A Memetic Algorithm for Multi-Level Redundancy Allocation

Zai Wang, *Student Member, IEEE*, Ke Tang, *Member, IEEE*, and Xin Yao, *Fellow, IEEE*

***Abstract*—Redundancy allocation problems (RAPs) have attracted much attention for the past thirty years due to its wide applications in improving the reliability of various engineering systems. Because RAP is an NP-hard problem, and exact methods are only applicable to small instances, various heuristic and meta-heuristic methods have been proposed to solve it. In the literature, most studies on RAPs have been conducted for single-level systems. However, real-world engineering systems usually contain multiple levels. In this paper, the RAP on multi-level systems is investigated. A novel memetic algorithm (MA) is proposed to solve this problem. Two genetic operators, namely breadth-first crossover and breadth-first mutation, and a local search method are designed for the MA. Comprehensive experimental studies have shown that the proposed MA outperformed the state-of-the-art approach significantly on two representative examples.**

***Index Terms*—Evolutionary algorithms, memetic algorithms, multi-level systems, redundancy allocation.**

Acronyms

RAP	Redundancy Allocation Problem
MLRAP	Multi-Level Redundancy Allocation Problem
MA	Memetic Algorithm
GA	Genetic Algorithm
HGA	Hierarchical Genetic Algorithm

Notations

R_{sys}	the reliability of a system
C_{sys}	the cost of a system
U_i	the ith unit, where a unit can refer to a system, a subsystem or a component
$U_{i,m}$	the mth child unit of U_i
R_i	the reliability of U_i
C_i	the cost of U_i
x_i	the redundancy of U_i
$\mathbf{x}$	a set of design variables x_i
$R(\mathbf{x})$	the reliability of a system or subsystem defined by design variables $\mathbf{x}$
$C(\mathbf{x})$	the cost of a system or subsystem defined by design variables $\mathbf{x}$
$\lambda_{i,m}$	the additional cost parameter of the mth child unit of U_i, which is at the second lowest level of the system
n_i	the number of child units of U_i
$U_{i,m}^j$	the jth redundant unit of mth child unit of U_i
$R_{i,m}^j$	the reliability of $U_{i,m}^j$
$C_{i,m}^j$	the cost of $U_{i,m}^j$
$x_{i,m}$	the redundancy of $U_{i,m}$

Manuscript received August 30, 2009; revised February 09, 2010 and March 17, 2010; accepted March 17, 2010. Date of publication July 26, 2010; date of current version November 30, 2010. This work was partially supported by EPSRC Grant EP/D052785/1 on "SEBASE: Software Engineering By Automated Search," National Natural Science Foundation of China Grant U0835002, and the Fund for Foreign Scholars in University Research and Teaching Programs under Grant B07033. Associate Editor: G. Levitin.

Z. Wang and K. Tang are with the Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, University of Science and Technology of China, Hefei, 230027, China (e-mail: wangzai@mail.ustc.edu.cn; ketang@ustc.edu.cn).

X. Yao is with the Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, University of Science and Technology of China, Hefei, 230027, China, and also with the CERCIA, School of Computer Science, University of Birmingham, Birmingham, B15 2TT, U.K. (e-mail: x.yao@cs.bham.ac.uk).

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.2010.2055927

I. Introduction

To maximize the reliability of a system, which is typically comprised of a number of components, either the component reliability can be enhanced, or redundant components can be added in parallel [1], [2]. In many real-world problems, the reliability of components utilized to construct a system are fixed, thus the only way to improve the system reliability is to increase the redundancy of utilized components. However, increasing the redundancy of components requires more resources. Thus, it is always important to optimally allocate redundancy to components under some resource constraints. This problem is referred to as the redundancy allocation problem (RAP) [3]. The RAP is one of the most important reliability optimization problems with regards to improving the reliability of real-world systems in the design phase. It has attracted many researchers in the past several decades due to reliability's critical importance in various kinds of systems, such as electrical systems, mechanical system, and software systems [1], [2].

The most commonly studied configuration of RAPs is a parallel-serial system with s-independent subsystems, as illustrated in Fig. 1 [1], [2]. The components in each subsystem are in

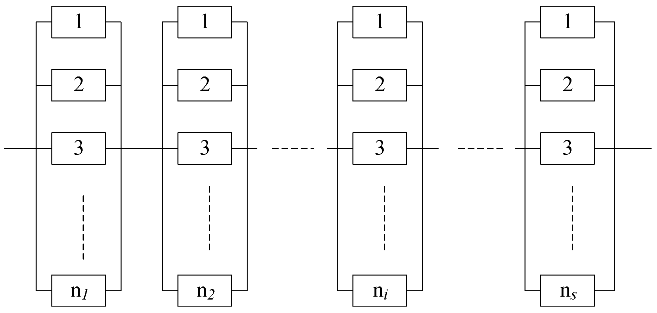

Fig. 1. The structure of a general parallel-serial system.

parallel, and a subsystem i is functioning if at least one of its n_i components is operational. For parallel-serial systems, an RAP is always formulated as a nonlinear integer programming problem with the objective to maximize the system reliability under some constraints, such as cost, volume, and weight. In early times, RAPs were relatively simple as all the components of a subsystem were considered to be of the same type. Later, progress was made by considering different types of components in one subsystem. This extension significantly increased the difficulty of RAPs. When an RAP was studied on parallel-serial systems, the redundancy could just be allocated to the components, which means that the basic structures of the parallel-serial systems were fixed. In other words, systems considered in this scenario only have a single level. Hence, the performance (e.g., reliability) of a system can be directly calculated based on the properties of components at the lowest level. However, real-world systems, such as communication systems, computing systems, control systems, and critical power systems, usually contain multiple levels. In these systems, the entire system is at the top level (system level), and the components are at the lowest level, while subsystems are at the levels between the entire system and the components. In a multi-level system, the entire system, subsystems, and components are all called units; and redundancy can be allocated to any level [4]. Fig. 2 illustrates a schematic diagram of a four-level system. An RAP with a multi-level system is referred to as a multi-level redundancy allocation problem (MLRAP). Specifically, the MLRAP investigated in this paper is formulated based on multi-level serial systems with the following assumptions [5].

- **Assumption 1**: For a unit that is not at the lowest level, its child units are in serial, and the number of these child units is fixed. (E.g. when a real system based on the configuration shown in Fig. 2 is constructed, every redundant unit U_1 must have three child units U_{11}, U_{12}, and U_{13}, and these three child units are in serial.)
- **Assumption 2**: The redundancy can be allocated to the units on any level.
- **Assumption 3**: The quality (reliability, cost) of each component is predefined. If one unit is not a component, its reliability, and cost are calculated based on its child units.

The RAP has been proved to be an NP-hard problem [6]. Thus, how to find effective approaches to it is a hot topic. There are many approaches proposed to cope with RAPs, including exact methods [7]–[10], max-min approaches [11], [12], dynamic programming [13], [14], heuristic methods [15], [16], and meta-heuristic algorithms [17]–[26]. Among them, meta-heuristic methods, especially genetic algorithms (GAs), are widely, successfully utilized due to their robustness, and strong ability of global search, even though sometimes they are time-consuming. Comprehensive literature reviews on RAP have been carried out in [1], [2]. From these reviews, it can be observed that, in spite of its importance in the real world, the MLRAP has rarely been investigated, and few approaches [5], [27], [28] have been proposed. In [27], Levitin proposed an algorithm based on a GA framework with a universal generating function technique for the system survivability evaluation. This method aims to solve multi-level protection cost minimization problems subject to survivability constraints. Later, Yun and Kim [28] proposed a restricted multi-level redundancy allocation model, and addressed a tri-level RAP using a customized GA. However, the customized GA employs a fixed-length vector to represent the solution to the MLRAP, and thus redundancy is required to be allocated to only one unit in a direct line, which is defined as a set of units from the system-level unit down to one component unit. Though both methods [27], [28] were designed to cope with MLRAPs, they were based on strong assumptions or rigid rules that do not accord with reality. Recently, Kumar et al. [5] re-formulated MLRAPs so that redundancy allocation can be carried out at any level of a multi-level system. In this work, a solution to an MLRAP was represented by a hierarchical structure, and a simple GA was adopted to tackle the MLRAP involved. Although the hierarchical genetic algorithm (HGA) in [5] has been shown to be effective via empirical studies, its effectiveness on the MLRAP can still be improved significantly.

