematical model of large-scale multiobjective discrete systems optimization, we consider multiobjective nonlinear integer programming problems with block-angular structures. By considering the vague nature of the decision maker's judgments, fuzzy goals of the decision maker are introduced, and the problem is interpreted as maximizing an overall degree of satisfaction with the multiple fuzzy goals. For deriving a satisficing solution for the decision maker, we develop an interactive fuzzy satisficing method. Realizing the block-angular structures that can be exploited in solving problems, we also propose genetic algorithms with decomposition procedures for nonlinear integer programming problems with block-angular structures.

The paper is organized as follows. Section 2 formulates multiobjective nonlinear integer programming problems with block-angular structures. Section 3 develops an interactive fuzzy satisficing method for deriving a satisficing solution for the decision maker. Section 4 proposes GADPCRRSU as an approximate solution method for nonlinear integer programming problems with block-angular structures. Section 5 provides illustrative numerical examples to demonstrate the feasibility and efficiency of the proposed method. Finally the conclusions are considered in Section 6 and the references.

2. Problem Formulation

Consider multiobjective nonlinear integer programming problems with block-angular structures formulated as

$$\begin{aligned}
&\text{minimize} \quad f_l(\mathbf{x}) = f_l(\mathbf{x}^1, \ldots, \mathbf{x}^P), \quad l = 1, 2, \ldots, k \\
&\text{subject to} \quad \mathbf{g}(\mathbf{x}) = \mathbf{g}(\mathbf{x}^1, \ldots, \mathbf{x}^P) \leq \mathbf{0} \\
&\qquad\qquad \mathbf{h}^1(\mathbf{x}^1) \leq \mathbf{0} \\
&\qquad\qquad \ddots \quad \vdots \\
&\qquad\qquad \mathbf{h}^P(\mathbf{x}^P) \leq \mathbf{0} \\
&x_j^J \in \left\{0, 1, \ldots, V_j^J\right\}, \quad J = 1, 2, \ldots, P, \; j = 1, 2, \ldots, n_J,
\end{aligned} \tag{2.1}$$

where $\mathbf{x}^J$, $J = 1, 2, \ldots, P$, are n_J dimensional integer decision variable column vectors and $\mathbf{x} = ((\mathbf{x}^1)^T, \ldots, (\mathbf{x}^P)^T)^T$. The constraints $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_{m_0}(\mathbf{x}))^T \leq \mathbf{0}$ are called as coupling constraints with m_0 dimension, while each of constraints $\mathbf{h}^J(\mathbf{x}^J) = (h_1^J(\mathbf{x}^1), \ldots, h_{m_J}^J(\mathbf{x}^J))^T \leq \mathbf{0}$, $J = 1, 2, \ldots, P$, is called as block constraints with m_J dimension. In (2.1), it is assumed that $f_l(\cdot)$, $g_i(\cdot)$, $h_i^J(\cdot)$ are general nonlinear functions. The positive integers V_j^J, $J = 1, 2, \ldots, P$, $j = 1, 2, \ldots, n_J$, represent upper bounds for x_j^J. In the following, for notational convenience, the feasible region of (2.1) is denoted by X.

As an example of nonlinear integer programming problems with block-angular structures in practical applications, Bretthauer et al. [27] formulated health care capacity planning, resource constrained production planning, and portfolio optimization with industry constraints.

3. An Interactive Fuzzy Satisficing Method

In order to consider the vague nature of the decision maker's judgments for each objective function in (2.1), if we introduce the fuzzy goals such as "$f_l(\mathbf{x})$ should be substantially less than or equal to a certain value," (2.1) can be rewritten as

$$\underset{\mathbf{x} \in X}{\text{maximize}} \; (\mu_1(f_1(\mathbf{x})), \ldots, \mu_k(f_k(\mathbf{x}))), \tag{3.1}$$

where $\mu_l(\cdot)$ is the membership function to quantify the fuzzy goal for the lth objective function in (2.1). To be more specific, if the decision maker feels that $f_l(\mathbf{x})$ should be less than or equal to at least f_l^0 and $f_l(\mathbf{x}) \leq f_l^1 (\leq f_l^0)$ is satisfactory, the shape of a typical membership function is shown in Figure 1.

Since (3.1) is regarded as a fuzzy multiobjective optimization problem, a complete optimal solution that simultaneously minimizes all of the multiple objective functions

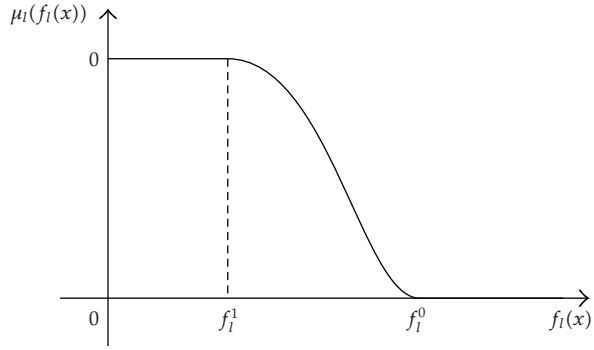

Figure 1: An example of membership functions.

does not always exist when the objective functions conflict with each other. Thus, instead of a complete optimal solution, as a natural extension of the Pareto optimality concept for ordinary multiobjective programming problems, Sakawa et al. [28, 29] introduced the concept of M-Pareto optimal solutions which is defined in terms of membership functions instead of objective functions, where M refers to membership.

