Multilevel Redundancy Allocation Optimization Using Hierarchical Genetic Algorithm

Ranjan Kumar, *Student Member, IEEE*, Kazuhiro Izui, Yoshimura Masataka, and Shinji Nishiwaki

***Abstract*—This paper proposes a generalized formulation for multilevel redundancy allocation problems that can handle redundancies for each unit in a hierarchical reliability system, with structures containing multiple layers of subsystems and components. Multilevel redundancy allocation is an especially powerful approach for improving the system reliability of such hierarchical configurations, and system optimization problems that take advantage of this approach are termed multilevel redundancy allocation optimization problems (MRAOP). Despite the growing interest in MRAOP, a survey of the literature indicates that most redundancy allocation schemes are mainly confined to a single level, and few problem-specific MRAOP have been proposed or solved. The design variables in MRAOP are hierarchically structured. This paper proposes a new variable coding method in which these hierarchical design variables are represented by two types of hierarchical genotype, termed ordinal node, and terminal node. These genotypes preserve the logical linkage among the hierarchical variables, and allow every possible combination of redundancy during the optimization process. Furthermore, this paper developed a hierarchical genetic algorithm (HGA) that uses special genetic operators to handle the hierarchical genotype representation of hierarchical design variables. For comparison, the customized HGA, and a conventional genetic algorithm (GA) in which design variables are coded in vector forms, are applied to solve MRAOP for series systems having two different configurations. The solutions obtained when using HGA are shown to be superior to the conventional GA solutions, indicating that the HGA here is especially suitable for solving MRAOP for series systems.**

***Index Terms*—Hierarchical genetic algorithm, hierarchical genotype, multilevel redundancy allocation, reliability optimization.**

NOTATION

U_s	system level unit
R_s	system reliability
U_i	i-th unit; a common name for system, subsystem, and component.
R_i	reliability of U_i
x_i	number of components used in U_i
$\boldsymbol{x}$	set of design variables x_i
$f(\boldsymbol{x})$	reliability function
$C(\boldsymbol{x})$	cost function
n_i	number of sub-units in U_i
U_{i,n_i}	n_ith sub-unit of U_i
$U_{i,m}^j$	j-th redundant unit of m-th sub-unit of U_i
$R_{i,m}^j$	reliability of $U_{i,m}^j$
$x_{i,m}^j$	number of sub-units of unit $U_{i,m}^j$, a nonnegative integer
$C_{i,m}^j$	cost of unit $U_{i,m}^j$
C_i	cost of i-th unit
C_s	system cost
C_0	threshold cost
λ_i	additional costs of i-th unit when adding a redundant unit to a unit
N_i	ordinal genotype node of i-th unit
N_t	terminal genotype node of i-th unit

ACRONYM[1]

GA	genetic algorithm
HGA	hierarchical genetic algorithm
MRAOP	multilevel redundancy allocation optimization problem
MS	multilevel series
RBD	reliability block diagram

Manuscript received September 30, 2007; revised April 05, 2008 and May 23, 2008; accepted May 25, 2008. First published November 07, 2008; current version published December 02, 2008, Associate Editor: G. Levitin.

The authors with the Department of Aeronautics and Astronautics, Kyoto University, Kyoto 606-8325, Japan.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TR.2008.2006079

I. INTRODUCTION

THE demand for higher reliability tends to make system designs increasingly complex. Typical systems contain multiple levels, with the entire system at the top level, subsystems at lower levels inside the system, and down to the components at the lowest levels inside the various subsystems. Hierarchical systems such as these are termed multilevel systems, and their reliability depends on the reliability values of lower subsystems. For example, if the lower subsystems of a bi-level system are connected serially, the system reliability is the product of the reliability values of the lower subsystems. Fig. 1 illustrates a schematic diagram of the multilevel configuration of a hierarchical reliability block diagram in a hierarchical product design.

The system reliability of a multilevel design configuration is usually optimized by allocating appropriate redundancy to less reliable subsystems or components at different levels, subject to

[1]The singular and plural of an acronym are always spelled the same.

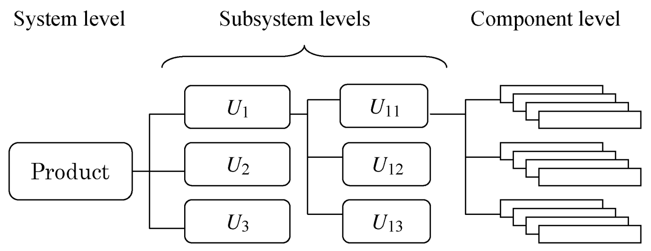


Fig. 1. A multilevel RBD.

certain constraints. This optimization technique is called multilevel redundancy allocation optimization (MRAO), and subsequently formulated problems are called multilevel redundancy allocation optimization problems (MRAOP).

MRAOP are particularly attractive because real world systems and products are increasingly complex, and the system reliability of the multilevel configurations of these complex designs can be significantly improved by using multilevel redundancy allocation techniques. Multilevel redundant designs are increasingly prevalent in many practical systems, such as communication systems, computing systems, control systems, and critical power systems [1]. Techniques for implementing redundancy span a wide spectrum in the design space, and can create high reliability systems. Moreover, recent progress in miniaturization has made it easier to provide redundancy at all levels, ranging from the system level down to component levels. This approach can boost system reliability remarkably because redundancy can be distributed to any component at any level without structural constraints.

The optimization of system reliability using multilevel redundancy allocation is widely practiced in industry. Most integrated memory circuits, and VLSI chips that include internal memory blocks, currently use a hierarchical redundancy allocation scheme to enhance reliability, and chip yields. Also, a significant advantage of multilevel or hierarchical allocation is that it permits a modular scheme of redundancy allocation. Koren *et al.* [2] described how such modular schemes are particularly applicable when designing fault-tolerant or self-repairing semiconductor devices. Multilevel architectures that provide physical protection are now commonly implemented to increase the survivability of real systems in adverse conditions [3]. For protecting archived data, multilevel redundant designs in redundant arrays of inexpensive disks (RAID) that provide fault tolerance against disk failures outperform other RAID designs [4]. Several examples of multilevel RBD structures can be found in the literature, such as hierarchical series, hierarchical series parallel, and others [5], [6].

Almost all previous research in redundancy allocation optimization problems has focused on system configurations such as series-parallel, parallel-series, general networks, *k-out-of-n*: G(F), and other unspecified configurations, classified by Tillman, Hwang, & Kuo [7]. Kuo & Zuo provided good details concerning optimal reliability modeling [8], and the review paper by Kuo & Prasad [9] presents an overview of system reliability optimization. However, a comprehensive examination of this literature reveals that multilevel redundancy allocation problems are seldom addressed in terms of the detailed modeling or appropriate optimization techniques that such problems actually require. Also, attention paid to redundancy allocation is mainly confined to a single level, principally due to the notion that single-level redundancy yields better system reliability. We feel that this is not always the case. Boland & EL-Neweihi [10] demonstrated that this result does not hold in cases of redundancy configurations using non-identical parts.

According to Chern, redundancy allocation optimization problems are nonlinear integer programming problems, and NP-hard [11]. Besides being NP-hard, MRAOP qualify as hierarchical optimization problems [12]. The optimization of such hierarchical optimization problems beyond two levels, however, is more difficult using heuristics or exact algorithms. This is because multilevel allocation optimization problems generate a very large search space, and searching for optimal solutions using exact methods or heuristics will necessarily be extremely time consuming. Therefore, metaheuristic algorithms, particularly genetic algorithm (GA), are suitable for solving the multilevel redundancy allocation optimization. The seminal work by Goldberg [13] demonstrated that GA are very useful for solving complex discrete optimization problems, and the multiple solutions that GA provide allow considerable, valuable flexibility when choosing the best solution. This is one reason that GA is popularly applied to a variety of reliability optimization problems [14]–[20]. However, none of the above-cited research specifically aims to optimize system reliability beyond two-level systems, and their subsystems.

