Decentralized Two-Level 0-1 Programming through Distributed Genetic Algorithms

Keiichi Niwa *, Tomohiro Hayashida†, Masatoshi Sakawa‡, Yishen Yang§

Abstract—**In this paper, we consider decentralized two-level 0-1 programming problems in which there are one decision maker (the leader) at the upper level and two or more decision makers (the followers) at the lower level, and decision variables of each decision maker are 0-1 variables. We assume that there is coordination among the followers while between the leader and the group of all the followers, there is no motivation to cooperate each other. We propose a modified computational method that solves problems related to computational methods for obtaining the Stackelberg solution. Specifically, in order to shorten the computational time of a computational method implementing a genetic algorithm (GA) proposed by us, a distributed genetic algorithm is introduced with respect to the upper level GA, which handles decision variables for the leader. Parallelization of the lower level GA is also performed along with parallelization of the upper level GA. In order to verify the effectiveness of the proposed method, we propose a comparison with the existing method by performing numerical experiments to verify both the accuracy of the solution and the time required for the computation.**

Keywords: Decentralized two-level 0-1 programming problem, Stackelberg solution, Distributed genetic algorithm

1 Introduction

In the real world, we can often encounter situations that there are two or more decision makers at the organization with hierarchical structures and they make their decisions in turn or at the same time in order to optimize their objectives. For two-level programming problems, the decision maker (the leader) at the upper level first specifies a decision and then the decision maker (the follower) at the lower level determines a decision so as to optimize an objective of the follower with full knowledge of the decision of the leader.

In this paper, we consider two-level programming problems in which there are one decision maker (the leader) at the upper level and two or more decision makers (the followers) at the lower level of a hierarchical structure, and decision variables of the leader and the followers are 0-1 variables, and call such problems decentralized two-level 0-1 programming problems.

As an overview of research dealing with two-level programming problems that include discrete variables, Bard *et al.* presented an algorithm based on the branch-and-bound approach in order to derive the Stackelberg solution for two-level 0-1 programming problems [3] and two-level mixed integer programming problems [2]. Wen *et al.* [14] have presented a computation method for obtaining the Stackelberg solution to two-level programming problems which have 0-1 parameters for the decision variables in the upper level and continuous parameters for the decision variables in the lower level.

Genetic algorithms (GAs) were proposed by Holland [8], as a new learning paradigm that models a natural evolution mechanism. GAs were not much known before Goldberg's book [6] had been published; however, many researchers in various fields have recently attracted much attention to GAs as a methodology for optimization, adaptation and learning [9, 13]. An example of research related to two-level programming problems using GAs is given by Anandalingam, et al. [1] which presents a method for deriving a Stackelberg solution for two-level linear programming problems. In order to derive a Stackelberg solution for 0-1 programming problems related to two-level decentralized systems, the authors [11] have also proposed a computational method that adopts the double string proposed by Sakawa, et al [13] as the individual representation. The authors have proposed computational methods that implement sharing [10] and cluster analysis [12] methods so as to improve the computational accuracy of the Stakelberg solution. Use of these methods allows for the derivation of approximate Stackelberg solutions with relatively high precision and in a relatively short time, but there is still room for improvement, particularly with regards to calculation times.

*Faculty of Economics, Hiroshima University of Economics, 5-37-1 Gion, Asaminami-ku, Hiroshima 731-0192, Japan Tel/Fax: +81-82-871-1048/1005 Email: ki-niwa@hue.ac.jp

†Graduate School of Engineering, Hiroshima University 1-4-1, Kagamiyama, Higashi-Hiroshima, 739-8527, Japan, Tel/Fax: +81-82-424-5267/422-7195 Email: hayashida@hiroshima-u.ac.jp

‡Graduate School of Engineering, Hiroshima University 1-4-1, Kagamiyama, Higashi-Hiroshima, 739-8527, Japan, Tel/Fax: +81-82-424-7694/422-7195 Email: sakawa@hiroshima-u.ac.jp