Definition 3.1 (M-Pareto optimality). A feasible solution $\mathbf{x}^* \in X$ is said to be M-Pareto optimal to a fuzzy multiobjective optimization problem if and only if there does not exist another feasible solution $\mathbf{x} \in X$ such as $\mu_l(f_l(\mathbf{x})) \geq \mu_l(f_l(\mathbf{x}^*))$, $l = 1, 2, \ldots, k$, and $\mu_j(f_j(\mathbf{x})) > \mu_j(f_j(\mathbf{x}^*))$ for at least one $j \in \{1, 2, \ldots, k\}$.

Introducing an aggregation function $\mu_D(\mathbf{x})$ for k membership functions in (3.1), the problem can be rewritten as

$$\underset{\mathbf{x}\in X}{\text{maximize}}\ \mu_D(\mathbf{x}), \tag{3.2}$$

where the aggregation function $\mu_D(\cdot)$ represents the degree of satisfaction or preference of the decision maker for the whole of k fuzzy goals. In the conventional fuzzy approaches, it has been implicitly assumed that the minimum operator is the proper representation of the decision maker's fuzzy preferences. However, it should be emphasized here that this approach is preferable only when the decision maker feels that the minimum operator is appropriate. In other words, in general decision situations, the decision maker does not always use the minimum operator when combining the fuzzy goals and/or constraints. Probably the most crucial problem in (3.2) is the identification of an appropriate aggregation function which well represents the decision maker's fuzzy preferences. If $\mu_D(\cdot)$ can be explicitly identified, then (3.2) reduces to a standard mathematical programming problem. However, this rarely happens, and as an alternative, an interaction with the decision maker is necessary to find a satisficing solution for (3.1).

In order to generate candidates of a satisficing solution which are M-Pareto optimal, the decision maker is asked to specify the aspiration levels of achievement for all membership functions, called reference membership levels. For reference membership levels given by the decision maker $\overline{\mu}_l$, $l = 1, 2, \ldots, k$, the corresponding M-Pareto optimal solution to $\overline{\boldsymbol{\mu}}$, which is the nearest to the requirements in the minimax sense or better than that if the reference

membership levels are attainable, is obtained by solving the following augmented minimax problem:

$$\underset{\mathbf{x}\in X}{\text{minimize}} \max_{l=1,2,\ldots,k} \left\{ \left(\overline{\mu}_l - \mu_l\left(f_l(\mathbf{x})\right)\right) + \rho \sum_{j=1}^{k} \left(\overline{\mu}_j - \mu_j\left(f_j(\mathbf{x})\right)\right) \right\}, \tag{3.3}$$

where ρ is a sufficiently small positive real number.

We can now construct an interactive algorithm in order to derive a satisficing solution for the decision maker from among the M-Pareto optimal solution set. The procedure of the interactive fuzzy satisficing method is summarized as follows.

3.1. An Interactive Fuzzy Satisficing Method

Step 1. Calculate the individual minimum and maximum of each objective function under the given constraints by solving the following problems:

$$\begin{aligned} &\underset{\mathbf{x}\in X}{\text{minimize}}\ f_l(\mathbf{x}), \quad l = 1, 2, \ldots, k, \\ &\underset{\mathbf{x}\in X}{\text{maximize}}\ f_l(\mathbf{x}), \quad l = 1, 2, \ldots, k. \end{aligned} \tag{3.4}$$

Step 2. By considering the individual minimum and maximum of each objective function, the decision maker subjectively specifies membership functions $\mu_l(f_l(\mathbf{x}))$, $l = 1, 2, \ldots, k$, to quantify fuzzy goals for objective functions.

Step 3. The decision maker sets initial reference membership levels $\overline{\mu}_l$, $l = 1, 2, \ldots, k$.

Step 4. For the current reference membership levels, solve the augmented minimax problem (3.3) to obtain the M-Pareto optimal solution and the membership function value.

Step 5. If the decision maker is satisfied with the current levels of the M-Pareto optimal solution, stop. Then the current M-Pareto optimal solution is the satisficing solution of the decision maker. Otherwise, ask the decision maker to update the current reference membership levels $\overline{\mu}_l$, $l = 1, 2, \ldots, k$, by considering the current values of the membership functions and return to Step 4.

In the interactive fuzzy satisficing method, it is required to solve nonlinear integer programming problems with block-angular structures (3.3) together with (3.4). It is significant to note that these problems are single objective integer programming problems with block-angular structures. Realizing this difficulty, in the next section, we propose genetic algorithms with decomposition procedures using continuous relaxation based on reference solution updating (GADPCRRSU).

4. Genetic Algorithms with Decomposition Procedures

As discussed above, in this section, we propose genetic algorithms with decomposition procedures using continuous relaxation based on reference solution updating (GADPCRRSU)

$\nu(1)$	$\nu(2)$	$\cdots$	$\nu(n)$
$y_{\nu(1)}$	$y_{\nu(2)}$	$\cdots$	$y_{\nu(n)}$

Figure 2: Double string.

as an approximate solution method for nonlinear integer programming problems with block-angular structures.

Consider single-objective nonlinear integer programming problems with block-angular structures formulated as

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) = f\left(\mathbf{x}^1, \ldots, \mathbf{x}^P\right) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}) = \mathbf{g}\left(\mathbf{x}^1, \ldots, \mathbf{x}^P\right) \leq 0 \\ & \mathbf{h}^1\left(\mathbf{x}^1\right) \leq 0 \\ & \ddots \quad \vdots \\ & \mathbf{h}^P\left(\mathbf{x}^P\right) \leq 0 \\ & x_j^J \in \left\{0, 1, \ldots, V_j^J\right\}, \quad J = 1, 2, \ldots, P, \ j = 1, 2, \ldots, n_J. \end{aligned} \tag{4.1}$$

Observe that this problem can be viewed as a single-objective version of the original problem (2.1).

