

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

Keiichi Niwa

Faculty of Economics

Hiroshima University of Economics

5-37-1, Asaminami-ku, Hiroshima,
731-0192, Japan

Email: ki-niwa@hue.ac.jp

Tomohiro Hayashida

Graduate School of Engineering

Hiroshima University

1-4-1, Kagamiyama, Higashi-Hiroshima,
739-8527, Japan

Email: hayashida@hiroshima-u.ac.jp

Masatoshi Sakawa

Graduate School of Engineering

Hiroshima University

1-4-1, Kagamiyama, Higashi-Hiroshima,
739-8527, Japan

Email: sakawa@hiroshima-u.ac.jp

Abstract—In this paper we focus on a multiobjective two-level 0-1 programming problem in which the decision maker at the upper level has an objective function and the decision maker at the lower level has multiple objective functions. We assume that there is not coordination between the decision maker at the upper level and the decision maker at the lower level. The decision maker at the upper level must take account of multiple rational responses of the decision maker at the lower level in the problem. We examine two kinds of situations based on anticipation of the decision maker at the upper level; an optimistic anticipation and a pessimistic anticipation. We show mathematical programming problems for obtaining the Stackelberg solutions based on two kinds of anticipation and propose computational methods using genetic algorithms for obtaining the Stackelberg solutions. In order to demonstrate feasibility and effectiveness of the proposed computational methods through genetic algorithms, we plan to conduct numerical experiments.

I. INTRODUCTION

In this paper, we deal with a multiobjective two-level 0-1 programming problem in which there is not coordination between the decision maker at the upper level (leader) and the decision maker at the lower level (follower). Each decision maker knows completely the objective functions and constraints of the opponent and self. For traditional two-level programming problems, the leader first specifies a decision and then the follower specifies a decision so as to optimize the objective with full knowledge of the decision of the leader. According to the rule, the leader also specifies the decision so as to optimize the objective function. We call the solution defined as the above mentioned procedure the Stackelberg solution [37].

It is often assumed that a set of rational responses $R(x)$ of the follower is a singleton in conventional two-level 0-1 programming problems. In multiobjective environments, however, the assumption is not realistic and consequently the leader takes into consideration multiple responses or an infinite number of responses of the followers with respect to the decision specified by the leader. In this paper we provide a prescriptive approach to the multiobjective two-level 0-1 programming problem from a viewpoint of the leader. We assume that the leader has some subjective anticipation or belief. Such anticipation or belief can be considered in two cases: (i) the leader anticipates that the follower will take a

decision desirable for the leader (optimistic anticipation); (ii) the leader anticipates that the follower will take a decision undesirable for the leader (pessimistic anticipation).

On the other hand, genetic algorithms (GAs) [19] have received a great deal of attention regarding their potential as optimization techniques for solving discrete optimization problems or other hard optimization problems. Although GAs were not much known at the beginning, after the publication of Goldberg's book [17], many researchers in various fields have recently attracted much attention to GAs as a methodology for optimization, adaptation and learning [21], [34].

In order to solve multiobjective optimization problems, various types of GAs are proposed by many researchers [20], [16], [43], [11], [42], [9], [13], [41]. As an early GA application, Shaffer [36] proposes vector evaluated genetic algorithms (VEGA). VEGA performs independent selection cycles according to each objective. Then the whole population is thoroughly shuffled to apply a crossover operator and a mutation operator. Afterward, multiobjective genetic algorithms (MOGAs) are proposed by C. M. Fonseca *et al.* [15] and a non-dominated sorting genetic algorithm (NSGA) based on the non-dominated sorting procedure [17] are proposed by Srinivas *et al.* [38]. Also, Tamaki *et al.* [39] propose to preserve solution non-dominated in the current population. Then, K. Deb *et al.* [10] propose a non-dominated sorting genetic algorithm II (NSGA-II) for solving some problems of NSGA and show the effectiveness of the proposed method by performing numerical experiments.

As an overview of research dealing with two-level programming problems that include discrete variables, Bard *et al.* develop algorithms based on the branch-and-bound techniques for obtaining Stackelberg solutions to two-level 0-1 programming problems [6] and two-level mixed integer programming problems [5]. Wen *et al.* [40] propose an exact computational method and a heuristic method for obtaining a Stackelberg solution to two-level mixed 0-1 programming problems with 0-1 decision variables of the leader and continuous decision variables of the follower. Anandalingam *et al.* [3] propose a computational method using GAs for deriving a Stackelberg solution for two-level linear programming problems. Niwa *et al.* [27], [28] propose computational methods through GAs

with double string proposed by Sakawa *et al.* [34] so as to derive a Stackelberg solution for 0-1 programming problems related to two-level decentralized systems. Also, Niwa *et al.* propose computational methods that implement sharing [23] and cluster analysis [30] methods so as to improve the computational accuracy of the Stakelberg solution. Moreover, Niwa *et al.* propose a computational method through parallel genetic algorithms (single-population master-slave GAs) for obtaining a Stackelberg solution to two-level 0-1 programming problems [31] and develope computational methods using distributed genetic algorithms so as to solve two-level 0-1 programming problems [26] and decentralized two-level 0-1 programming problems [25]. Specifically, in these computational methods, parallel genetic algorithms and distributed genetic algorithms are introduced with respect to the upper level GA, which handles decision variables for the upper level decision maker. Use of these methods allows for the derivation of approximate Stackelberg solutions with high precision and in a short time.