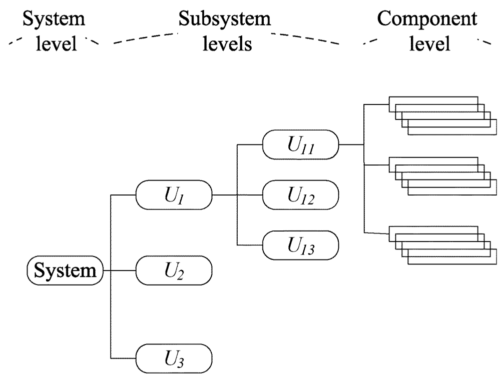

Fig. 2. The structure of a tri-level system.

In this paper, a novel memetic algorithm (MA) is proposed for MLRAPs. As an emerging area of evolutionary computation, MAs are population-based meta-heuristic search methods that combine global search strategies with local search heuristics [29]. MAs have been reported to not only converge to high quality solutions, but also search more efficiently than conventional GAs. The success of MAs have been demonstrated on a wide variety of real-world problems [30]. Compared to conventional GAs, there are two key issues for the success of MAs. One is an appropriate balance between global and local search, and the other is a cost effective coordination of local search. Hence, the local search procedure, which is usually designed

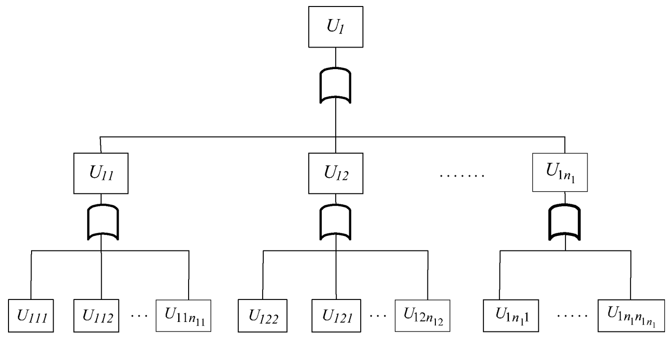

Fig. 3. A general multi-level serial system.

to utilize the domain knowledge of the problem, plays one of the most important roles in MAs. In our paper, based on the hierarchical genotype representation of the variables employed in [5], two new breadth-first-search genetic operators (breadth-first-search crossover, and breadth-first-search mutation), and a novel problem-specific local search operator are proposed. Then, the newly proposed genetic operators, and the local search operator, are incorporated into the MA framework; and a new MA is developed for MLRAPs. The proposed MA is compared with HGA on two multi-level systems. Experimental results show that the newly proposed MA outperformed HGA on both multi-level systems.

The rest of this paper is organized as follows. Section II introduces the background of this work, including the formulation of MLRAPs, and the hierarchical genotype representation for MLRAPs. Section III proposes our new memetic algorithm (MA). Then, empirical comparisons between the MA and HGA are presented in Section IV. Section V concludes this paper.

II. Background

A. Problem Formulation

We define the parts of a multi-level serial system hierarchically as the entire system at the topmost level, the subsystems at lower levels, and the components at the lowest level. The system, subsystems, and components are all referred to as units. Each unit except a component has a fixed number of serial units at the immediately lower level, which are called its child units. Each unit except the system unit relates to a unique unit at the immediately higher level, which is called its parent unit. Fig. 3 illustrates an example of multi-level serial systems. In this figure, U_1 is a system unit containing n_1 serial child units (U_{11} to U_{1n_1}). Similarly, U_{11} has n_{11} serial child units, represented as U_{111} to $U_{11n_{11}}$. This structure applies to all of the units except for the component units.

Given a multi-level serial system, the redundancy allocation procedure starts from the system level, and moves to the component level. One is allowed to replicate any unit in the system, and the original unit and its replicated units are combined together in parallel. Fig. 4 describes an example of redundancy allocation on a bi-level serial system, which consists of a system unit U_1, and two child units U_{11} and U_{12}. After allocating redundancy to this system, the redundancy of U_1 is two. Under the parent unit U_1^1, the redundancy of U_{11}, and U_{12} are three, and

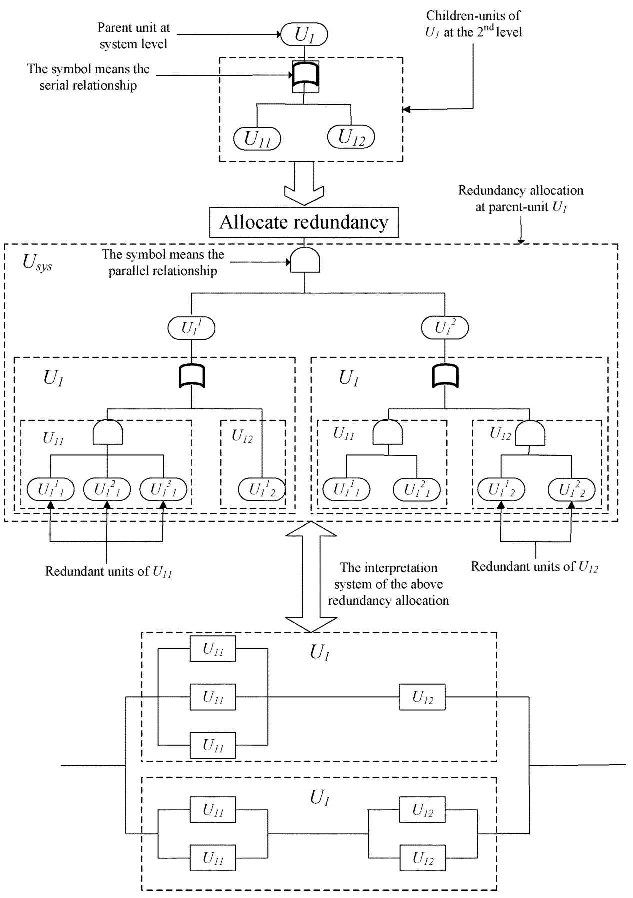
Parent unit at system level
U_1
Children-units of U_1 at the 2nd level
The symbol means the serial relationship
U_{11}
U_{12}
Allocate redundancy
Redundancy allocation at parent-unit U_1
U_{sys}
The symbol means the parallel relationship
U_1^1
U_1^2
U_1
U_{11}
U_{12}
U_{11}^1
U_{11}^2
U_{11}^3
U_{12}^1
U_{12}^2
Redundant units of U_{11}
The interpretation system of the above redundancy allocation
Redundant units of U_{12}

Fig. 4. An example of redundancy allocation on a bi-level serial system. $U_{i,m}^j$ represents the jth redundant unit of mth child unit of U_i.

one, respectively. Under the parent unit U_1^2, the redundancy of U_{11}, and U_{12} are both two. The corresponding final system is illustrated at the bottom of Fig. 4.

In a multi-level serial system, the reliability of the units at the lowest level (i.e., components) are predefined. For each unit at a higher level, its reliability can be calculated based on its child units directly. Assume a unit U_i has n_i child units, and denote the redundancy of a child unit $U_{i,m}$ as $x_{i,m}$. The reliability of U_i can be calculated with (1).

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

Using (1), the reliability of the units at the second lowest level can be calculated on the basis of the components. Then the reliability of the entire system can be calculated by repeating this procedure. For example, the reliability of the final system given in Fig. 4 is

$$R_{sys} = \left[1-\left(1-\left(1-\prod_{j=1}^{3}\left(1-R_{1,1}^j\right)\right)\left(R_{1,2}^1\right)\right)\right.$$
$$\left.\times\left(1-\left(1-\prod_{j=1}^{2}\left(1-R_{1,1}^j\right)\right)\left(1-\prod_{j=1}^{2}\left(1-R_{1,2}^j\right)\right)\right)\right]. \tag{2}$$