Sakawa et al. [24] have already studied genetic algorithms with double strings using continuous relaxation based on reference solution updating (GADSCRRSU) for ordinary nonlinear integer programming problems formulated as

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \ldots, m, \\ & x_j \in \{0, 1, \ldots, V_j\}, \quad j = 1, 2, \ldots, n, \end{aligned} \tag{4.2}$$

where an individual is represented by a double string. In a double string as is shown in Figure 2, for a certain j, $\nu(j) \in \{1, 2, \ldots, n\}$ represents an index of a variable in the solution space, while $y_{\nu(j)}$, $j = 1, 2, \ldots, n$, does the value among $\{0, 1, \ldots, V_{\nu(j)}\}$ of the $\nu(j)$th variable $x_{\nu(j)}$.

In view of the block-angular structure of (4.1), it seems to be quite reasonable to define an individual $\mathbf{S}$ as an aggregation of P subindividuals $\mathbf{s}^J$, $J = 1, 2, \ldots, P$, corresponding to the block constraint $\mathbf{h}^J(\mathbf{x}^J) \leq 0$ as shown in Figure 3.

If these subindividuals are represented by double strings, for each of subindividuals $\mathbf{s}^J$, $J = 1, 2, \ldots, P$, a phenotype (subsolution) satisfying each of the block constraints can be obtained by the decoding algorithm in GADSCRRSU.

Unfortunately, however, the simple combination of these subsolutions does not always satisfy the coupling constraints g(x) $\leq$ 0. To cope with this problem, a triple string representation as shown in Figure 4 and the corresponding decoding algorithm are presented as an extension of the double string representation and the corresponding decoding

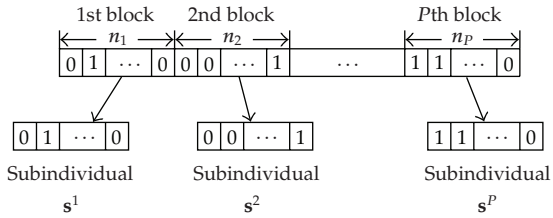

Figure 3: Division of an individual into P subindividuals.

$\mathbf{S}^J =$

r^J			
$v^J(1)$	$v^J(2)$	$\cdots$	$v^J(nJ)$
$y^J_{v^J(1)}$	$y^J_{v^J(2)}$	$\cdots$	$y^J_{v^J(nJ)}$

Figure 4: Triple string.

algorithm. By using the proposed representation and decoding algorithm, a phenotype (solution) satisfying both the block constraints and coupling constraints can be obtained for each individual $\mathbf{S} = (\mathbf{s}^1, \mathbf{s}^2, \ldots, \mathbf{s}^P)$.

To be more specific, in a triple string which represents a subindividual corresponding to the Jth block, $r^J \in \{1, 2, \ldots, P\}$ represents the priority of the Jth block, each $v^J(j) \in \{1, 2, \ldots, n_J\}$ is an index of a variable in phenotype and each $y^J_{v^J(j)}$ takes an integer value among $\{0, 1, \ldots, V^J_{v^J(j)}\}$. As in GADSCRRSU, a feasible solution, called a reference solution, is necessary for decoding of triple strings. In our proposed GADPCRRSU, the reference solution is obtained as a solution $\mathbf{x}^*$ to a minimization problem of constraint violation. In the following, we summarize the decoding algorithm for triple strings using a reference solution $\mathbf{x}^*$, where N is the number of individuals and I is a counter for the individual number.

4.1. Decoding Algorithm for Triple String

Step 1. Let $I := 1$.

Step 2. If $1 \leq I \leq \lfloor N/2 \rfloor$, go to Step 3. Otherwise, go to Step 11.

Step 3. Let $\mathbf{x} := \mathbf{0}$, $r := 1$, $L := 0$.

Step 4. Find $J \in \{1, 2, \ldots, P\}$ such that $r^J = r$. Let $j := 1$, $l := 0$.

Step 5. Let $x^J_{v^J(j)} := y^J_{v^J(j)}$.

Step 6. If $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ and $\mathbf{h}^J(\mathbf{x}^J) \leq \mathbf{0}$, let $L := r$, $l := j$, $j := j + 1$ and go to Step 7. Otherwise, let $j := j + 1$ and go to Step 7.

Step 7. If $j > n_J$, let $r := r + 1$ and go to Step 8. Otherwise, go to Step 5.

Step 8. If $r > P$, go to Step 9. Otherwise, go to Step 4.

Step 9. If $L = 0$ and $l = 0$, go to Step 11. Otherwise, go to Step 10.

Step 10. Find $J(r)$ such that $r^{J(r)} = r$ for $r = 1, \ldots, L-1$. Then, let $x^{J(r)}_{\nu^{J(r)}(j)} := y^{J(r)}_{\nu^{J(r)}(j)}$, $j = 1, 2, \ldots, n_{J(r)}$. Furthermore, find $J(L)$ such that $r^{J(L)} = L$ and let $x^{J(L)}_{\nu^{J(L)}(j)} := y^{J(L)}_{\nu^{J(L)}(j)}$, $j = 1, 2, \ldots, l$. The remainder elements of $\mathbf{x}$ are set to 0. Terminate the decoding process.