§Faculty of Economics, Hiroshima University of Economics, 5-37-1 Gion, Asaminami-ku, Hiroshima 731-0192, Japan, Tel/Fax: +81-82-871-1031/1005 Email: ys-yang@hue.ac.jp

ISBN: 978-988-18210-5-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

This paper focuses on decentralized two-level 0-1 programming problems, and proposes an improved computational method that addresses problems related to the computational method proposed by the authors for deriving the Stackelberg solution. Specifically, in order to shorten the computational time of a computational method implementing a genetic algorithm (GA) proposed by the authors, a distributed genetic algorithm is introduced with respect to the upper level GA, which handles decision variables for the leader. Parallelization of the lower level GA is also performed along with parallelization of the upper level GA. In order to verify the effectiveness of the proposed method, we propose a comparison with the existing method by performing numerical experiments to verify both the accuracy of the solution and the time required for the computation.

2 Decentralized two-level zero-one programming problem

We consider two-level decision making situations where there are one leader at the upper level and k followers at the lower level, and decision variables of the leader and the followers are integer variables. Let DM0 denote the leader and DM1, ..., DMk denote the followers.

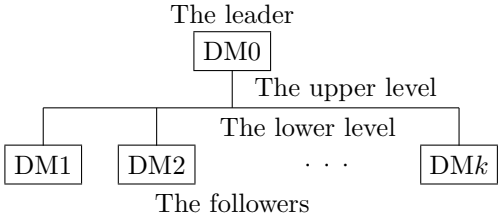

Figure 1: Decentralized two-level structure

The decentralized two-level 0-1 programming problem is expressed as

$$\left.\begin{array}{ll} \underset{\boldsymbol{x}}{\text{maximize}} & z_0(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{c}_0\boldsymbol{x} + \boldsymbol{d}_{01}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{0k}\boldsymbol{y}_k \\ \text{where} & (\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k) \text{ solves} \\ \underset{\boldsymbol{y}_1}{\text{maximize}} & z_1(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{c}_1\boldsymbol{x} + \boldsymbol{d}_{11}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{1k}\boldsymbol{y}_k \\ & \cdots\cdots\cdots\cdots \\ \underset{\boldsymbol{y}_k}{\text{maximize}} & z_k(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{c}_k\boldsymbol{x} + \boldsymbol{d}_{k1}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{kk}\boldsymbol{y}_k \\ \text{subject to} & A\boldsymbol{x} + B_1\boldsymbol{y}_1 + \cdots + B_k\boldsymbol{y}_k \leqq \boldsymbol{b} \\ & \boldsymbol{x} \in \{0,1\}^{n_0} \\ & \boldsymbol{y}_j \in \{0,1\}^{n_j}, j = 1,\ldots,k, \end{array}\right\} \quad (1)$$

where $\boldsymbol{c}_i$, $i = 0,1,\ldots,k$ are n_0-dimensional row coefficient vectors; $\boldsymbol{d}_{ij}, i = 0,1,\ldots,k$, $j = 1,\ldots,k$ are n_j-dimensional row coefficient vectors; A is an $m \times n_0$ coefficient matrix; B_j, $j = 1,\ldots,k$ are $m \times n_j$ coefficient matrices; and $\boldsymbol{b}$ is an m-dimensional column constant vector; $\boldsymbol{x}$ is an n_0-dimensional column decision variable vector of the leader (DM0); $\boldsymbol{y}_j$, $j = 1,\ldots,k$ are n_j-dimensional column decision variable vectors of the jth follower (DMj); for the sake of simplicity, $\boldsymbol{y} = (\boldsymbol{y}_1^T,\ldots,\boldsymbol{y}_k^T)^T$, where T denotes transposition; z_0 is a objective function of DM0; z_j, $j = 1,\ldots,k$ are objective functions of DMj.