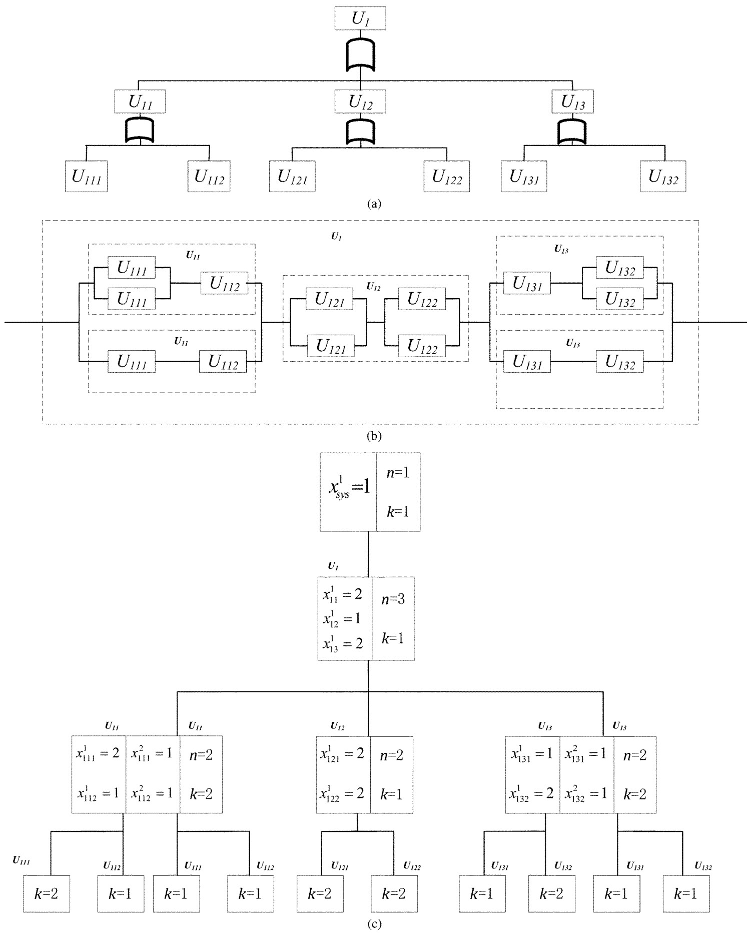

Fig. 5. An example of the hierarchical representation scheme on a tri-level serial system: (a) basic configuration; (b) system based on the redundancy allocation on (a); (c) hierarchical representation.

The cost of the system is another important issue in the reliability optimization field. It is usually considered as the constraint in the design phase of a system (i.e., the total cost of a system should not exceed a predefined value). In a traditional RAP, the cost of a system is simply the aggregate of the cost of each component in the system. However, in a multi-level system, additional costs need to be taken into account to reflect the hierarchical structure. Specifically, such additional cost is introduced in the second lowest level, and the cost of a unit U_i in a multi-level serial system can thus be calculated as

$$C_i = \begin{cases} \sum_{m=1}^{n_i} \sum_{j=1}^{x_{i,m}} C^j_{i,m}, & \text{if } U_i \text{ is not at the second lowest level} \\ \sum_{m=1}^{n_i} \sum_{j=1}^{x_{i,m}} C^j_{i,m} + {\lambda_{i,m}}^{x_{i,m}}, & \text{if } U_i \text{ is at the second lowest level} \end{cases} \quad (3)$$

For example, the cost of the final system in Fig. 4 is

$$C_{sys} = \left(\sum_{j=1}^{3} C^j_{1,1} + {\lambda_{1,1}}^3 + C^1_{1,2} + \lambda_{1,2} \right) + \left(\sum_{j=1}^{2} C^j_{1,1} + {\lambda_{1,1}}^2 + \sum_{j=1}^{2} C^j_{1,2} + {\lambda_{1,2}}^2 \right). \quad (4)$$

Let $\mathbf{x}$ be the variable of an MLRAP, consisting of all the $x_{i,m}$ that state the redundancy of all the units in the system. The MLRAP studied in this paper is formally defined as

$$\begin{aligned} \text{Maximize :} \quad & R_{sys} = R(\mathbf{x}) \\ \text{s.t.} \quad & C_{sys} = C(\mathbf{x}) \le C_0 \\ & 1 \le x_{i,m} \le p_i \end{aligned} \quad (5)$$

where C_0 is the maximum cost allowed, and p_i is the predefined maximum redundancy of U_i.

B. Solution Representation

When applying conventional GAs to an optimization problem, a solution is usually represented as a fixed-length vector. Each element of it corresponds to the value of a decision variable (e.g., the redundancy of a unit). That means the number of decision variables is fixed during the problem-solving process. In an MLRAP, the number of decision variables may vary due to the changes of the redundancy allocated to a unit. For this reason, a hierarchical structure was proposed in [5] to represent a solution to an MLRAP. This representation scheme does not restrict the number of decision variables to a fixed value, and is capable of representing all possible solutions to the MLRAP. Its advantages can be observed from [5]. Therefore, it is employed in the work here.

```
1: **Initialization**: Generate an initial population
2: **while** stopping criteria are not satisfied **do**
3:     Evaluate all individuals in the population
4:     Evolve a new population using evolutionary operators
5:     **for** Each individual **do**
6:         Perform local search around it with probability $P$
7:     **end for**
8: **end while**
```

Fig. 6. General framework of MA.

Fig. 5 illustrates an example of the hierarchical representation scheme on a tri-level serial system. The basic structure of the system is given in Fig. 5(a). A system obtained after redundancy allocation is shown in Fig. 5(b), and is represented in the hierarchical structure in Fig. 5(c). For each unit in the figure, three types of variables are given in the box associated with it. Here, k stands for the redundancy allocated to it, and n stands for the number of its child units. A variable in the form of $x_{i,m}^{e}$ represents the redundancy allocated to the mth child unit of the eth redundant unit of U_i. In other words, the variables $x_{i,m}^{e}$ are to be optimized in an MLRAP. Because the units at the lowest level (i.e., the component level) have no child unit, only the redundancy allocated to themselves are given.

III. Proposed Memetic Algorithm

An MA consists of two types of operators: the genetic operator for global exploration, and the local search operator for exploitation. Without loss of generality, the framework of MA is summarized by Fig. 6.

A. Fitness Function

How to evaluate the quality of a solution is one of the most important issues in an MA. Because an MLRAP is formulated as a constrained optimization problem, violations to the constraints must be considered together with the value of objective functions when evaluating a solution. In the literature, there are many useful constraint handling methods such as penalty techniques, repair techniques, separation techniques, and hybrid techniques [31], [32]. In this work, a penalty function proposed by Gen and Cheng [31] is employed. Concretely, the fitness function $f(\mathbf{x})$ for evaluating the quality of a solution during the search process of our MA is defined by the form

$$f(\mathbf{x}) = R(\mathbf{x}) \times \psi(\mathbf{x}), \tag{6}$$

$\psi(\mathbf{x})$ is a penalty function that measures the extent of the solution violating the constraints. It always lies in the region of [1]. The larger the value of $\psi(\mathbf{x})$, the less the solution violates the constraints.

B. Initial Population

To initialize our MA, a population of solutions (usually called individuals in the literature of MA) need to be generated first. These initial solutions are randomly generated. Like the calculation of the reliability of a system, an individual is generated in an iterative manner. This procedure starts from the system level. Given a multi-level serial system, an integer k is first generated randomly as the redundancy of the system unit. Assume each system unit consists of n child units. Then $n \times k$ integers need to be generated as the redundancy of the units at the second level. Repeating this procedure until the second lowest level, an individual will be obtained.

C. Genetic Operators

In [5], two genetic operators (i.e., crossover, and mutation) were proposed for the hierarchical solution representation. At each iteration of the algorithm, a unit in the hierarchical representation needs to be selected. Then, the two operators will be applied to this unit. Hence, the performance of the algorithm largely depends on how this unit is selected. When applying the crossover operator in [5] to two solutions, a unit will be selected only when the redundancy of its parent unit in the two solutions are the same. This approach makes it possible that some units will never be selected, while crossover is applicable to the other units at the same level. However, from the system design viewpoint, the units at the same level should be treated equally, regardless of the status of their parent units. Therefore, two new genetic operators are proposed in our paper. Both of them treat the units at the same level equally.

Crossover between two individuals is conducted through three steps. First, an intermediate level between the system and component levels is chosen based on some predefined probability. A higher level is assigned with a higher probability of being selected. After that, a unit at the selected level of each individual is chosen randomly. Finally, the chosen units, and their lower structures are exchanged to generate two new individuals. In the above second step, all units at the same level have the same probability of being selected for crossover; such a strategy is similar to the breadth-first search process. Hence, it is named the breadth-first crossover (BFC). Fig. 7 illustrates the final step of the crossover. In this case, the crossover occurs at the second level.