Step 11. Let $\mathbf{x} := \mathbf{x}^*$, $r := 1$ and go to Step 12.

Step 12. Find $J \in \{1, 2, \ldots, P\}$ such that $r^J = r$ and let $j := 1$.

Step 13. Let $x^J_{\nu^J(j)} := y^J_{\nu^J(j)}$. If $y^J_{\nu^J(j)} = x^{*J}_{\nu^J(j)}$, let $j := j + 1$ and go to Step 15. If $y^J_{\nu^J(j)} \neq x^{*J}_{\nu^J(j)}$, go to Step 14.

Step 14. If $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ and $\mathbf{h}^J(\mathbf{x}^J) \leq \mathbf{0}$, let $j := j + 1$ and go to Step 15. Otherwise, let $x^J_{\nu^J(j)} := x^{*J}_{\nu^J(j)}$, $j := j + 1$ and go to Step 15.

Step 15. If $j \leq n_J$, go to Step 13. Otherwise, let $r := r + 1$ and go to Step 16.

Step 16. If $r \leq P$, go to Step 12. Otherwise, $I := I + 1$ and go to Step 17.

Step 17. If $I \leq N$, go to Step 2. Otherwise, terminate the decoding process.

It is expected that an optimal solution to the continuous relaxation problem becomes a good approximate optimal solution of the original nonlinear integer programming problem. In the proposed method, after obtaining an (approximate) optimal solution $\hat{x}^J_j$, $J = 1, 2, \ldots, P$, $j = 1, 2, \ldots, n_J$ to the continuous relaxation problem, we suppose that each decision variable x^J_j takes exactly or approximately the same value that $\hat{x}^J_j$ does. In particular, decision variables x^J_j such as $\hat{x}^J_j = 0$ are very likely to be equal to 0.

To be more specific, the information of the (approximate) optimal solution $\hat{\mathbf{x}}$ to the continuous relaxation problem of (4.1) is used when generating the initial population and performing mutation. In order to generate the initial population, when we determine the value of each $y^J_{\nu^J(j)}$ in the lowest row of a triple string, we use a Gaussian random variable with mean $\hat{x}^J_{\nu^J(j)}$ and variance σ. In mutation, when we change the value of $y^J_{\nu^J(j)}$ for some (J, j), we also use a Gaussian random variable with mean $\hat{x}^J_{\nu^J(j)}$ and variance τ.

Various kinds of reproduction methods have been proposed. Among them, Sakawa et al. [14] investigated the performance of each of six reproduction operators, that is, ranking selection, elitist ranking selection, expected value selection, elitist expected value selection, roulette wheel selection, and elitist roulette wheel selection, and as a result confirmed that elitist expected value selection is relatively efficient for multiobjective 0-1 programming problems incorporating the fuzzy goals of the decision maker. Thereby, the elitist expected value selection—elitism and expected value selection combined together—is adopted. Here, elitism and expected value selection are summarized as follows.

Elitism

If the fitness of an individual in the past populations is larger than that of every individual in the current population, preserve this string into the current generation.

Expected Value Selection

For a population consisting of N individuals, the expected number of each $\mathbf{s}_n^J$, $J = 1, 2, \ldots, P$, each subindividual of the nth individual $\mathbf{S}_n$, in the next population, is given by

$$N_n = \frac{f(\mathbf{S}_n)}{\sum_{n=1}^{N} f(\mathbf{S}_n)} \times N. \tag{4.3}$$

Then, the integral part of $N_n (= \lfloor N_n \rfloor)$ denotes the definite number of $\mathbf{s}_n^J$ preserved in the next population. While, using the decimal part of $N_n (= N_n - \lfloor N_n \rfloor)$, the probability to preserve $\mathbf{s}_n^J$, $J = 1, 2, \ldots, P$, in the next population is determined by

$$\frac{N_n - \lfloor N_n \rfloor}{\sum_{n=1}^{N} (N_n - \lfloor N_n \rfloor)}. \tag{4.4}$$

If a single-point crossover or multipoint crossover is directly applied to upper or middle string of individuals of triple string type, the kth element of the string of an offspring may take the same number that the k'th element takes. The same violation occurs in solving the traveling salesman problems or scheduling problems through genetic algorithms. In order to avoid this violation, a crossover method called partially matched crossover (PMX) is modified to be suitable for triple strings. PMX is applied as usual for upper strings, whereas, for a couple of middle string and lower string, PMX for double strings [14] is applied to every subindividual.

It is now appropriate to present the detailed procedures of the crossover method for triple strings.

4.2. Partially Matched Crossover (PMX) for Upper String

Let

$$X = r_X^1,\ r_X^2, \ldots,\ r_X^P \tag{4.5}$$

be the upper string of an individual and let

$$Y = r_Y^1,\ r_Y^2, \ldots,\ r_Y^P \tag{4.6}$$

be the upper string of another individual. 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. Set $i := h$ and repeat the following procedures.

(a) Find J such that $r_{X'}^J = r_Y^i$. Then, interchange $r_{X'}^i$ with $r_{X'}^J$ and set $i := i + 1$.

(b) If $i > k$, stop and let X' be the offspring of X. Otherwise, return to (a).

Step 2 is carried out for Y' in the same manner, as shown in Figure 5.