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

Let S denote the feasible region of the problem (1), and then, the individual minimum of the objective function for DMi, $i = 0,1,\ldots,k$ is

$$z_i^{\min} = \min_{\boldsymbol{x},\boldsymbol{y}\in S} \boldsymbol{c}_i\boldsymbol{x} + \boldsymbol{d}_{i1}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{ik}\boldsymbol{y}_k, \quad (2)$$

and the individual maximum of the objective function for DMi is

$$z_i^{\max} = \max_{\boldsymbol{x},\boldsymbol{y}\in S} \boldsymbol{c}_i\boldsymbol{x} + \boldsymbol{d}_{i1}\boldsymbol{y}_1 + \cdots + \boldsymbol{d}_{ik}\boldsymbol{y}_k. \quad (3)$$

The individual minimum and the individual maximum are helpful for DMi to identify a membership function prescribing the fuzzy goal for his/her objective function z_i. Consulting the variation ratio of degree of satisfaction in the interval between the individual minimum (2) and the individual maximum (3), DMi determines the membership function $\mu_i(z_i)$, which is strictly monotone increasing for z_i. The domain of the membership function is the interval $[z_i^{\min}, z_i^{\max}]$, $i = 0,1,\ldots,k$, and DMi specifies the value z_i^0 of the objective function such that the degree of satisfaction is 0, $\mu_i(z_i^0) = 0$, and the value z_i^1 of the objective function such that the degree of satisfaction is 1, $\mu_i(z_i^1) = 1$. For the value undesired (smaller) than z_i^0, it is defined that $\mu_i(z_i) = 0$, and for the value desired (larger) than z_i^1, it is defined that $\mu_i(z_i) = 1$.

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

$$\mu_i(z_i) = \begin{cases} 0, & \text{if } z_i < z_i^0 \\ \dfrac{z_i - z_i^0}{z_i^1 - z_i^0}, & \text{if } z_i^0 \leqq z_i < z_i^1 \\ 1, & \text{if } z_i \geqq z_i^1, \end{cases} \quad (4)$$

and it is depicted in Figure 2.

For the leader and the followers, suppose that DMi determines his/her linear membership function by choosing $z_i^0 = z_i^{\min}$ and $z_i^1 = z_i^{\max}$.

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

ISBN: 978-988-18210-5-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

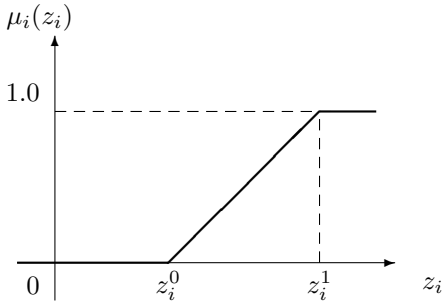

Figure 2: Linear membership function.

gramming problem:

$$\left.\begin{array}{l} \underset{\boldsymbol{x}}{\text{maximize}}\ \mu_0(\boldsymbol{c}_0\boldsymbol{x}+\boldsymbol{d}_{01}\boldsymbol{y}_1+\cdots+\boldsymbol{d}_{0k}\boldsymbol{y}_k) \\ \text{where } \boldsymbol{y} \text{ solves} \\ \underset{\boldsymbol{y}}{\text{maximize}}\ \min_{j=1,\ldots,k}\mu_j(\boldsymbol{c}_j\boldsymbol{x}+\boldsymbol{d}_{j1}\boldsymbol{y}_1+\cdots+\boldsymbol{d}_{jk}\boldsymbol{y}_k) \\ \text{subject to } A\boldsymbol{x}+B_1\boldsymbol{y}_1+\cdots+B_k\boldsymbol{y}_k \leqq \boldsymbol{b}, \\ \quad \boldsymbol{x}\in\{0,1\}^{n_0} \\ \quad \boldsymbol{y}_j\in\{0,1\}^{n_j}, j=1,\ldots,k. \end{array}\right\} \quad (5)$$

In this formulation, the followers (DM1, ..., DMk) maximize their aggregated degree of satisfaction $\min_{j=1,\ldots,k}$ $\mu_j(\boldsymbol{c}_j\boldsymbol{x}+\boldsymbol{d}_{j1}\boldsymbol{y}_1+\cdots+\boldsymbol{d}_{jk}\boldsymbol{y}_k)$. This aggregated degree of satisfaction is nothing else but the fuzzy decision proposed by Bellman, et al [4], which is often employed as a solution concept in fuzzy environments.