For the multiobjective two-level linear programming problems, Nishizaki *et al.* [22] propose solution concepts with the leader anticipation and develop computational methods based on the idea of the k th best method proposed by Bialas *et al.* [7] and a computational method by using branch-and-bound technique proposed by Bard *et al.* [4]. Niwa *et al.* develop computational methods through the vector evaluated genetic algorithms for obtaining Stackelberg solutions to multiobjective two-level 0-1 programming problems [29] and multiobjective two-level integer programming problems [24].

In this paper, therefore, we focus on a multiobjective two-level 0-1 programming problem and propose two types of computational methods for obtaining the Stackelberg solution. Specifically, in order to reduce computational time, we consider parallelization of a genetic algorithm handling the decision of the leader and adopt a distributed genetic algorithm as the genetic algorithm. Also, we consider parallelization of the genetic algorithm handling the decision of the follower. Moreover, a first type of proposed computational method adopts the vector evaluated genetic algorithms [36], [39] for deriving multiple rational responses of the follower. A second type of proposed computational method introduces NSGA-II [10] so as to derive multiple rational responses of the follower. In order to demonstrate the effectiveness and the feasibility of the proposed computational methods, we plan to conduct numerical experiments.

II. MULTIOBJECTIVE TWO-LEVEL 0-1 PROGRAMMING PROBLEM

In this paper, we consider a multiobjective two-level programming problem in which the leader has an objective function and the follower has multiple objective functions and decision variables of the leader and the follower are 0-1 variables. A multiobjective two-level 0-1 programming problem for obtaining a Stackelberg solution is represented

as

$$\left. \begin{array}{l} \text{minimize}_{\mathbf{x}} z_1(\mathbf{x}, \mathbf{y}) = \mathbf{c}_1 \mathbf{x} + \mathbf{d}_1 \mathbf{y} \\ \text{where } \mathbf{y} \text{ solves} \\ \text{minimize}_{\mathbf{y}} z_{21}(\mathbf{x}, \mathbf{y}) = \mathbf{c}_{21} \mathbf{x} + \mathbf{d}_{21} \mathbf{y} \\ \dots \\ \text{minimize}_{\mathbf{y}} z_{2k_2}(\mathbf{x}, \mathbf{y}) = \mathbf{c}_{2k_2} \mathbf{x} + \mathbf{d}_{2k_2} \mathbf{y} \\ \text{subject to } A\mathbf{x} + B\mathbf{y} \leq \mathbf{b} \\ x_i \in \{0, 1\}, i = 1, \dots, n_1 \\ y_j \in \{0, 1\}, j = 1, \dots, n_2, \end{array} \right\} \quad (1)$$

where $\mathbf{c}_1, \mathbf{c}_{2k}, k = 1, \dots, k_2$ are n_1 -dimensional constant row vectors; $\mathbf{d}_1, \mathbf{d}_{2k}, k = 1, \dots, k_2$ are n_2 -dimensional constant row vectors, respectively. $\mathbf{b}$ is an m -dimensional constant column vector; A is an $m \times n_1$ constant matrix; B is an $m \times n_2$ constant matrix; these coefficients are assumed to be integer constants; $z_1(\mathbf{x}, \mathbf{y})$ is the leader's objective function; $z_{2k}(\mathbf{x}, \mathbf{y}), k = 1, \dots, k_2$ are the follower's objective functions; $\mathbf{x}$ and $\mathbf{y}$ are the leader's and the follower's 0-1 decision variables, respectively.

For conventional two-level 0-1 programming problems, the leader first makes a decision and then the follower specifies a decision in order to optimize the objective function of the follower with full knowledge of the decision of the leader. According to the rule, the leader also determines a decision so as to optimize the objective function of self. We call the solution defined as the above mentioned procedure the Stackelberg solution. In this case, it is often assumed that the set of rational responses $R(\mathbf{x})$ for each of all the possible decisions $\mathbf{x}$ of the leader is a singleton. However, in multiobjective environment, the above assumption becomes meaningless and hence the leader takes into consideration multiple responses or an infinite number of responses of the follower with respect to the decision specified by the leader. It seems reasonable to suppose that at least the set of rational responses $R(\mathbf{x})$ is imposed is the Pareto optimality.