4.3. Partially Matched Crossover (PMX) for Double String

Let

$$X = \begin{matrix} v_X^J(1),\ v_X^J(2), \ldots, v_X^J(n_J) \\ y^J_{v_X^J(1)},\ y^J_{v_X^J(2)}, \cdots,\ y^J_{v_X^J(n_J)} \end{matrix} \tag{4.7}$$

be the middle and lower part of a subindividual in the Jth subpopulation, and

$$Y = \begin{matrix} v_Y^J(1),\ v_Y^J(2), \ldots,\ v_Y^J(n_J) \\ y^J_{v_Y^J(1)},\ y^J_{v_Y^J(2)}, \cdots,\ y^J_{v_Y^J(n_J)} \end{matrix} \tag{4.8}$$

be the middle and lower parts of another subindividual in the Jth subpopulation. First, 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. Set $i := h$ and repeat the following procedures.

(a) Find i' such that $v^J_{X'}(i') = v^J_Y(i)$. Then, interchange $(v^J_{X'}(i),\ y^J_{v^J_{X'}(i)})^T$ with $(v^J_{X'}(i'),\ y^J_{v^J_{X'}(i')})^T$ and set $i := i + 1$.

(b) If $i > k$, stop. Otherwise, return to (a).

Step 3. Replace the part from h to k of X' with that of Y and let X' be the offspring of X.

This procedure is carried out for Y' and X in the same manner, as shown in Figure 6.

It is considered that mutation plays the role of local random search in genetic algorithms. Only for the lower string of a triple string, mutation of bit-reverse type is adopted and applied to every subindividual.

For the upper string and for the middle and lower string of the triple string, inversion defined by the following algorithm is adopted

Step 1. After determining two inversion points h and k $(h < k)$, pick out the part of the string from h to k.

Step 2. Arrange the substring in reverse order.

Step 3. Put the arranged substring back in the string.

Figure 7 illustrates examples of mutation.

Now we are ready to introduce the genetic algorithm with decomposition procedures as an approximate solution method for nonlinear integer programming problems with block angular structures. The outline of procedures is shown in Figure 8.

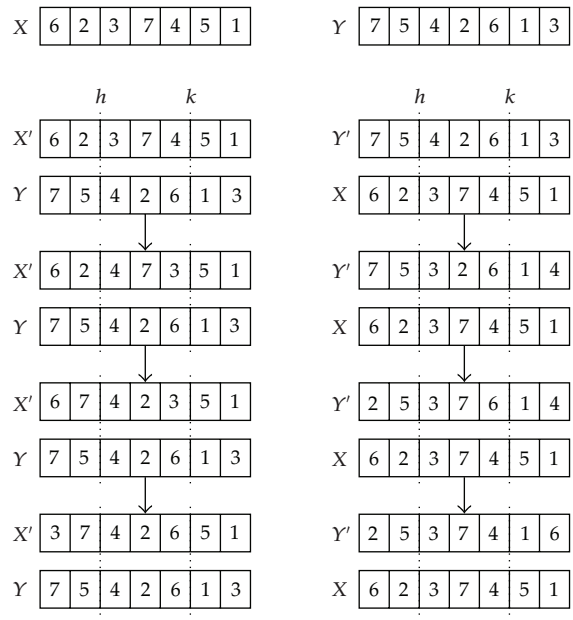

Figure 5: An example of PMX for upper string.

4.4. Computational Procedures

Step 1. Set an iteration index (generation) $t = 0$ and determine the parameter values for the population size N, the probability of crossover p_c, the probability of mutation p_m, the probability of inversion p_i, variances σ, τ, the minimal search generation $I_{\min}$ and the maximal search generation $I_{\max}$.

Step 2. Generate N individuals whose subindividuals are of triple string type at random.

Step 3. Evaluate each individual (subindividual) on the basis of phenotype obtained by the decoding algorithm and calculate the mean fitness f_{mean} and the maximal fitness $f_{\max}$ of the population. If $t > I_{\min}$ and $(f_{\max} - f_{\text{mean}})/f_{\max} < \varepsilon$, or, if $t > I_{\max}$, regard an individual with the maximal fitness as an optimal individual and terminate this program. Otherwise, set $t = t + 1$ and proceed to Step 4.

Step 4. Apply the reproduction operator to all subpopulations $\{\mathbf{s}_n^J \mid n = 1, 2, \ldots, N\}$, $J = 1, 2, \ldots, P$.

Step 5. Apply the PMX for double strings to the middle and lower part of every subindividual according to the probability of crossover p_c.

Step 6. Apply the mutation operator of the bit-reverse type to the lower part of every subindividual according to the probability of mutation p_m and apply the inversion operator for the middle and lower parts of every subindividual according to the probability of inversion p_i.

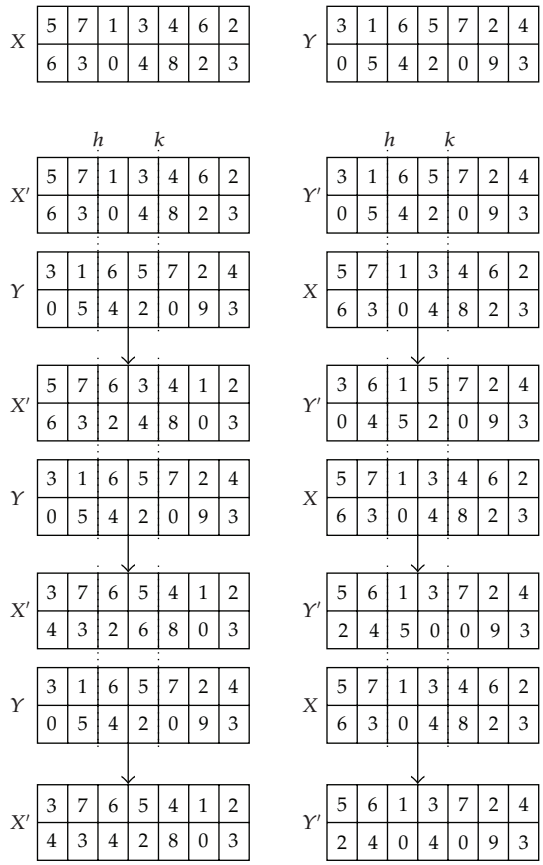

Figure 6: An example of PMX for double string.