3 GA based computational method

We propose a computational method through GAs in order to obtain Stackelberg solutions to the decentralized two-level 0-1 programming problems. In this section, we describe fundamental elements of GAs, which are coding procedure, a decoding procedure and genetic operators, used in the proposed computational method.

3.1 Coding and decoding

Binary strings are usually adopted to express individuals [7, 6] when solving 0-1 programming problems using GAs. However, under this representation it is possible that infeasible individuals that do not satisfy the constraints may be generated, so there is a danger that the performance of the GAs may degrade. Thus, in this paper, in order to derive only feasible solutions, a double string [13] is used which is composed of the substring corresponding to the decision of the leader (DM0), $\boldsymbol{x}$, and the substring corresponding to the decisions of the followers (DM1, ..., DMk), $\boldsymbol{y}$, as shown in Fig.3. The decisions of the leader (DM0) and the followers (DM1, ..., DMk) are handled by performing genetic operators on each sub-individual. In this paper, the GA handling the decision of the leader (DM0) is called the upper level GA, and the GA handling the decisions of the followers (DM1, ..., DMk) is called the lower level GA.

← Individual for $\boldsymbol{x}$ →			← Individual for $\boldsymbol{y}$ →		
$i_x(1)$	$\cdots$	$i_x(n_0)$	$i_y(1)$	$\cdots$	$i_y(n_1+\cdots+n_k)$
$S_{i_x(1)}$	$\cdots$	$S_{i_x(n_0)}$	$S_{i_y(1)}$	$\cdots$	$S_{i_y(n_1+\cdots+n_k)}$

Figure 3: Double string

$s_{i_x(m)} \in \{0,1\}, i_x(m) \in \{1,\ldots,n_0\}$, and for $m \neq m'$ it is assumed that $i_x(m) \neq i_x(m')$. Similarly, $s_{i_y(m)} \in \{0,1\}, i_y(m) \in \{1,\ldots,n_1+\cdots+n_k\}$, and for $m \neq m'$ it is assumed that $i_y(m) \neq i_y(m')$. Also, in the double string, regarding $i_x(k)$, $i_y(k)$ and $s_{i_x(k)}$, $s_{i_y(k)}$ as the index of an element in a solution vector and the value of the element respectively, a string $\boldsymbol{s}$ can be transformed into a solution $\boldsymbol{x}=(x_1,\ldots,x_{n_0})$ and $\boldsymbol{y}_j=(y_{j1},\ldots,y_{j\,n_j})$, $j=1,\ldots,k$ as:

$$\begin{array}{ll} x_{i(m)} & = s_{i_x(m)},\ m=1,\ldots,n_0, \\ y_{1i(m)} & = s_{i_y(m)} \text{ for } 1 \leqq i_y(m) \leqq n_1, \\ y_{2i(m)} & = s_{i_y(m)} \text{ for } n_1+1 \leqq i_y(m) \leqq n_2, \\ & \cdots\cdots\cdots \\ y_{ki(m)} & = s_{i_y(m)} \text{ for } n_{m-1}+1 \leqq i_y(m) \leqq n_k. \end{array}$$

In this paper, a decoding algorithm propoesd by the authors [11] is also applied to the upper level and lower level GA, generating only feasible solutions.

3.2 Reproduction

First, we describe the reproduction operator of the lower level GA. In the lower level GA, by using the value of $\boldsymbol{y}$ obtained by decoding individuals in the lower level GA, and the given values of the decision variables in the upper level GA, $\boldsymbol{x}$, a value of the aggregated fuzzy goal $\min_{j=1,\ldots,k}$ $\mu_j(\boldsymbol{c}_j\boldsymbol{x}+\boldsymbol{d}_{j1}\boldsymbol{y}_1+\cdots+\boldsymbol{d}_{jk}\boldsymbol{y}_k)$ is evaluated, and a evaluation value for each individual is thus obtained. Next, the fitness value for each individual is derived using linear scaling, and the individuals remaining in the next generation are determined by applying elitist expected value selection.