Recently, the growing research interest in multilevel reliability modeling, and multilevel optimization using GA, is reflected in the literature, due to the practical importance of these techniques. Levitin [3] proposed an algorithm for solving multilevel protection cost minimization problems subject to survivability constraints. This algorithm is based on a universal generating function technique used for system survivability evaluation, and on a genetic algorithm used as an optimization engine. Later, Yun & Kim [21] proposed a restricted multilevel redundancy allocation model, and optimized a three-level series redundancy allocation problem using a customized GA. However, this model allows redundancy allocation to only one unit at a given level in a direct line, which is defined as a set of units in which every unit except the system has a parent unit, and no other cousin units, the other units at the same level, are present in that set. Direct line concepts are explained by an example in a later section of this paper. The purpose of using direct lines is to transform the multilevel design variables into vector representations, because conventional GA use one-dimensional representations of design variables. Unfortunately, the additional constraints imposed when transforming hierarchical design variables into vector design variables artificially constrict the feasible design region, often leading to suboptimal solutions.

Several genetic algorithms use a hierarchical approach to solve classes of hierarchical optimization problems. The hierarchical features offer the potential to address large problems efficiently [22]. De Jong *et al.* [23] delineated classes of hierarchical problems, and described a framework for Hierarchical Genetic Algorithms (HGA), genetic algorithms that can exploit the structure present in hierarchical problems to achieve improvements in efficiency. These HGA exploit hierarchical

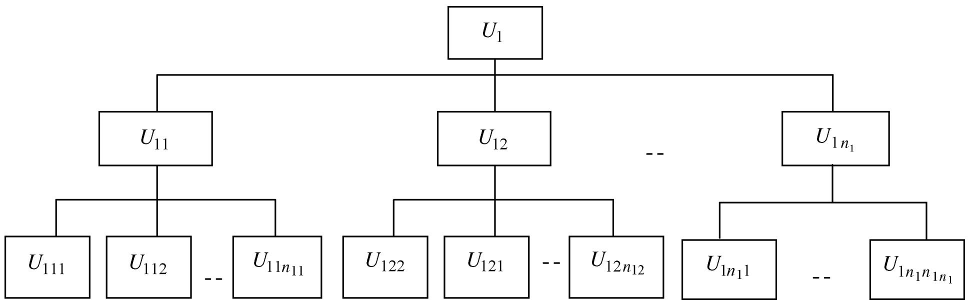


Fig. 2. A general multilevel redundancy allocation configuration.

features in different ways depending on the problem, such as the use of a fitness-based hierarchy of populations [24], problem-specific subdivision of an algorithm into multiple levels [25], and the use of hierarchical representation by using control genes that regulate other genes [26]. Sefrioui & Periaux [27] developed HGA in which they used a hierarchical topology for the layout of sub-populations, achieving higher efficiency than conventional GA. Further, Yoshimura & Izui [28] proposed a genetic algorithm in which hierarchical genotype coding representation is used to exactly express the internal structure, and related hierarchical details. New crossover and mutation operators have been developed to handle these hierarchical genotypes during optimization processes.

The genotype coding representation used in the genetic algorithms proposed by Yoshimura & Izui [28] aims to represent the hierarchical design variables in design optimization problems for mechanical structures. However, the MROAP require a problem specific coding method for handling the logical linkages among the hierarchical design variables, and thus the coding scheme proposed by them cannot be applicable directly to solving MRAOP. Therefore, this paper proposes a new variable coding method for the HGA first proposed by Yoshimura & Izui [28]. In this coding method, the phenotypes of hierarchical design variables are coded using two newly designed hierarchical nodal genotypes: the ordinal, and the terminal. These two nodal genotypes can be used as building blocks to codify most of the MRAOP hierarchical configurations. Thus, there is no need to transform the hierarchical design variables because these nodal genotypes preserve the exact hierarchical relationships within each design variable. The novelty of these hierarchical nodal genotypes is that they can express every possible combination of multilevel redundancy allocation, so that the optimization has a high probability of yielding nearly global optimal solutions.

The rest of the paper is organized as follows. Section II describes the detailed mathematical formulation of our multilevel series redundancy allocation optimization model. In Section III, the HGA concepts are explained, and a HGA coding method for HS problems is proposed. In Section IV, we solve two series problems, a three-level problem, and a four-level problem. The optimal solutions obtained when using a conventional GA are compared with those obtained with the custom-coded HGA, and the resulting configurations are presented. Finally, the results are discussed in Section V, while Section VI concludes the paper.

II. Problem Description

A. Multilevel Series Redundancy Allocation Model

The proposed redundancy model contains multiple hierarchical levels. The system level is the topmost level, and the component level is the lowest. Subsystem or module levels are located between the top, and second lowest levels. Each system, module, and component is here termed a unit. Every unit except components can have any number of subordinate elements, such as modules that make up a system, or components that make up a module. These subordinate elements are called sub-units, whereas the next highest hierarchical unit of a sub-unit is called a parent unit. The proposed redundancy allocation model can provide redundancy for all units of a multilevel reliability system. Fig. 2 represents the schematic diagram of a generalized hierarchical redundancy allocation model. The connecting lines in the diagram imply the logical relationships among the units at different levels, relationships that may be in series, in parallel, or combinations of these two. Redundancy at all levels is assumed to be active, and failures are s-independent.

Fig. 3 explains the redundancy allocation scheme in a bi-level series system, and the distinction between sub-units and redundant units. In Fig. 3(a), U_1 is a unit at the system level that has two sub-units U_{11} & U_{12} at the next lowest level in the basic configuration. Fig. 3(b) illustrates the redundancy allocation in U_1, which has two redundant units at system level U_1^1 & U_1^2. Similarly, sub-units U_{11}, and U_{12} have 3, and 1 redundant units, respectively, in parent unit U_1^1, and so on.

Thus, in a multilevel redundancy allocation model, each unit U_i can have x_i redundant units, and n_i sub-units, so there are $n_i x_i$ sub-units in the level below a parent unit. The sub-units n_i are different for each parent unit in the model described here. For example, as shown in Fig. 2, U_1 is a system unit containing U_{11} to U_{1n_1} units as modules at its next lowest hierarchical level. Similarly, the U_{11} module contains n_{11} sub-units as modules or components at its next lowest level, represented as U_{111} to $U_{11n_{11}}$, which is actually the second level of the system hierarchy. This structure is replicated until the lowest level of the system hierarchy is reached.

Thus, the reliability R_i of unit U_i for multilevel series configurations can be calculated using

$$R_i = \prod_{m=1}^{n_i} \left[1 - \prod_{j=1}^{x_i} \left(1 - R_{i,m}^j \right) \right] \tag{1}$$

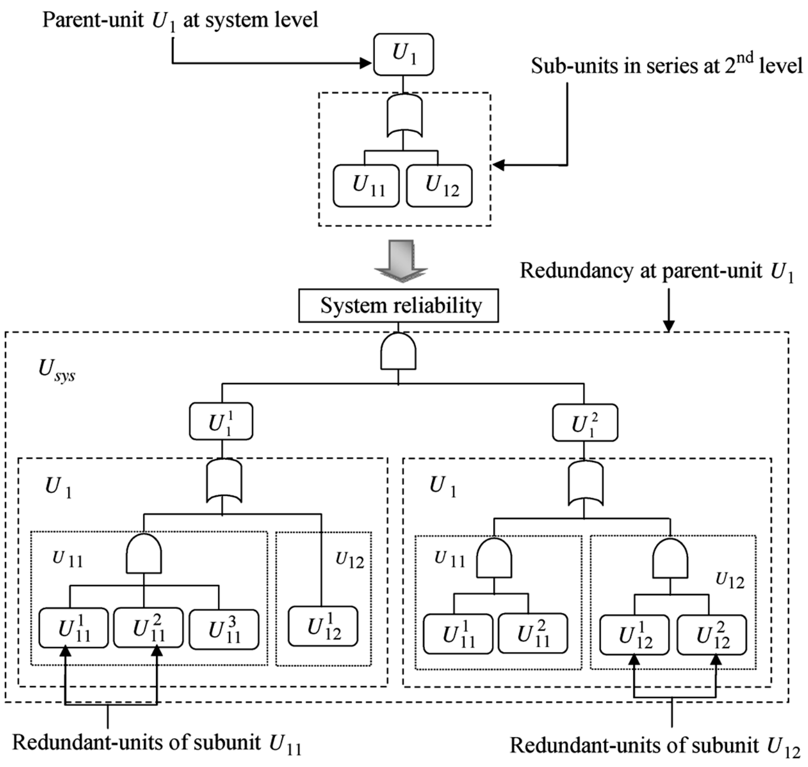

Parent-unit U_1 at system level
U_1
Sub-units in series at 2nd level
U_{11}
U_{12}
Redundancy at parent-unit U_1
System reliability
U_{sys}
U_1^1
U_1^2
U_1
U_{11}
U_{12}
U_{11}^1
U_{11}^2
U_{11}^3
U_{12}^1
U_{12}^2
Redundant-units of subunit U_{11}
Redundant-units of subunit U_{12}