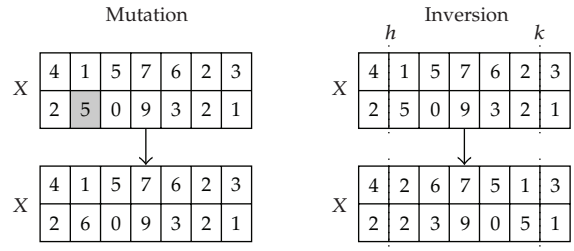

Figure 7: Examples of mutation.

Step 7. Apply the PMX for upper strings according to p_c.

Step 8. Apply the inversion operator for upper strings according to p_i and return to Step 3.

It should be noted here that, in the algorithm, the operations in the Steps 4, 5, and 6 can be applied to every subindividual of all individuals independently. As a result, it is theoretically possible to reduce the amount of working memory needed to solve the problem and carry out parallel processing.

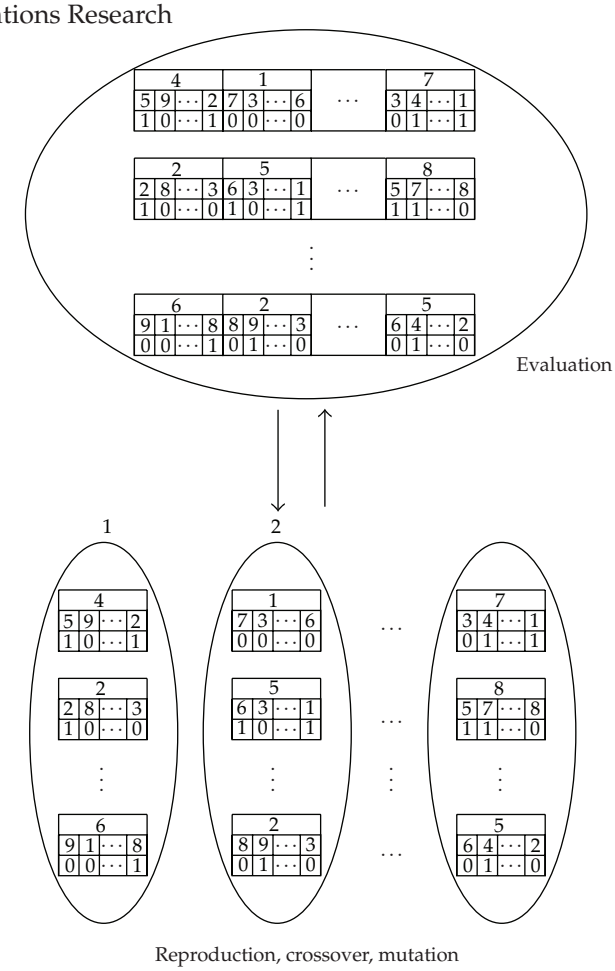

Figure 8: The outline of procedures.

Table 1: The whole process of interaction.

Interaction	1st	2nd	3rd
$\overline{\mu}_1$	1	1	1
$\overline{\mu}_2$	1	0.900	0.900
$\overline{\mu}_3$	1	1	0.900
$\mu_1(f_1(\mathbf{x}))$	0.496	0.552	0.554
$\mu_2(f_2(\mathbf{x}))$	0.497	0.450	0.474
$\mu_3(f_3(\mathbf{x}))$	0.491	0.558	0.524
$f_1(\mathbf{x})$	1500050	1335423	1326906
$f_2(\mathbf{x})$	−1629427	−1475077	−1553513
$f_3(\mathbf{x})$	158226	86012	123127
Computation time (sec)			
GADPCRRSU (proposed method)	26.7	32.8	24.2
GADSCRRSU (no decomposition)	539.6	584.7	503.3

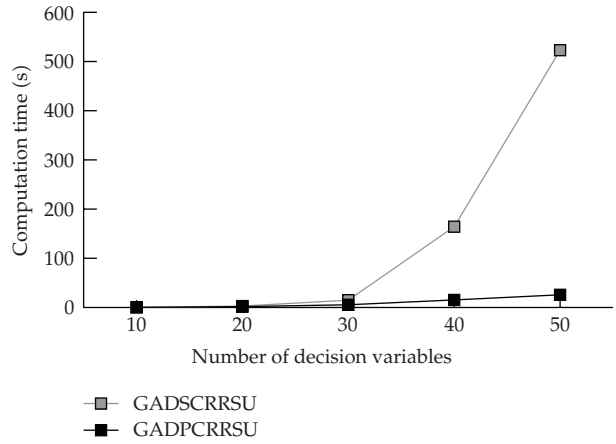

Figure 9: The comparison of computation time.