Next, we show the reproduction operator of the upper level GA. In the upper level GA, by using the value of $\boldsymbol{x}$ obtained by decoding individuals in the upper level GA and the value of the rational reaction obtained by applying the lower level GA, a value of fuzzy goal of the leader $\mu_0(\boldsymbol{c}_0\boldsymbol{x}+\boldsymbol{d}_{01}\boldsymbol{y}_1+\cdots+\boldsymbol{d}_{0k}\boldsymbol{y}_k)$ is evaluated, and a evaluation value for each individual is obtained. Furthermore, the fitness value for each individual is calculated by applying linear scaling and adopting a clustering method. The individuals remaining in the next generation are determined by applying elitist expected value selection based on these fitness values.

3.3 Crossover and mutation

If single-point or multi-point crossover operators are applied to double string individuals, then there is a possibil-

ISBN: 978-988-18210-5-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

ity that infeasible individuals may be generated because the indexes occurring in the offspring, $i_x(m)$, $i_x(m')$, $m \neq m'$ or $i_y(m)$, $i_y(m')$, $m \neq m'$, may have the same number. Recall that the same violation occurs in solving traveling salesman problems or scheduling problems through GAs. Partially matched crossovers (PMX) have been devised to stop this violation. In this paper, a modified version of PMX is used in order to handle the double strings proposed by Sakawa *et al.* [13]. Also, when determining whether or not to apply the crossover operator, a probability p_c is used. Its value is set in advance.

The mutation operator is thought to fulfill the role of a local random search in genetic algorithms. For double strings, the index string expresses the priority of the parameters. For binary strings, since the value of the 0-1 parameters themselves are expressed, strings with differing properties coexist in a single string, and it is necessary to apply mutations to each string. In this paper, the mutation operator is applied to each string, and inversion is used for additional strings. For binary strings, bit-reverse is introduced. When applying the mutation operator to individuals, it is first determined whether or not the mutation operator will be applied to an individual according to the mutation probability p_m. In the case that mutation is applied, it is then determined whether to apply inversion or bit-reverse according to the mutation selection constant M_{Pum}.

3.4 Application of the parallel genetic algorithm

In this study, we aim for reductions in computational time, and consider parallelization of the upper level GA and the lower level GA implemented by the computational method proposed by the authors [11].

In GAs, it is possible to perform parallel processing in the greater part of the operations included in the algorithm. In reproduction operations, however, because it is necessary to calculate evaluation values for each individual in a population, and based on that value determine the fitness of each individual, direct application of parallel processing is difficult. Research related to the parallelization of GAs started with improvements to such barriers to the implementation of parallelization, and a variety of types of models have been proposed and their effectiveness noted by numerous researchers [5]. Today, GAs that implement parallel processing have come to be called parallel genetic algorithms.

Broadly classified, there are four types of parallel genetic algorithms:

1) **Single-population master-slave GAs** In these models, the population is not divided, and reproduction and crossover are performed globally, with only individual evaluation performed on multiple processors.

2) **Multiple-population GAs** In these models, the population is divided into multiple partial populations, and the partial populations are assigned to multiple processors. Reproduction, crossover and mutation are then performed on the assigned processors, and at some fixed period an operation called migration is performed to swap individuals among populations. These models are also known as distributed GA.

3) **Fine-grained GAs** In these models, each processor is assigned one or a very small number of individuals.

4) **Hierarchical hybrids** These models combine multiple-population GAs or fine-grained GAs.

The computational method proposed by the authors includes an upper level GA and a lower level GA, and parallelization of each must be considered. We will describe the parallelization of upper level GAs. In this paper, we will employ multiple-population GAs. First, in our proposed method, the population of the upper level GA is divided into multiple subpopulations, and the subpopulations are assigned to multiple processors. Next, reproduction operator, crossover operator and mutation operator are applied on the assigned processors, and migration is performed to swap individuals among subpopulations.