Fig. 3. An example of redundancy allocation in bi-level series configuration.

where $R_{i,m}^j$ are reliability values for sub-units $U_{i,m}^j$, a unit in the j-th redundant unit of the m-th sub-unit of U_i. Each $R_{i,m}^j$ value is calculated using (1) at the level immediately below the unit, and these calculations are recursively iterated to the level just above the very lowest hierarchical level. At the very lowest level, where there are no sub-units belonging to unit U_i, the reliability can be obtained as

$$R_i = 1 - \prod_{j=1}^{x_i} \left(1 - R_i^j\right) \tag{2}$$

The multilevel reliability allocation model presented here allows redundancy for any unit at any level, and it is thus possible to achieve redundancy schemes that function at both the component, and modular levels. This mixed redundancy scheme allows the units to have redundancy not only at the same level, but also simultaneously for sub-units at lower levels.

The cost constraint of a multilevel redundancy allocation model also reveals hierarchical relationships among the multilevel units. The system cost is essentially the sum of the cost of subsystems and modules, and the cost of a module is the sum of all modules or component costs therein, when there are parallel units at the immediate lower level. In practical systems, it is assumed that multilevel redundancy incurs additional cost due to the adding or duplication of redundant units to modules, and the increased number of components. In general, the redundancy cost of R_i can be expressed mathematically as

$$C_i = \sum_{m=1}^{n_i} \sum_{j=1}^{x_i} C_{i,m}^j \times x_i + additional\ costs \tag{3}$$

Note that there are definite advantages to using modular redundancy in multilevel redundancy allocation, because the cost of adding, duplicating, or repairing a module is lower than carrying out a similar action upon a component. This result holds because the lower the level in a system, the more costly the repair job.

B. Redundancy Allocation Optimization Problem

The redundancy allocation optimization problem in a reliability system consisting of a set of design variables is expressed as

$$\text{Maximize} \quad R_s = f(\boldsymbol{x}) \tag{4}$$
$$\text{Subject to} \quad C(\boldsymbol{x}) \le C_0 \tag{5}$$

In a set of design variables $\boldsymbol{x}$, each design variable has a minimum, and maximum redundancy value. C_0 is a given, fixed positive value for the cost constraint. For example, the problem of optimizing a 2-level series redundancy allocation, as shown in Fig. 3, can be stated mathematically as

$$\begin{aligned} R_{sys} = \Bigg[1 - \Bigg(1 - \Bigg\{ \Bigg(1 - \prod_{j=1}^{3} \left(1 - R_{11}^j\right) \Bigg) \left(R_{12}^1\right) \Bigg\} \Bigg) \\ \times \Bigg(1 - \Bigg\{ \Bigg(1 - \prod_{j=1}^{3} \left(1 - R_{11}^j\right) \Bigg) \\ \times \Bigg(1 - \prod_{j=1}^{2} \left(1 - R_{12}^j\right) \Bigg) \Bigg\} \Bigg) \Bigg] \end{aligned} \tag{6}$$

The cost function used in this paper for the cost constraint is described by (3).

TABLE I
HIERARCHICAL GENOTYPE REPRESENTATION

	Ordinal genotype node N_i	Terminal genotype node N_{t_i}
Design variable	$x_{i,m}^{j}$: the number of sub-unit for the m-th unit	
Parameter	k_i: the redundancy for unit U_i n_i: the number of sub-unit	k_i: the redundancy for component U_i r_i: unit reliability c_i: unit cost

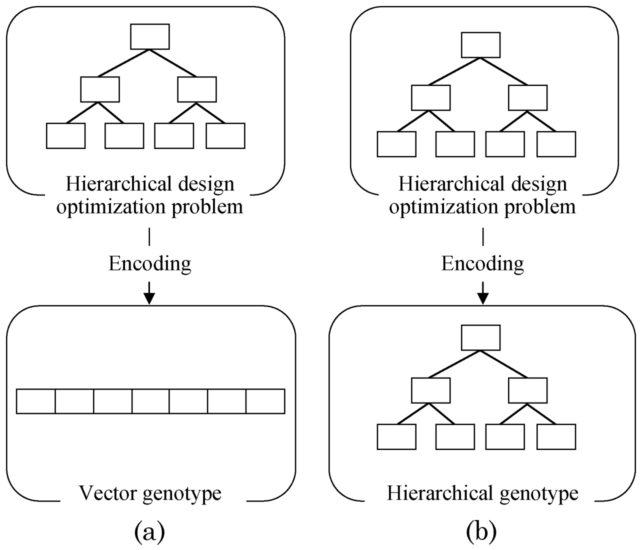

Hierarchical design optimization problem
Encoding
Vector genotype
(a)
Hierarchical design optimization problem
Encoding
Hierarchical genotype
(b)

Fig. 4. Representation schemes of design variables in GA, and HGA: (a) conventional GA; (b) HGA.

III. HIERARCHICAL GENETIC ALGORITHM

A HGA [28] is an advanced genetic algorithm that can represent hierarchical and constraint relationships among design variables using hierarchical genotypes, and can optimize hierarchical problems in a single optimization process. This HGA is further customized with a new variable coding method, and subsequently applied to solve the MRAOP in this paper. Fig. 4 illustrates that conventional GA [13] use vector genotype structures, in contrast to HGA that use hierarchical genotype structures.

The hierarchical redundancy allocation optimization problems here involve hierarchical relationships among design variables. Such hierarchical relationships can be handled well using hierarchical genotype representation. Because the HGA has special types of genotype structures, new crossover, and mutation operators have to be applied. The HGA allows lower branches of the hierarchical structure to be exchanged, in addition to the exchange of genes. Using such genetic operations, new individuals are produced, and optimal hierarchical structures can then be obtained.

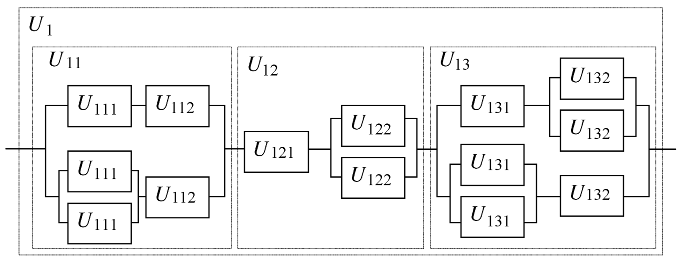
A. Solution Encoding

A hierarchical genotype is represented using two types of nodes, ordinal, and terminal nodes, as shown in Table I. Ordinal node N_i corresponds to redundancy unit U_i, and is characterized by several parameters, and design variables. Parameters k_i, and n_i stand for the redundancy of unit U_i, and the number of sub-units, respectively. Here, k_i is given by a design variable at an upper node, while the parameter n_i is a fixed value that depends on the optimization problems to be solved. $x_{i,m}^{j}$ is a design variable denoting the redundancy for the m-th sub-unit of the j-th redundancy unit, where j varies from 1 to k. Therefore, there are $n_i k_i$ design variables in unit U_i. A terminal node N_{t_i} corresponds to one of the lowest units, and incorporates design variable k_i, unit reliability r_i, and unit cost c_i. Because there are no sub-units, this terminal node does not contain parameter n_i, or design variable $x_{i,m}^{j}$. Using these two genotypes, all possible optimal solutions for series reliability allocation problems can be represented.