Therefore, we introduce the following set of Pareto optimal responses $P(\mathbf{x})$ as a substitute for the set of rational responses $R(\mathbf{x})$.

$$P(\mathbf{x}) = \{\mathbf{y} \in S(\mathbf{x}) \mid \text{there does not exist another } \mathbf{y}' \in S(\mathbf{x}) \text{ such that } z_{2k}(\mathbf{x}, \mathbf{y}') \leq z_{2k}(\mathbf{x}, \mathbf{y}), \forall k, k = 1, \dots, k_2 \text{ and } z_{2\ell}(\mathbf{x}, \mathbf{y}') \neq z_{2\ell}(\mathbf{x}, \mathbf{y}) \text{ for at least one } \ell.\} \quad (2)$$

where $S(\mathbf{x}) = \{\mathbf{y} \mid B\mathbf{y} \leq \mathbf{b} - A\mathbf{x}, y_j \in \{0, 1\}, j = 1, \dots, n_2\}$.

For a decision $\mathbf{x}$ determined by the leader, the leader can presume only that a response $\mathbf{y}$ of the follower with respect to $\mathbf{x}$ belongs to the set of Pareto optimal responses $P(\mathbf{x})$.

As done in a conventional two-level 0-1 programming problem, the leader first has specified a decision $\mathbf{x}$ and then the follower takes a decision $\mathbf{y}$ which belongs to the set of Pareto optimal responses $P(\mathbf{x})$. However, the leader does not know which a response is chosen from the set of Pareto optimal responses by the follower. Then we can provide reasonable decisions of the leader in accordance with some subjective anticipation or belief if a prescriptive approach should be employed. Such anticipation or belief can be considered in two cases: i) a case where the leader anticipates that the follower

will take a decision desirable for the leader (the optimistic anticipation); ii) a case where the leader anticipates that the follower will take a decision undesirable for the leader (the pessimistic anticipation). When the leader does not know the preference of the follower, the leader would make a decision by consulting outcomes yielded from the optimistic anticipation and the pessimistic anticipation. We show multiobjective two-level 0-1 programming problems based on the optimistic anticipation and the pessimistic anticipation.

A. Optimistic Stackelberg solution

After the leader has chosen a decision $\hat{x}$, the follower makes a decision y with full knowledge of the decision $\hat{x}$ of the leader by solving the multiobjective 0-1 programming problem:

$$\left. \begin{array}{ll} \text{minimize}_{\mathbf{y}} & z_{21}(\hat{x}, \mathbf{y}) = \mathbf{d}_{21}\mathbf{y} + \mathbf{c}_{21}\hat{x} \\ & \dots\dots\dots \\ \text{minimize}_{\mathbf{y}} & z_{2k_2}(\hat{x}, \mathbf{y}) = \mathbf{d}_{2k_2}\mathbf{y} + \mathbf{c}_{2k_2}\hat{x} \\ \text{subject to} & \mathbf{B}\mathbf{y} \leq \mathbf{b} - \mathbf{A}\hat{x} \\ & y_j \in \{0, 1\}, j = 1, \dots, n_2. \end{array} \right\} \quad (3)$$

Let $P(\hat{x})$ denote the set of Pareto optimal responses (solutions) of the multiobjective integer programming problem (3).

Hereafter, we call a Stackelberg solution based on the optimistic anticipation the optimistic Stackelberg solution simply. The optimistic Stackelberg solution is an optimal solution to the following problem (4).

$$\left. \begin{array}{ll} \min_{\mathbf{x}} \min_{\mathbf{y}} & z_1(\mathbf{x}) = \mathbf{c}_1\mathbf{x} + \mathbf{d}_1\mathbf{y} \\ \text{subject to} & \mathbf{y} \in P(\mathbf{x}) \\ & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b} \\ & x_i \in \{0, 1\}, i = 1, \dots, n_1 \end{array} \right\} \quad (4)$$

Problem (4) expresses a situation where the leader chooses a decision $\mathbf{x}$ so as to minimize the leader's objective function on the assumption that, for the given $\mathbf{x}$ of the leader, the follower takes a decision $\mathbf{y}$ in the set of Pareto optimal responses $P(\mathbf{x})$ such that the leader's objective function is minimized.

B. Pessimistic Stackelberg solution

Suppose that the leader makes a decision $\mathbf{x}$ and anticipates that the follower chooses a decision undesirable for the leader in the set of Pareto optimal responses $P(\mathbf{x})$. We call a Stackelberg solution based on the pessimistic anticipation the pessimistic Stackelberg solution. The pessimistic Stackelberg solution can be obtained by solving the problem

$$\left. \begin{array}{ll} \min_{\mathbf{x}} \max_{\mathbf{y}} & z_1(\mathbf{x}) = \mathbf{c}_1\mathbf{x} + \mathbf{d}_1\mathbf{y} \\ \text{subject to} & \mathbf{y} \in P(\mathbf{x}) \\ & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b} \\ & x_i \in \{0, 1\}, i = 1, \dots, n_1. \end{array} \right\} \quad (5)$$

This problem can be solved by an algorithm similar to the algorithm for the optimistic Stackelberg solutions.