We will describe the parallelization of the lower level GA. A lower level GA is used to obtain the rational reaction $\boldsymbol{y}(\boldsymbol{x})$ to a given upper level GA individual $\boldsymbol{x}$. For each individual $\boldsymbol{x}$ of the upper level GA therefore there is an independently operating lower level GA, and so it is possible to divide the lower level GA operations and assign them across multiple processors.

It is possible to reduce calculation times and improve calculation precision by employing such a model.

3.5 Lower level GA avoidance procedures

By introducing a lower level GA avoidance procedure it is possible to reduce the number of unneeded rational reaction calculations, and so calculation times are reduced. In this paper, we introduce storage regions as shown in Fig. 4.

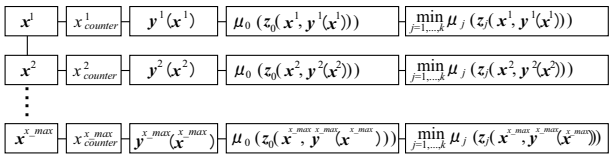

Figure 4: Storage for saving $\boldsymbol{x}$ and $\boldsymbol{y}(\boldsymbol{x})$

$\boldsymbol{x}^i, i = 1, \ldots, x\_max$ indicates those values of $\boldsymbol{x}$ that were

ISBN: 978-988-18210-5-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

used in the past for handling individuals of the upper level GA, and $\boldsymbol{y}^i(\boldsymbol{x}^i), i = 1, \ldots, x\_max$ indicates the values of the rational reactions associated with $\boldsymbol{x}^i$ obtained by the lower level GA. $x^i_{counter} \in \{1, 2, \ldots, y\_max\}, i = 1, \ldots, x\_max$ indicates the number of times that the lower level GA was used to find the rational reaction $\boldsymbol{y}^i(\boldsymbol{x}^i)$ for $\boldsymbol{x}^i$. x_max indicates the maximum number of the leader's (DM0) decisions $\boldsymbol{x}$ saved, and y_max indicates the maximum number of times that the lower level GA can be repeatedly used to find the rational reaction $\boldsymbol{y}^i(\boldsymbol{x}^i)$ for $\boldsymbol{x}^i$. $\mu_0(z_0(\boldsymbol{x}^i, \boldsymbol{y}^i))$ is a value that is obtaiend by substituting stored $\boldsymbol{x}^i$, $\boldsymbol{y}^i(\boldsymbol{x}^i)$ values into fuzzy goal of the leader. Also, $\min_{j=1,\ldots,k} \mu_j(z_j(\boldsymbol{x}^i, \boldsymbol{y}^i))$ is a value that is obtained by substituting stored $\boldsymbol{x}^i$ and $\boldsymbol{y}^i(\boldsymbol{x}^i)$ values into aggregated fuzzy goal of the flollowers. By furthermore using an algorithm like the following, the number of applications of the lower level GA is reduced, and unnecessary calculation times eliminated.

Storage of the rational reaction $\boldsymbol{y}(\boldsymbol{x})$ and lower level GA avoidance procedures

Step 1 If there exists in $\boldsymbol{x}^i$ an upper level GA individual $\bar{\boldsymbol{x}}$, proceed to Step 2. If one does not exist, then check if the number of $\boldsymbol{x}^i$ has reached x_max, and if so continue on to Step 3. If not, proceed to Step 4.

Step 2 If $x^i_{counter}$ has reached y_max, then the saved $\boldsymbol{y}^i(\boldsymbol{x}^i)$ is returned to the upper level GA as the rational reaction and the algorithm terminates. If not reached, proceed to Step 4.