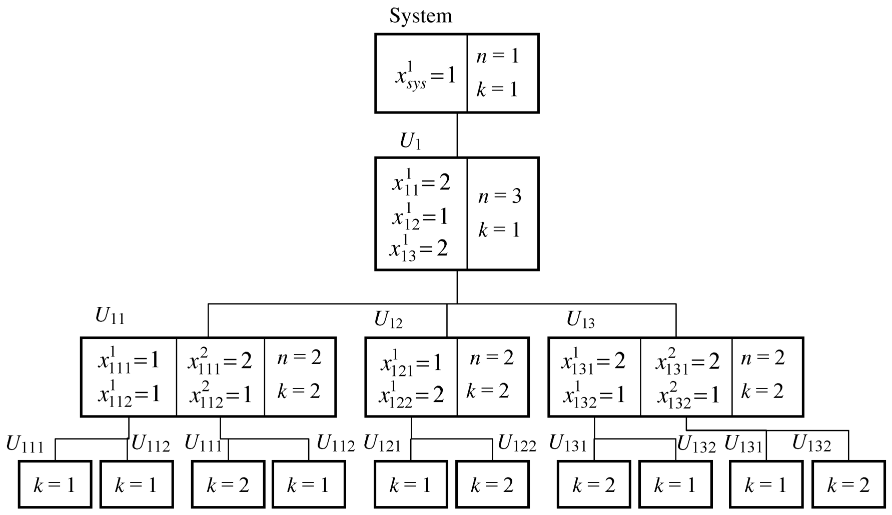
Furthermore, the ordinal, and the terminal genotypes each have two functions, namely, reliability, and cost. When the reliability function in the ordinal genotype is called, a calculation is conducted using (1). The particular equation selected depends on whether the unit is in series, or in parallel. When calculating either of these two equations, the reliability values of the lower units, $R_{i,m}^{j}$, are required; and these are obtained by calling the reliability function of the lower units. Finally, the reliability function of the terminal genotype returns its unit reliability r_i. Thus, the reliability functions are recursively called, and the total system reliability can be obtained. Similarly, the system cost can be obtained by calling the cost function embedded in each genotype.

Fig. 5 illustrates an example of the genotype encoding for a three-level series redundancy configuration. Fig. 5(a) shows an optimal redundancy configuration for a system U_1 consisting three modules, U_{11}, U_{12}, and U_{13}, at the second level. The ordinal, and terminal nodes are assigned to represent modules, and component units at each level. Note that unit features, such as the redundancy and configuration, series or parallel, are expressed in the corresponding upper node.

The HGA example shown in Fig. 5(b) illustrates that genotypes using fixed arrays, which are frequently used in various optimization problems, are not applicable to this problem because the number of design variables varies according to the number of redundant units. In other words, the number of genes varies dynamically based on the proposed solution configuration. In this case, the two design variables, x_{111}^{1}, and x_{111}^{2}, represent the redundancy of U_{111}, because there are two redundant units for U_{11}, which is the unit above U_{111} in the hierarchy. If the number of redundant units for U_{11} increases, the number of design variables for U_{111} will also increase. The solution encoding scheme proposed in this paper can successfully represent different numbers of design variables at every hierarchical level.



(a)



(b)

Fig. 5. Hierarchical genotype representation in system U_1. (a) An example of a multilevel reliability system U_1; (b) design variable values at each ordinal, and terminal node.

B. Objective Function

A penalty function method has been applied to transform the constrained problem into an unconstrained problem, by penalizing infeasible solutions via a penalty term added to the objective function for any violation of the constraints. In this paper, we used Gen & Cheng's method [29], which applies a severe penalty to infeasible solutions. The fitness function, $eval(\boldsymbol{x})$, is calculated using

$$eval(\boldsymbol{x}) = f(\boldsymbol{x}) \times p(\boldsymbol{x}) \tag{7}$$

where, $f(\boldsymbol{x})$, $p(\boldsymbol{x})$, and $\boldsymbol{x}$ are the system reliability, penalty function, and a set of design variables, respectively. We calculate the value of $p(\boldsymbol{x})$ using Gen & Cheng's penalty function for each individual; and for highly constrained optimization problems, the infeasible solutions occupy relatively large portions of the population at each generation. The penalty approach here adjusts the ratio of penalties adaptively at each generation to achieve a balance between the preservation of information, the selective pressure for infeasibility, and the avoidance of excessive penalization.

C. Crossover Operators

Crossover operations between individuals are conducted among each corresponding set of genes, using a two-step procedure. For the initial step, any other individual is first selected as the crossover partner, and crossover operators then exchange the corresponding genes of the two individuals. Here, when a gene of an alternative for a substructure is exchanged with the corresponding gene of another alternative, all corresponding lower substructures are also exchanged, to preserve consistency in the selection of alternatives. If this operation were not conducted in this way, meaningless lower structures might be generated in the lower positions of the exchanged substructures. The algorithmic procedures are as follows.

Step 1 Select two individuals for crossover operations, then find the set of genes at the highest level of the multilevel structural system for each of the two individuals, and start the crossover operation with probability p_{c_1}.

Step 2.1 If the gene $x^j_{i,m}$ of individual 1, and that of individual 2, are different, then conduct a crossover operation for $x^j_{i,m}$ with probability p_{c_2}. This operation is the same as a uniform crossover of simple genetic algorithms with p_{c_2} set to 0.5. Then, proceed to Step 2.3. If crossover operations are not conducted, proceed to Step 2.4. If the genes of both individuals are the same, proceed to Step 2.2.

Step 2.2 If $x^j_{i,m}$ contains a subordinate set of genes, it will be examined for possible crossover operations in Step 2.1. Otherwise, proceed to Step 2.4.

Step 2.3 When $x^j_{i,m}$ genes are exchanged between individuals 1 and 2, the lower substructures of each individual are also exchanged.

Step 2.4 Increment m by 1. When $m > n$, set $m = 1$, and increment j by 1. When $j > k_i$, end the crossover

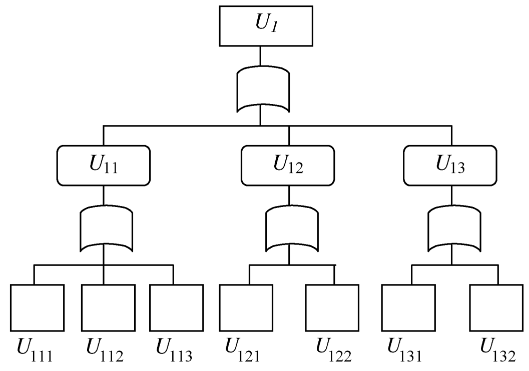


Fig. 6. *Problem-A* (a three level MS system [21]).

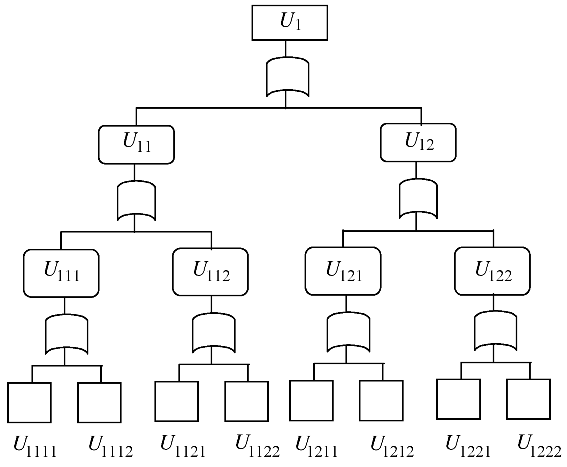


Fig. 7. *Problem-B* (a four level MS system).

operations because the set of genes has been exhausted, and return to the crossover operations for the parent set of genes.

D. Mutation Operators

In mutation operations, mutation operators are first applied to the set of genes at the highest level of the multilevel structural system, and mutation operators are recursively applied to their child sets of genes in the same way as for crossover operators. The algorithmic procedures are as follows.

Step 1 Examine the substructure at the highest level.

Step 2.1 Determine whether or not a mutation operation should be conducted, with mutation probability p_m for the gene $x_{i,m}^j$. If the mutation is conducted, proceed to Step 2.3. Otherwise, proceed to Step 2.2.