5. Numerical Examples

In order to demonstrate the feasibility and efficiency of the proposed method, consider the following multiobjective quadratic integer programming problem with block-angular structures:

$$\begin{aligned}
&\text{minimize} \quad f_l(\mathbf{x}) = \sum_{J=1}^{P}\left(\mathbf{c}_l^J\mathbf{x}^J + \left(\frac{1}{2}\right)\left(\mathbf{x}^J\right)^T C_l^J\mathbf{x}^J\right), \quad l = 1,2,\ldots,k,\\
&\text{subject to} \quad g_i(\mathbf{x}) = -\sum_{J=1}^{P}\left(a_i^J\mathbf{x}^J + \left(\frac{1}{2}\right)\left(\mathbf{x}^J\right)^T A_i^J\mathbf{x}^J\right) + b_i^0 \le 0, \quad i = 1,2,\ldots,m_0,\\
&\qquad h_i^J\left(\mathbf{x}^J\right) = -\left(d_i^J\mathbf{x}^J + \left(\frac{1}{2}\right)\left(\mathbf{x}^J\right)^T D_i^J\mathbf{x}^J\right) + b_i^J \le 0,\\
&\qquad\qquad J = 1,2,\ldots,P,\ i = 1,2,\ldots,m_J,\\
&\qquad x_j^J \in \left\{0,1,\ldots,V_j^J\right\}, \quad J = 1,2,\ldots,P,\ j = 1,2,\ldots,n_J,
\end{aligned} \tag{5.1}$$

For comparison, genetic algorithms with double strings using continuous relaxation based on reference solution updating (GADSCRRSU) [24] are also adopted. It is significant to note here that decomposition procedures are not involved in GADSCRRSU.

For this problem, we set $k = 3$, $P = 5$, $n_1 = n_2 = \cdots = n_5 = 10$, $m_0 = 2$ and $m_1 = m_2 = \cdots = m_5 = 5$, $V_j^J = 30$, $J = 1,2,\ldots,5$, $j = 1,2,\ldots,10$. Elements of $c_l^J, C_l^J, a_i^J, A_i^J, d_i^J$ and D_i^J in objectives and constraints of the above problem are determined by uniform random number on $[-100,100]$, and those of b^J in constraints are determined so that the feasible region is not empty.

Numerical experiments are performed on a personal computer (CPU: Intel Celeron Processor, 900 MHz, Memory: 256 MB, C_Compiler: Microsoft Visual C++ 6.0).

Parameter values of GADPCRRSU are set as: population size $N = 100$, crossover rate $p_c = 0.9$, mutation rate $p_m = 0.05$, inversion rate $p_i = 0.05$, variances $\sigma = 2.0$, $\tau = 3.0$, minimal search generation number $I_{\min} = 500$, and maximal search generation number $I_{\max} = 1000$.

In this numerical example, for the sake of simplicity, the linear membership function

$$\mu_l(f_l(\mathbf{x})) = \begin{cases} 1, & f_l(\mathbf{x}) < f_{l,1}, \\ \dfrac{f_l(\mathbf{x}) - f_{l,0}}{f_{l,1} - f_{l,0}}, & f_{l,1} \le f_l(\mathbf{x}) \le f_{l,0}, \\ 0, & f_l(\mathbf{x}) > f_{l,0} \end{cases} \tag{5.2}$$

is adopted, and the parameter values are determined as [30]

$$\begin{aligned} f_{l,1} = f_{l,\min} = f_l\left(\mathbf{x}^l_{\min}\right) = \min_{\mathbf{x}\in X} f_l(\mathbf{x}), \quad l = 1, 2, \ldots, k, \\ f_{l,0} = \max\left\{ f_l\left(\mathbf{x}^1_{\min}\right), \ldots, f_l\left(\mathbf{x}^{l-1}_{\min}\right), f_l\left(\mathbf{x}^{l+1}_{\min}\right), \ldots, f_l\left(\mathbf{x}^k_{\min}\right) \right\}, \quad l = 1, 2, \ldots, k. \end{aligned} \tag{5.3}$$

For the initial reference levels $(1, 1, 1)$, the augmented minimax problem (3.3) is solved. The obtained solutions are shown at the second column in Table 1. Assume that the hypothetical decision maker is not satisfied with the current solution and he feels that $\mu_1(f_1(\mathbf{x}))$ and $\mu_3(f_3(\mathbf{x}))$ should be improved at the expense of $\mu_2(f_2(\mathbf{x}))$. Then, the decision maker updates the reference membership levels to $(1, 0.9000, 1)$. The result for the updated reference membership levels is shown at the third column in Table 1. Since the decision maker is not satisfied with the current solution, he updates the reference membership levels to $(1, 0.900, 0.900)$ for obtaining better value of $\mu_1(f_1(\mathbf{x}))$. A similar procedure continues in this way and, in this example, a satisficing solution for the decision maker is derived at the third interaction.

Table 1 shows that the proposed interactive method using GADPCRRSU with decomposition procedures can find an (approximate) optimal solution at each interaction in shorter time than that using GADSCRRSU without decomposition procedures.

Furthermore, in order to see how the computation time changes with the increased size of block-angular nonlinear integer programming problems, typical problems with 10, 20, 30, 40, and 50 variables are solved by GADPCRRSU and GADSCRRSU. As depicted in Figure 9, it can be seen that the computation time of the proposed GADPCRRSU increases almost linearly with the size of the problem while that of GADSCRRSU increases rapidly and nonlinearly.