III. COMPUTATIONAL METHODS THROUGH GENETIC ALGORITHMS

In this paper, we propose computational methods through genetic algorithms for obtaining Stackelberg solutions based on the optimistic anticipation and the pessimistic anticipation to multiobjective two-level 0-1 programming problems. The proposed computational methods are described as follows.

We propose two types of computational methods using GAs for obtaining Stackelberg solutions to the problem (4) and (5). The proposed computational methods use the double string [32] which composed of the substring corresponding to the decision of the leader, and the substring corresponding to the decision of the follower. In order to solve the problem (4) and (5), the decisions of the leader and the follower are handled by performing genetic operators on each substring. Specifically, in order to reduce computational time, we consider parallelization of a genetic algorithm handling the decision of the leader and adopt a distributed genetic algorithm as the genetic algorithm. Also, we consider parallelization of the genetic algorithm handling the decision of the follower. The above-mentioned parallelization approaches are improvements common to two types of the proposed computational methods. Moreover, a first type of the proposed computational method adopts VEGA [36], [39] for deriving multiple rational responses of the follower. Also, a second type of proposed computational method introduces NSGA-II [10] so as to derive multiple rational responses of the follower.

In the following sections III-A, III-B, III-C, III-D, we show common operators or procedures used for the proposed computational methods. In section III-E, we describe a reproduction operator used for each computational method in detail. Finally, we show algorithms used in the computational methods after improvement in section III-F.

A. Coding and decoding

For solving 0-1 programming problems using genetic algorithms, an individual is usually represented by a binary 0-1 string [18], [17], [21]. However, this representation may weaken the ability of genetic algorithms since an individual whose phenotype is feasible is scarcely generated under this representation.

For multidimensional 0-1 knapsack problems, Sakawa *et al.* [32] propose a double string representation and a decoding algorithm for eliminating infeasible solutions. In this paper, a double string which is composed of the substring corresponding to the decision of the leader, $\mathbf{x}$, and the substring corresponding to the decision of the follower, $\mathbf{y}$, as shown in Fig.1 is used so as to derive only feasible solutions.

The decisions of the leader and the follower are handled by applying genetic operators on each substring. We call the genetic algorithm operating to the decision $\mathbf{x}$ of the leader the upper level GA, and the genetic algorithm operating to the decision $\mathbf{y}$ of the follower the lower level GA.

$s_{i_x(m)} \in \{0, 1\}$, $i_x(m) \in \{1, \dots, n_1\}$, and for $m \neq m'$ it is assumed that $i_x(m) \neq i_x(m')$. Similarly, $s_{i_y(m)} \in \{0, 1\}$, $i_y(m) \in \{1, \dots, n_2\}$, and for $m \neq m'$ it is assumed

← Individual for x →			← Individual for y →		
$i_x(1)$	...	$i_x(n_1)$	$i_y(1)$	...	$i_y(n_2)$
$S_{i_x(1)}$	...	$S_{i_x(n_1)}$	$S_{i_y(1)}$	...	$S_{i_y(n_2)}$

Fig. 1. Double string

that $i_y(m) \neq i_y(m')$. Also, in the double string, $i_x(m)$, $i_y(m)$ and $s_{i_x(m)}$, $s_{i_y(m)}$ express indexes of the elements of each solution vector respectively, and their values.

Also, a decoding algorithm proposed by the Niwa *et al.* [27] is also applied to the upper level GA and the lower level GA so as to generate only feasible solutions.

B. Crossover and mutation

If a one-point or multi-point crossover operator is applied to individuals represented by double strings then there is a possibility 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. When solving the traveling salesman problem or the scheduling problem through GAs, this kind of violation occurs. One possible approach to circumvent such violation, a crossover method called partially matched crossover (PMX) is useful [18]. In this paper, a modified version of PMX is used in order to handle the double string proposed by Sakawa *et al.* [34]. Also, when determining whether or not to apply the crossover operator, the prespecified probability p_c is used. Its value is set in advance.

A mutation operator is thought to fulfill the role of local random search in genetic algorithms. In this paper, for the lower string of a double string, mutation of bit-reverse type is adopted. For the upper string of a double string, an inversion is used. 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 prespecified probability p_m . In the case that mutation is applied, it is then determined whether to apply an inversion or mutation of bit-reverse type according to the prespecified probability M_{Pum} .