Step 2.2 If $x_{i,m}^j$ contains a child set of genes, proceed to Step 2.1, and examine the child set of genes. If not, proceed to Step 2.5.

Step 2.3 Randomly generate $x_{i,m}^j$.

Step 2.4 Randomly reconstruct the genes of all sub-units for the selected alternative.

Step 2.5 Increment m by 1. When $m > n$, set $m = 1$, and increment j by 1. When $j > k_i$, end the crossover operations because the set of genes has been exhausted, and return to the crossover operations for the parent set of genes.

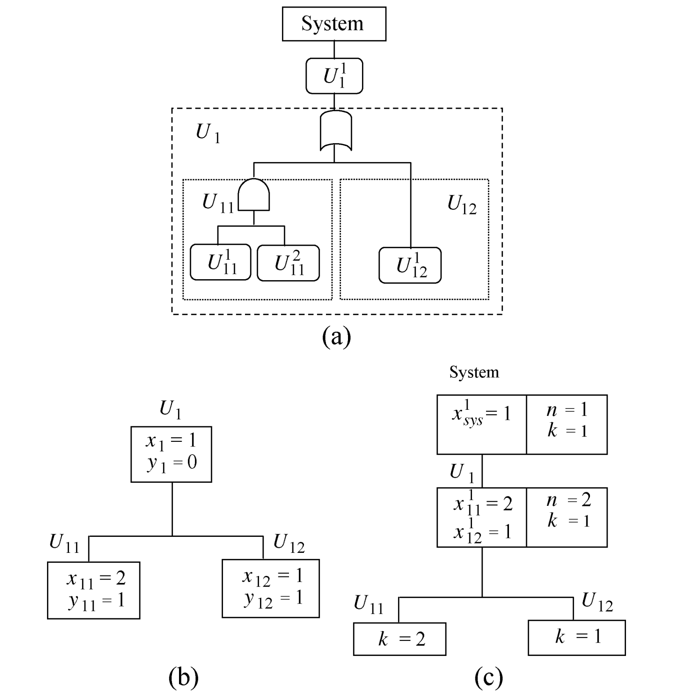


Fig. 8. Coding schemes in conventional GA, and HGA for a bi-level series configuration. (a) Redundancy allocation in unit U_1; (b) conventional GA coding of unit U_1; (c) HGA coding of unit U_1.

TABLE II
HGA PARAMETERS

Cases	Parameters		Average Fitness	Best Fitness
	Crossover	Mutation	(20-runs)	(20-runs)
1	0.7	0.05	0.96047	0.97628
2	0.9	0.05	0.96155	0.97628
3	0.8	0.05	0.9621	0.97639
4	1.0	0.05	0.96117	0.97628
5	0.8	0.01	0.92826	0.97254
6	0.8	0.1	0.94484	0.96422
7	0.8	0.2	0.931904	0.950827

IV. NUMERICAL EXAMPLES

A. Hierarchical Series Redundancy Allocation Problem

The HGA was applied to optimize multilevel series redundancy allocation problems having two different configurations. The first configuration is called *problem-A*, and is similar to the problem described in Yum & Kim [21], while the second configuration is called *problem-B*. Figs. 6 and 7 respectively represent *problem-A*, and *problem-B*.

Problem-A contains three levels, and *problem-B* contains four levels. All units of these configurations are in series. We applied the GA, proposed by Yum & Kim [21] to solve MRAOP to compare the obtained solutions with those obtained by the HGA. We call this GA a conventional GA because it uses vector coding of the design variables, and applies a special crossover & mutation operator to handle such coding.

The genotypes for the conventional GA [21] are encoded as an ordered couple of a design variable, $x_{i,m}$, and an indicator variable, $y_{i,m}$; $v_i = (x_{i,m}, y_{i,m})$, where the subscript i is the

TABLE III
INPUT DATA

Problem-A				*Problem-B*			
Unit	Reliability	Cost	λ	Unit	Reliability	Cost	λ
U_1	0.4003	72	2	U_1	0.2198	102	2
U_{11}	0.7267	26	2	U_{11}	0.5130	48	2
U_{12}	0.7650	19	3	U_{12}	0.4284	50	2
U_{13}	0.7200	21	2	U_{111}	0.7200	21	3
U_{111}	0.9000	5	3	U_{112}	0.7125	21	3
U_{112}	0.9500	6	4	U_{121}	0.6300	23	3
U_{113}	0.8500	5	3	U_{122}	0.6800	21	3
U_{121}	0.9000	6	4	U_{1111}	0.9000	7	4
U_{122}	0.8500	7	4	U_{1112}	0.8000	6	4
U_{131}	0.9000	8	3	U_{1121}	0.7500	8	4
U_{132}	0.8000	7	4	U_{1122}	0.9500	5	4
				U_{1211}	0.7000	9	4
				U_{1212}	0.9000	6	4
				U_{1221}	0.8500	5	4
				U_{1222}	0.8000	8	4

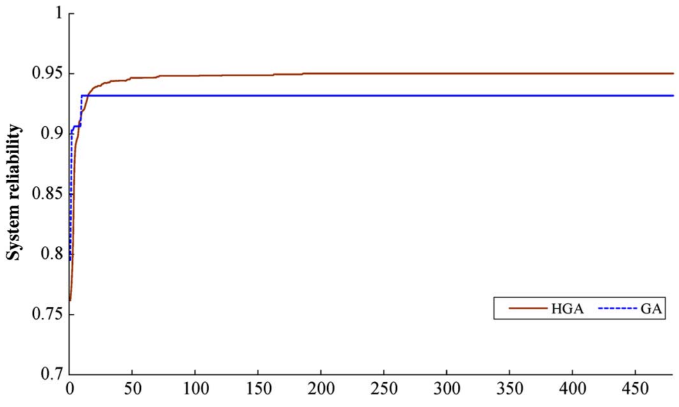


Fig. 9. Convergence of GA, and HGA in *problem-A*.

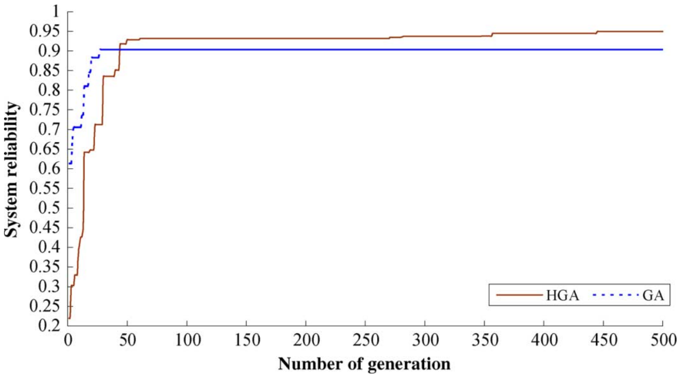


Fig. 10. Convergence of GA and HGA in *problem-B*.

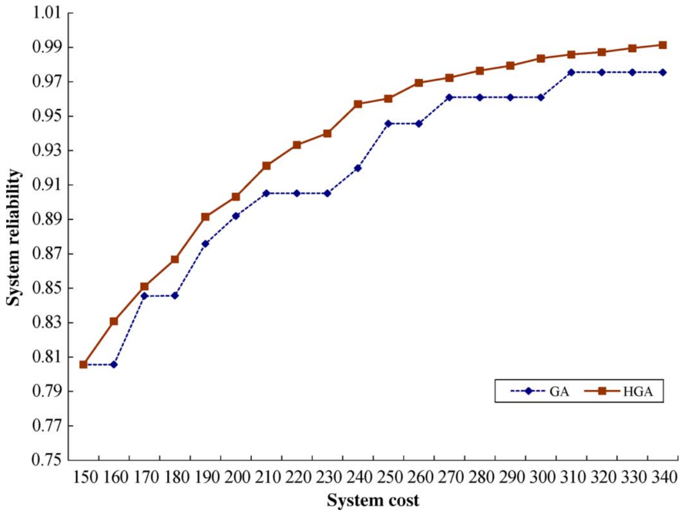


Fig. 11. Optimal solutions for *problem-A* obtained using GA, and HGA.