The mutation operator of our MA is named the breadth-first mutation operator (BFM). It also contains three steps, but is applied to a single individual. The first two steps are similar to those of the crossover operator. First, an intermediate level is selected. Then, a unit at this level is selected randomly. After that, the redundancy of the selected unit is replaced by a randomly generated integer. Because changing the redundancy of a unit will change the structure at the lower levels, the redundancy of the units involved in the changed structure is finally re-generated following the procedures used for generating initial individuals. Fig. 8 presents an example of BFM on an individual P_1. The first unit on the second level is selected as a mutation node, and the redundancy of it $(k = 2)$ is changed by a randomly generated integer number $(k = 3)$. Then, the lower

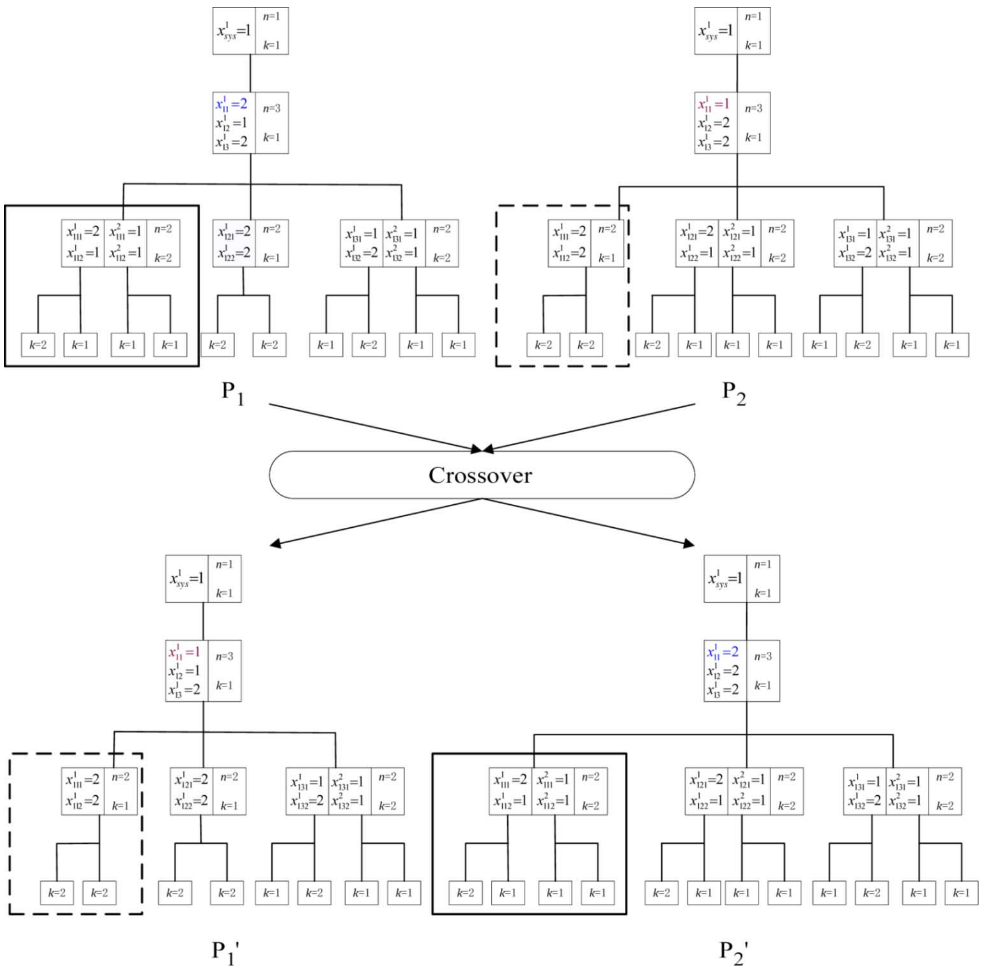

Fig. 7. An example of crossover operator between two parents (P_1 and P_2).

structure of this unit is reconstructed. In this way, a new individual P_1' is obtained. Note that when the redundancy of a unit is changed, the corresponding decision variable in its parent unit should also be updated.

D. Local Search Method

As mentioned before, local search is an important procedure in an MA. In our MA, because both BFC and BFM explore the solution space globally for promising areas, local search is indispensable to carry out exploitation (i.e., local refinement) once a promising area is reached. Specifically, local search in our MA is implemented in three steps.

First, an individual is selected from the population, and local search is carried out based on it. Concretely, prior to the local search, all individuals are evaluated by the metric

$$\gamma(\mathbf{x}) = \frac{R(\mathbf{x})}{C(\mathbf{x})}. \tag{7}$$

Then, the individual with the largest $\gamma(\mathbf{x})$ is chosen for carrying out local search. The rationale behind this strategy is that, for an MLRAP, it is desired that the optimal system be of the highest reliability, while satisfying the constraints on the cost. The smaller the cost, the better. Hence, the ratio of the reliability to cost serves as a natural metric of the potential of an individual.

Second, the local search operator is applied to the selected individual. As can be demonstrated by the descriptions of BFC and BFM in the hierarchical structure, modifications to a unit at higher levels are likely to result in more significant changes to the structure of the whole system. Local search, however, requires modifying the structure of the system slightly. Hence, our local search operator only works on the component level (i.e., the lowest level). Assume the selected individual corresponds to a system consisting of n components. The local search operator randomly selects ten pairs of components. For each pair, the redundancy of the components are modified following the schemes presented in Alg. 1. During the local search, a newly generated individual is preserved in a temporary archive if it does not violate the cost constraint. Otherwise, it is discarded.

Finally, all the new individuals preserved in the local search process are combined with the current population. All individuals in this combined population are then sorted in the descending order with respect to their fitness, which can be calculated using (6). The top individuals survive into the next generation, and the others are discarded. The pseudo-code of the local search procedure is presented in Alg. 1.

Alg. 1. Pseudo-Code of the local search procedure on an individual $\mathbf{x}$

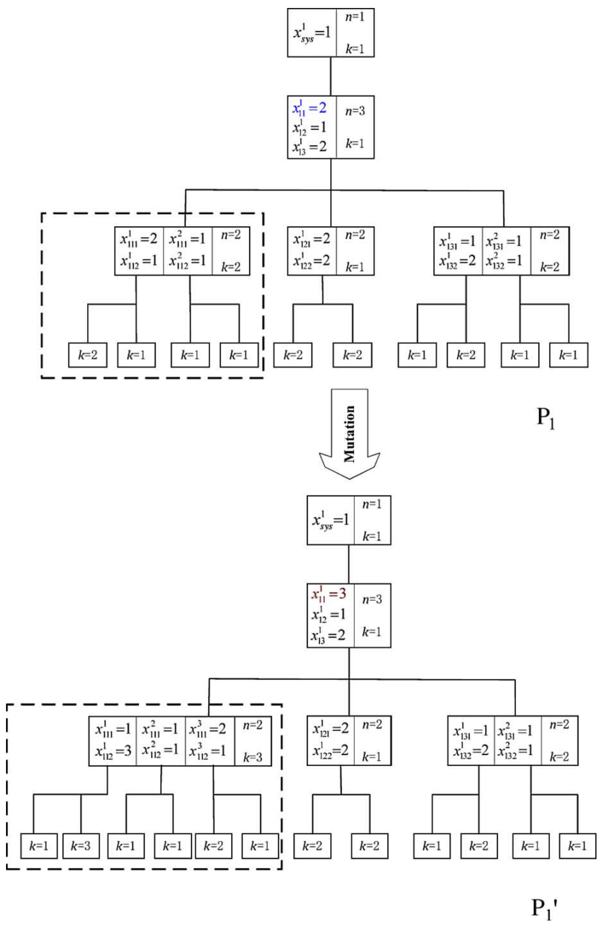

Fig. 8. An example of mutation operator on P_1.

```
1: Randomly select ten pairs of components of $\mathbf{x}$, denoted as $\{(c_{01}, c_{02}), (c_{11}, c_{12}), \ldots, (c_{91}, c_{92})\}$.
2: Initialize an archive $A = \phi$. This archive will be used to store the feasible solutions obtained by local search.
3: for $i$=0:9 do
4:     Decrease the redundancy of each components by 1 (i.e., $k(c_{i1}) = k(c_{i1}) - 1$ or $k(c_{i2}) = k(c_{i2}) - 1$) to get two new individuals $\mathbf{x}_{i1}$ and $\mathbf{x}_{i2}$.
5:     Increase the redundancy of component $c_{i1}$ by 1, and decrease the redundancy of component $c_{i2}$ by 1, then obtain a new individual $\mathbf{x}_{i3}$.
6:Increase the redundancy of component $c_{i2}$ by 1, and decrease the redundancy of component $c_{i1}$ by 1, then obtain a new individual $\mathbf{x}_{i4}$.
7:     Check whether the four new individuals violate the cost constraint, and include the ones satisfying the constraint into $A$.
end for
Output $A$.
```

Pseudo-Code of the local search procedure on an individual $\mathbf{x}$.