Step 3 Select the least of the values $\mu_0(z_0(\boldsymbol{x}^i, \boldsymbol{y}^i(\boldsymbol{x}^i)))$ from the saved $\boldsymbol{x}^i$, and take that $\boldsymbol{x}^i$ value as $\boldsymbol{x}^k$. After applying the lower level GA and thus obtaining the rational reaction $\boldsymbol{y}(\bar{\boldsymbol{x}})$ for $\bar{\boldsymbol{x}}$, if $\mu_0(z_0(\boldsymbol{x}^k, \boldsymbol{y}^k(\boldsymbol{x}^k))) \leqq \mu_0(z_0(\bar{\boldsymbol{x}}, \boldsymbol{y}(\bar{\boldsymbol{x}})))$, save $\bar{\boldsymbol{x}}$, $\boldsymbol{y}(\bar{\boldsymbol{x}})$, $\mu_0(z_0(\bar{\boldsymbol{x}}, \boldsymbol{y}(\bar{\boldsymbol{x}})))$, $\min_{j=1,\ldots,k} \mu_j(z_j(\bar{\boldsymbol{x}}, \boldsymbol{y}(\bar{\boldsymbol{x}})))$ in the storage region $\boldsymbol{x}^k$, and terminate the algorithm.

Step 4 After obtaining the rational reaction $\boldsymbol{y}(\bar{\boldsymbol{x}})$ for $\bar{\boldsymbol{x}}$ by applying the lower level GA, save $\bar{\boldsymbol{x}}$, $\boldsymbol{y}(\bar{\boldsymbol{x}})$, $\mu_0(z_0(\bar{\boldsymbol{x}}, \boldsymbol{y}(\bar{\boldsymbol{x}})))$, $\min_{j=1,\ldots,k} \mu_j(z_j(\bar{\boldsymbol{x}}, \boldsymbol{y}(\bar{\boldsymbol{x}})))$, and terminate the algorithm.

3.6 The algorithm for the improved computational method

The following is a summary of the algorithm used in the computational method after improvement. Here, the number of processors used in the experiment is taken to be p.

Step 1 For each processor $q, q = 1, \ldots, p$, apply the upper level GA operations on Step1 through Step7. Taking the generation of the upper level GA as $t_{uq} := 0$, N_u initial individuals are randomly generated.

Step 2 For each individual $\boldsymbol{x}$ in the upper level GA, determine whether or not to apply the lower level GA, and find the number of lower level GA to apply N_{ul}. For those individuals to which the lower level GA will be applied, apply the lower level GA operations in Step 2-1 through Step 2-3, and obtain the rational reaction $\boldsymbol{y}(\boldsymbol{x})$. For those $N_u - N_{ul}$ individuals to which the lower level GA will not be applied, take the saved $\boldsymbol{y}(\boldsymbol{x})$ as the rational reaction, and proceed to Step 4.

Step 2-1 Set $t_l := 0$. Randomly generate N_l lower level GA individuals $\boldsymbol{y}$, and take these as the initial population of the lower level GA . Proceed to Step 2-2.

Step 2-2 Use $\boldsymbol{x}$ given as the upper level GA individual and $\boldsymbol{y}$ generated by the lower level GA to calculate the value of the aggregated fuzzy goal $\min_{j=1,\ldots,k} \mu_j(z_j(\boldsymbol{x}, \boldsymbol{y}))$, and after applying linear scaling use that value to generate an individual. Proceed to Step 2-3.

Step 2-3 If t_l has exceeded the previously defined a maximum number of generation M_l, take the individual with the best fitness value as the optimal individual $\boldsymbol{y}(\boldsymbol{x})$, and proceed to Step 3. Otherwise, apply crossover operator and mutation operator to each lower level GA individual, let $t_l = t_l + 1$, and proceed to Step 2-2.

Step 3 Calculate the value of fuzzy goal of the leader (DM0) $\mu_0(z_0(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x})))$ and the value of the aggregated fuzzy goal $\min_{j=1,\ldots,k} \mu_j(z_j(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x})))$ using the lower level rational reaction $\boldsymbol{y}(\boldsymbol{x})$ obtained by operation of the lower level GA, and the individual $\boldsymbol{x}$ of the upper level GA. Perform the procedures required to save $\boldsymbol{x}$ and its rational reactions $\boldsymbol{y}(\boldsymbol{x})$ to the storage region, and proceed to Step 4.

Step 4 Calculate the value of fuzzy goal of the leader (DM0) $\mu_0(z_0(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x})))$ for each upper level GA individual $\boldsymbol{x}$, and after performing linear scaling apply the clustering method to measure the level of convergence of the individuals. Depending upon the degree of convergence, calculate the fitness value of each individual. Proceed to Step 5.