C. Application of parallel genetic algorithms

It is possible to divide the operations without a reproduction operator included in genetic algorithms, and to solve those operations simultaneously using multiple processors. However, in a reproduction operator, an evaluation value (ex. objective function value) for each individual is calculated, and fitness values for individuals are computed by using the evaluation values. Because it is necessary to divide this process in a reproduction operator, 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. In general, parallel genetic algorithms are classified broadly into four types: 1) Single-population master-slave GAs, 2) Multiple-population GAs, 3) Fine-grained GAs, and 4) Hierarchical hybrids [8]. Single-population master-slave GAs have a single population. The master performs a

selection operator a crossover operator and a mutation operator. Slaves only evaluate the fitness of the individuals. Multiple-population GAs or multiple-deme GAs consist of several subpopulations that exchange individuals every so often. This exchange of individuals is called migration. The subpopulations are assigned to multiple processors. A reproduction operator, a crossover operator and a mutation operator are performed on each subpopulation. These models are also known as distributed GAs or island parallel GAs. Fine-grained GAs consist of a single spatially-distributed population. Each processor has a very small number of individuals. Ideally, an individual is assigned to one processor and all individuals are evaluated simultaneously. Selection and mating are applied to a small neighborhood around each individual. The neighborhoods overlap, so that eventually the good characters of a superior individual can spread to the whole population. Hierarchical hybrids combine multiple-population GAs with single-population master-slave GAs or fine-grained GAs.

The single-population master-slave GAs executes parallel processing by not dividing the population, and using multiple processors only for the evaluation of individuals [1], [14]. This model may be used in a situation which spends too much time calculating the fitness of each individual. By adopting this model, calculated load may be able to distribute. Another characteristic is that it also implements a simpler algorithm than the other three models, and because there are no changes to the population due to parallelization, there will be no effects on the search for a solution.

A computational method proposed by Niwa *et al.* [31] adopts the single-population master-slave GAs as the upper level GA. In the computational method, a master only executes a reproduction operator and slaves evaluate the fitness of the individuals and performs a crossover operator and a mutation operator. Also, the computational method divides operations of the lower level GA and assigns them across multiple processors. By assigning calculations of fitness for individuals, a crossover operator and a mutation operator to multiple processors, calculation times are reduced. We call the computational method the computational method using master-slave GAs (MGA).

Moreover, Niwa *et al.* proposed computational methods through multiple-population GAs (distributed GAs) in order to solve a two-level 0-1 programming problem [26] and a decentralized two-level 0-1 programming problem [25]. We call these computational methods the computational methods using distributed GAs (DGA). In order to verify the effectiveness of DGA, numerical experiments are carried out. From the results, DGA is superior to MGA and the existing computational method through GAs [27], both from a standpoint of solution precision and required computational time.

In this paper, therefore, we employ computational methods thorough distributed GAs so as to reduce calculation times and obtain good approximate Stackelberg solutions.

D. Lower level GA avoidance procedures

We introduce a storage region shown in Fig. 2.

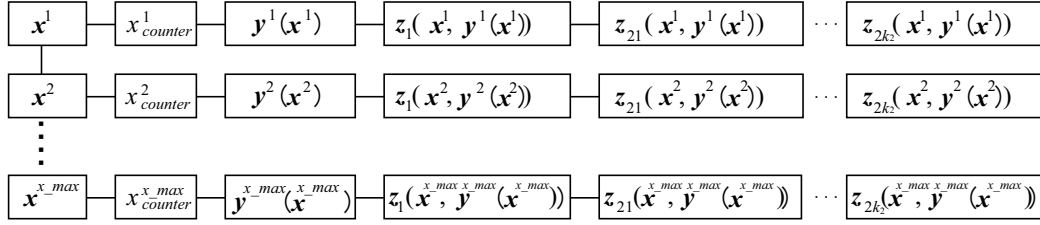


Fig. 2. Storage for saving $\mathbf{x}$ and $\mathbf{y}(\mathbf{x})$

Here, $\mathbf{x}^i, i = 1, \dots, x_max$ indicate the values of $\mathbf{x}$ already handled as individuals by the upper level GA, and $\mathbf{y}^i(\mathbf{x}^i), i = 1, \dots, x_max$ indicate the Pareto optimal responses associated with $\mathbf{x}^i$ obtained by the lower level GA. $x_counter^i \in \{1, 2, \dots, y_max\}, i = 1, \dots, x_max$ indicate the number of times that the lower level GA was used to find the Pareto optimal response $\mathbf{y}^i(\mathbf{x}^i)$ for $\mathbf{x}^i$. x_max indicates the maximum number of stored decisions $\mathbf{x}$, and y_max indicates the maximum number of times that the lower level GA can be repeatedly used to find the Pareto optimal response $\mathbf{y}^i(\mathbf{x}^i)$ for $\mathbf{x}^i$. $z_1(\mathbf{x}^i, \mathbf{y}^i(\mathbf{x}^i))$ is the value of the leader's objective function for $\mathbf{x}^i$ and $\mathbf{y}^i(\mathbf{x}^i)$. $z_{21}(\mathbf{x}^i, \mathbf{y}^i(\mathbf{x}^i)), \dots, z_{2k_2}(\mathbf{x}^i, \mathbf{y}^i(\mathbf{x}^i))$ are the values of the follower's objective functions for $\mathbf{x}^i$ and $\mathbf{y}^i(\mathbf{x}^i)$. By using the following algorithm, the number of applications of the lower level GA is reduced, and unnecessary calculation times eliminated.