E. Summary of the Proposed MA

To summarize, the proposed MA starts by initializing a population of individuals. Then, new individuals are generated iteratively. At each generation, the BFC, and BFM operators are first applied to the current population of individuals, generating a set of new individuals. Then, local search is applied to a number of these new individuals to further improve them. After that, all the individuals generated by BFC, BFM, and local search are combined with the current population, and the best part of them are selected to survive into the next generation. This process is repeated for a predefined number of generations, and the best individual in the final population will be the final solution to the MLRAP. The pseudo-code of the MA is presented in Alg. 2, where $psize$, $csize$, and $msize$ are three parameters to be predefined. Variable $psize$ is the population size; $csize$, and $msize$ are the number of individuals to which the crossover, and mutation will be applied, respectively.

Alg. 2. Pseudo-Code of Our MA.

```
1: Set the parent population $X = \phi$, the offspring population $C = \phi$, and the generation counter $t = 0$.
2: Initialize the population $X$ with $psize$ individuals $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{psize}\}$.
3:while $t < t_{max}$ (maximum generation number) do
4:    Evaluate the population $X$, and sort $X$ based on the their fitness.
5:    for $i = 1 : csize/2$ do
6:        Select two individuals $\mathbf{x}_m$, and $\mathbf{x}_n$ from $X$, where $\mathbf{x}_m$ is randomly selected from the top ten individuals in $X$, and $\mathbf{x}_n$ is randomly selected from the other individuals.
7:        Apply BFC to $\mathbf{x}_m$, and $\mathbf{x}_n$ to generate two offsprings $\mathbf{x}'_m$, and $\mathbf{x}'_n$.
8:        Include $\mathbf{x}'_m$, and $\mathbf{x}'_n$ into $C$.
9:    end for
10:    for $j = 1 : msize$ do
11:        Randomly select one individual $\mathbf{x}_m$ from $X$.
12:        Apply BFM to $\mathbf{x}_m$ to generate an offspring $\mathbf{x}'_m$.
13:        Include $\mathbf{x}'_m$ into $C$.
14:    end for
15:    Sort all the individuals in $C$ with respect to (7), and select the individual with the largest $\gamma(\mathbf{x})$.
16:    Apply the local-search operator to the selected individual.
17:    Include the feasible solutions obtained in the local search into $C$.
18:    Combine $X$ and $C$, evaluate this combined population, and select the $psize$ best individuals as the new $X$.
```

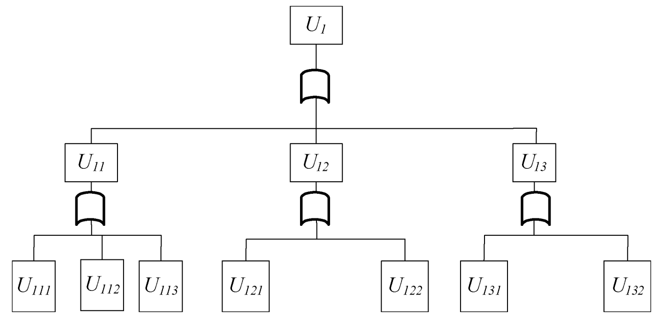

Fig. 9. The basic multi-level configuration of *Problem-A*.

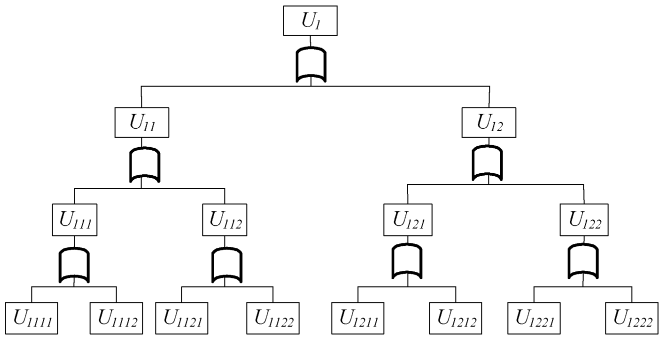

Fig. 10. The basic multi-level configuration of *Problem-B*.

```
19:     Set $C = \phi$.
20:     Set $t = t + 1$.
21: end while
22: The best individual in $X$ is the final solution.
```

IV. Empirical Studies

In this section, the performance of our MA is evaluated on two MLRAP examples. The results are compared with HGA [5], which is the most recent algorithm in the literature. The structures of the two multi-level serial systems are illustrated in Figs. 9 and 10. The first system is comprised of three levels, and the second consists of four levels. For both systems, the redundancy that can be allocated to a single unit is between one and five. That is, five choices are available for each unit. Hereafter, the MLRAP for the first system is referred to as *Problem-A*, and the MLRAP for the second system is referred to as *Problem-B*. Both of them have been used in [5] to evaluate the performance of HGA.

A. Experimental Design, and Results on Problem-A

Both HGA, and our MA involve a number of control parameters to be set in advance. Because HGA has been studied on both *Problem-A*, and *Problem-B*, the parameter settings used in [5] were used directly in our experiments. Specifically, the population size was set to 100, the maximum generation was 500, the crossover rate was set to 0.8, and the mutation rate was set to 0.05. To make a fair comparison, the same population size, and maximum generation numbers were used in our MA. At each

TABLE I
The Input Data of the Basic Multi-Level Systems on *Problem-A*, and *Problem-B*

Problem-A				*Problem-B*			
Unit	Reliability	Cost	λ	Unit	Reliability	Cost	λ
U_{111}	0.9000	5	3	U_{1111}	0.9000	7	4
U_{112}	0.9500	6	4	U_{1112}	0.8000	6	4
U_{113}	0.8500	5	4	U_{1121}	0.7500	8	4
U_{121}	0.9000	6	4	U_{1122}	0.9500	5	4
U_{122}	0.8500	7	4	U_{1211}	0.7000	9	4
U_{131}	0.9000	8	3	U_{1212}	0.9000	6	4
U_{132}	0.8000	7	4	U_{1221}	0.8500	5	4
				U_{1222}	0.8000	8	4

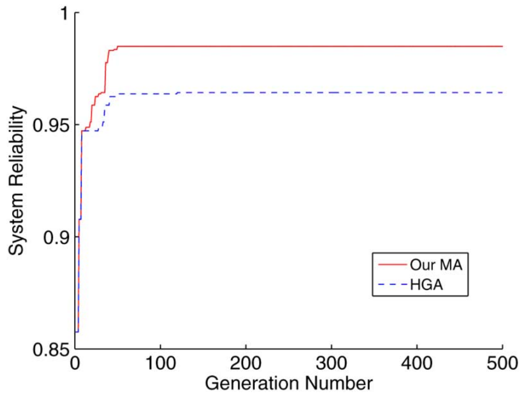

Fig. 11. Convergence plot of HGA and our MA on *Problem-A*.

generation, 50, and 10 offsprings were generated by crossover, and mutation operators, respectively. This setting was obtained on the basis of some preliminary experiments. As will be shown by the experimental results, these parameter settings works well on most test instances. Hence, we recommend them as default settings.

1) Single Case Study: In this experiment, the aim is to evaluate the convergence behavior of our MA on a case study. Because HGA has also been studied from this perspective in [5], the same settings were used in our experiment. That is, the cost constraint value (C_0) was set to 300, and the parameters of the components of the system are given in Table I. The system parameters of the other units can be calculated following the method described in Section II. The best solution obtained in each generation was recorded, and the reliability of the corresponding system was calculated. The convergence curve of both MA, and HGA can thus be plotted in Fig. 11, where the x-axis represents the number of generations, and the y-axis represents the system reliability. Both methods converged rather fast (within 100 generations), and the solution obtained by our MA was significantly better than that obtained by HGA.