Step 5 If t_{uq} has exceeded the previously set a maximum number of generation M_u, then terminate the algorithm. In that case, the individual obtained up to that generation with the best fitness value is taken as the optimal individual $(\boldsymbol{x}, \boldsymbol{y})$. Otherwise, proceed to Step 6.

Step 6 Reproduction operator is performed using the fitness values of each individual of the upper level GA. Apply crossover operator and mutation operator to each upper level GA individual, and proceed to Step7.

ISBN: 978-988-18210-5-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Step 7 If t_{uq} mod m_i (migration interval) = 0, after performing synchronization between the processors, apply migration. Return to Step 2 with $t_{uq} := t_{uq} + 1$.

4 Conclusion

In this paper, we have focused on decentralized two-level 0-1 programming problems, and have proposed a modified computational method that solves problems related to computational methods for obtaining the Stackelberg solution. Specifically, in order to shorten the computational time of a computational method implementing a GA proposed by the authors, a distributed genetic algorithm has been introduced with respect to the upper level GA. Also, parallelization of the lower level GA has been performed along with parallelization of the upper level GA. In order to verify the effectiveness of the proposed method, it is intended to conduct numerical experiments into both the solution precision and computation time, and thus compare the proposed method with the existing method.

References

[1] G. Anandalingam, R. Mathieu, C.L. Pittard, and N. Sinha, "Artificial intelligence based approaches for solving hierarchical optimization problems," in: Sharda, Golden, Wasil, Balci and Stewart (eds.), *Impacts of Recent Computer Advances on Operations Research,* North-Holland, pp. 289–301 (1989).

[2] J. Bard and J. Moore, "The mixed integer linear bilevel programming problem," *Operations Research,* Vol. 38, pp. 911–921 (1990).

[3] J. Bard and J. Moore, "An algorithm for the discrete bilevel programming problem," *Naval Research Logistics,* Vol. 39, pp. 419–435 (1992).

[4] R.E. Bellman and L.A. Zadeh, "Decision making in a fuzzy environment," *Management Science,* Vol. 17, pp. 141-164 (1970).

[5] E. Cantú-Paz: *Efficient and Accurate Parallel Genetic Algorithms,* Kluwer Academic Publishers, Norwell, Massachussets (2000).

[6] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning,* Addison Wesley, Massachusetts (1989).

[7] D.E. Goldberg and R. Lingle: "Alleles, loci, and the traveling salesman problem," *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 154–159 (1985).

[8] J.H. Holland, *Adaptation in Natural and Artificial Systems,* University of Michigan Press (1975), MIT Press, Cambridge (1992).

[9] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs,* Springer-Verlag, Third, revised and extended edition, Berlin, (1996).

[10] K. Niwa: "Revised computational methods for using genetic algorithms for obtaining Stackelberg solutions to two-level 0-1 programming problems," *Essays and Studies in Commemoration of the 40th Anniversary of the Founding of Hiroshima University of Economics,* Hiroshima University Economics, pp. 771–794, in Japanese (2007)

[11] K. Niwa, I. Nishizaki and M. Sakawa, "Decentralized two-level 0-1 programming through genetic algorithms with double strings," in: L.C. Jain and R.K. Jain (eds.), *In proceedings of Second International Conference on Knowledge-Based Intelligent Electronic Systems,* vol. 2, pp. 278–283 (1998).

[12] K. Niwa, I. Nishizaki and M. Sakawa: "Two-Level 0-1 Programming Using Genetic Algorithms and a Sharing Scheme Based on Cluster Analysis," *International MultiConference of Engineers and Computer Scientists 2008 Proceedings*, pp. 1931–1936 (2008).

[13] M. Sakawa, M. Tanaka: *Genetic Algorithms*, Asakura Publishing, in Japanese (1995).

[14] W.P. Wen, and Y.H. Yang: "Algorithms for solving the mixed integer two-level linear programming problem," *Computers and Operations Research*, Vol. 17, pp. 133–142 (1990).

ISBN: 978-988-18210-5-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)