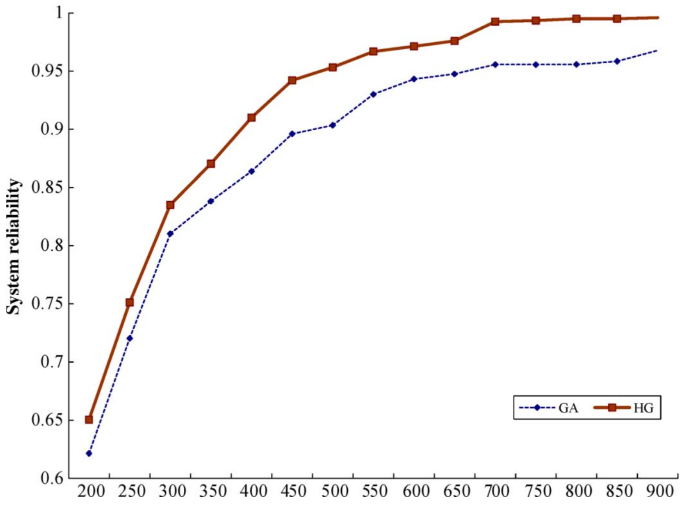


Fig. 12. Optimal solutions for *problem-B* obtained using GA, and HGA.

index of the chromosome to which the gene belongs, and subscript m denotes units. A chromosome is represented as

$$v_i = [(x_{i1}, y_{i1})(x_{i2}, y_{i2}) \dots (x_{in}, y_{in})]$$

The value of the indicator variables for a unit is 1 when that unit is subject to redundancy, and 0 when that unit is not allowed to have redundancy. Only one unit among the set of units in a direct line is selected to have redundancy so that the sum of the indicator variables of units along a direct line is 1. On the other hand, we used hierarchical genotype encoding when applying the HGA to solve the MRAOP. Coding schemes for conventional GA, and HGA can be understood more clearly by examining an example of redundancy allocation in a bi-level series unit U_1 having two sub-units, U_{11}, and U_{12}, as shown in Fig. 8.

The redundancy values for sub-units U_{11}, and U_{12} are 2, and 1, respectively. Note that there are two direct lines, $(U_1 - U_{11})$, and $(U_1 - U_{12})$ in Fig. 8(b). Because only a unit at a level is selected to have redundancy among the set of units in a direct line, unit U_1 cannot have redundancy if units U_{11} and U_{12} are subject to redundancy. Thus, the GA coding scheme does not allow redundancy at two levels simultaneously. In contrast, the HGA allows redundancy at two levels simultaneously. Fig. 8(c) shows the reliability for both the system, and the sub-units.

Both a conventional GA, and a HGA were applied when optimizing multilevel series redundancy allocation problems having two different configurations, to evaluate their applicability for solving multilevel allocation problems. The cost function $C(x) = cx + \lambda^x$, described in [21], is used as a

TABLE IV
OPTIMAL HIERARCHICAL CONFIGURATIONS

	Problem-A					
	GA			HGA		
Cases	Reliability	Cost	Optimal Allocation $[x_1x_{11}x_{12}x_{13}x_{111}x_{112}x_{113}x_{121}x_{122}x_{131}x_{132}]$ $[y_1y_{11}y_{12}y_{13}y_{111}y_{112}y_{113}y_{121}y_{122}y_{131}y_{132}]$	Reliability	Cost	Optimal Allocation $[(x_1)(x_{11}x_{12}x_{13})$ $(x_{111}x_{112}x_{113})(x_{121}x_{122})(x_{131}x_{132})]$
1	0.9276	289	[14333242224]	0.9742	290	[(1)(222)
			[01110000000]			(211322)(2121)(1122)]
2	0.7822	278	[24241232212]	0.8537	289	[(1)(221)
			[01010001100]			(222212)(1122)(32)]
3	0.9557	275	[54333343552]	0.9622	297	[(1)(122)
			[01110000000]			(223)(2211)(2122)]
4	0.7989	278	[34542212225]	0.8734	292	[(1)(212)
			[01010001100]			(122122)(22)(2222)]
5	0.8447	291	[45333232231]	0.9029	290	[(1)(221)
			[01010001100]			(221221)(2122)(32)]
6	0.8506	292	[25321433131]	0.9102	286	[(1)(221)
			[01110000000]			(322212)(2211)(22)]
7	0.8986	275	[34335144225]	0.9187	300	[(1)(212)
			[01110000000]			(212212)(32)(1122)]
8	0.9272	298	[43243533234]	0.9579	294	[(1)(122)
			[01010001100]			(322)(2222)(1212)]
9	0.9262	270	[53341311213]	0.9433	300	[(1)(132)
			[01110000000]			(322)(121111)(2132)]
10	0.9185	278	[34544552214]	0.9467	297	[(1)(212)
			[01010001100]			(221222)(22)(2222)]

constraint for these two optimization problems. The symbols x, cx, and λ respectively represent the number of parallel units, the unit cost, and the additional cost.

B. Input Data

Suitable parameters for optimizing the two allocation problems were selected based on several experimental runs using a conventional GA, and the HGA we created. We observed the convergence of fitness functions, and selected suitable GA operator values for subsequent use in the optimization process. Table II provides a summary of the average, and best fitness values for different HGA parameters obtained during 20 runs with 500 generations in each run. The best crossover, and mutation rate values for solving these problems when using a conventional GA were 0.8, and 0.1, respectively. Similarly, when using the HGA, these best values were respectively 0.8, and 0.05. An initial population of 100 individuals was generated randomly when using both the GA, and HGA. This population size was selected based on the performance evaluation of the algorithms with different population sizes.

Twenty two design variables were used with the conventional GA, which is the sum of the redundancy numbers plus the constraints for direct lines. In contrast, the number of design variables used with the HGA was 11. The number of generations was 500 in each case, and a maximum redundancy number of five was imposed for both the modular, and component redundancy schemes. The unit reliability, and the unit cost at the very lowest level in the multilevel redundancy allocation problems were used when calculating the unit reliability, and the unit cost of upper level units, up to the system level. Table III summarizes the unit reliability, and unit cost of the components at the very lowest level in both problems. Note that we used the same data for *problem-A* that Yum & Kim [21] used, to enable a comparison of the optimal solutions obtained by the HGA with those provided by a conventional GA.

C. Computational Results

We separately applied the HGA, and GA when solving the *problem-A,* and *problem-B* allocation optimization problems. First, we checked the convergence of the optimal solutions when using the GA, and HGA; and Figs. 9 and 10 show the results when using the two different types of algorithm.

The x-axis represents the number of generations, and the y-axis represents the system reliability. The cost constraints for these two graphs were 240 for *problem-A*, and 500 for *problem-B*. These figures indicate that the HGA yields a better optimal solution than the conventional GA under the same computational environment.

To assess the influence of cost constraints upon the optimal solutions, 20 cases for a 3-level problem, and 15 cases for a 4-level problem, were examined. Ten 500-generation trials were performed using each algorithm type, and the best solution of the ten-trial set was chosen as the optimal solution in each of these cases. Figs. 11 and 12 show the trends of optimal solutions obtained using the GA, and HGA. The x-axis represents the cost constraint, and the y-axis represents the optimal system reliability.