2) Performance Over Different Constraint Values: The previous sub-section only provides a case study on which the MA outperformed HGA, and it is insufficient to draw a more general conclusion. Hence, further comparisons between the two methods under different conditions by varying the cost constraint values were carried out. To be specific, 20 cost constraint values were sampled between the interval 150 to 340. All the

TABLE II
RESULTS OF THE PROPOSED MA, AND HGA ON *PROBLEM-A* WITH DIFFERENT CONSTRAINT VALUES, WHERE "BEST SOLUTIONS" INDICATE THE BEST SOLUTION FOUND FROM 10 RUNS, AND "OVERALL PERFORMANCE" INDICATES THE AVERAGES OVER THE 10 RUNS

Cost Constraint	Best Solutions					Overall Performance			
	HGA		MA			HGA		MA	
	Reliability	Cost	Reliability	Cost	Detailed Solutions $[(x_1)(x_{11}x_{12}x_{13})(x_{111}x_{112}x_{113})(x_{121}x_{122})(x_{131}x_{132})]$	Mean	Variance	Mean	Variance
150	0.764087	148	0.800473	141	[(1)(122)(212)(1111)(1111)]	0.758664	2.64E-4	**0.794405**	2.53E-4
160	0.825487	152	0.840942	159	[(1)(222)(111111)(1111)(1112)]	0.814682	1.73E-4	**0.839620**	3.85E-4
170	0.846681	169	0.866762	170	[(1)(121)(222)(1111)(22)]	0.836681	2.57E-4	**0.860763**	3.02E-4
180	0.861791	173	0.878124	179	[(1)(221)(212111)(1111)(22)]	0.853320	9.64E-5	**0.876084**	1.37E-4
190	0.873937	186	0.891501	189	[(1)(121)(222)(1211)(22)]	0.872088	8.42E-5	**0.891501**	0
200	0.889995	198	0.903187	198	[(1)(221)(212111)(1211)(22)]	0.887392	8.32E-5	**0.901123**	7.32E-5
210	0.907725	208	0.921117	208	[(1)(112)(222)(22)(1122)]	0.904971	1.53E-4	**0.921117**	0
220	0.919424	217	0.937125	220	[(1)(222)(111212)(1211)(1122)]	0.915212	6.92E-6	**0.933345**	1.49E-5
230	0.927116	225	0.944680	229	[(1)(122)(222)(2211)(1122)]	0.923866	7.73E-6	**0.940280**	8.25E-6
240	0.937243	239	0.957063	238	[(1)(222)(212111)(1122)(1122)]	0.934465	4.56E-5	**0.956063**	7.34E-6
250	0.947030	250	0.962800	249	[(1)(222)(212211)(1122)(1122)]	0.947030	0	**0.959702**	5.97E-5
260	0.952400	256	0.969355	256	[(1)(222)(222111)(2211)(2211)]	0.950043	2.39E-5	**0.967522**	2.46E-5
270	0.958907	267	0.973986	269	[(1)(232)(212212)(111111)(2212)]	0.954633	4.02E-5	**0.970031**	3.58E-5
280	0.964743	279	0.979184	278	[(1)(223)(222111)(2211)(111122)]	0.963582	1.35E-5	**0.977263**	2.65E-5
290	0.970363	285	0.982124	288	[(1)(223)(212212)(1122)(111122)]	0.968372	2.83E-5	**0.979924**	5.69E-5
300	0.973004	292	0.984909	299	[(1)(233)(111222)(111122)(221111)]	0.973004	0	**0.984058**	1.36E-5
310	0.976853	307	0.986322	310	[(1)(232)(111222)(112211)(2222)]	0.974927	3.05E-6	**0.985073**	3.34E-6
320	0.980995	316	0.989283	320	[(1)(232)(212212)(221111)(2222)]	0.978425	8.27E-5	**0.989283**	0
330	0.981885	328	0.989469	325	[(1)(223)(222212)(2212)(111122)]	0.980379	4.85E-5	**0.989469**	0
340	0.983592	337	0.992975	338	[(1)(232)(222212)(221111)(2222)]	0.982274	5.24E-6	**0.992324**	1.65E-6

system parameters were kept the same as the previous sub-section. For each cost constraint value, both MA, and HGA were applied 10 times. Table II summarizes the results of this experiment. The columns headed "Best solutions" present the reliability, and cost of the best solutions obtained by the MA, and HGA in the 10 independent runs. The results in these columns are also plotted in Fig. 12. In addition, the details of these best solutions are also provided. Fig. 13 illustrates the system corresponding to the best solution obtained with the cost constraint value 150. The columns headed "Overall Performance" present the average reliability of the solutions obtained in the 10 runs. The corresponding variances are also given. Furthermore, the Wilcoxon rank sum test (with a significance level 0.05) has been employed to compare the overall performance of the two methods. For each cost constraint value, the results that are significantly better (i.e., the reliability of the system obtained) are highlighted in **boldface**.

From Table II, and Fig. 12, it is clear that the proposed MA outperformed HGA on all the 20 cases, in terms of both the quality of the best solutions, and overall performance. In particular, the best solutions found by our MA are always better than those found by HGA. The MA even achieved solutions with a higher reliability but a lower cost on five cases (i.e., the cases whose cost constraint values are 150, 240, 250, 280, and 330).

3) Comparison of Two Methods Using Different System Parameters: The third experiment was conducted to examine whether the advantage of the proposed MA holds over a variety of different system parameters. Because the major system parameters of a multi-level serial system are the parameters of the components, a set of test instances were randomly generated by

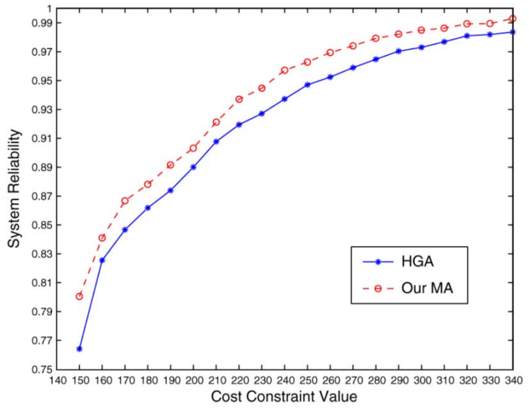

Fig. 12. Comparison between the MA and HGA under different constraint values on *Problem-A*.

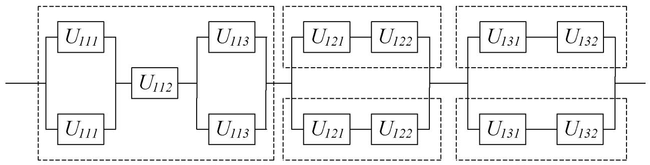

Fig. 13. The detailed solution in the first line of Table II.

modifying the reliability of the components. Specifically, the reliability of each component was randomly chosen from the set {0.80, 0.85, 0.90, 0.95}. Ten test instances were obtained in this way; and both the MA, and HGA were applied to each instance for ten independent runs. The best results, and average results over the ten runs are presented in Table III. The

TABLE III
RESULTS OF THE PROPOSED MA, AND HGA ON *Problem-A* WITH DIFFERENT SYSTEM PARAMETERS, WHERE "BEST SOLUTIONS" INDICATE THE BEST SOLUTION FOUND FROM 10 RUNS, AND "OVERALL PERFORMANCE" INDICATES THE AVERAGES OVER THE 10 RUNS

Case	Best Solutions				Overall Performance			
	HGA		MA		HGA		MA	
Number	Reliability	Cost	Reliability	Cost	Mean	Variance	Mean	Variance
1	0.973004	292	0.984909	299	0.962134	7.43E-4	**0.982537**	1.46E-5
2	0.975782	293	0.983661	293	0.970384	4.60E-5	**0.980023**	3.25E-5
3	0.976467	300	0.984903	297	0.973712	9.48E-5	**0.981135**	5.69E-5
4	0.975613	293	0.984186	298	0.970028	4.82E-5	**0.983066**	1.47E-6
5	0.967198	294	0.978494	300	0.959832	8.78E-4	**0.972073**	1.95E-5
6	0.982991	296	0.991684	298	0.973971	3.72E-4	**0.987201**	5.82E-4
7	0.969423	292	0.981200	300	0.959620	2.63E-4	**0.975923**	6.92E-5
8	0.974554	299	0.983096	295	0.970471	5.82E-4	**0.983096**	0
9	0.950217	293	0.970801	299	0.938612	8.54E-3	**0.966396**	3.71E-5
10	0.923802	299	0.948863	298	0.912242	3.82E-4	**0.942201**	6.92E-5

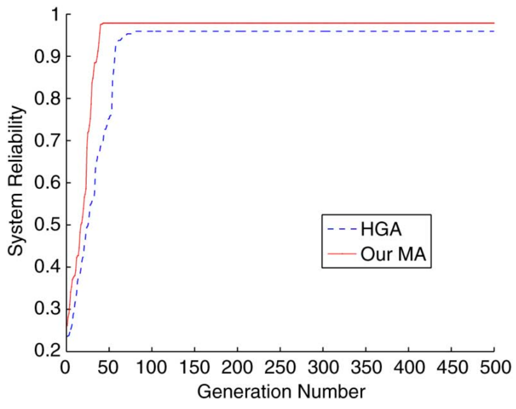

Fig. 14. Convergence plot of HGA, and our MA on *Problem-B*.

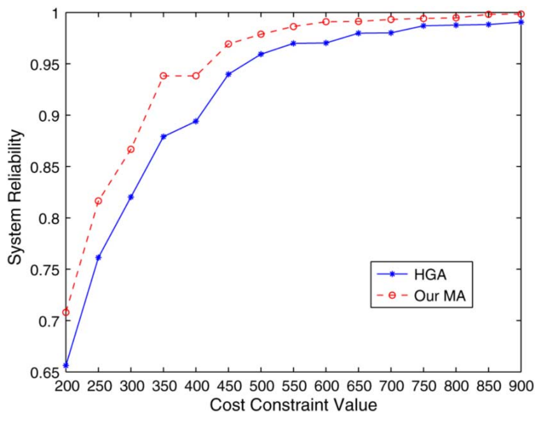

Fig. 15. Comparison between the MA and HGA under different constraint values on *Problem-B*.