6. Conclusions

In this paper, as a typical mathematical model of large-scale discrete systems optimization, we considered multiobjective nonlinear integer programming with block-angular structures. Taking into account vagueness of judgments of the decision makers, fuzzy goals of the decision maker were introduced, and the problem was interpreted as maximizing an overall degree of satisfaction with the multiple fuzzy goals. An interactive fuzzy satisficing method was developed for deriving a satisficing solution for the decision maker. Realizing the block-angular structures that can be exploited, we also propose genetic algorithms

with decomposition procedures for solving nonlinear integer programming problems with block-angular structures. Illustrative numerical examples were provided to demonstrate the feasibility and efficiency of the proposed method. Extensions to multiobjective two-level integer programming problems with block-angular structures will be considered elsewhere. Also extensions to stochastic multiobjective two-level integer programming problems with block-angular structures will be required in the near future.

References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, Reading, Mass, USA, 1989.

[3] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Artificial Intelligence, Springer, Berlin, Germany, 1992.

[4] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, Germany, 2nd edition, 1994.

[5] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, Germany, 3rd edition, 1996.

[6] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, The Clarendon Press, Oxford University Press, New York, NY, USA, 1996.

[7] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*, Institute of Physics, Bristol, UK, 1997.

[8] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley-Interscience Series in Systems and Optimization, John Wiley & Sons, Chichester, UK, 2001.

[9] M. Sakawa, *Large Scale Interactive Fuzzy Multiobjective Programming*, Physica, Heidelberg, Germany, 2000.

[10] M. Sakawa, *Genetic Algorithms and Fuzzy Multiobjective Optimization*, vol. 14 of *Operations Research/Computer Science Interfaces Series*, Kluwer Academic Publishers, Boston, Mass, USA, 2002.

[11] C. A. C. Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York, NY, USA, 2002.

[12] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Natural Computing Series, Springer, Berlin, Germany, 2003.

[13] Z. Hua and F. Huang, "A variable-grouping based genetic algorithm for large-scale integer programming," *Information Sciences*, vol. 176, no. 19, pp. 2869–2885, 2006.

[14] 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.

[15] M. Sakawa, K. Kato, S. Ushiro, and K. Ooura, "Fuzzy programming for general multiobjective 0-1 programming problems through genetic algorithms with double strings," in *Proceedings of IEEE International Fuzzy Systems Conference*, vol. 3, pp. 1522–1527, 1999.

[16] M. Sakawa, K. Kato, T. Shibano, and K. Hirose, "Fuzzy multiobjective integer programs through genetic algorithms using double string representation and information about solutions of continuous relaxation problems," in *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pp. 967–972, 1999.

[17] M. Sakawa and K. Kato, "Integer programming through genetic algorithms with double strings based on reference solution updating," in *Proceedings of IEEE International Conference on Industrial Electronics, Control and Instrumentation*, pp. 2915–2920, 2000.

[18] P. Hansen, "Quadratic zero-one programming by implicit enumeration," in *Numerical Methods for Non-Linear Optimization (Conf., Univ. Dundee, Dundee, 1971)*, F. A. Lootsma, Ed., pp. 265–278, Academic Press, London, UK, 1972.

[19] J. Li, "A bound heuristic algorithm for solving reliability redundancy optimization," *Microelectronics and Reliability*, vol. 36, pp. 335–339, 1996.

[20] D. Li, J. Wang, and X. L. Sun, "Computing exact solution to nonlinear integer programming: convergent Lagrangian and objective level cut method," *Journal of Global Optimization*, vol. 39, no. 1, pp. 127–154, 2007.

[21] R. H. Nickel, I. Mikolic-Torreira, and J. W. Tolle, "Computing aviation sparing policies: solving a large nonlinear integer program," *Computational Optimization and Applications*, vol. 35, no. 1, pp. 109–126, 2006.
[22] M. S. Sabbagh, "A partial enumeration algorithm for pure nonlinear integer programming," *Applied Mathematical Modelling*, vol. 32, no. 12, pp. 2560–2569, 2008.
[23] W. Zhu and H. Fan, "A discrete dynamic convexized method for nonlinear integer programming," *Journal of Computational and Applied Mathematics*, vol. 223, no. 1, pp. 356–373, 2009.
[24] M. Sakawa, K. Kato, M. A. K. Azad, and R. Watanabe, "A genetic algorithm with double string for nonlinear integer programming problems," in *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pp. 3281–3286, 2005.
[25] L. S. Lasdon, *Optimization Theory for Large Systems*, The Macmillian, New York, NY, USA, 1970.
[26] K. Kato and M. Sakawa, "Genetic algorithms with decomposition procedures for multidimensional 0-1 knapsack problems with block angular structures," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 33, pp. 410–419, 2003.
[27] K. M. Bretthauer, B. Shetty, and S. Syam, "A specially structured nonlinear integer resource allocation problem," *Naval Research Logistics*, vol. 50, no. 7, pp. 770–792, 2003.
[28] M. Sakawa, H. Yano, and T. Yumine, "An interactive fuzzy satisficing method for multiobjective linear-programming problems and its application," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 17, no. 4, pp. 654–661, 1987.
[29] M. Sakawa, *Fuzzy Sets and Interactive Multiobjective Optimization*, Applied Information Technology, Plenum Press, New York, NY, USA, 1993.
[30] H.-J. Zimmermann, "Fuzzy programming and linear programming with several objective functions," *Fuzzy Sets and Systems*, vol. 1, no. 1, pp. 45–55, 1978.