Next, we examined ten cases in which the unit reliability values were varied while the cost constraint was held to a value of 300 for *problem-A*, and 500 for *problem-B*. In the same manner as before, ten 500-generation trials for each of these ten cases were carried out, and the best solution was chosen as the optimal solution for each case. Note that the number of function calls in each case considered here was the same for both GA, and HGA. Tables IV and V summarize the optimal solutions

TABLE V
OPTIMAL HIERARCHICAL CONFIGURATIONS

Cases	*Problem-B*					
	GA			HGA		
	Reliability	Cost	Optimal Allocation $[x_1 x_{11} x_{12} x_{111} x_{112} x_{121} x_{122}$ $x_{1111} x_{1112} x_{1121} x_{1122}$ $x_{1211} x_{1212} x_{1221} x_{1222}]$ $[y_1 y_{11} y_{12} y_{111} y_{112} y_{121} y_{122}$ $y_{1111} y_{1112} y_{1121} y_{1122}$ $y_{1211} y_{1212} y_{1221} y_{1222}]$	Reliability	Cost	Optimal Allocation $[(x_1)(x_{11} x_{12})(x_{111} x_{112})(x_{121} x_{122})$ $(x_{1111} x_{1112})(x_{1121} x_{1122})$ $(x_{1211} x_{1212})(x_{1221} x_{1222})]$
1	0.9568	484	[155132223244513] [001010011000000]	0.9775	499	[(1)(22)(11)(21) (1222)(2222)(32) (33)]
2	0.7810	485	[143143432441532] [000011011000011]	0.8677	496	[(1)(11)(21)(12) (2212)(33)(23) (2222)]
3	0.9568	484	[155132223244513] [001010011000000]	0.9777	486	[(1)(11)(22)(12) (2212)(2122)(33) (2222)]
4	0.8202	486	[224335554332233] [000100000111111]	0.8870	460	[(1)(11)(12)(12) (32)(2223)(22) (2222)]
5	0.8586	485	[213431342143232] [000110000001111]	0.9368	454	[(1)(11)(12)(12) (32)(2223)(22) (2222)]
6	0.8767	462	[135443341233234] [000101100110000]	0.9508	491	[(1)(11)(32)(21) (112222)(2123)(2211) (32)]
7	0.9124	481	[253452354243211] [010000100001100]	0.9538	467	[(1)(11)(12)(22) (23)(1212)(2132)(2222)]
8	0.9515	486	[111324452224143] [000101100110000]	0.9741	491	[(1)(11)(12)(22) (23)(2122)(2232) (32212)]
9	0.9122	462	[153433255123132] [000111000000011]	0.95021	467	[(1)(11)(22)(21) (2222)(2222)(2222) (32)]
10	0.8941	441	[134433314313215] [000111100000000]	0.9645	494	[(1)(11)(23)(22) (2122)(112112)(2222) (2132)]

obtained when using the GA, and HGA for *problem-A*, and *problem-B*, respectively.

An interpretation of the optimal solution data summarized in these two tables is provided in Fig. 13, which shows the arrangement of the units in *problem-A*, and *problem-B*. It is a graphic representation of the optimal solutions for the fourth case listed in Table IV.

Fig. 13(a) illustrates the optimal arrangement of the modules and components in the system obtained when using the GA, and Fig. 13(b) illustrates the optimal arrangement obtained when using the HGA.

V. DISCUSSION

The numerical examples solved in the previous section demonstrate that hierarchical genotype representations of hierarchical design variables provide superior solutions in comparison to vector representation. The most suitable GA, and HGA parameters were selected from the results of a number of preliminary runs; and Table II shows that the most useful HGA crossover, and mutation rates are 0.8, and 0.05, respectively, determined by twenty 500-generation runs. We observe in Fig. 9, and Fig. 10 that the HGA offers superior convergence, and that this advantage is achieved more smoothly by searching a larger feasible design space than when a conventional GA is used.

Moreover, Figs. 11 and 12 indicate that the optimal solution obtained using the HGA is superior to its conventional GA counterpart. After examining the solution data, we find that there is an approximately 4% maximum improvement in the 3-level series allocation problem, and a 5% improvement in the 4-level series allocation problem. Similarly, in Tables IV and V, we see that the HGA yielded average improvements of 4.7%, and 5.82% over the conventional GA. Moreover, the maximum improvement in the optimal solutions when using the HGA was found to be 9.23% in the 3-level problem, and 11% in the 4-level problem. The improved reliability obtained using the HGA is achieved without incurring additional material or parts costs. This is an important milestone because, in high reliability applications, even very small improvements in reliability are often difficult to obtain. Thus, it appears incontrovertible that the hierarchical genotype scheme typical of HGA is better suited for optimizing multilevel allocation problems than the one-dimensional vector schemes of conventional GA.

The reason why the GA yielded inferior solutions in comparison to the HGA is that the GA requires vector transformation of the hierarchical design variables. The vector transformation of hierarchical design variables into one dimensional array representations actually reduces the feasible design space, and the GA may consequently fail to find superior solutions that exist just beyond its feasible design space. Because HGA do not require vector transformation, the feasible design space remains unaffected, and this leads to better optimal solutions during the searching process. Additionally, the hierarchical coding method proposed in this paper can express the exact internal structure with series linkage.

Furthermore, the simultaneous allocation of redundancy at two or more levels also leads to better solutions than those provided by conventional GA. Allocated resources can be appropriately shared at all levels, and one such optimal arrangement of redundant units is graphically illustrated in Fig. 13. We see that the optimal HGA solution contains two parallel modules for unit U_{11}; and sub-units U_{111}, U_{112}, and U_{113} have single, double, and double redundancy, respectively. On the other hand, the optimal solution obtained using the conventional GA contains four

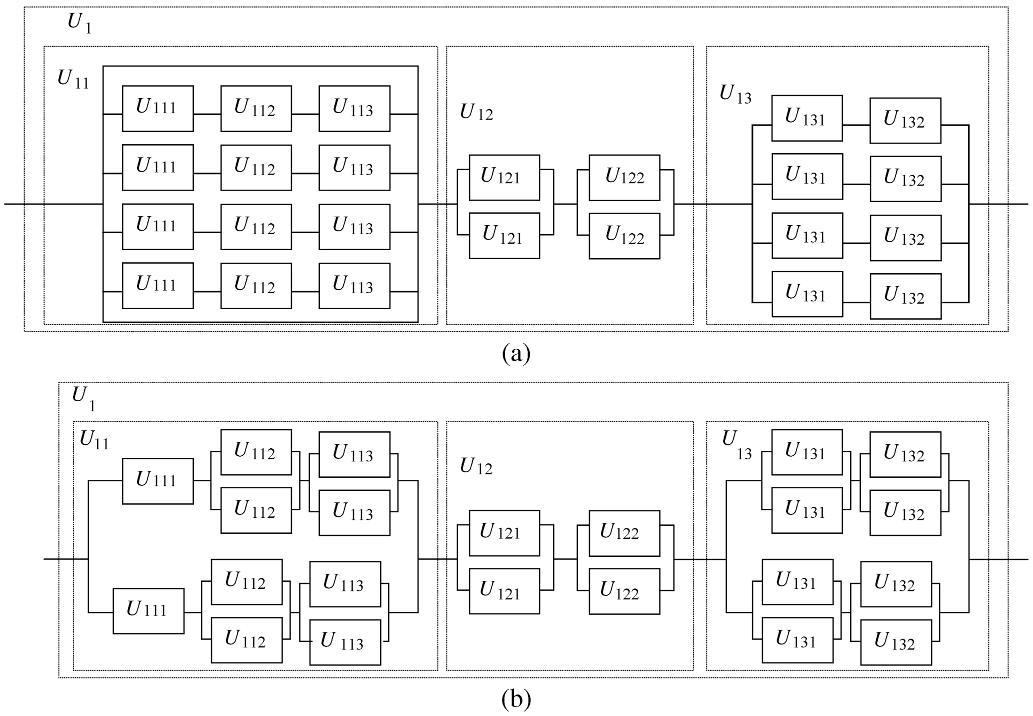


Fig. 13. Optimal solutions for the fourth case listed in Table IV. (a) Optimal configuration obtained using GA; (b) optimal configuration obtained using HGA.

parallel U_{11} modules, and all the sub-units have only single redundancy. A similar pattern can be seen concerning the other two U_1 modular units. Hence, an additional significant advantage that the use of HGA provides is that redundancy at both the unit, and the sub-unit level can be achieved simultaneously.

The performance of the HGA in solving the two examples here indicates that hierarchical genotype representation is not only capable of solving multilevel reliability optimization problems of any size, but also that it allows significant flexibility so that every possible redundancy combination can be evaluated. This flexibility in redundancy optimization seems impossible to achieve when using conventional GA. Another useful feature of hierarchical genotype representation is that optimal redundancy values are given hierarchically for each module, and component. This is highly desirable in a complex system, when the goal of ensuring optimum reliability depends on determining exactly how many redundancies are required for a particular module at a particular level in the hierarchical system.