TABLE IV
RESULTS OF THE PROPOSED MA, AND HGA ON *Problem-B* WITH DIFFERENT CONSTRAINT VALUES, WHERE "BEST SOLUTIONS" INDICATE THE BEST SOLUTION FOUND FROM 10 RUNS, AND "OVERALL PERFORMANCE" INDICATES THE AVERAGES OVER THE 10 RUNS

Cost	Best Solutions					Overall Performance			
	HGA		MA			HGA		MA	
Constraint	Reliability	Cost	Reliability	Cost	Detailed Solutions $[(x_1)(x_{11}x_{12})(x_{111}x_{112})(x_{121}x_{122})$ $(x_{1111}x_{1112})(x_{1121}x_{1122})$ $(x_{1211}x_{1212})(x_{1221}x_{1222})]$	Mean	Variance	Mean	Variance
200	0.656196	190	0.708032	193	[(1)(11)(22)(22) (1111)(1111)(2111)(1111)]	0.613872	2.73E-4	0.652099	9.26E-4
250	0.761348	244	0.816424	250	[(1)(11)(22)(22) (1111)(1121)(1121)(1122)]	0.728640	3.59E-4	0.755391	5.29E-4
300	0.820286	300	0.866775	300	[(1)(11)(21)(22) (1212)(22)(2211)(1122)]	0.794485	2.84E-4	**0.837821**	1.84E-4
350	0.879036	345	0.938285	350	[(1)(11)(32)(22) (111111)(2121)(2221)(2212)]	0.848625	8.36E-5	**0.896301**	2.85E-5
400	0.894051	397	0.938241	399	[(1)(21)(1111)(32) (22)(22)(22)(11)(112211)(2122)]	0.873082	5.62E-5	**0.913927**	6.73E-6
450	0.939884	449	0.969320	438	[(1)(11)(22)(32) (2212)(2222)(222111)(2122)]	0.925863	2.95E-6	**0.960071**	2.54E-6
500	0.959363	493	0.978447	492	[(1)(11)(22)(22) (2212)(2222)(2132)(2222)]	0.949327	3.74E-5	**0.971538**	4.82E-5
550	0.969910	547	0.986362	539	[(1)(11)(23)(33) (2222)(212121)(222112)(111132)]	0.964592	8.63E-6	**0.983201**	7.74E-6
600	0.970285	583	0.990953	597	[(1)(11)(33)(33) (122221)(212121)(222112)(221132)]	0.969342	4.82E-5	**0.988241**	5.38E-6
650	0.979855	644	0.991272	643	[(1)(11)(22)(32) (2222)(3222)(321121)(2232)]	0.976292	2.85E-6	**0.990735**	3.82E-6
700	0.980132	694	0.993212	699	[(1)(11)(32)(22) (222222)(3222)(3222)(2223)]	0.979821	1.84E-5	**0.992402**	1.34E-5
750	0.987066	739	0.994254	744	[(1)(11)(33)(32) (222212)(322111)(222231)(1133)]	0.982074	2.56E-6	**0.993225**	1.82E-6
800	0.987723	797	0.994736	800	[(1)(11)(33)(32) (112222)(322112)(223231)(1133)]	0.986392	7.32E-6	**0.994736**	0
850	0.988248	835	0.998219	848	[(1)(12)(33)(1232) (222222)(323211)(22)(1222)(122222)(1222)]	0.987359	2.79E-6	**0.996497**	2.14E-6
900	0.990542	897	0.998399	883	[(1)(11)(33)(1232) (222222)(322221)(23)(2222)(2222)(1222)]	0.988321	4.62E-6	**0.997921**	1.73E-7

Wilcoxon rank sum test (with a significance level 0.05) was employed to compare the two methods. For each instance, the average result of the significantly better method is highlighted in **boldface**. It is shown that the proposed MA also consistently outperformed HGA in this scenario.

B. *Experimental Design, and Results on Problem-B*

The experiments on *Problem-B* were designed in exactly the same way as those on *Problem-A*. That is, the proposed MA was compared to HGA from three perspectives: a single case study, the performance over different constraint values, and the performance on test instances with different system parameters. For the first experiment, the cost constraint value (C_0) was set to 500 following the previous work [5], and the system parameters are given in Table I. The maximum generation number was set to 500 for both methods. The results of this experiment are presented in Fig. 14. For the second experiment, 15 different cost constraint values were sampled between the interval 200 to 900. The same system parameters as in Table I were used. The results are presented in Table IV, and Fig. 15 in the same form as Table II, and Fig. 12, respectively. For the third experiment, ten test instances were generated by randomly setting the reliability of each component to a value of the set $\{0.70, 0.75, 0.80, 0.85, 0.90\}$. Both the MA, and HGA were run on these instances for ten times. The results are presented in Table V. In general, similar conclusions can be drawn from the results on *Problem-B*, and those on *Problem-A*. Therefore, the superiority of the proposed MA is further demonstrated.

TABLE V
RESULTS OF THE PROPOSED MA, AND HGA ON *Problem-B* WITH DIFFERENT SYSTEM PARAMETERS, WHERE "BEST SOLUTIONS" INDICATE THE BEST SOLUTION FOUND FROM 10 RUNS, AND "OVERALL PERFORMANCE" INDICATES THE AVERAGES OVER THE 10 RUNS

Case	Best Solutions				Overall Performance			
	HGA		MA		HGA		MA	
Number	Reliability	Cost	Reliability	Cost	Mean	Variance	Mean	Variance
1	0.950429	500	0.964249	499	0.947293	3.64E-5	0.952084	1.41E-4
2	0.951790	499	0.955361	500	0.942630	6.42E-5	**0.950388**	6.84E-5
3	0.961935	485	0.987239	495	0.958302	4.74E-5	**0.980582**	3.65E-5
4	0.954075	497	0.962872	500	0.952205	5.72E-6	**0.960353**	9.64E-6
5	0.950373	476	0.963714	500	0.948291	8.75E-5	**0.961842**	4.62E-5
6	0.967201	491	0.989973	496	0.962863	6.28E-5	**0.984032**	1.64E-5
7	0.959306	492	0.970569	495	0.954032	2.39E-5	**0.968145**	2.77E-6
8	0.964437	497	0.981231	500	0.964437	0	0.972845	9.49E-5
9	0.944596	489	0.976990	495	0.940382	4.18E-5	**0.942990**	5.52E-5
10	0.956614	478	0.979008	495 0	0.953214	2.39E-5	**0.975843**	2.53E-5

V. CONCLUSIONS

In the literature, the RAP has been intensively investigated on single-level systems. However, practical systems usually have multiple levels. Limited studies have been conducted in this scenario. In this paper, the RAP for multi-level systems was investigated, and a novel MA was proposed to solve this type of problem. Two new genetic operators (BFC, and BFM), and a local search method have been proposed based on the hierarchical solution representation used in [5]. The proposed MA combines the advantages of the two genetic operators with the local search method to maintain a good trade-off between local exploitation and global exploration. Experimental studies on two examples demonstrated that our MA consistently outperformed a state-of-the-art algorithm named HGA [5].

Two issues deserve further discussion and investigation. First, in the experimental study, only multi-level serial systems were considered. Changing the structure of a system will lead to different reliability and cost functions of the corresponding optimization problem. Because the proposed MA has been shown to be effective on multi-level serial systems, it should be extended to multi-level systems with more complex structures in the future. Second, the studied MLRAP is just a single-objective problem with the only goal to maximize the system reliability, while the designing cost is considered as a constraint. In fact, a multi-objective problem can be formulated, which aims to seek multiple solutions that represent different trade-off between the system reliability and cost. Such a problem definition, as well as the problem-solving approach, will be investigated in the future.

REFERENCES

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

[2] W. Kuo and R. Wang, "Recent advances in optimal reliability allocation," *IEEE Trans. Systems, Man, and Cybernetics–Part A: Systems and Humans*, vol. 37, no. 2, pp. 143–156, 2007.

[3] D. E. Fyffe, W. W. Hines, and N. K. Lee, "System reliability allocation and a computational algorithm," *IEEE Trans. Reliability*, vol. 17, no. 2, pp. 64–69, Jun. 1968.