Storage of Pareto optimal response $\mathbf{y}(\mathbf{x})$ and lower level GA avoidance procedures

- Step 1 If an individual $\bar{\mathbf{x}}$ of the upper level GA exists in $\mathbf{x}^i$, proceed to Step 2. If such an individual does not exist, then check if the number of $\mathbf{x}^i$ has reached x_max , and if so continue on to Step 3. If not, proceed to Step 4.
- Step 2 If $x_counter^i$ has reached y_max , then the saved $\mathbf{y}^i(\mathbf{x}^i)$ is returned to the upper level GA as the Pareto optimal response and the algorithm terminates. If not reached, proceed to Step 4.
- Step 3 Select the worst value $z_1(\mathbf{x}^i, \mathbf{y}^i(\mathbf{x}^i))$ from the stored $\mathbf{x}^i$, and take that $\mathbf{x}^i$ value as $\mathbf{x}^k$. After applying the lower level GA and thus obtaining the Pareto optimal response $\mathbf{y}(\bar{\mathbf{x}})$ for $\bar{\mathbf{x}}$, if $z_1(\mathbf{x}^k, \mathbf{y}^k(\mathbf{x}^k)) \geq z_1(\bar{\mathbf{x}}, \mathbf{y}(\bar{\mathbf{x}}))$, save $\bar{\mathbf{x}}, \mathbf{y}(\bar{\mathbf{x}}), z_1(\bar{\mathbf{x}}, \mathbf{y}(\bar{\mathbf{x}})), z_{21}(\bar{\mathbf{x}}, \mathbf{y}(\bar{\mathbf{x}})), \dots, z_{2k_2}(\bar{\mathbf{x}}, \mathbf{y}(\bar{\mathbf{x}}))$ in the storage region $\mathbf{x}^k$, and terminate the algorithm.
- Step 4 After obtaining the Pareto optimal response $\mathbf{y}(\bar{\mathbf{x}})$ for $\bar{\mathbf{x}}$ by applying the lower level GA, $\bar{\mathbf{x}}, \mathbf{y}(\bar{\mathbf{x}}), z_1(\bar{\mathbf{x}}, \mathbf{y}(\bar{\mathbf{x}})), z_{21}(\bar{\mathbf{x}}, \mathbf{y}(\bar{\mathbf{x}})), \dots, z_{2k_2}(\bar{\mathbf{x}}, \mathbf{y}(\bar{\mathbf{x}}))$ are saved, and the algorithm is terminated.

E. Reproduction

We first describe the reproduction operator of the upper level GA. In the upper level GA, the value of $\mathbf{x}$ obtained by decoding individuals in the upper level GA and the follower's Pareto optimal responses $\mathbf{y}(\mathbf{x}) \in P(\mathbf{x})$ obtained by applying the lower level GA are substituted into the objective function of the leader $z_1(\mathbf{x}, \mathbf{y})$, and the value of the evaluation function for each individual is thus obtained. Next, 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.

Next, we describe the reproduction operator of the lower level GA. In the lower level GA, the value of $\mathbf{y}$ obtained by decoding individuals in the lower level GA, and the given values of the decision variables in the upper level GA, $\mathbf{x}$, are substituted into the follower's objective functions $z_{2k}(\mathbf{x}, \mathbf{y}), k = 1, \dots, k_2$, and the value of the evaluation function for each individual is thus obtained. Then, in the first type of the proposed computational method, the reproduction operator of VEGA [36], [39] is adopted by using the value of the evaluation function for deriving multiple rational responses of the follower. Also, in the second type of the proposed computational method, the reproduction operator of NSGA-II [10] is applied by using the value of the evaluation function.

F. Algorithms for the proposed computational methods

Initially, an algorithm used in the first type of the proposed computational method after improvement can be summarized as follows. N_p denotes the number of processors.

- Step 1 For each processor $q, q = 1, \dots, N_p$, apply the upper level GA operations on Step1 through Step7. Generate N_u individuals at random to form the initial population. Set the generation index $t_{uq} := 0$ and determine the value of the maximal generation T_u .
- Step 2 For each individual $\mathbf{x}$ in the upper level GA, determine whether or not to apply the operations of the lower level GA. If it is applied, then apply the lower level GA operations in Step 2-a through Step 2-c,

and obtain a Pareto optimal response $\mathbf{y}(\mathbf{x})$. If it is not applied, then a stored $\mathbf{y}(\mathbf{x})$ is taken as a Pareto optimal response, and proceed to Step 4.

Step a Randomly generate N_l lower level GA individuals $\mathbf{y}$, and take these as the initial population of the lower level GA. Set the generation index $t_l := 0$ and determine the value of the maximal generation T_l . Proceed to Step 2-b.

Step b By using $\mathbf{x}$ given as the upper level GA individual and $\mathbf{y}$ generated by the lower level GA, the follower's objective function values are calculated. As the reproduction operator, the parallel selection method is applied. Preserve non-dominated solutions in the population. Proceed to Step 2-c.