VI. Conclusions

Multilevel redundancy allocation optimization problems are frequently encountered in complex system designs. This paper proposed a general formulation for multilevel redundancy allocation optimization problems that aim to maximize system reliability. These multilevel optimization problems have hierarchical design variables, so we proposed a new coding method for use in a HGA, in which hierarchical design variables of MRAOP are represented using two types of hierarchical genotype: nodal, and terminal. We applied the newly developed HGA, and a conventional GA separately, to solve two multilevel series redundancy allocation optimization problems having three, and four levels. The optimal solutions for these two problems demonstrated that the proposed HGA provides optimal system reliability that is superior to the conventional GA results, because it does not depend on the use of vector coding to represent the hierarchical variables, and can preserve the original design space. HGA using the new variable coding method presented here can be applied in other hierarchical optimization problems, but the efficiency of such algorithms must be investigated. We hope to extend our approach for optimizing the system reliability of other multilevel structures such as hierarchical series-parallel, multilevel network, and other multilevel configurations in future work.

Acknowledgment

The authors would like to thank the reviewers for their comments for improving this paper.

References

[1] W. Wang, N. J. Loman, and P. Vassiliou, "Reliability importance of components in a complex system," in *Proceedings of the Annual Reliability and Maintainability Symposium*, LA, 2004, pp. 6–11.
[2] I. Koren and Z. Koren, "Defect tolerance in VLSI circuits: Techniques and yield analysis," *Proceedings of the IEEE*, vol. 86, no. 9, pp. 1819–1837, 1998.
[3] G. Levitin, "Optimal multilevel protection in series-parallel systems," *Reliability Engineering & System Safety*, vol. 81, no. 1, pp. 93–102, 2003.
[4] S. H. Baek, B. W. Kim, E. J. Joung, and C. W. Park, "Reliability and performance of hierarchical RAID with multiple controllers," in *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing*, Newport, Rhode Island, 2001, pp. 246–254.
[5] C. Ha and W. Kuo, "Reliability redundancy allocation: An improved realization for nonconvex nonlinear programming problems," *European Journal of Operational Research*, vol. 171, pp. 24–38, 2006.
[6] W. Kuo, V. R. Prasad, F. A. Tillman, and C. L. Hwang, *Optimal Reliability Design: Fundamentals and Applications*. Cambridge: Cambridge University Press, 2001.
[7] F. A. Tillman, C. L. Hwang, and W. Kuo, "Optimization techniques for systems reliability with redundancy—A review," *IEEE Trans. Reliability*, vol. 26, pp. 148–155, 1977.

[8] W. Kuo and M. J. Zuo, *Optimal Reliability Modeling: Principles and Applications*. New York: Wiley, 2003.

[9] W. Kuo and V. R. Prasad, "An annotated overview of system-reliability optimization," *IEEE Trans. Reliability*, vol. 49, no. 2, pp. 176–187, 2000.

[10] P. Boland and E. EL-Neweihi, "Component redundancy vs. system redundancy in the hazard rate ordering," *IEEE Trans. Reliability*, vol. 44, no. 4, pp. 614–619, 1995.

[11] M. S. Chern, "On the computational complexity of reliability redundancy allocation in a series system," *Operations Research Letters*, vol. 11, pp. 309–315, 1992.

[12] G. Anandlingam and T. L. Friesz, "Hierarchical optimization: An introduction," *Annals of Operations Research*, vol. 34, no. 1–4, pp. 1–11, 1992.

[13] D. E. Goldberg, *Genetic Algorithms in Search Optimization & Machine Learning*. , MA: Addison-Wesley, 1989.

[14] L. Painton and J. Campbell, "Genetic algorithms in optimization of system reliability," *IEEE Trans. Reliability*, vol. R-44, no. 2, pp. 172–178, 1995.

[15] T. Yokata, M. Gen, and K. Ida, "System reliability of optimization problems with several failure modes by genetic algorithm," *Japanese Journal of Fuzzy Theory and Systems*, vol. 7, no. 1, pp. 117–135, 1995.

[16] K. Ida, M. Gen, and T. Yokata, "System reliability optimization with several failure modes by genetic algorithm," in *Proceedings of the 16th International Conference on Computers and Industrial Engineering*, Ashikaga, Japan, 1994, pp. 349–352.

[17] D. W. Coit and A. E. Smith, "Reliability optimization of series-parallel systems using a genetic algorithm," *IEEE Trans. Reliability*, vol. R-45, no. 2, pp. 254–260, 1996.

[18] D. W. Coit and A. E. Smith, "Considering risk profiles in design optimization for series-parallel systems," in *Proceedings of the Annual Reliability and Maintainability Symposium*, Philadelphia, PA, 1997, pp. 271–277.

[19] Y. C. Hsieh, T. C. Chen, and D. L. Bricker, Genetic Algorithms for Reliability Design Problems, Department of Industrial Engineering, University of Iowa, 1997, Technical Report.

[20] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*. New York: Wiley, 1997.

[21] W. Y. Yun and J. W. Kim, "Multi-level redundancy optimization in series systems," *Computers & Industrial Engineering*, vol. 46, pp. 337–346, 2004.

[22] H. A. Simon, *The Sciences of the Artificial*. Cambridge, MA: The MIT Press, 1968.

[23] E. D. DeJong, D. Thierens, and R. A. Watson, "Hierarchical genetic algorithms," *Parallel Problem Solving From Nature-PPSNVIII*, pp. 232–241, 2004.

[24] J. Hu and E. D. Goodman, "The hierarchical fair competition (HFC) model for parallel evolutionary algorithms," in *Proceedings of the Congress on Evolutionary Computation (CEC-2002)*, 2002, pp. 49–54.

[25] M. Gulsen and A. E. Smith, "A hierarchical genetic algorithm for system identification and curve fitting with a supercomputer implementation," in *Evolutionary Algorithms*. New York: Springer, 1999, pp. 111–137.

[26] K. Tang, K. Man, and R. Istepanian, "Teleoperation controller design using hierarchical genetic algorithms," in *Proceedings of the IEEE International Conference on Industrial Technology*, 2000, pp. 707–711.

[27] M. Sefrioui and J. Périaux, "A hierarchical genetic algorithm using multiple models for optimization," in *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, 2000, pp. 879–888.

[28] M. Yoshimura and K. Izui, "Smart optimization of machine systems using hierarchical genotype representations," *ASME Journal of Mechanical Design*, vol. 124, no. 3, pp. 375–384, 2002.

[29] M. Gen and R. Cheng, "A survey of penalty technique in genetic algorithms," in *Proceedings of the International Conference on Evolutionary Computation*, Japan, 1996, pp. 804–809, Nagoya University.

Ranjan Kumar is currently a Graduate Student in the Department of Aeronautics and Astronautics at Kyoto University, Japan. He received a Bachelor of Engineering Degree from Birsa Institute of Technology at Sindri, and a Master Degree in Reliability Engineering from Indian Institute of Technology at Kharagpur. His research interests include design for reliability and maintainability, and system design optimization using evolutionary computation. He is a student member of the IEEE.

Kazuhiro Izui is an Assistant Professor in the Department of Aeronautics and Astronautics at Kyoto University. He earned his Bachelor of Engineering Degree from Kyoto University. He received his Master Degree, and Ph.D. in Precision Engineering from Kyoto University. His research interests include product design optimization using evolutional computing, and structural optimization methods.

Yoshimura Masataka is a Professor in the Department of Aeronautics and Astronautics at Kyoto University. He earned his Bachelor of Engineering Degree in Mechanical Engineering, Master of Engineering Degree in Precision Engineering, and Doctor of Engineering Degree from Kyoto University. His research interests include concurrent optimization of product design and manufacturing, information systems for manufacturing, collaborative optimization, concurrent engineering, and dynamics of machine tools and industrial robots.

Shinji Nishiwaki is an Associate Professor in the Department of Aeronautics and Astronautics at Kyoto University. He earned his Bachelor of Engineering, and Master Degree in Precision Engineering from Kyoto University. He received his Ph.D. in Mechanical Engineering from the University of Michigan. His research interests include concurrent optimization of product design, structural and system optimization, and automotive engineering.