[4] 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*, Los Angeles, California, Jan. 26–29, 2004, pp. 6–11.

[5] R. Kumar, K. Izui, M. Yoshimura, and S. Nishiwaki, "Multilevel redundancy allocation optimization using hierarchical genetic algorithm," *IEEE Trans. Reliability*, vol. 57, no. 4, pp. 650–661, Dec. 2008.

[6] M. S. Chen, "On the computational complexity of reliability redundancy allocation in a series system," *Operations Research Letters*, vol. 11, no. 5, pp. 309–315, 1992.

[7] M. Djerdjour and K. Rekab, "A branch and bound algorithm for designing reliable systems at a minimum cost," *Applied Mathematics and Computation*, vol. 118, no. 2/3, pp. 247–259, 2001.

[8] C. Ha and W. Kuo, "Reliability redundancy allocation: An improved realization for nonconvex nonlinear programming problems," *European Journal of Operational Research*, vol. 171, no. 1, pp. 24–38, 2006.

[9] V. R. Prasad and W. Kuo, "Maximization of a percentile life of a series system through component redundancy allocation," *IIE Trans.*, vol. 33, no. 12, pp. 1071–1079, 2001.

[10] J. Onishi, S. Kimura, R. J. W. James, and Y. J. Nakagawa, "Solving the redundancy allocation problem with a mix of components using the improved surrogate constraint method," *IEEE Trans. Reliability*, vol. 56, no. 1, pp. 94–101, Mar. 2007.

[11] H. Lee, W. Kuo, and C. Ha, "Comparison of max-min approach and NN method for reliability optimization of series-parallel systems," *Journal of System Science and System Engineering*, vol. 12, no. 1, pp. 39–48, 2003.

[12] J. E. Ramirez-Marques, D. W. Coit, and A. Konak, "Redundancy allocation for series-parallel systems using a max-min approach," *IIE Trans.*, vol. 36, no. 9, pp. 891–898, 2004.

[13] K. Y. K. Ng and N. G. F. Sancho, "A hybrid dynamic programming/depth-first search algorithm with an application to redundancy allocation," *IIE Trans.*, vol. 33, no. 12, pp. 1047–1058, 2001.

[14] A. Yalaoui, E. Chatelet, and C. Chu, "A new dynamic programming method for reliability and redundancy allocation in parallel-series system," *IEEE Trans. Reliability*, vol. 54, no. 2, pp. 254–261, Jun. 2005.

[15] J. E. Ramirez-Marques and D. W. Coit, "A heuristic for solving the redundancy allocation problem for multi-state series-parallel system," *Reliability Engineering and System Safety*, vol. 83, no. 3, pp. 341–349, 2004.

[16] C. Elegbede, C. Chu, K. Adajallah, and F. Yalaoui, "Reliability allocation through cost minimization," *IEEE Trans. Reliability*, vol. 52, no. 1, pp. 106–111, Mar. 2003.

[17] D. W. Coit and A. E. Smith, "Genetic algorithm to maximize a lower-bound for system time-to-failure with uncertain component Weibull parameters," *Computers and Industrial Engineering*, vol. 41, no. 4, pp. 423–440, 2002.

[18] G. Levitin and A. Lisnianski, "Optimal separation of elements in vulnerable multi–state systems," *Reliability Engineering and System Safety*, vol. 73, no. 1, pp. 55–66, 2001.

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

[20] Z. Wang, T. Chen, K. Tang, and X. Yao, "A multi-objective approach to redundancy allocation problem in parallel-series systems," in *Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC2009)*, Trondheim, Norway, May 18–21, 2009, pp. 582–589.

[21] G. Levitin, "Optimal allocation of multi-state elements in a linear consecutively–connected system," *IEEE Trans. Reliability*, vol. 52, no. 2, pp. 192–199, Jun. 2003.

[22] Y. C. Liang and A. E. Smith, "An ant colony optimization algorithm for the redundancy allocation problem," *IEEE Trans. Reliability*, vol. 53, no. 3, pp. 417–423, Sep. 2004.

[23] N. Wattanapongsakorn and S. P. Levitan, "Reliability optimization models for embedded systems with multiple applications," *IEEE Trans. Reliability*, vol. 53, no. 3, pp. 406–416, Sep. 2004.

[24] N. Nahas, M. Nourelfath, and D. Ait-Kadi, "Coupling ant colony and the degraded ceiling algorithm for the redundancy allocation problem of series–parallel systems," *Reliability Engineering and System Safety*, vol. 92, no. 2, pp. 211–222, 2007.

[25] M. Nourelfath and D. Ait-Kadi, "Optimization of series–parallel multi–state systems under maintenance policies," *Reliability Engineering and System Safety*, vol. 92, no. 12, pp. 1620–1626, 2007.

[26] N. Nahas, M. Nourelfath, and D. Ait-Kadi, “Ant colonies for structure optimisation in a failure prone series–parallel production system,” *Journal of Quality in Maintenance Engineering*, vol. 14, no. 1, pp. 7–33, 2008.

[27] G. Levitin, “Optimal multilevel protection in serial-parallel systems,” *Reliability Engineering and System Safety*, vol. 81, no. 1, pp. 93–102, 2003.

[28] W. Y. Yun and J. W. kim, “Multilevel redundancy optimization in series systems,” *Computers and Industrial Engineering*, vol. 46, pp. 337–346, 2004.

[29] P. Moscato, On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithm CalTech, Pasadena, CA, CalTech Concurrent Computation Program Report 826, 1989.

[30] N. Krasnogor and J. Smith, “A tutorial for competent memetic algorithms: Model, taxonomy, and design issues,” *IEEE Trans. Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, 2005.

[31] 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.

[32] C. A. C. Coello, “Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art,” *Comput Methods Appl Mech Eng*, vol. 8, no. 2, pp. 1245–1287, 2002.

Zai Wang (S’09) received the B.S. degree in computer science from the School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China, in 2006; and is currently working toward the Ph.D. degree at the Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, USTC.

Ke Tang (S’05–M’07) received the B.Eng. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2002; and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2007.

Since 2007, he has been an Associate Professor with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. He is the coauthor of more than 40 refereed publications. His major research interests include machine learning, pattern analysis, evolutionary computation, data mining, metaheuristic algorithms, and real-world applications.

Dr. Tang is an Associate Editor of IEEE Computational Intelligence Magazine, editorial board member of three international journals, and the Chair of IEEE Task Force on Large Scale Global Optimization.

Xin Yao (M’91–SM’96–F’03) received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, China, in 1982; the M.Sc. degree from the North China Institute of Computing Technology, Beijing, China, in 1985; and the Ph.D. degree from USTC in 1990.

From 1985 to 1990, he was an Associate Lecturer and Lecturer with USTC, while working toward the Ph.D. degree in simulated annealing and evolutionary algorithms. In 1990, he was a Postdoctoral Fellow with the Computer Sciences Laboratory, Australian National University, Canberra, Australia, where he continued his work on simulated annealing and evolutionary algorithms. In 1991, he was with the Knowledge-Based Systems Group, Commonwealth Scientific and Industrial Research Organization, Division of Building, Construction and Engineering, Melbourne, Australia, where he worked primarily on an industrial project on automatic inspection of sewage pipes. In 1992, he returned to Canberra to take up a lectureship in the School of Computer Science, University College, University of New South Wales, Australian Defense Force Academy, Sydney, Australia, where he was later promoted to a Senior Lecturer and Associate Professor. Since April 1999, he has been a Professor (Chair) of computer science in the University of Birmingham, Birmingham, U.K. He is currently the Director of the Center of Excellence for Research in Computational Intelligence and Applications, School of Computer Science, University of Birmingham, Birmingham, U.K. and also a Changjiang (Visiting) Chair Professor (Cheung Kong Scholar) with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, USTC. He has given more than 50 invited keynote and plenary speeches at conferences and workshops worldwide. He has more than 300 referenced publications. His major research interests include evolutionary artificial neural networks, automatic modularization of machine learning systems, evolutionary optimization, constraint-handling techniques, computational time complexity of evolutionary algorithms, coevolution, iterated prisoner’s dilemma, data mining, and real-world applications.

Dr. Yao was the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2003 to 2008, an Associate Editor or editorial board member of 12 other journals, and the Editor of the World Scientific Book Series on Advances in Natural Computation. He was the recipient of the President’s Award for Outstanding Thesis by the Chinese Academy of Sciences for his Ph.D. work on simulated annealing and evolutionary algorithms in 1989. He was the recipient of the 2001 IEEE Donald G. Fink Prize Paper Award for his work on evolutionary artificial neural networks.