Step c If t_l has exceeded the maximal generation T_l , take the individual with the best fitness value as the optimal individual $\mathbf{y}(\mathbf{x})$, and proceed to Step 3. Otherwise, apply a crossover operator and a mutation operator to each lower level GA individual. Select individuals from the population at random, and exchange the individuals for the non-dominated solutions (individuals). Let $t_l = t_l + 1$, and proceed to Step 2-b.

Step 3 By using the follower's Pareto optimal response $\mathbf{y}(\mathbf{x})$ obtained by operation of the lower level GA and the individual $\mathbf{x}$ of the upper level GA, calculate objective function values for the leader and the follower. Perform the procedures required to save $\mathbf{x}$ and its Pareto optimal response $\mathbf{y}(\mathbf{x})$ to the storage region and proceed to Step 4.

Step 4 Calculate the leader's objective function value for each upper level GA individual $\mathbf{x}$, 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 maximal generation T_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 $(\mathbf{x}, \mathbf{y})$. Otherwise, proceed to Step 6.

Step 6 A reproduction operator is performed using the fitness values of each individual of the upper level GA. Apply a crossover operator and a mutation operator to each upper level GA individual, and proceed to Step 7.

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

Next, we show an algorithm used for the second type of the proposed computational method. Because the difference of

two computational methods is only procedures to apply the lower level GA, an algorithm for the lower level GA is shown as follows.

Step a Set the generation index $t_l := 0$ and determine the value of the maximal generation T_l . Generate N_l lower level GA individuals $\mathbf{y}$ at random, and take these individuals as the initial population $P(t_l)$ of the lower level GA. Proceed to Step 2-b.

Step b If t_l has exceeded the maximal generation T_l , take the individual with the best fitness value as the optimal individual $\mathbf{y}(\mathbf{x})$, and proceed to Step 3. Otherwise, apply a crowded tournament selection operator, a crossover operator and a mutation operator to each lower level GA individual in $P(t_l)$ and form a population $Q(t_l)$ by use of these individuals.

Step c Let $R(t_l) = P(t_l) \cup Q(t_l)$. By using $\mathbf{x}$ given as an individual of the upper level GA and $\mathbf{y}$ given as an individual of the lower level GA in $R(t_l)$, the follower's objective function values are calculated. Based on these values, a non-dominated sorting is applied as a reproduction operator. Then, create the new population $P(t_l + 1)$ of the next generation $t_l + 1$. Let $t_l = t_l + 1$, and proceed to Step 2-b.

By using these computational methods, it is possible to obtain the Stackelberg solutions to the problem (4) and (5).

IV. CONCLUSION

This paper has focused on a multiobjective two-level 0-1 programming problem in which there is not coordination between the decision maker at the upper level and the decision maker at the lower level. We have considered two kinds of situations based on anticipation of the decision maker at the upper level; an optimistic anticipation and a pessimistic anticipation. Mathematical programming problems for obtaining the Stackelberg solutions based on two kinds of anticipation have been shown and two types of computational methods for obtaining the Stackelberg solutions have been proposed. In order to demonstrate the effectiveness of the proposed methods, it is intended to conduct numerical experiments into both the solution precision and computation time, and thus compare the proposed computational methods with previous methods.

REFERENCES

- [1] D. Abramson and J. Abela, "A parallel genetic algorithm for solving the school timetabling problem," *Proceedings of the Fifteenth Australian Computer Science Conference*, Vol. 14, pp. 1-11, 1992.
- [2] G. Anandalingam, "A mathematical programming model of decentralized multi-Level systems," *Journal of Operational Research Society*, Vol. 39, pp. 1021-1033, 1988.
- [3] 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.
- [4] J. Bard and J. Moore, "A branch and bound algorithm for the bilevel programming problem," *SIAM Journal on Scientific and Statistical Computing*, Vol. 11, pp. 281-292, 1990.
- [5] J. Bard and J. Moore, "The mixed integer linear bilevel programming problem," *Operations Research*, Vol. 38, pp. 911-921, 1990.

- [6] J. Bard and J. Moore, "An algorithm for the discrete bilevel programming problem," *Naval Research Logistics*, Vol. 39, pp. 419–435, 1992.
- [7] W.F. Bialas and M.H. Karwan, "Two-level linear programming," *Management Science*, Vol. 30, pp. 1004–1020, 1984.
- [8] E. Cantú-Paz: *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer Academic Publishers, Norwell, Massachusetts, 2000.
- [9] N. Chaiyaratana, T. Piroonratana and N. Sangkawelert, "Effects of diversity control in single-objective and multi-objective genetic algorithms," *Journal of Heuristics*, Vol. 13, No. 1, pp. 1–34, 2007.
- [10] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 182–197, 2002.
- [11] K. Deb and T. Goel, "Controlled elitist non-dominated sorting genetic algorithms for better convergence," *Lecture Notes in Computer Science*, Vol. 1993, pp. 67–81, 2001.
- [12] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 182–197, 2002.
- [13] K. Deb, L. Thiele, L. Laumanns and E. Zitzler, "Scalable test problems for evolutionary multi-objective optimization," *Evolutionary Multiobjective Optimization*, pp. 105–145, 2005.
- [14] T.C. Fogarty and R. Huang, "Implementing the genetic Algorithm on Transputer Based Parallel Processing Systems," *Proceedings of the First International Conference on Parallel Problem Solving from Nature*, pp. 145–149, 1990.
- [15] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: formulation, discussion and generalization," *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 416–423, 1993.
- [16] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms-Part 1: A united formulation," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, Vol. 28, No. 1, pp. 26–37, 1998.
- [17] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, Massachusetts, 1989.
- [18] 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.
- [19] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975, MIT Press, Cambridge, 1992.
- [20] J. Horn and N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," *Proceedings of the First IEEE Conference on Evolutionary Computation*, pp. 82–87, 1994.
- [21] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolutionary Programs*, Third, revised and extended edition, Springer-Verlag, 1996.
- [22] I. Nishizaki and M. Sakawa, "Stackelberg solutions to multiobjective two-level linear programming problems," *Journal of Optimization Theory and Applications*, Vol. 103, pp. 161–182, 1999.
- [23] 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, 2007, (in Japanese).
- [24] K. Niwa, T. Hayashida and M. Sakawa, "Multiobjective Two-Level Integer Programming through Genetic Algorithms," *Joint 5th International Conference on Soft Computing and Intelligent Systems and 11th International Symposium on Advanced Intelligent Systems Conference Proceedings*, pp. 1038–1043, 2010.
- [25] K. Niwa, T. Hayashida, M. Sakawa and Y. Yang, "Decentralized Two-Level 0-1 Programming through Distributed Genetic Algorithms," *Lecture Notes in Engineering and Computer Science: Proceedings of The International MultiConference of Engineers and Computer Scientists 2010*, Vol. 3, pp. 2163–2168, 2010.
- [26] K. Niwa, T. Hayashida and M. Sakawa, "Computational Methods for Two-Level 0-1 Programming Problems through Distributed Genetic Algorithms," *Journal of Telecommunications and Information Technology*, Vol. 2, pp. 78–87, 2010.
- [27] 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.
- [28] K. Niwa, I. Nishizaki and M. Sakawa, "Computational methods for decentralized two-level zero-one programming problems through genetic algorithms," *Proceedings of the Eighth International Fuzzy Systems Association World Congress*, Vol. 2, pp. 986–990, 1999.
- [29] K. Niwa, I. Nishizaki and M. Sakawa, "Multiobjective two-level zero-one programming problems through genetic algorithms," *Proceedings of the Second Asia-Pacific Conference on Industrial Engineering and Management Systems*, pp. 493–496, 1999.
- [30] 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.
- [31] K. Niwa, I. Nishizaki and M. Sakawa, "Two-Level 0-1 Programming through Parallel Genetic Algorithms," *International MultiConference of Engineers and Computer Scientists 2009 Proceedings*, Vol. 2, pp. 2000–2005, 2009.
- [32] 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. 8, pp. 149–158, 1997.
- [33] M. Sakawa and K. Kato, "Integer Programming through Genetic Algorithms with Double Strings Based on Reference Solution Updating," *IECON-2000*, pp. 2744–2749, 2000.
- [34] M. Sakawa, M. Tanaka, *Genetic Algorithms*, Asakura Publishing, 1995, (in Japanese).
- [35] K. Niwa, I. Nishizaki and M. Sakawa, "Multiobjective two-level zero-one programming problems through genetic algorithms," *Proceedings of the Second Asia-Pacific Conference on Industrial Engineering and Management Systems*, pp. 493–496, 1999.
- [36] J.D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," *Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications*, Lawrence Erlbaum Associates Publishers, New Jersey, pp. 93–100, 1985.
- [37] M. Simaan and J.B. IR. Cruz, "On the Stackelberg strategy in nonzero-sum games," *Journal of Optimization Theory and Applications*, Vol. 11, pp. 533–555, 1973.
- [38] N. Srinivas and K. Deb, "Multi-objective optimization using non-dominated sorting genetic algorithms," *Evolutionary Computation*, Vol. 2, No. 3, pp. 221–248, 1994.
- [39] H. Tamaki, M. Mori and M. Araki, "Generation of a set of Pareto-optimal solutions by genetic algorithms," *Transactions of the Society of Instrument and Control Engineers*, Vol. 31, pp. 1185–1192, 1995 (in Japanese).
- [40] 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.
- [41] E. Zitzler, K. Deb and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical Results," *Evolutionary Computation*, Vol. 8, No. 2, pp. 173–195, 2000.
- [42] E. Zitzler, M. Laumanns and L. Thiele: SPEA2: "Improving the strength Pareto evolutionary algorithm for multiobjective optimization," *Evolutionary Methods for Design Optimization and Control*, pp. 95–100, 2002.
- [43] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 4, pp. 257–271, 1999.